UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Módulo Gestión del Control Interno para el Sistema de gestión de los procesos administrativos de CEIGE

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

Autor: Yoandry Freire Pérez
Tutores: Ing. Raymond Weeden Gamboa
Ing. Mailín Ayala Rodríguez
Ing. Leidy Ramos González

La Habana, 2017
"Año 58 de la Revolución"

AGRADECIMIENTOS Y DEDICATORIA

Agradezco a:

A todos los que me apoyaron y me dieron la posibilidad de estar hoy aquí.

A mis tutores, por el apoyo brindado, sin importar la hora o las cosas mal que hacía, por aguantar los dolores de cabeza. A ustedes muchas gracias, si hoy estoy aquí se lo debo a ustedes, nunca los voy a olvidar.

A la familia, por estar siempre presente en el día a día, molestándome en el teléfono para saber como me iba.

A mis amigos por demostrar su preocupación cuando me veían decaído.

Gracias a todos.

Dedico a:

A la Pelo una vieja amiga que siempre estuvo presente, enviándome mensajes oportunos y sacándome una sonrisa cuando realmente estaba punto de romper en llanto.

En especial, se lo dedico a mi vieja que siempre se preguntaba si estaría aquí este día. Aquí estamos mi viejita y seguiremos estando por más tiempo. Te quiero mucho.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yoandry Freire Pérez
_____
Autor

Ing. Raymond Weeden Gamboa
_____
Tutor

Ing. Mailín Ayala Rodríguez
_____
Tutor

Ing. Leidy Ramos González
_____
Tutor

DATOS DE CONTACTO

Autor: Yoandry Freire Pérez
Correo electrónico:
yfreire@estudiantes.uci.cu

Tutor: Ing. Mailín Ayala Rodríguez
Correo electrónico: mayala@uci.cu

Tutor: Ing. Raymond Weeden Gamboa
Correo electrónico: rweeden@uci.cu

Tutor: Ing. Leidy Ramos González
Correo electrónico: lramosg@uci.cu

INDICE

RESUMEN

La presente investigación comprende los procesos de control interno (Medios Básicos, Dietas y Útiles-Herramientas) en el Centro de Informatización de Entidades. En la actualidad el proceso de control de los útiles se realiza de forma manual, incurriendo en altos tiempos de ejecución. Además el registro de las dietas se realiza de forma personal, lo que genera gran cantidad de documentos almacenados teniendo en cuenta el número de especialistas que reciben las mismas. Existe entonces la necesidad de buscar una solución que integre e informatice estos procesos, valiéndose de técnicas y tecnologías actuales. Luego de obtener los datos mediante entrevistas y estudio de sistemas existentes, se trazó como objetivo desarrollar un módulo para la gestión y control de dietas, medios básicos y útiles-herramientas. Se utilizó como guía para el trabajo la metodología de desarrollo de software AUP-UCI y como principal tecnología de desarrollo el marco de trabajo Symfony 2.8. El sistema fue validado mediante las pruebas de caja negra y caja blanca así como por la aceptación satisfactoria del cliente.

Palabras clave: dietas, gestión, herramientas, medios básicos, útiles.

INTRODUCCIÓN

En el entorno empresarial y de negocios el interés de los directivos por una administración eficiente ha traído incontables procesos de transformación. Los valores del mercado, la evolución acelerada del mundo tecnológico y la presión constante de la competencia han provocado grandes cambios en el ambiente de control como uno de los 5 componentes principales del control interno de toda institución. Los factores anteriormente enunciados tienen cierta influencia, ya sea directa o indirectamente, en los procesos, operaciones y actividades que desarrolla cualquier organización y por ello están sujetas a la posibilidad de ocurrencia de hechos o acontecimientos como: las fisuras en el diseño y funcionamiento en las organizaciones; negligencias en los funcionarios en cuanto al control y custodia de los recursos del Estado permitiendo la ocurrencia de presuntos hechos delictivos y de corrupción; escapes y la proliferación de hechos delictivos; etc. Todo esto afectando el desarrollo de los objetivos empresariales de cada empresa como entidad social.

A partir de los cambios surgidos en los Lineamientos de la Política Económica y Social del Partido, el país se encuentra en constante actualización de su modelo económico, por lo que resulta imprescindible ser rigurosos con el respeto a la legalidad y la actuación ética de cuadros y trabajadores. Por ello se debe continuar fortaleciendo el Sistema de Control Interno en aras de obtener los resultados esperados con eficiencia, orden y disciplina, entre ellos, el referido al plan de la economía y el presupuesto. A partir de las Normas del Sistema de Control Interno aprobadas mediante la Resolución No. 60/201 por la Contraloría General de la República; es necesario que todas las entidades logren adecuar y aplicar su Guía de Autocontrol en correspondencia con tales normativas. Este documento es una herramienta de trabajo, donde se relacionan también las principales disposiciones vigentes para que los cuadros de conjunto con todos los trabajadores, implementen su Sistema de Control Interno armonizado por componentes y normas (1). Dentro del conjunto de actividades que son necesarias conocer y dominar en la dirección de una empresa, ocupa un lugar importante el Control Interno, el cual reúne los requerimientos fundamentales de todas las especialidades contables, financieras, administrativas, técnicas, productivas y de cuantas actividades posea la empresa, además incluye el plan de organización, todos los métodos coordinados y medidas adoptadas (2).

En el año 2002 surge la Universidad de las Ciencias Informáticas (UCI) como modelo de universidad productiva, con la misión de formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática con carácter dinámico, flexible y creativo (3). Los centros de desarrollo son parte de la estructura de la UCI. Tienen como misión fundamental el desarrollo de aplicaciones y servicios informáticos para colaborar en la solución de necesidades y problemas

existentes en la sociedad; orientada por el planteamiento del Comandante cuando expresa que la idea fundamental es que se convierta la informática en la rama más productiva, aportadora de recursos para la nación es el empleo a fondo de la inteligencia y del capital humano que tenemos y, principalmente del que se puede crear casi como espina dorsal de la economía. (4).

El Centro de Informatización de Entidades (CEIGE) tiene como misión desarrollar productos, servicios y soluciones informáticas para la gestión de entidades, contribuyendo a la formación de estudiantes y profesionales, permitiendo un posicionamiento en el mercado nacional e internacional (5). Para su desarrollo rige su trabajo por lo establecido en la Contraloría General de la República respecto al Control Interno trabajando de forma sistemática en el mismo. Su visión se enmarca en ser un centro de referencia nacional en el desarrollo de soluciones para la gestión de entidades, certificado en CMMI Nivel 3.

CEIGE cuenta con una estructura organizacional distribuida de la siguiente forma:

• Dirección
• Subdirección de I+D+I
• Departamento de Práctica Profesional (PP)
• Departamento de Desarrollo de Componentes (DC)
• Departamento de Aplicaciones de Gestión Empresarial (AGE)
• Departamento de Aplicaciones Financieras y Aduanales (AFA)

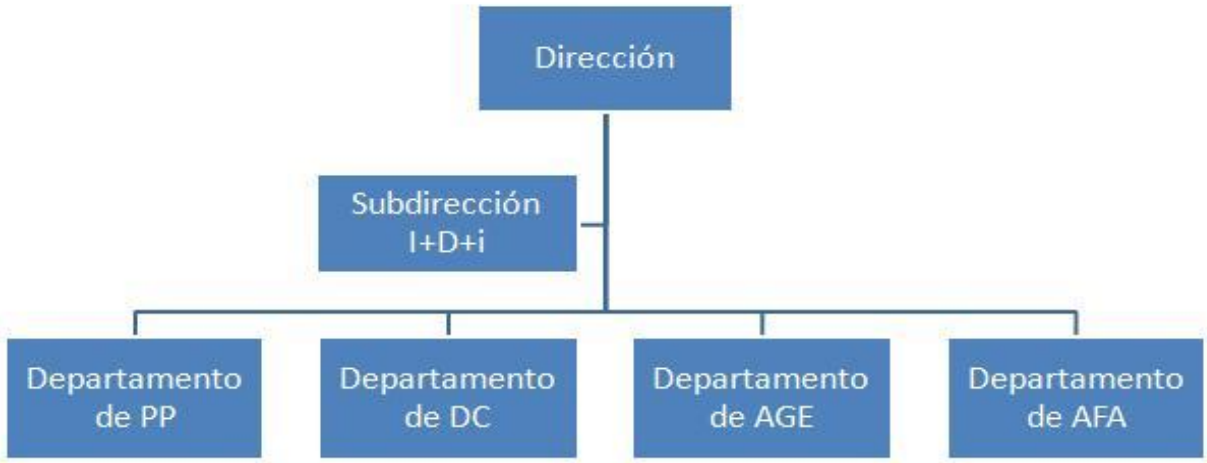

Figura 1. Estructura del CEIGE

El centro para su funcionamiento desarrolla un conjunto de procesos transversales que involucran a todos los departamentos. Los procesos fundamentales son los siguientes:

- Gestión del control interno
- Gestión de los técnicos de laboratorio

- Gestión de la planificación y control de los proyectos
- Gestión de la mercadotecnia

El centro por la amplitud y la gama de procesos que en él concurren, tiene la necesidad de agilizarlos con el fin de controlar de forma más rápida y certera la información que se maneja en cada uno de ellos. Los procesos de gestión de control interno constituyen los más engorrosos y difíciles de procesar porque se hace a través de una comprobación manual de medios, útiles y herramientas.

Actualmente CEIGE presenta deficiencias respecto a tiempo de ejecución en los procesos de gestión de control interno, dietas, medios básicos y útiles, donde el cúmulo de información que se maneja en el desarrollo de los mismos los transforma en tareas engorrosas prolongando sus tiempos de ejecución considerablemente.

Durante el estudio de estos procesos de gestión se detectaron los siguientes problemas:

- El control de los medios básicos se hace de forma manual utilizando un formato duro, lo que implica una prolongación notable del tiempo de ejecución del mismo, involucrando como mínimo dos personas para cada control.
- El número de medios que se encuentra en un local del centro es relativamente grande, necesitando comparar cada medio registrado en el papel con el existente en el local, esto propicia en gran medida la posibilidad de error además que extiende innecesariamente el tiempo de ejecución del proceso.
- El control de cada uno de los activos es engorroso dado que este proceso se efectúa a través de su identificador (cadena extensa Ejemplo: MB00102169). Este proceso de control también incluye el registro de entrada, salida, alta o traslado de cada medio, lo que puede provocar duplicidad en los registros de los medios si no se actualiza manualmente cada uno de los papeles: origen y destino de dicho medio en movimiento, siendo posible el movimiento de estos en cualquier momento de su vida útil.
- La consultoría de los registros de los medios básicos se convierte en un proceso difícil teniendo en cuenta la cantidad de documentos a filtrar.
- El proceso de dietas se hace a través de un registro mensual para todas las dietas emitidas siendo de forma independiente para cada mes. El registro de cada dieta se efectúa por persona, lugar al cual se va a destinar la dieta, cantidad de efectivo asignado y demás campos necesarios para llevar un control y emitir partes sobre el cumplimiento de las mismas. Esto genera incontables documentos almacenados teniendo en cuenta el número de especialistas que reciben las mismas.

- El proceso de control de los útiles de igual manera se hace de forma manual, incurriendo en altos tiempos de ejecución. Estos poseen un indicador emitido por el centro CEIGE para cada uno, por el cual son controlados porque no tienen número de inventario

establecido solo poseen un código para destacar el tipo de producto y la cantidad. Resultando ser un proceso difícil para la revisión, control, entrega y movimiento de cada uno de ellos.

Partiendo de lo anteriormente expresado se define el siguiente **problema a resolver:** ¿Cómo disminuir el tiempo de ejecución de los procesos control interno de forma eficaz del Centro de Informatización de Entidades?

A partir del problema se define como **objeto de estudio**: Los procesos de control interno (Medios Básicos, Dietas y Útiles).

Teniendo en cuenta el objeto de estudio se tiene como **campo de acción:** Los procesos control interno en el Centro de Informatización de Entidades.

Se define como **objetivo general:** Desarrollar un módulo para optimizar el tiempo de ejecución de los procesos de control interno del Centro de Informatización de Entidades.

Con el fin de dar cumplimiento al objetivo general se definen los siguientes **objetivos específicos:**

- Elaborar el marco teórico relativo a los sistemas de gestión contables de dietas y medios básicos para obtener un estado del arte que permita establecer el precedente de la investigación.
- Realizar la ingeniería de requisitos para lograr una mayor capacidad de predecir cronogramas de proyectos, así como sus resultados, proporcionando un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- Implementar la solución modelada de manera que cumpla con las necesidades del cliente.
- Validar la propuesta de solución en función de corroborar su correspondencia con los requisitos del cliente y el cumplimiento del objetivo general de la investigación.

