

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS



**Universidad de las Ciencias
Informáticas**

FACULTAD DE CIENCIAS Y TECNOLOGÍAS COMPUTACIONALES

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

Autores:

Karel Luis Dorzón Fonstecilla

Raimel Torres Ledesma

Título:

Sistema para la Gestión de Viáticos

Tutores:

MSc. Omar Mar Cornelio

Ing. Yoan Céspedes Williams

La Habana, junio 2017

“Año 59 de la Revolución”

“Si hay un secreto del buen éxito reside en la capacidad para apreciar el punto de vista del prójimo y ver las cosas desde ese punto de vista, así como del propio”

Henry Ford

Declaración de autoría

Declaramos ser los autores de la presente tesis: “Sistema para la Gestión de Viáticos” y se otorga a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo.

Para que así conste se firma la presente a los días____el mes_____de del año.

Raimel Torres Ledesma

Karel Luis Dorzón Fontecilla

Firma del Autor

Firma del Autor

MSc. Omar Mar Cornelio

Ing. Yoan Céspedes Williams

Firma del Tutor

Firma del Tutor

Datos de Contacto

Autor:

Raimel Torres Ledesma

Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: rtledesma@estudiantes.uci.cu

Autor:

Karel Luis Dorzón Fontecilla

Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: kldorzon@estudiantes.uci.cu

Tutor:

MSc. Omar Mar Cornelio

Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: omarmar@uci.cu

Tutor:

Ing. Yoan Céspedes Williams

Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: ycespedesw@uci.cu

Agradecimientos

Raimel Torres Ledesma: Quiero agradecer a toda mi familia por el apoyo incondicional que me dieron. A mi madre por siempre estar ahí en todo momento, por ser mi madre, ser como un padre, ser mi guía, mi mejor amiga, por serlo todo para mí. A mi papá por darme todos esos buenos consejos que me dio y sigue dando, por estar disponible cada vez que me hacía falta. A mi tío Ramoncito por ser mi segundo padre, siempre dispuesto a escucharme, darme consejo. A todos mis hermanos por de una forma u otra ser las personas que me han guiado a ser una mejor persona, tanto moral como intelectualmente, por jalarme las orejas cuando hacía falta y darme una palmadita en la espalda cuando lo acreditaba. A mi padrastro por la preocupación que ha mostrado hacia mi persona a pesar que no somos familia. A todas mis amistades de mi barrio, por estar siempre ahí en mi casa cada viernes que yo salía, por alegrarme todos los días que pasaba con ellos. A las amistades que conocí en esta escuela, por hacerme ver la vida de otra manera, por hacerme grata la mi estancia en la UCI. En resumen, gracias a todos por lo momentos que me han regalado, son parte de los mejores momentos que he podido tener en mi vida.

Karel Luis Dorzón Fonstecilla: Quiero agradecer a mi familia especialmente a mi abuela por su amor y comprensión. A mi madre por siempre estar siempre a mi lado, por ser madre y padre, por serlo todo para mí, doy gracias a Dios por su existencia. A mi papá por estar presente cada vez que me hacía falta y por su gran preocupación por mis estudios. A todas mis amistades que conocí en esta escuela, gracias por el apoyo cuando lo necesité.

Dedicatoria

Dedico este trabajo de diploma principalmente a mi madre, a mi padre, hermanos, tío ramoncito y a mi hija. Gracias a todos por darme lo mejor de cada uno. **Raimel.**

Dedico este trabajo de diploma a mi Familia. Especialmente a mi madre y. Gracias por apoyarme siempre, para ti todo mi amor y cariño. **Karel**

Resumen

En la actualidad las empresas e instituciones se inclinan por informatizar los procesos que las soportan, la dinámica de los cambios tecnológicos se acelera exponencialmente, permitiendo realizar propuestas de nuevos sistemas de Gestión, los cuales pueden trabajar de forma automática o semiautomática en procesos que anteriormente eran manuales. Enfocados en automatizar estos procesos surge la tarea de crear un Sistema informático para la gestión de viáticos o gastos de viaje. En el presente trabajo de diploma se realiza el análisis, diseño e implementación de las funcionalidades que permiten automatizar el proceso de Gestión de Viáticos en la Facultad de Ciencias y Tecnologías Computacionales. Dicho proceso se realiza de manera manual, acarreado como consecuencia que se cometan errores en los datos de modelos oficiales aprobados, sumado a esto se encuentra el hecho de que no es posible darle seguimiento al proceso de liquidación de viáticos, careciéndose de mecanismos que permitan generar reportes sobre el comportamiento de los viáticos. Para llevar a cabo la propuesta de solución se utilizó como metodología de desarrollo ágil AUP en su variante para la UCI, como Entorno de desarrollo Integrado PHPStorm 9.0, como lenguaje de modelado UML 2.0, como Sistema Gestor de Bases de Datos PostgreSQL 9.4 y como marco de trabajo Symfony 2.8 en conjunto con el marco de trabajo Xedro perteneciente a la Estrategia Marcaria de los productos de la UCI haciendo uso de PHP 5.5 como lenguaje de programación.