Se determina como **idea a defender**: Si se desarrolla una aplicacón que informatice los procesos de gestión contables: Medios Básicos y Dietas en el Centro de Informatizaciòn de Entidades, entonces se optimizará el tiempo de ejecución de estas actividades.

Para dar cumplimiento a los objetivos planteados se emplearon los siguientes **métodos científicos**:

- ✓ **Teóricos**

- **Analítico-Sintético:** Permitió realizar un estudio de las herramientas, tecnologías y metodologías a emplear para el desarrollo de la solución al problema a resolver.
- **Inductivo-Deductivo:** para la identificación de la problemática, así como sus variantes de solución.
- **Modelación:** A través de este método se crearon modelos con vistas a representar los principales conceptos y elementos a tener en cuenta para resolver el problema. Así como el modelado de esquemas y diagramas que permiten obtener una mejor representación del objeto de estudio.

✓ **Empíricos**

- **Análisis Documental:** Permitió realizar el análisis de la bibliografía referente a las temáticas de control interno y el marco teórico.
- **Entrevista:** Permitió a partir de la persona encargada de gestionar el control de medios básicos, dietas y útiles identificar el problema a resolver y precisar el modo de funcionamiento de la misma.

El trabajo de diploma se estructura de la siguiente forma:

Capítulo I: "FUNDAMENTACIÓN TEÓRICA"
En este capítulo se realiza un estudio de algunas soluciones informáticas correspondientes a sistemas de control interno. Además se describen las herramientas y tecnologías a utilizar para el desarrollo de la solución.

Capítulo II: "ANÁLISIS Y DISEÑO DEL SISTEMA"
En este capítulo se exponen los requisitos funcionales y no funcionales del sistema, una representación gráfica de los diagramas de clases e interacción del diseño, así como el diseño de la base de datos. Se establecen además las funcionalidades del sistema y la arquitectura que se emplea en el proyecto para la implementación de la solución informática.

Capítulo III: "IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA"
En este capítulo se generan el modelo de despliegue y los diagramas de componentes, se implementa el sistema y se concluye realizando pruebas de funcionalidad para comprobar si la aplicación cumple las expectativas del cliente.

CAPÍTULO I: “FUNDAMENTACIÓN TEÓRICA”

Introducción

En este capítulo se explica el funcionamiento de los procesos de gestión contables: medios básicos, dietas y útiles en el Centro de Informatización de Entidades. Se realiza un análisis para determinar cada una de las herramientas, tecnologías y lenguajes a emplear en la implementación del sistema, con una breve descripción que justifique la elección de las mismas.

1.1. Marco conceptual

Control Interno: En la Resolución No 60/2011, “Normas del Sistema de Control Interno”, emitida por la Contraloría General de la República, se define el control interno como el proceso integrado a las operaciones con un enfoque de mejoramiento continuo, extendido a todas las actividades inherentes a la gestión, efectuado por la dirección y el resto del personal. Se implementa mediante un sistema integrado de normas y procedimientos, que contribuyen a prever y limitar los riesgos internos y externos; proporciona una seguridad razonable al logro de los objetivos institucionales y una adecuada rendición de cuentas (1).

El control interno consta de cinco componentes que se encuentran interrelacionados entre sí:

- **Ambiente de control**: sienta las pautas para el funcionamiento legal y armónico de los órganos, organizaciones y demás entidades para el desarrollo de sus acciones que permitan reflejar la actitud asumida por la máxima dirección y el resto del colectivo en relación con la importancia del Sistema de Control Interno (1).
- **Gestión y prevención de riesgos**: toda organización debe conocer los riesgos a los que enfrenta, estableciendo mecanismos para identificarlos, analizarlos y tratarlos (1).
- **Actividades de control**: establecimiento y ejecución de las políticas y procedimientos que sirvan para alcanzar los objetivos de la organización (1).
- **Información y comunicación**: los sistemas de información y comunicación permiten que el personal capte e intercambie la información requerida para desarrollar, gestionar y controlar sus operaciones (1).
- **Supervisión y monitoreo**: para que un sistema reaccione ágil y flexiblemente de acuerdo con las circunstancias, deber ser supervisado (1).

Activos: son los bienes, derechos y otros recursos controlados económicamente por la entidad, resultantes de sucesos pasados de los que se espera obtener beneficios o rendimientos económicos en el futuro. Un activo es un bien que la entidad posee y que pueden convertirse en dinero u otros medios líquidos equivalentes (12).

1.2. Estudio de sistemas similares

A pesar de que la universidad cuenta hoy con un sistema para el procesamiento de los resultados de la aplicación de la guía de control interno y un sistema para el control de los activos fijos tangibles y útiles, no es suficiente para el trabajo que se realiza diariamente en el centro, debido a que solo se puede obtener un documento excel del mismo, sin permiso a nada más.

ASSET

Este es un sistema comercializado por la firma panameña D´MARCO S.A. y distribuido en Cuba por INFOMASTER, entidad informática perteneciente a la Empresa Nacional de Producción y Servicios a la Educación Superior del MES2 (15). ASSETS, es un sistema de gestión integral estándar y parametrizado que permite controlar, realizar y contabilizar todas las transacciones comunes relacionadas con el proceso de compra-venta, e incluye así mismo los procedimientos necesarios para registrar los movimientos de los AF y de los útiles y herramientas. El sistema permite (15):

Controlar, por centros de costos, los medios básicos. Permite definir además, la ubicación física de los mismos.

Controlar los procesos de compras, alquiler, altas, bajas, préstamo y traspasos hacia otras áreas dentro y fuera de la entidad.

Realiza reparaciones, realizar ajustes y controlar la depreciación acumulada de cada activo. Permite además realizar revalorizaciones de los activos y emitir, al cierre del mes, el comprobante de operaciones por la depreciación de los activos, cada movimiento genera automáticamente el asiento contable correspondiente.

Este sistema solo es utilizado en la universidad para los procesos centrales de la institución y no para las particularidades de cada centro. Además de ser un sistema privativo, solo permite el acceso a los módulos para los que se posee licencia.No gestiona el proceso de dietas ni está permitido su uso para centros de costos internos solo para la UCI de forma general.

SISCONT5

La historia del sistema comienza desde el año 1985 donde nace como un Sistema Uniforme de Contabilidad Automatizado, denominado SISCONT, el cual ha permitido durante todos estos años que toda la información contable que se desarrolla en el organismo se realice sobre una base uniforme, posibilitando que los análisis a los distintos niveles se puedan realizar bajo los mismos criterios de consolidación o desagregación, con la fortaleza además de mantener actualizada la contabilidad con todos los cambios que durante los últimos 20 años se han operado en la Economía Nacional (17).Los módulos en explotación: Contabilidad General, Contabilidad, Estados Financieros, Contabilidad de Costos, Recursos Materiales, Activos Fijos, Inventarios, Útiles y Herramientas, Recursos Financieros, Cobros y Pagos, Recursos Humanos, Nóminas.

Pese a la potencia presenciada en este sistema, no es viable debido que solo es compatible con Windows, no posee módulos para la gestión de dietas y es privativo. Además se comporta de forma no satisfactoria en la mayoría los indicadores evaluados en la investigación.

VERSAT SARASOLA

Software desarrollado por la entidad cubana TEICO de Villa Clara, constituye un sistema que automatiza prácticamente todas las actividades de planificación, control y análisis Económico de cualquier tipo de entidad, ya que es configurable. Se caracteriza por ser una aplicación de escritorio. Implementado en Delphi. Trabaja sobre el sistema operativo Windows y presenta como soporte para bases de datos SQL Server 2000 por lo que no cumple con los requisitos de software libre como política tecnológica por la que aboga el país (14).

SICAR (Punto de Venta)

Este sistema ofrece crecimiento continuo con herramientas sencillas y con un gran estilo. Ayuda a mejorar control de inventarios desde el primer momento. Se especializa en el control de inventarios basándose en este como el corazón de cualquier empresa que se dedique a la compra y venta de bienes o servicios. SICAR Punto de Venta permite saber cuanto entra y sale de la empresa de una forma sencilla y práctica, además de contar con herramientas que pueden ayudar a monitorear los movimientos de la mercancía. Brinda las herramientas para vender y descontar del inventario con cada venta, además de al realizar compras ingresar los productos a tu inventario. Requiere comprar la licencia de SICARP Punto de Venta (30).

Beneficios del Control de Inventarios con SICAR:

- Consigue monitorear tus inventarios de forma rápida y sencilla.
- Conocer en tiempo y forma cuál es la situación de tu empresa con un par de clicks.
- Manejar cambios de precio según las necesidades de tu empresa y mejora cotizaciones.
- Podrás realizar devoluciones de mercancía sin que te afecte en tu inventario.
- Control total de los movimientos de caja para mejorar la toma de decisiones.
- Contar con reportes que te digan cómo está tu empresa en cualquier momento.
- Podrás saber cuando pedir mercancía y que tipo de mercancía tienes que pedir.
- Comprar al mejor proveedor a los mejores precios.
- Realizar ajustes de inventarios de forma periódica.
- Tendrás reportes de cortes de caja, movimientos de caja y mucho más.

SOFTWARE PARA VIATICOS

Escaneado y digitalización de los recibos.

Permite subir los recibos de los gastos desde una aplicación móvil. Software para control de

viáticos escanea los recibos y lee todos los datos pertinentes de los gastos realizados. Puedes agregar más información del proyecto o categoría y tus gastos son creados en la plataforma.

Administración eficiente de gastos.

Utiliza tu cuenta en línea para administrar tus gastos, los avances, el kilometraje y el tiempo de seguimiento. Las PYME pueden establecer sucursales y grupos para mejorar la política de gastos de las empresas. Tendrás control de viáticos en formato de cualquier tipo.

Informes y herramientas de manera sencilla.

Genera informes de gastos de viaje de negocios y reembolso en tan sólo unos pocos clics. Exporta tus gastos a PDF, CSV y un efectivo control de viáticos en Excel con nuestro software para viáticos. Las herramientas que se manejan en el software son amigables para las empresas. Software para control de viáticos es una solución móvil para la administración de gastos basada en web revolucionaria que elimina los informes de gastos como los conocemos hoy en día. Esto mediante la automatización de todos los pasos del proceso de gestión de gastos, haciendo uso de la exploración recepción y aprobación integrada en el nivel de gasto. Ahora es posible gestionar todos los gastos diarios. Software para viáticos es una solución utilizada por muchas empresas y lo manejan, empleados, CFO y dueños de negocios. Debido a todas estas facilidades todos pueden estar involucrados en el proceso de administración de gastos, tratamos de que la vida de todos sea más sencilla. (31)

AKADEMOS (Activos)

Sistema empleado en la Universidad de Ciencias Informáticas para el proceso de elaboración de los documentos necesarios para efectuar la solicitud de gestión de medios básicos y útiles o herramientas en la universidad. Los permisos para trabajar en el sistema solo son conferidos para el director y no para aquellas personas involucradas en el proceso de medios básicos.

No tiene el módulo de gestión de dietas donde este trabajo enmarca una de sus premisas fundamentales.

En este sistema se tomaron ideas para la gestión de estos procesos y la forma de darle tratamientos a los mismos en una aplicación web. Sería una buena solución temporal para las necesidades del cliente.

Con la intención de realizar una comparación específica de estos sistemas con la necesidad del cliente se definen los siguientes indicadores: la implementación de las funcionalidades Gestión de Dietas, Medios Básicos, Útiles-Herramientas, Sistamas multiplataforma y que respondan a la política de software libre. En la siguiente tabla queda evidenciado el resultado de la comparación:

Tabla 1. Estudio de sistemas similares

	MB	U-H	Dietas	Multiplataforma	Libre
ASSET	SI	SI	-	SI	-
VERSAT SARASOLA	SI	SI	-	-	-
SISCONT5	SI	SI	-	-	-
SICAR (Punto de venta)	SI	SI	-	SI	-
SOFTWARE PARA VIÁTICOS	-	-	SI	SI	-
AKADEMOS (Activos)	SI	SI	-	SI	-

Luego de haber realizado un estudio de los sistemas descritos anteriormente con el objetivo de analizar la forma en que se gestionan los procesos de Gestión de medios básicos, dietas, útiles-herramientas aun cuando las características y funciones de estos sistemas son semejantes a lo que se desea lograr, no se pueden usar como solución al problema existentes dado que se enfocan en el aspecto económico de los mismos, pero cabe señalar que sirvieron para tomar ideas respecto a la forma de elaborar dichas funcionalidades. Estos sistemas presentan como principal desventaja que son de licencia privativa y su utilización no está en correspondencia con las políticas de soberanía tecnológica del país. Teniendo en cuenta todo lo expresado anteriormente se decide desarrollar un módulo que dé respuesta al problema planteado.

1.3. Metodología

“Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.” (16). La aplicación a desarrollar será sustentada sobre la metodología de desarrollo para la actividad productiva de la Universidad de las Ciencias Informáticas. La cual tiene entre sus objetivos aumentar la calidad del software, basándose en el Modelo CMMI-DEV v1.3 y en la metodología “Proceso Unificado Ágil” (AUP).

1.3.1. Descripción de las fases y disciplinas

AUP-UCI, como se le conoce propone para el ciclo de vida de los proyectos 3 fases fundamentales (Inicio, Ejecución y Cierre), de las 4 que propone AUP (Inicio, Elaboración, Construcción y Transición), donde se mantiene la fase Inicio, se unifican las 3 últimas fases en Ejecución y se agrega la fase de Cierre.

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos

de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y Diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).

Todas las disciplinas antes definidas (desde modelado de negocio hasta pruebas de aceptación) se desarrollan en la fase de ejecución, de ahí que en la misma se realicen iteraciones y se obtengan resultados incrementales. En una iteración se repite el flujo de trabajo de las disciplinas, Requisitos, Análisis y Diseño, Implementación y Pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración.

1.3.2. Escenarios para la disciplina Requisitos

Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto (16).

Propone 4 escenarios del cual empleará el número 3.

Escenario 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

1.4. Herramienta de modelado

Se seleccionó Visual Paradigm para el modelado de diagramas, es una herramienta que utiliza UML (Lenguaje Unificado de Modelado), multiplataforma que soporta el ciclo de vida completo del desarrollo de software. Visual Paradigm permite dibujar todos los tipos de diagramas de clases y generar código inverso desde diagramas. Dentro de sus características fundamentales se encuentra la posibilidad de usar BPMN (Business Proccess Model Notation) y la presencia de una potente funcionalidad para la creación de interfaces de usuarios de las aplicaciones. Aunque el centro no emplea diagramas de caso de uso dado que su desarrollo parte de los diferentes requisitos funcionales y no funcionales. Esta herramienta fue creada para utilizarse tanto en el entorno de Windows como en el entorno de Linux, facilita la realización de los diagramas de modelado como los diagramas de clases y código inverso además de que soporta el ciclo de vida completo del desarrollo de software (13).

1.5. Lenguaje de programación

PHP 5.5

Es un lenguaje de código abierto y del lado del servidor lo que implica la necesidad de un servidor web para poder visualizar el contenido; el mismo es muy popular especialmente adecuado para desarrollo web y puede ser incrustado en HTML de manera transparente para los usuarios Al ser un lenguaje gratuito y de uso libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas (7):

Soporte para diferentes bases de datos como: MySQL, PostgreSQL, Oracle, MS SQL Server, SybasemSQL, Informix, entre otras. Además soportado por una amplia comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un grupo grande de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. Es un lenguaje versátil ya que puede usarse con la mayoría de sistemas operativos, ya sea basados en UNIX (Linux, Solares, FreeBSD), como con Windows, el sistema operativo de Microsoft.

1.6. Tecnología

Symfony2.8

Symfony2 como entorno de trabajo PHP. Emplea la arquitectura Modelo Vista Controlador (MVC), aunque tiene su propia estructura y forma especializada de trabajo, así como el desarrollo de las vistas a través de extensiones Twig y HTML 5 dando una nueva, fácil y cómoda forma de trabajo para el desarrollo de aplicaciones web. Sigue las buenas prácticas y patrones de diseño para la web. Empleo de CSS3 en sus estilos y diseños generando un ambiente más elegante en el proyecto. Como lenguaje de frente al servidor PHP 5.5, dentro de sus ventajas se encuentran fácil aprendizaje, multiplataforma, multiparadigma, y su compatibilidad con la mayoría de los gestores de base de datos e incluye una gran cantidad de funciones. Su uso está muy difundido en el mundo de los programadores debido a su flexibilidad (8).

Apache 2.2

Servidor web de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras. Apache es altamente configurable, tiene bases de datos de autenticación y negociado de contenido. Entre las ventajas que presenta el mismo se pueden citar que es modular, de código abierto, multi-plataforma y extensible. Es usado principalmente para lanzar páginas web estáticas y dinámica a la World Wide Web (Gran Red Mundial) (9).

NetBeans IDE 8.0

IDE de código abierto y multiplataforma, emplea plugins para proporcionar todas sus funcionalidades. La arquitectura de plugins permite integrar diversos lenguajes, introducir otras

aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo (10).

PostgresSQL 9.3

Es un gestor de base de datos de código abierto, siendo uno de los primeros en implementar un motor de base de datos relacional. Dicho gestor maneja una serie de características y funcionalidades que permiten un trabajo flexible sobre los datos, como son la alta concurrencia, claves ajenas y disparadores (11).

1.7. Técnicas para la obtención de requisitos

Las técnicas empleadas para la obtención de requisitos de software ayudan a identificar las necesidades de negocio de los clientes con facilidad y rapidez, y permiten un mejor entendimiento entre el cliente y el programador, posibilitando la obtención de la información necesaria para formar los requisitos funcionales y no funcionales de un sistema (22).

- **Entrevista:** Consiste en establecer una conversación entre personas de ambas partes para obtener información sobre el negocio y a partir de éstos se definen los requisitos (23).
- **Estudio de documentación:** Consiste en realizar una lectura basada en documentos sobre el dominio del negocio de la organización o sus prácticas profesionales. Ejemplos de algunos documentos que se pueden consultar son: manuales de usuarios del sistema actual, normativas y legislaciones (23).
- **Sistemas existentes:** Consiste en analizar distintos sistemas ya desarrollados que estén relacionados con el proceso que se intenta informatizar. Pueden ser analizadas las interfaces de usuario y las distintas salidas que los sistemas producen (listados, consultas, reportes), ya que pueden surgir nuevas ideas sobre la base de éstas (23).

1.8. Validación de requisitos

El objetivo principal de la actividad "validar los requisitos del sistema" es comprobar que el sistema software descrito por la Especificación de Requisitos del Sistema (ERS) se corresponde con las necesidades de negocio de clientes y usuarios, obteniendo su aprobación y permitiendo generar una línea base de los requisitos en la actividad Gestionar las líneas base y las peticiones de cambio a los requisitos del sistema del procedimiento Gestionar los requisitos del sistema software a desarrollar (25). A continuación se presentan tipos de técnicas para la validación de requisitos:

- **Auditorías**: La revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de chequeos predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.
- **Matrices de trazabilidad**: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre

cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

- **Técnica de prototipo**: el prototipo de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un sistema software que permite a clientes y usuarios entender fácilmente la propuesta de solución para resolver sus problemas de negocio (25).
- **Diseño de casos de prueba**: el objetivo principal del diseño de casos de prueba es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software (26)
- **Revisiones Técnicas Formales**: es el filtro más efectivo desde el punto de vista de garantía de calidad. El objetivo fundamental de estas revisiones es encontrar errores durante el proceso de desarrollo de software. Se deben realizar desde el principio del proyecto para evitar la propagación de los errores a las etapas siguientes del proceso de software (26).

1.9. Patrón Arquitectónico Modelo-Vista-Controlador

Los patrones arquitectónicos: expresan un esquema fundamental de organización estructural para sistemas de software. Proveen subsistemas predefinidos, especificando sus responsabilidades, e incluyen reglas y guías para organizar las relaciones entre ellos (18).

Symfony está basado en el patrón clásico del diseño web conocido como arquitectura Modelo Vista Control (MVC), que está formado por tres niveles:

El ***Modelo*** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.

La ***Vista*** transforma el modelo en una página web que permite al usuario interactuar con ella.

El ***Controlador*** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Figura 2. Arquitectura Modelo Vista Controlador (18)

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de

aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (19).

1.10. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Está compuesto por Clases, Instancias, Roles, Colaboraciones y la Distribución de Responsabilidades (18).

Patrones GRASP: son patrones generales de software para asignar responsabilidades a objetos:

- **Controlador:** consiste en asignar la responsabilidad a una clase de manejar los mensajes correspondientes a eventos de un sistema. El controlador es un intermediario entre la interfaz de usuario y el núcleo donde reside la lógica de la aplicación. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación para aumentar la reutilización del código (27).
- **Experto:** este patrón consiste en asignar una responsabilidad al experto en información, es decir, a la clase que cuenta con la información necesaria para cumplir con la responsabilidad. Lo que permite reforzar el encapsulamiento y favorecer el bajo acoplamiento (27).
- **Creador:** es el responsable de la creación o instanciación de nuevos objetos o clases. Éste patrón refleja que la nueva instancia podrá ser creada por una clase si (27):
    - Clase B contiene o agrega la clase A.
    - Clase B es una agregación o composición de la clase A.
    - Almacena la instancia en algún sitio (por ejemplo una base de datos).
    - Tiene la información necesaria para realizar la creación del objeto (es 'Experta').
- **Alta cohesión:** su objetivo es asignar una responsabilidad, de modo que la cohesión siga siendo alta. Las clases con alta cohesión se caracterizan por tener responsabilidades estrechamente relacionadas y no realizar un trabajo enorme. Una clase con alta cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla (27).
- **Bajo acoplamiento:** pretende asignar una responsabilidad para mantener el bajo acoplamiento, es decir, el diseño de clases más independientes que no se relacionen con muchas otras y reduzcan el impacto de los cambios, siendo más reutilizables y acrecienten la oportunidad de una mayor productividad (27).

Patrones GoF: a principios de los años 90 con la publicación del libro Design Patterns, se establecen veintitrés patrones de diseño GoF. Los patrones GoF surgen como una forma

indispensable de enfrentarse a la programación una vez publicado el libro “Design Patterns-Elements of Reusable Software” de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, para ser conocidos estos patrones como los patrones de la pandilla de los cuatro (GoF, gang of four) (26).

Patrones GoF: Los patrones de diseño pueden tener propósito creacional, estructural o de comportamiento (18):

- **Patrones creacionales:** Se encargan de las formas de crear instancias en los objetos. Su objetivo es: abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. Ejemplos de patrones de creación: Abstract Factory, Builder, Prototype y Singleton (18).
- **Patrones estructurales:** Están relacionados y se aprecia cómo las clases y los objetos se combinan para formar nuevas estructuras más complejas y proporcionar nuevas funcionalidades. Ejemplos de patrones estructurales: Adapter, Bridge, Proxy, Decorator y Facade (18).
- **Patrones de comportamiento:** Están relacionados con algoritmos y asignación de responsabilidades a los objetos. Describen no solamente patrones de objetos o clases, sino también patrones de comunicación entre ellos. Ejemplos de patrones de comportamiento: Chain of Responsibility, Mediator, Observer y Visitor (18).

1.11. Métricas para la validación del diseño

Las métricas basadas en clases según Lorenz y Kidd se dividen cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones, para luego promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clases examinan la cohesión y asuntos relacionados con el código así como las métricas orientadas a los valores externos examinan el acoplamiento y la reutilización. Para la validación del diseño se seleccionaron las métricas Tamaño Operacional de Clase y Relaciones entre Clases que a continuación se describen:

Evalúan los siguientes atributos de calidad:

- **Responsabilidad**: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación**: Consiste en el grado de dificultad que tiene implementado un diseño de clases determinado.
- **Reutilización**: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento**: Consiste en el grado de dependencia o interconexión de una clase o

estructura de clase, con otras, está muy ligada a la característica de Reutilización.

- **Complejidad del mantenimiento**: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas**: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

1.11.1. Métrica Tamaño Operacional de las Clases

Métrica Tamaño Operacional de las Clases (TOC): La métrica TOC está dada por la cantidad de funcionalidades contenidas en las clases, a partir de las cuales se determina la afectación que ejerce en el diseño. La misma comprende los siguientes atributos de calidad (37).

- **Responsabilidad:** Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación:** Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización:** Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 2. Atributos de la métrica TOC

Atributo	Categorías	Criterios
Responsabilidad	Baja	< =Prom
	Media	Entre Prom. y 2* Prom
	Alta	> 2* Prom
Complejidad de implementación	Baja	< =Prom
	Media	Entre Prom. y 2* Prom
	Alta	> 2* Prom
Reutilización	Baja	> 2* Prom
	Media	Entre Prom. y 2* Prom
	Alta	< =Prom

Promedio: Sumatoria de la cantidad de operaciones por clase entre la cantidad de clases.

1.11.2. Métrica Relaciones entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- **Acoplamiento:** Un aumento del RC implica un aumento del Acoplamiento de la clase.
- **Complejidad de mantenimiento:** Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.

- **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 3. Atributos de la métrica RC

Atributo	Categorías	Criterios
Acoplamiento	Baja	RC=1
	Media	RC=2
	Alta	RC>2
Complejidad de mantenimiento	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	> 2*Prom
Cantidad de pruebas	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	> 2*Prom
Reutilización	Baja	> 2*Prom
	Media	Entre Prom. y 2*Prom
	Alta	<= Prom

Promedio: sumatoria de la cantidad de relaciones de uso por clase entre la cantidad de clases.

1.12. Diseño de la Base de Datos

El modelo Entidad-Relación (ER) es un modelo de datos que permite representar cualquier abstracción, percepción y conocimiento en un sistema de información formado por un conjunto de objetos denominados entidades y relaciones, incorporando una representación visual conocida como diagrama entidad-relación (20).

Entidad. La entidad es cualquier clase de objeto o conjunto de elementos presentes o no, en un contexto determinado dado por el sistema de información o las funciones y procesos que se definen en un plan de automatización (20).

Atributos - Intención. Son las características, rasgos y propiedades de una entidad, que toman como valor una instancia particular. Es decir, los atributos de una tabla son en realidad sus campos descriptivos, el predicado que permite definir lo que decimos de un determinado sujeto (20).

Relación. Vínculo que permite definir una dependencia entre los conjuntos de dos o más entidades. Esto es la relación entre la información contenida en los registros de varias tablas (20).

Interrelación. Las interrelaciones las constituyen los vínculos entre entidades, de forma tal que representan las relaciones definidas en el esquema relacional de forma efectiva. Esto no sólo la

relación de los registros sino de sus tablas y de las características de la interrelación entre las entidades, a través de un campo clave que actúa como código de identificación y referencia para relacionar (es decir, como nexo de unión y articulación de la relación) (20).

Entidades fuertes. Lo constituyen las tablas principales de la base de datos que contienen los registros principales del sistema de información y que requieren de entidades o tablas auxiliares para completar su descripción o información (20).

Entidades débiles. Son entidades débiles a las tablas auxiliares de una tabla principal a la que completan o complementan con la información de sus registros relacionados (20).

1.13. Pruebas de Software

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo (36).

1.13.1. Pruebas de caja blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba (26).
Las pruebas de caja blanca intentan garantizar que:

- Todas las rutas independientes dentro de un módulo se revisaron al menos una vez.
- Se revisen todas las decisiones lógicas en sus lados verdadero y falso.
- Se ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas.
- Se revisen todas las estructuras de datos internas.

Las técnicas empleadas para ejecutar dichas pruebas son:

- **Camino básico:** permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.
- **Prueba de condición:** ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.
- **Prueba de Flujo de Datos:** se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.

- **Prueba de Bucles:** se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución todos los bucles en sus límites operacionales.

1.13.2. Pruebas de caja negra

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa (26).

Las pruebas de caja negra intentan encontrar errores en las categorías siguientes:

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en las estructuras de datos o en el acceso a bases de datos externas.
- Errores de comportamiento o rendimiento.
- Errores de inicialización y terminación.

Para desarrollar la prueba de caja negra existen técnicas las cuales se describen a continuación:

- **Partición de Equivalencia:** esta técnica divide el campo de estrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Análisis de Valores Límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Grafos de Causa-Efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Conclusiones parciales

- Se elaboró el diseño teórico de la investigación quedando debidamente expuestos los principales conceptos sobre el tema, así como las diferentes herramientas y tecnologías a utilizar.
- Los sistemas estudiados no manejan de forma conjunta la gestión y control de medios básicos, dietas y útiles.
- La selección de las tecnologías, herramientas y metodología a utilizar permitió establecer el entorno de desarrollo de la propuesta de solución.

CAPITULO II: “ANÁLISIS Y DISEÑO DEL SISTEMA”

Introducción

En el presente capítulo se describen los elementos esenciales para el desarrollo del sistema a partir del análisis previamente realizado en el capítulo anterior. Se presentan los siguientes artefactos definidos por la metodología: la especificación de requisitos funcionales y no funcionales que tendrá la propuesta de solución. Además, se abarcan la composición de la arquitectura, los diagramas de clases y diseño de la base de datos.

2.1. Descripción de la propuesta de solución

El objetivo del módulo a desarrollar como propuesta de solución es lograr la informatización de los procesos de control interno: Control de medios básicos, Gestión de dietas y Control de útiles y herramientas. Entre las funcionalidades del sistema está la posibilidad de gestionar un conjunto de reportes tabulados en formato PDF y EXCEL respondiendo a la necesidad de información del cliente; la gestión centralizada de la información referente a cada uno de los procesos con la intención de un mejor almacenamiento y organización de la misma. El nombre que recibe el módulo a desarrollar es Módulo para la gestión y control de medios básicos, dietas, y útiles-herramientas.

2.2. Procesos del negocio

Entre los procesos de control interno que se llevan a cabo en el Centro de Informatización de Entidades se encuentran los siguientes:

Medios básicos: Proceso que consta de un registro de los medios en formato duro y digital, empleo de excel, dicho control incluye registrar desde la llegada de cada uno a los centros, así como sus traslados a otras áreas, ya sea de forma temporal o permanente, además de una revisión constante de los mismos.

Útiles-herramientas: Proceso que consta de un registro de útiles y herramientas en formato duro y digital, a través del empleo de ficheros excel, cuyo control incluye el registro desde la entrada del útil o herramienta al centro hasta sus posteriores traslados a las diferentes áreas donde son asignados a un determinado especialista, incluyendo un control ciclico sobre los mismos.

Dietas: Sumas de dinero para resarcir determinados gastos en los que estos han incurrido por la realización y desempeño de trabajos fuera del centro donde no se les puede garantizar hospedaje ni alimentación.

Para procesar una dieta se consta de 2 posibilidades.

Por anticipo:

- Se planifica la dieta a los trabajadores en un mes dado que trabajarán fuera de la universidad (Esto se registra en un documento).
- Se aprueba(n) la(s) dieta(s) de dicha entidad en la caja.
- Adquiere la dieta el individuo en la caja.
- Reintegra de la dieta lo que no se consumió. (Se registra en un documento).

Por liquidación:

- En el caso de liquidación se efectúan las actividades encontradas en anticipo, pero con la diferencia de que el individuo solo adquiere la dieta luego de efectuar el trabajo.

2.3. Modelo conceptual

Un modelo conceptual tiene como objetivo identificar y explicar los conceptos significativos en un dominio de problema, identificando los atributos y las asociaciones existentes entre ellos. Puede ser visto, también como una representación de las cosas, entidades, ideas, conceptos u objetos del “mundo real” o dominio de interés. Es importante diferenciar que dichas entidades u objetos no son componentes de software. También podría ser considerado como un diccionario visual de abstracciones, conceptos, vocabulario e información del dominio de problema. No es absolutamente correcta o incorrecta, su intención en ser útil sirviendo como una herramienta de comunicación. (21)

En la siguiente figura se muestra el modelo conceptual que representa los conceptos del contexto del negocio referente a los procesos administrativos: Gestión de Medios Básicos, Gestión de Dietas y Gestión de Útiles y Herramientas.

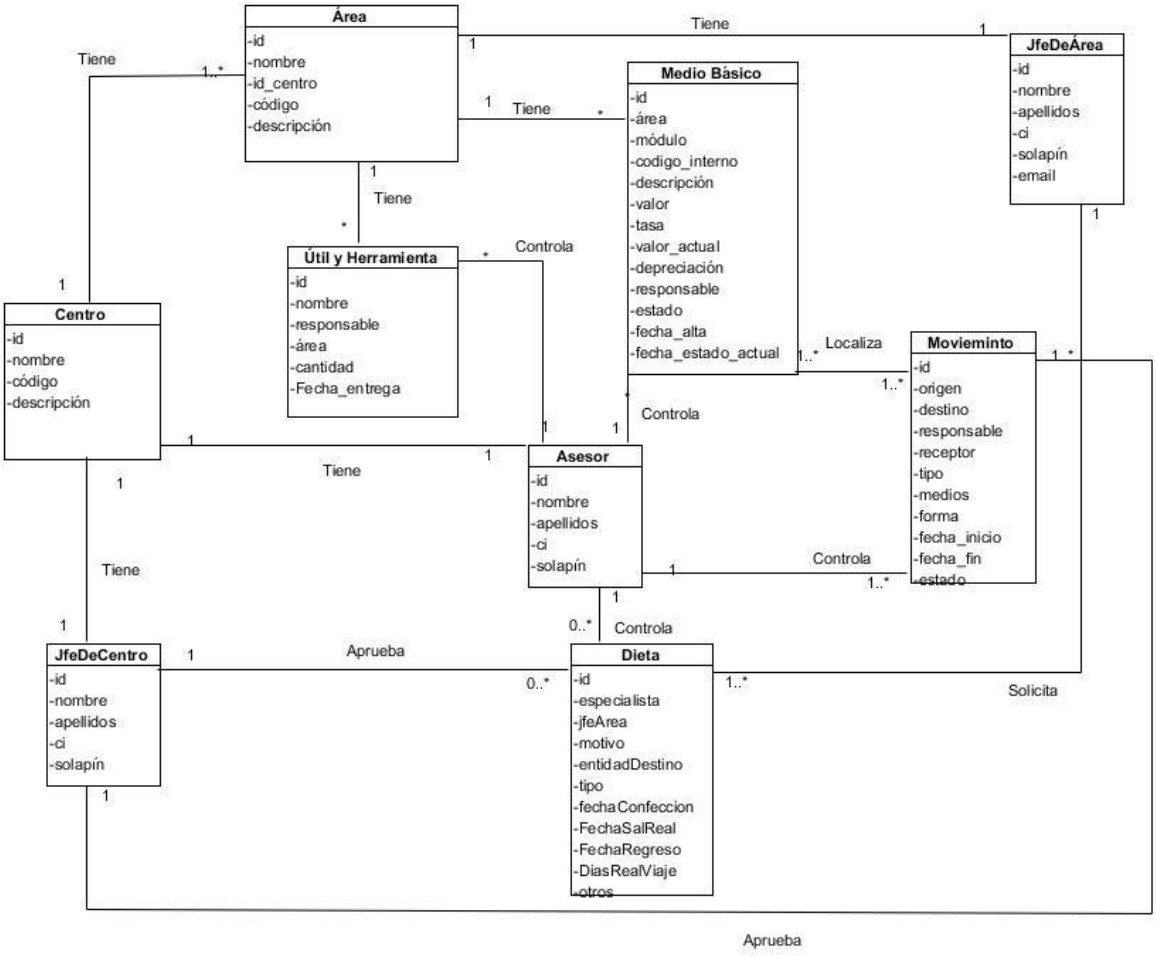

Figura 3. Modelo conceptual

Los conceptos del negocio anteriormente modelado son los siguientes:

- Asesor: es la persona que controla todos los procesos del negocio (dietas, medios básicos, útiles-herramientas).
- Jefe de Centro: es la persona que autoriza la ejecución de cada uno de los procesos del negocio (movimiento de medios, elaboración de dietas, asignación de útiles).
- Jefe de Área: es la persona que efectúa la solicitud de las dietas y movimiento de medios.
- Centro: es la entidad interesada en efectuar cada uno de los procesos administrativos que se llevan a cabo en sus áreas.
- Área: es el local donde se gestiona cada uno de los procesos administrativos.
- Dieta: es el registro de la información referente al proceso de gestión de dietas.
- Medio Básico: es el registro de cada uno de los medios existentes en cada área.
- Movimiento: es el registro de la información referente a la localización de un medio una vez efectuado el proceso de movimiento del mismo.
- Útil y Herramienta: es el registro de la información referente a la asignación de los útiles a los trabajadores.

El modelo conceptual elaborado se utiliza como base para la definición de las funcionalidades y restricciones que debe cumplir el sistema. El tratamiento a estos requisitos indispensables para cumplir con los requerimientos del cliente y serán abordados en la disciplina de Requisitos.

2.4. Requisitos de software

Los requisitos de software son características que deben incluirse en un sistema o aplicación, se pueden clasificar según Sommerville en requisitos funcionales y requisitos no funcionales.

2.4.1. Técnicas aplicadas para la obtención de requisitos

Las técnicas empleadas para la obtención de requisitos de software durante la investigación fueron:

Entrevista: Se realizaron entrevistas informales con el cliente, realizándole preguntas cerradas y abiertas en aras de comprender de forma general sus necesidades y el contexto en que se desarrolla el problema a solucionar. Además fueron entrevistados los jefes de áreas y el director del centro para conocer los detalles de los procesos administrativos que se trabajarán en la investigación.

Sistemas existentes: Se analizaron los detalles de los sistemas similares los que ayudaron a concebir nuevas ideas para la futura solución.

2.4.2. Requisitos Funcionales (RF)

Los requisitos funcionales de un software describen lo que este debe hacer. Dependen del tipo de software que se desarrolle, de los posibles usuarios del software y de las necesidades que posea el cliente. Son las declaraciones de los servicios que el sistema debe proporcionar, de la manera en que debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones determinadas (24).

A continuación la tabla muestra los RF del módulo a desarrollar.

Tabla 4. Requisitos Funcionales

Agrupación de requisitos	Numeración	Requisito funcional	Complejidad
Gestionar Medio Básico	RF. 1	Consultar Medio	Alta
	RF. 2	Dar baja al Medio	Media
	RF. 3	Listar Medios Básicos	Alta
Gestionar útiles y herramientas (UH)	RF. 4	Consultar Útil-Herramienta	Media
	RF. 5	Listar UH	Alta
Gestionar dietas	RF. 6	Adicionar Dieta	Alta
	RF. 7	Consultar Dieta	Alta
	RF. 8	Eliminar Dieta	Media

	RF. 9	Listar Dietas	Alta
Gestionar movimientos	RF. 10	Adicionar Movimiento	Alta
	RF. 11	Consultar Movimiento	Alta
	RF. 12	Editar Movimiento	Media
	RF. 13	Rechazar Movimiento	Media
	RF. 14	Listar Movimientos	Alta
Gestionar usuarios	RF. 15	Editar Usuario	Baja
	RF. 16	Listar Usuarios	Alta
	RF. 17	Asignar rol a Usuario	Baja
	RF. 18	Autenticar Usuario	Alta
	RF. 19	Exportar en formato PDF listado de Medios básicos	Alta
	RF. 20	Exportar en formato PDF listado de Útiles y Herramientas	Alta
	RF. 21	Exportar en formato PDF listado de Dietas	Alta
	RF. 22	Exportar en formato PDF listado de movimientos	Alta
	RF. 23	Cargar ASSET de Medios Básicos	Alta
	RF. 24	Cargar ASSET de Útiles y Herramientas	Alta
	RF. 25	Exportar Medio Básico en formato PDF	Alta
	RF. 26	Exportar UH en formato PDF	Media
	RF. 27	Exportar dieta en formato PDF	Alta
	RF. 28	Exportar movimiento en formato PDF	Alta

2.4.3. Descripción de los requisitos

A continuación se muestra la descripción del Listar Medios Básicos.

Tabla 5. Descripción del requisito Listar Medios Básicos

Precondiciones	Debe existir el listado de los medios básicos. Debe estar autenticado el usuario.
Flujo de eventos	
Flujo básico “Listar Medios Básicos”	
1	El asesor selecciona la opción Lista de MB en el menú Gestión de MB
2	El sistema muestra un listado con los medios registrados en la base de datos

Pos-condiciones		
N/A	N/A	
Flujos alternativos		
N/A	N/A	
Pos-condiciones		
N/A	N/A	
Validaciones		
N/A		
Conceptos	Atributos visibles en la interfaz	Atributos usados internamente
Medio Básico	Área	id
	Rótulo	
	Código interno	
	Descripción	
	Responsable	
	Fecha Alta	
	Estado	
Requisitos Especiales	Debe existir el listado de los medios básicos.	

2.4.4. Requisitos No Funcionales (RNF)

Los requisitos no funcionales son restricciones de las funciones o servicios ofrecidos por el sistema. No se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este. Definen restricciones sobre el sistema y el proceso de desarrollo de software (24). Los requisitos no funcionales definidos son:

RNF. 1 Usabilidad

El idioma de todas las interfaces de la aplicación será español. Los errores cometidos por el usuario les serán notificados. El sistema expondrá el menú general desde cualquiera de sus páginas.

RNF. 2 Confiabilidad

Ante el fallo de una funcionalidad del sistema, el resto de las funcionalidades que no dependen de esta deberán seguir funcionando.

RNF. 3 Mantenibilidad

La codificación del sistema será estándar y las funcionalidades serán comentadas.

RNF.4 Software

RNF. 4.1 Las estaciones de trabajo deberán poseer un navegador web. Mozilla Firefox 34.0 o una versión superior.

RNF. 4.2 El servidor de aplicaciones web deberá ser Apache 2.0.

RNF. 4.3 El servidor de base de datos deberá ser PostgreSQL 9.3.

2.4.5. Validación de requisitos

Validar los requisitos del sistema es comprobar que el sistema software descrito por la Especificación de Requisitos del Sistema (ERS) se corresponde con las necesidades de negocio de clientes y usuarios, obteniendo su aprobación.

Los requisitos de la presente investigación fueron validados mediante la técnica de prototipo. La siguiente figura muestra el prototipo del requisito Listar Medios Básicos.

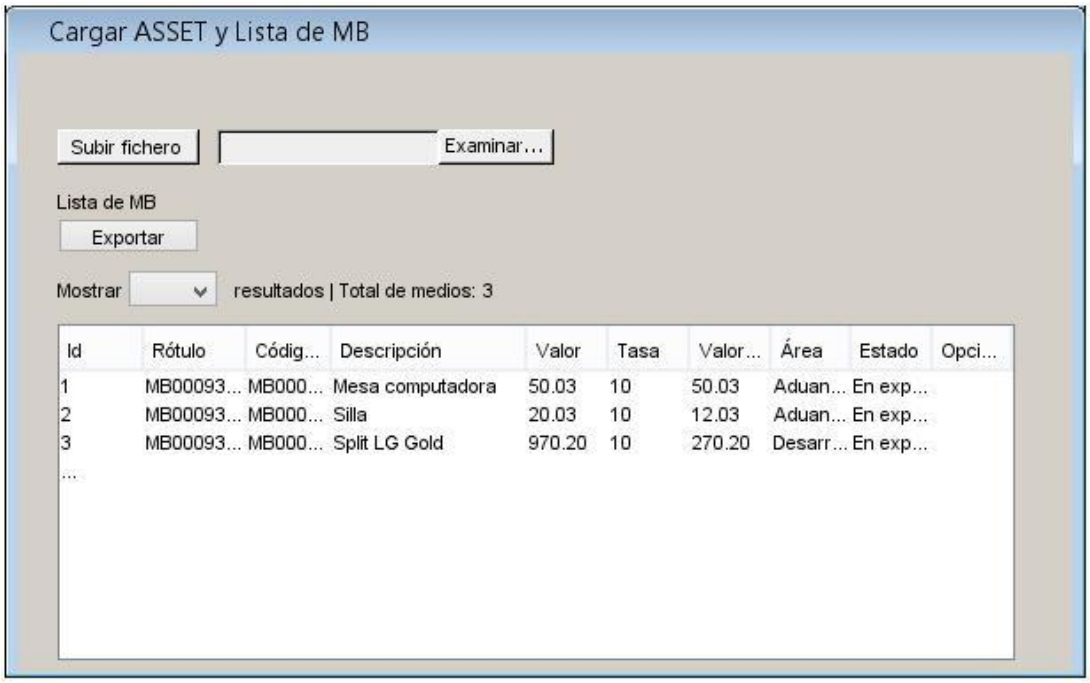

Figura 4. Prototipo de requisito funcional Listar Medios Básicos

2.5. Patrones utilizados en el diseño de la solución

2.5.1. Patrón Arquitectónico

Para el desarrollo de la solución fue utilizado el patrón arquitectónico MVC. Las siguientes figuras ilustran parte de su utilización en esta investigación.

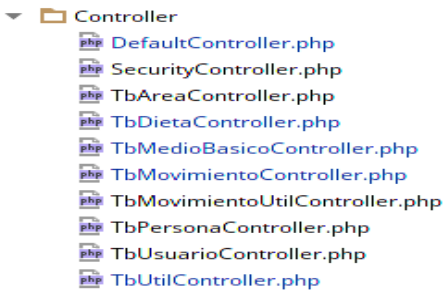

Figura 5. Capa Controladora

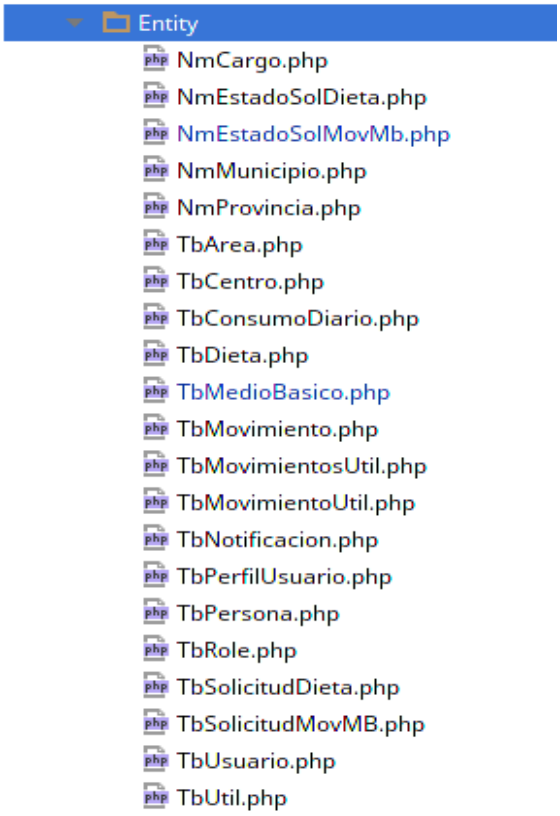

Figura 6. Capa Modelo

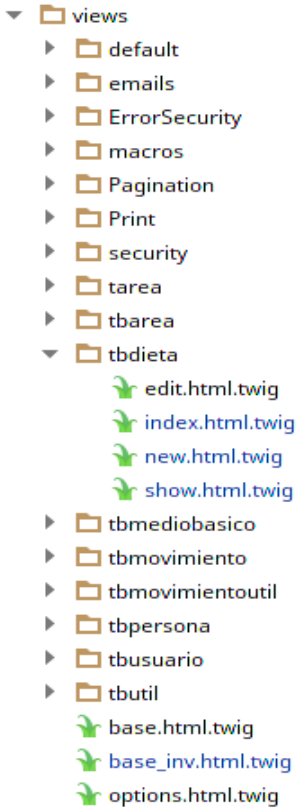

Figura 7. Capa Vista

2.5.2. Patrones de diseño

Los patrones generales de software para asignar responsabilidades a objetos utilizados en el desarrollo del diseño fueron:

Controlador: Este patrón se evidencia en las clases controladoras del sistema. Ejemplo:

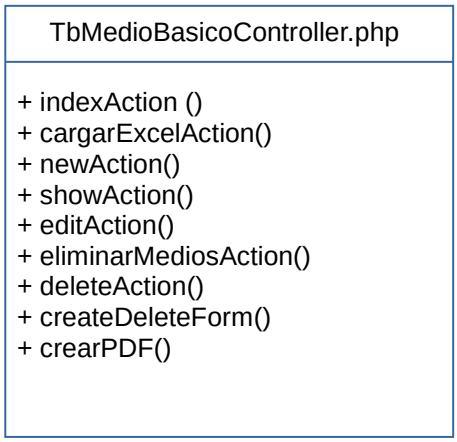

Figura 8. Utilización del patrón Controlador

Experto: Este patrón se evidencia en la clase MedioBasico.php la cual se encarga de toda la información referente a los medios básicos.

Figura 9. Utilización del patrón Experto

Creador: Este patrón se evidencia en el empleo de la clase MovimientoController.php donde se crea una instancia de la clase entityManager.php para acceder a la base de datos a través de doctrine.

Alta cohesión: Su empleo se evidencia en las clases controladoras, aunque cada clase es experta en su responsabilidad, puede cooperar con otras clases para poder realizar algunas de sus tareas. En la clase *TbDietaController* cuando se elimina una dieta se hace necesario saber si hay alguna dependencia *TbconsumosDiarios*.

Bajo acoplamiento: En la aplicación se evidencia este patrón en la baja interdependencia que existe entre las clases de las vistas y las controladoras, lo que permite cambios en las mismas sin que las otras sufran grandes afectaciones. Ejemplo la vista mediobasico.html.twig y la controladora TbMedioBasicoController.php

Los patrones GoF aplicados durante el desarrollo del diseño son:

Decorador (Decorator): Se evidencia en el empleo de una plantilla base.html.twig donde esta almacena el contenido común del resto de las páginas, las cuales extienden de la misma, evitando repetir código innecesariamente.

Inyección de dependencia (Dependency injection): Este patrón queda evidenciado a partir del empleo de servicios donde una vez especificados los datos necesarios se obtendrán los datos resultantes al servicio especificado. Ejemplo:

```
    /**
     * Cargar excel
     *
     * @Route("/", name="cargar_excel")
     * @Method("POST")
     */
    public function cargarexcelAction()
    {
        $rol= $this->get('security.context');

        if(!$rol->isGranted(['ROLE_JAREA','ROLE_JCENTRO','ROLE_ADMIN','ROLE_ASESOR']))
            return $this->render('ErrorSecurity/acceso_denegado');
        $session = new Session();
        $form = $this->createForm('AppBundle\Form\CargarExcelType');
        $form->handleRequest($this->getRequest());
        if ($form->isValid()) {

            $f = $form->get('CargarExcel')->getData();
            $ext = $f->getClientOriginalExtension();
            if($ext!='xls')
             $session->getFlashBag()->add('incorrecto', 'El fichero no tiene el formato correcto.');
             else{
                $DIR = '../data/';
                $FILE_NAME = 'submayor-' . date('Ymd') . '.' . $ext;
                $f->move($DIR, $FILE_NAME);
                try{
                $submayor = $this->get('submayor_manager');

                }catch (\Exception $exec){
                    $session->getFlashBag()->add('incorrecto', 'El fichero no tiene el formato correcto.');

                }
                $submayor->cargaSubMayor($DIR . $FILE_NAME);
            }

            return $this->redirectToRoute('tbmediobasico_index');
        }
    }
```

Figura 10. Utilización del patrón Inyección de dependencia

2.6. Diseño de la Base de Datos

El modelo de datos elaborado para el desarrollo del módulo fue el siguiente:

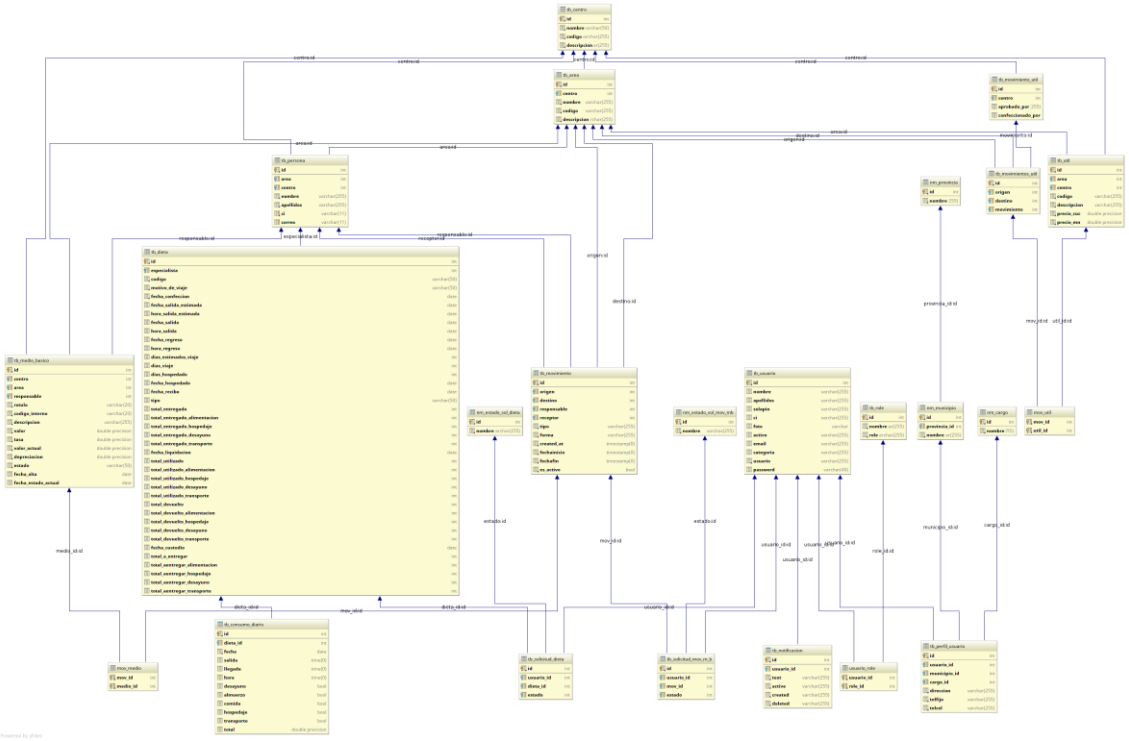

Figura. 11 Modelo de datos

2.7. Validación del diseño

2.7.1. Tamaño operacional de clase (TOC)

Para validar el diseño de la investigación fueron utilizadas varias métricas. Los resultados que a continuación se exponen corresponden a la métrica TOC.

La siguiente figura muestra los resultados de la evaluación de la métrica TOC en el atributo responsabilidad. El 67% de las clases poseen una baja responsabilidad permitiendo que en caso de fallos, como la responsabilidad está distribuida de forma equilibrada ninguna clase es demasiado crítica como para dejar al sistema fuera de servicio.

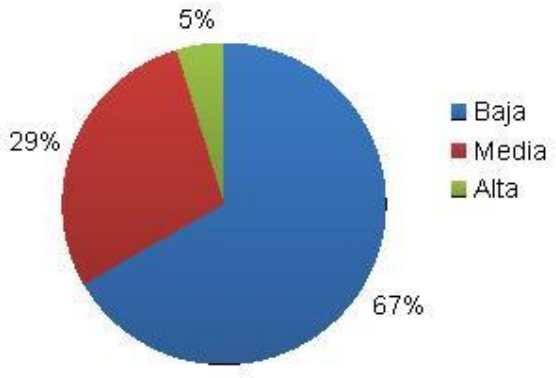

Figura 12. Resultado del atributo Responsabilidad

En la figura 13 se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo complejidad de implementación. El gráfico muestra un resultado satisfactorio pues más del 67% de las clases poseen una baja complejidad de implementación. Esta característica permite mejorar el mantenimiento y soporte de estas clases.

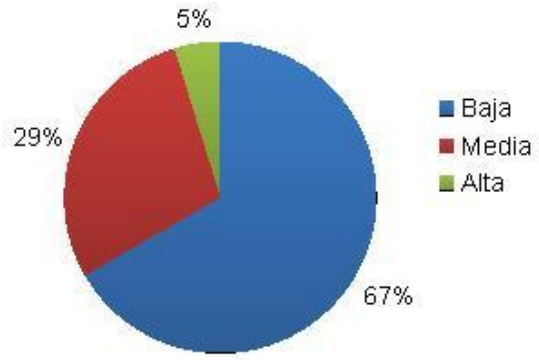

Figura 13. Resultado del atributo Complejidad de Implementación

La figura 14 muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización. Queda demostrado que el diseño de la solución es eficiente pues solamente el 18% del total de las clases poseen una baja reutilización.

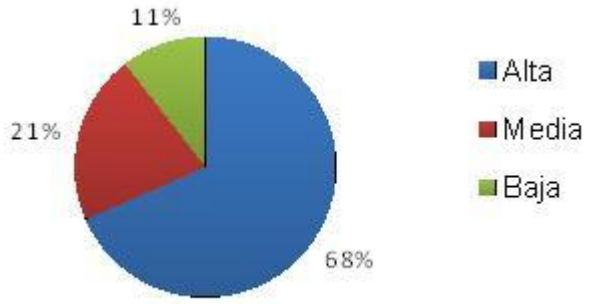

Figura 14. Resultado del atributo Reutilización

Luego de realizado un análisis de los resultados obtenidos para los atributos de la métrica TOC, se observa que la mayoría de las clases que conforman el módulo para los atributos responsabilidad y complejidad están dentro de la categoría media y baja para un 96% del total. Esto demuestra que las clases cuentan con un bajo nivel de dependencias y pocos procedimientos en su implementación, mientras que el atributo reutilización cuenta con 89% en las categorías alta y media mostrando así que el módulo cuenta con una elevada reutilización. Se concluye que los resultados obtenidos según esta métrica son satisfactorios.

2.7.2. Relaciones entre clases (RC)

La figura 15 muestra la representación de la incidencia de los resultados de la evaluación de la

métrica RC en el atributo acoplamiento. Se evidencia un bajo acoplamiento entre las clases pues el 76% de las clases presentan una dependencia con otra. Este resultado es muy favorable para el diseño de la funcionalidad pues al existir poca dependencia entre las clases aumenta el grado de reutilización de la misma.

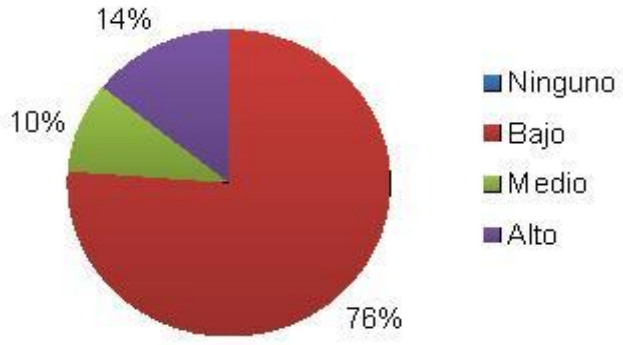

Figura 15. Resultado del atributo Acoplamiento

La figura 16 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento. El gráfico refleja un resultado aceptable del atributo pues el 76% de las clases presentan una baja complejidad de mantenimiento.

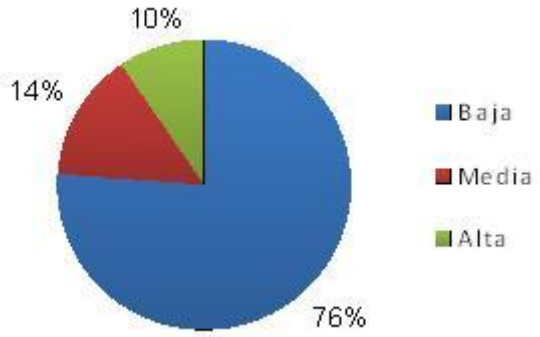

Figura 16. Resultado del atributo Complejidad de Mantenimiento

La figura 17 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas.

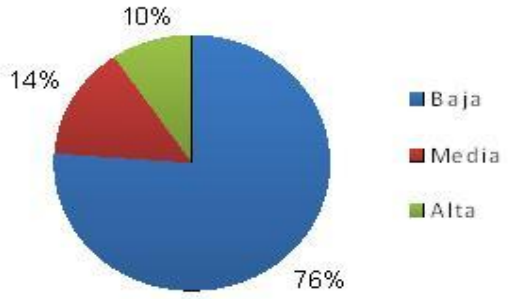

Figura 17. Resultado del atributo Cantidad de Pruebas

La figura 18 muestra la representación de la incidencia de los resultados de la evaluación del atributo reutilización. Esto evidencia que el 67% de las clases poseen una alta reutilización lo que es un factor fundamental que debe ser tenido en cuenta en el desarrollo de software.

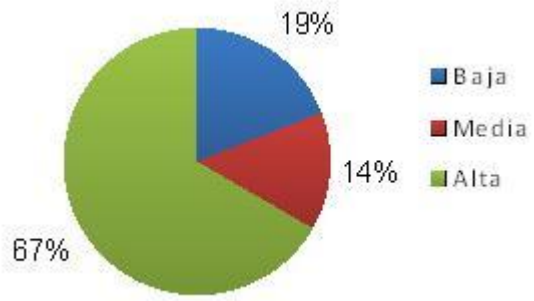

Figura 18. Resultado del atributo Reutilización

Al analizar los resultados obtenidos de la métrica RC, se puede concluir que el diseño del módulo tiene una calidad aceptable. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 76% de las clases el nivel de acoplamiento es bajo. La complejidad de mantenimiento y la cantidad de pruebas son bajas a un 76%, mientras que, el atributo reutilización cuenta con 67% en la categoría alta, lo que representa valores favorables para el diseño realizado.

Conclusiones parciales

- Se generaron los artefactos modelo conceptual y descripción de requisitos por procesos, correspondientes al escenario tres de la metodología empleada.
- A partir de los artefactos generados se logró una mayor comprensión del negocio definiendo los requisitos funcionales y no funcionales, permitiendo conocer en profundidad, cuáles son las preferencias del cliente y cuáles debe cumplir el producto final.
- Como resultado del empleo de patrones de diseños GRASP y GoF; y arquitectónico MVC se logró una mayor reutilización de código en el sistema y un mejor mantenimiento del mismo evidenciado en los resultados obtenidos al aplicar las métricas de diseño al módulo.
- Se validó además el diseño propuesto a partir de la aplicación de métricas (Tamaño Operacional de Clase y Relaciones entre Clases) que permitieron evaluar aspectos como la complejidad de la implementación, la responsabilidad y reutilización de las clases.

CAPÍTULO III: “IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA”

Introducción

En el presente capítulo se describen los estándares de codificación por los que se regirá el código fuente, la estrategia de integración a seguir y el desarrollo de la arquitectura del sistema. Además serán detalladas las pruebas de caja blanca y caja negra, realizadas garantizar que la solución cumpla con las métricas de calidad establecidas y corregir posibles errores existentes.

3.1. Implementación

En este flujo de trabajo se desarrolla la arquitectura del sistema, partiendo de los resultados del diseño, se reflejan los diagramas correspondientes como el diagrama de despliegue y los diagramas de componentes, se implementa el sistema y se realizan pruebas para comprobar si la aplicación cumple sus objetivos.

3.2. Diagrama de componente

Este diagrama representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. (32)

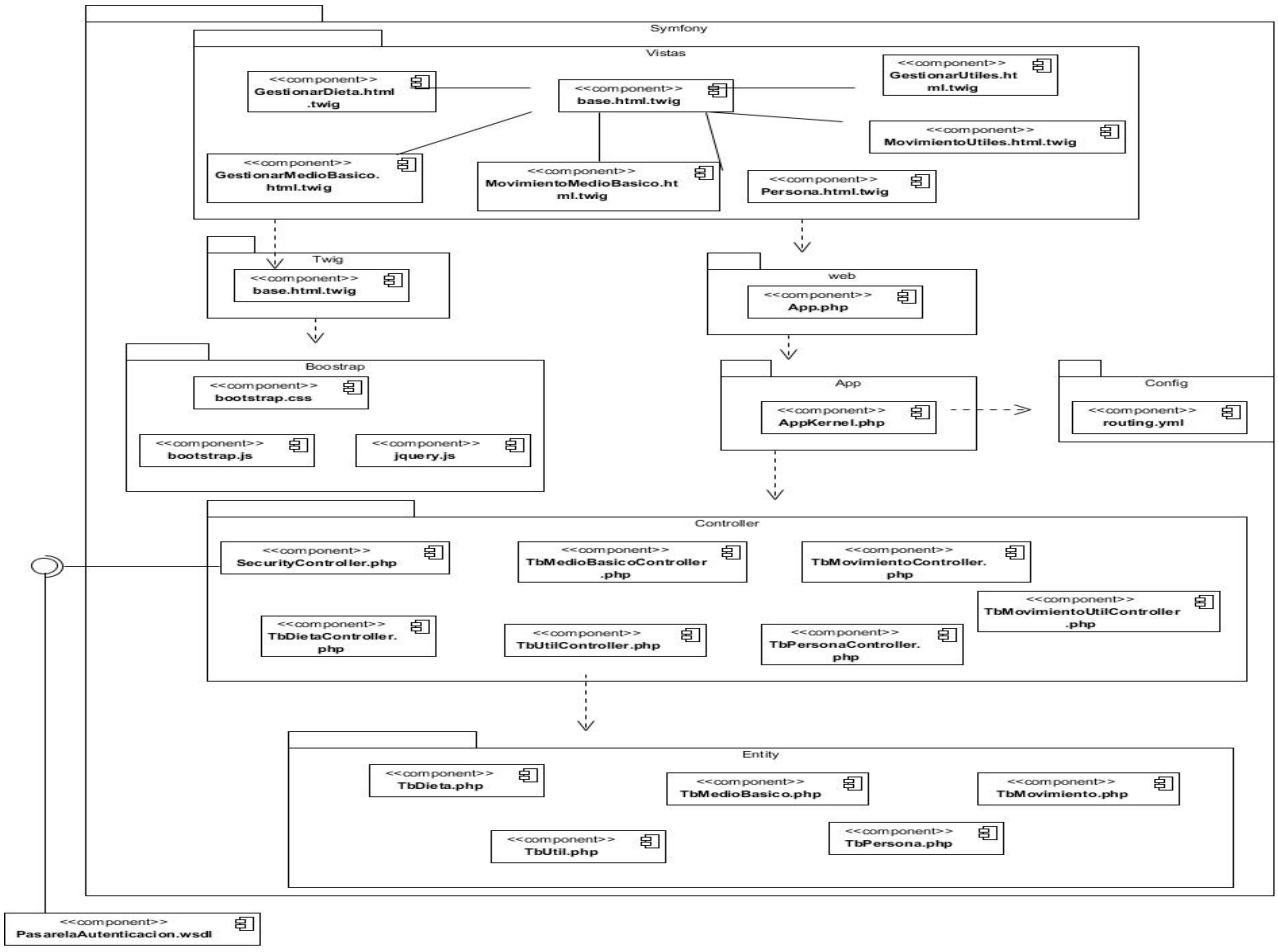

Figura 19. Diagrama de componentes

3.3. Diagrama de despliegue

Los diagramas de despliegue muestran la configuración en funcionamiento del sistema incluyendo su software y su hardware. (33)

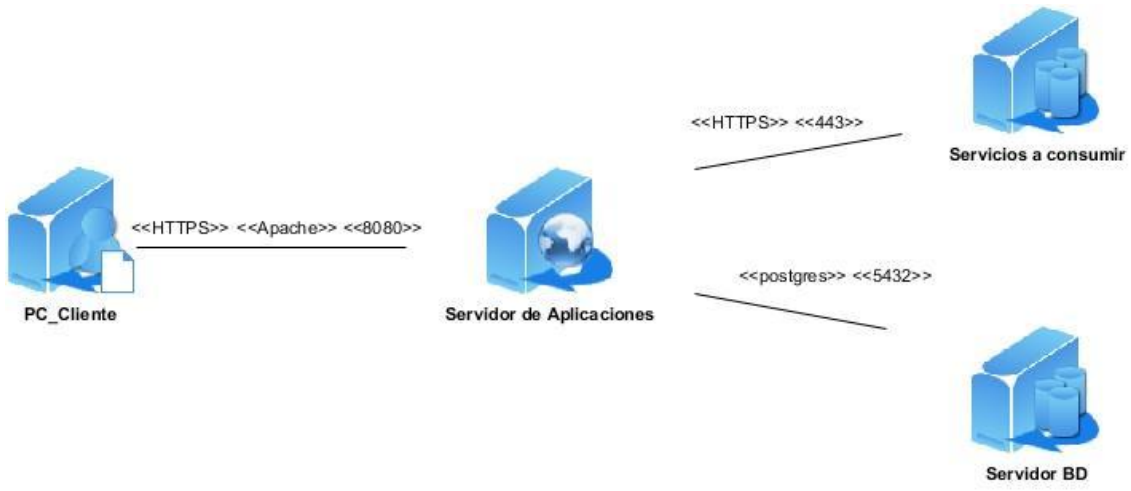

Figura 20. Diagrama de despliegue

3.4. Estándar de codificación

Los estándares de codificación son patrones de programación que no están orientados a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Existen diversas formas de estandarizar el código de acuerdo a cada lenguaje, sin embargo, se puede hacer una generalización de los puntos más importantes a considerar, entre los que se encuentran:

Estructura del código

- Añadir un solo espacio alrededor de los operadores (==, &&,...).
- Añadir una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- Añadir una línea en blanco antes de las declaraciones *return*, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones (tal como una declaración *if*).
- Declarar las propiedades de clase antes que los métodos.
- Usa llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga;
- Usar paréntesis cuando se instancie una clase a pesar del número de argumentos que su constructor posea.

Nomenclaturas utilizadas

- Nomenclaturas de las clases: se emplea la notación CamelCase en su variante lowerCamelCase, que se aplica a frases o palabras compuestas, para esta variante la primera palabra siempre inicia en minúsculas.
- Nomenclatura de los controladores: los controladores después del nombre llevan el sufijo Controller. Por ejemplo EstudianteController.
- Nomenclatura de las variables: El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación lowerCamelCasing y comenzando con un prefijo según el tipo de datos. Ejemplo: $nombre y $temaInvestigacón.
- Usa guiones bajos para nombres de opción y nombres de parámetro.
- Sufija las interfaces con *Interface*.
- Sufija las excepciones con *Exception*.
- Utiliza caracteres alfanuméricos y guiones bajos para los nombres de archivo.

```php
/**
 * Creates a new tbMedioBasico entity.
 *
 * @Route("/new", name="tbMedioBasico_new")
 * @Method({"GET", "POST"})
 */
public function newAction(Request $request)
{
    $tbMedioBasico = new Tbmediobasico();
    $form = $this -> createForm('AppBundle\Form\TbMedioBasicoType', $tbMedioBasico);
    $form -> handleRequest($request);

    if ($form -> isSubmitted() && $form->isValid()) {
        $em = $this -> getDoctrine() -> getManager();
        $em -> persist($tbMedioBasico);
        $em -> flush($tbMedioBasico);

        return $this -> redirectToRoute('tbMedioBasico_show', array('id' => $tbMedioBasico->getId()));
    }

    return $this -> render('tbMedioBasico/new.html.twig', array(
        'tbMedioBasico' => $tbMedioBasico,
        'form' => $form -> createView(),
    ));
}
```

Figura 21. Utilización de los estándares de codificación

3.5. Pruebas de software

Las pruebas aplicadas para garantizar la calidad de la propuesta de solución y el cumplimiento con los intereses del cliente, son las pruebas de caja blanca y caja negra; para ello se establecen las siguientes metas:

- Verificar la implementación de la solución.

- Verificar la integración adecuada d la solución.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar los errores y asegurar que estos sean corregidos.

Pruebas de caja blanca:

Con el objetivo de valorar la calidad con la que se llevó a cabo la implementación de la funcionalidad “Actualizar Movimiento”, fue necesario aplicar una de las técnicas descritas anteriormente: **camino básico.**

Para ello fue necesario seguir los siguientes pasos básicos:

- A partir del diseño o del código fuente, dibujar el grafo de flujo asociado.
- Calcular la complejidad ciclomática del grafo.
- Determinar un conjunto básico de caminos independientes.
- Preparar los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Para dar cumplimiento a los pasos anteriores se enumeran cada una de las sentencias de código de uno del procedimiento revertirMovimiento de la clase MovimientoController.

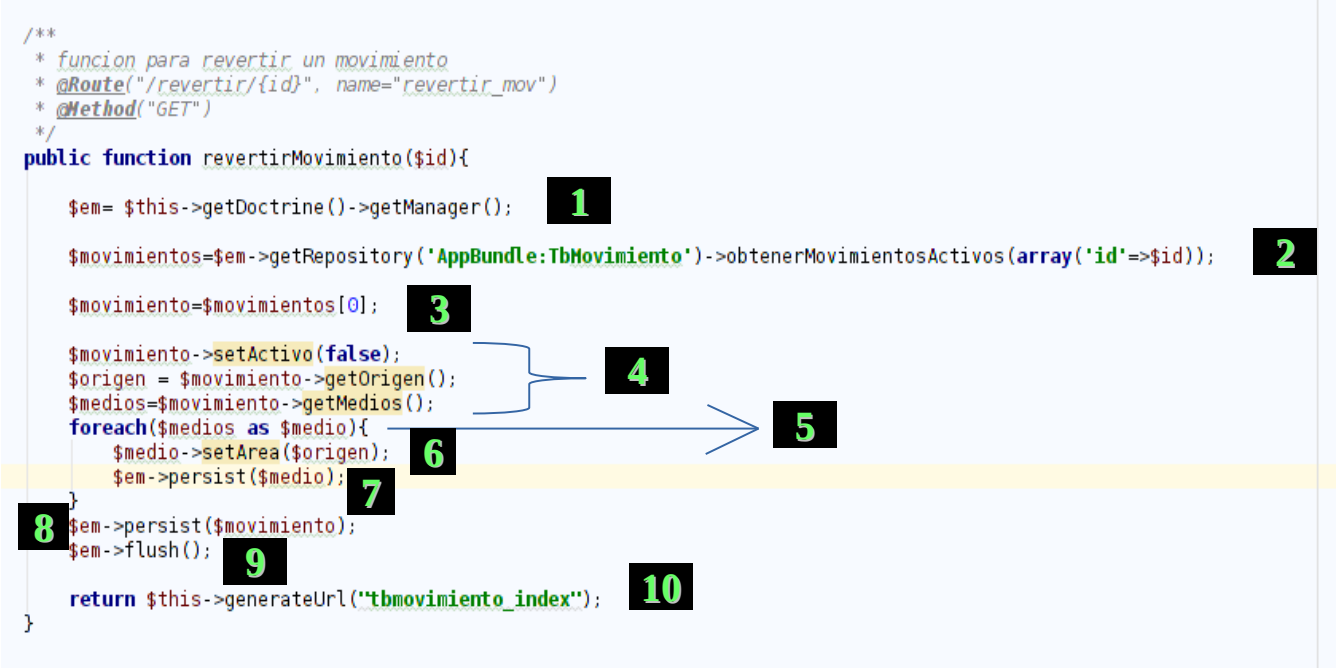

Figura 22. Enumeración de las sentencias de código

Elementos de los grafos de flujo:

- **Nodos:** Son los círculos en el grafo, representan una o más secuencias del procedimiento.
- **Aristas:** Son las saetas y unen los Nodos representando el flujo de control del procedimiento.
- **Regiones:** Son las áreas delimitadas por las aristas y nodos.

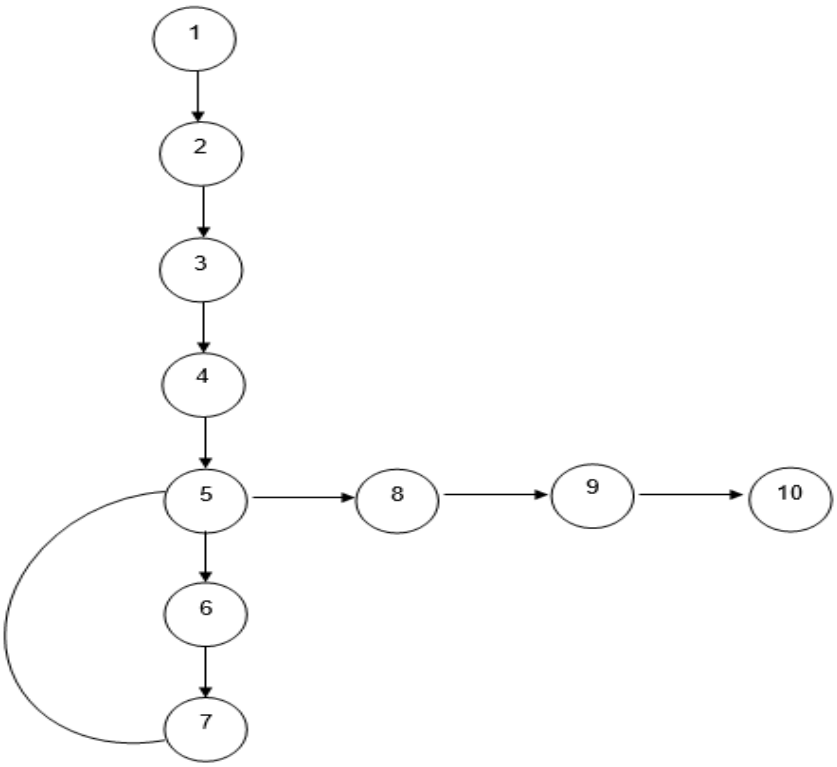

Figura 23. Grafo del código fuente

Una vez construido el grafo de flujo asociado al procedimiento anterior se procede a determinar la complejidad ciclomática, para ellos se procede a determinar sus 3 vías de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

1. V (G) = (A - N) + 2

Siendo “A” la cantidad total de aristas y “N” la cantidad de nodos.

V (G) = (10 - 10) + 2

V (G) = 2.

2. V (G) = P + 1

Siendo “P” la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 1 + 1

V (G) = 2.

3. V (G) = R

Siendo “R” la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

V (G) = 2.

Como resultado de las fórmulas antes presentadas se obtiene una complejidad ciclomática de valor 2, de manera que existen dos posibles caminos por donde el flujo puede circular, este valor

representa el número mínimo de casos de pruebas para el procedimiento tratado.

Caminos básicos posibles a tomar por el algoritmo durante su ejecución:
Camino básico 1: 1-2-3-4-5-8-9-10.
Camino básico 2: 1-2-3-4-5–6-7-8-9-10.

Ejecución de los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo.

Tabla 6. Camino básico 1: 1-2-3-4-5–8-9-10

Camino básico 1: 1-2-3-4-5–8-9-10	
Descripción	Sucede cuando solo hay un medio en movimiento.
Condición de ejecución	Ocurre cuando el usuario decide actualizar un movimiento que ha concluido su tiempo de efectividad.
Entrada	$id = 2
Resultado esperado	El sistema actualiza el estado del movimiento.

Tabla 7. Camino básico 2: 1-2-3-4-5-6-7–8-9-10

Camino básico 2: 1-2-3-4-5-6-7–8-9-10	
Descripción	Sucede cuando hay varios medios en movimiento.
Condición de ejecución	Ocurre cuando el usuario decide actualizar un movimiento que ha concluido su tiempo de efectividad.
Entrada	$id = 2
Resultado esperado	El sistema actualiza el estado del movimiento.

Luego de aplicar los casos de pruebas correspondientes a cada camino básico identificado en el grafo de flujo asociado, se pudo comprobar el correcto funcionamiento del método.

Pruebas de caja negra

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales. Se centra en lo que se espera de un programa, es decir, intenta encontrar casos en el que el producto o sistema no brinde como respuesta el resultado esperado. Dicha comprobación estudia la salida, sin preocuparse del funcionamiento interno.

Se aplica la prueba caja negra utilizando la técnica **Partición de Equivalencia** aplicada al requisito funcional Crear movimiento. Para ello se definieron un conjunto de estados válidos y no válidos para las condiciones de entrada del sistema.

Como parte de la ejecución de las pruebas y teniendo en cuenta como objetivo esencial el de identificar en qué medida satisface la aplicación las funcionalidades establecidas, se realizó una primera iteración de pruebas, donde fueron aplicados los diseños de casos de pruebas realizados.

En una primera iteración se identificaron un total de 12 No Conformidades, clasificadas en 4 no significativas y 8 significativas. Luego de corregidas estas deficiencias se procedió a la realización de una segunda iteración de pruebas donde no se encontraron No Conformidades, por lo que se determina no realizar nuevas iteraciones dando por concluidas las pruebas evidenciando que el módulo cumple con los requisitos propuestos.

Pruebas de Aceptación

Además como parte de las pruebas al software, el cliente realizó una primera revisión exhaustiva de la aplicación, detectando 4 no conformidades significativas. Luego de corregidas estas no conformidades se procedió a una segunda revisión con el cliente, quedando totalmente satisfecho con el trabajo realizado, sin encontrar no conformidades y cumpliendo sus expectativas iniciales.

Conclusiones parciales

- La definición de los principales aspectos que influyeron en la implementación de la solución propuesta fue crucial para el desarrollo del módulo que dé respuesta al problema existente.
- Se definió la estructura física de la solución haciendo también referencia a los estándares de codificación utilizados.
- La aplicación de pruebas de caja blanca y caja negra posibilitó verificar el correcto funcionamiento de la funcionalidad.
- La aceptación del cliente constituyó una prueba certera del módulo desarrollado.

CONCLUSIONES GENERALES

El desarrollo del presente trabajo de diploma y los resultados obtenidos con el mismo permitieron arribar a las siguientes conclusiones generales:

- El estudio del arte sobre los sistemas de gestión de medios básicos, dietas, útiles-herramientas arrojó que estas soluciones no satisfacen la problemática, lo cual evidencia la necesidad de esta investigación.
- Como parte del proceso de análisis y diseño ejecutado se obtuvo como resultado los diagramas y artefactos establecidos por el tercer escenario de la metodología AUP-UCI para guiar el desarrollo del módulo.
- Se desarrolló el módulo cumpliendo con las necesidades del cliente.
- Las pruebas realizadas para validar las funcionalidades que fueron implementadas demuestran que el módulo desarrollado cumple satisfactoriamente con los requisitos que garantizan su correcto funcionamiento.

RECOMENDACIONES

- Se recomienda aplicar esta solución a otros entornos de trabajo de la universidad.
- Realizar el control de consumos diarios del proceso de Dietas en un calendario de tareas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Portela, Gladys María Bejerano.** Resolución No. 60/11. [En línea] 1 de marzo de 2011. [Citado el: 14 de noviembre de 2016.] http://bvs.sld.cu/revistas/infd/n1311/4%20Res.%2060-11%20Sobre%20las%20Normas%20del%20Sistema%20de%20Control%20Interno.pdf.

2. **MSC. Mariuska Leon Beruvides/Lic. Yuliet Lopez Guerra**.(2015).El control Interno, un reto en la actualidad. Universidad de Matanzas Camilo Cienfuegos.

3. **Universidad de las Ciencias Informáticas** (UCI). Disponible en Historia. Visitado el 6 de noviembre de 2016.

4. **López Jiménez, T.:"**Selección de contenidos del Proyecto Estratégico de la UCI para el período 2008 – 2012"; Ciudad de La Habana, Cuba, 2008.

5. **Objetivos y Metas**. [Visitado el: 06 de noviembre de 2016.] *https://gespro.ceige.prod.uci.cu/*

6. Ley del Impuesto Sobre la Renta de las Personas Físicas(LRPF).

7. **Achour, M. B.** (2010). *Manual de PHP.* Recuperado el 2016

8. **Potencier, F., & Zaninotto, F.** (2008). *Symfony la Guía Definitiva.* Recuperado el 2016

9. **Servidor HTTP apache**. [En línea] The Apache Software Foundation., 1999-2015. [Visitado el: 30 de noviembre de 2016.] http://www.apache.org/foundation/license-faq.html#Marks.
10. **Netbeans**. [En línea] [Visitado el: 4 de diciembre de 2016.] https://netbeans.org/features.
11. **Postgresql**.[En línea] [Visitado el: 4 de diciembre de 2016.] http://www.postgresql.org.es/sobre_postgresql.
12. **Contabilidad Financiera**. [En línea] 2011. [Visitado el: 7 de diciembre de 2016.] https://prezi.com/lachcdr_iwn6/contabilidad-financiera.
13. **Visual Paradigm.** [En línea] 2012. [Visitado el: 7 de diciembre de 2016.] http:// www.visual-paradigm.com.
14. **Versat_Sarasola. Versat_Sarasola.** [En línea] Casa Consultora DISAIC, 2014. [Visitado el: 7de diciembre de 2016.] http://www.disaic.cu/index.php.
15. **S.A., D´MARCO.** Sistema de Gestón Integral ASSETS. Assets. [En línea] 2014. [Visitado el: 10 de diciembre de 2016.] http://www.assets.co.cu.
16. **Rodríguez Sánchez, Tamara.** Metodología de desarrollo para la Actividad productiva de la UCI. Habana: s.n., 2015.

17. **Siscont.[En linea] 2008.** [Visitado el: 13 de diciembre de 2016.] http://www.siscont.cl/ManualSiscontGold.pdf,http://www.cipimm.co.cu/sitio-info/SISCONT5.php
18. **Larman, Craig**. UML Y PATRONES. México: Prentice Hall, 1999.
19. **Potencier, Fabien. 2015.** librosweb. [En línea] 2015. [Citado el: 6 de Febrero de 2016.] http://librosweb.es/libro/symfony_1_1/capitulo_2/el_patron_mvc.html.
20. **Prof. Dr. Manuel Blázquez Ochando**. Fundamentos y Diseño de Bases de Datos. [En línea] 20 de febrero de 2014. [Citado el: 29 de enero de 2017.] http://ccdoc-basesdedatos.blogspot.com/2013/02/modelo-entidad-relacion-er.html.
21. **Zaldívar, MsC. Yoenis Pantoja**. Instituto Superior de Tecnologias de Información y Comunicación (ISUTIC). [En línea] 15 de marzo de 2016. [Citado el: 2017 de febrero de 23.] https://engenhariasoftwareisutic.files.wordpress.com/2016/05/mc.pdf.
22. **Rojas, Msc. Juan Carlos Olivera**. SlideShare- Técnicas para la obtención de requerimientos. [En línea] 13 de marzo de 2012. [Citado el: 2 de marzo de 2017.] https://es.slideshare.net/PedroGutirrezCuadra/tcnicas-para-la-obtencin-de-requerimientos.
23. **García, Cesar Arturo Guerra.** SG Buzz-Obtención de Requerimientos. Técnicas y Estrategias. [En línea] 10 de octubre de 2007. [Citado el: 3 de marzo de 2017.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.
24. **Sommerville, I**. Ingeniería de software. España : Pearson, 2011. ISBN: 9786073206044.
25. **MODEJA:** Validar los requisitos del sistema. Marco de Desarrollo de la Junta de Andalucía. [En línea] 6 de octubre de 2011. [Citado el: 10 de marzo de 2017.http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/185.
26. **Pressman, R.S.,** Software engineering: a practitioner's approach. 7th ed. McGraw-Hil. New York, EUA.: 2010. ISBN:978-0-07-337597.
27. **Usaola, Macario Polo.** Escuela Superior de Informática-ESI. [En línea] http://www.inf-cr.uclm.es/www/mpolo/asig/0304/0102/patronesgrasp.pdf.
28. **Patrones-de-Diseno-Decorador-Decorator.** [En línea] 2014. [Citado el: 10 de marzo de 2017.] http://www.michael-pratt.com/blog/14/Patrones-de-Diseno-Decorador-Decorator.
29. **Patrones-de-Diseno-Inyeccion-de-dependencias.** [En línea] 2014. [Citado el: 10 de marzo de 2017.] http://www.michael-pratt.com/blog/14/Patrones-de-Diseno-Inyeccion-de-dependencias.
30. **Navarro, Autlán de**. SOFTWARE PARA VIATICOS. [En línea] junio de 2016. [Citado el: 3 de mayo de 2017.] https://www.sicar.mx/control-de-inventarios.
31. **Software para viáticos**. [En línea] 2015. [Citado el: 3 de mayo de 2017.] http://viaticos.com.mx**/**

32. **Diagrama de componentes**. [En línea] 2015. [Citado el: 30 de mayo de 2017.] https://www.altova.com/es/umodel/uml-component-diagrams.html
33. **Diagrama de despliegue.** [En línea] 2015. [Citado el: 30 de mayo de 2017.] https://www.ecured.cu/Diagrama_de_despliegue
34. **Métrica de diseño**. [Citado el: 30 de mayo de 2017]. https://www.ecured.cu/Métrica_de_diseño.
35. **Tenorio, Roberto Ruiz.** Las Pruebas de Software y su Importancia en las Organizaciones.
36. **Pruebas_de_software.** [Citado el: 30 de mayo de 2017]. https://www.ecured.cu/Pruebas_de_software
37. **García Ramírez, Christian y Pavón Reyes, Raydel. 2013.** Componente dinámico para la interoperabilidad de procesos. La Habana : s.n., 2013.