Palabras claves: gasto, gestión, liquidación, presupuesto, viático.

Abstract

Today companies and institutions are inclined to computerize the processes that support them, the dynamics of technological changes are accelerated exponentially, allowing proposals for new management systems, which can work automatically or semi-automatically in processes that were previously Manual. Focused on automating these processes arises the task of creating a computer system for the management of travel expenses or travel expenses. In the present work of University Degree, the analysis, design and implementation of the functionalities are carried out that allow to automate the process of Viatical Management in the Faculty of Sciences and Computational Technologies. This process is done manually, resulting in errors in the data of approved official models, added to this is the fact that it is not possible to follow the process of liquidation of viatical, lacking mechanisms to generate Reports on the behavior of travel expenses. In order to carry out the proposed solution, the AUP agile development methodology was used in its version for the UCI, as Integrated Development Environment PHPStorm 9.0, as a UML 8.0 modeling language, as PostgreSQL 9.4 Database Management System and as a framework of work Symfony 2.8 in conjunction with the framework Xedro belonging to the mark strategy of the products of the UCI making use of PHP 5.5 like programming language.

Key words: budget, expenditure, liquidation, management, viatical.

Índice

Introducción	1
Capítulo 1: Fundamentación Teórica del Sistema para la Gestión de Viáticos	7
1.1 Conceptos asociados al dominio del problema	7
<i>Presupuesto</i>	7
<i>Liquidación de Sueldo</i>	7
<i>Gastos financieros</i>	8
<i>Viático</i>	8
1.2 Análisis de Sistemas de Gestión de Viáticos	8
1.2.1 <i>Sistemas Extranjeros</i>	8
Conclusiones del estudio de las soluciones existentes	10
1.3 Herramientas y tecnologías a utilizar	11
1.3.1 <i>Herramienta para el modelado</i>	11
1.3.2 <i>Lenguajes de Programación</i>	12
1.3.3 <i>Entorno de Desarrollo Integrado</i>	13
1.3.4 <i>Marco de trabajo</i>	14
1.3.5 <i>Sistema Gestor de Base de Datos</i>	15
1.3.6 <i>Tecnologías web</i>	16
1.4 Marco de Trabajo Xedro	16
1.5 Metodología de Desarrollo de Software	16
1.5.1 <i>Metodología de desarrollo para la actividad productiva de la UCI</i>	17
Resultados esperados	19
Conclusiones Parciales	19
Capítulo 2: Modelado análisis y diseño del Sistema para la Gestión de Viáticos	20
2.1 Reglas del negocio	21
2.2 Actores del negocio	23
2.3 Trabajadores del negocio	23
2.4 Diagrama de Caso de Uso del Negocio	23
2.5 Casos de uso del negocio	24
2.6 Diagrama de actividades	26

2.7 Requisitos funcionales	27
2.7.1 Requisitos funcionales	28
2.8 Requisitos no funcionales	28
2.8.1 Requisitos no funcionales del sistema	29
2.9 Modelo de casos de uso del sistema	30
2.9.1 Diagrama de caso de uso del sistema	31
2.10 Patrones de caso de uso	31
2.11 Modelo de diseño	38
Diagrama de clases del diseño	40
2.11.1 Diagrama de clases del diseño del caso de uso Gestionar Viáticos	41
2.12 Patrones de diseño	42
2.13 Modelo de datos	43
2.14 Diagrama de despliegue	45
Conclusiones Parciales	46
Capítulo 3: Implementación y pruebas del Sistema para la Gestión de Viáticos	47
3.1 Modelo de implementación	47
3.1.1 Diagrama de Componentes	47
3.2 Código Fuente	48
3.2.1 Estándares de codificación	49
3.3 Pruebas del software	50
3.3.1. Niveles de pruebas	50
3.3.2 Tipos de pruebas	51
3.4 Métodos de pruebas	51
3.5 Diseño de casos de prueba	55
3.6 Resultado de las pruebas	65
Conclusiones parciales	65
Conclusiones generales	66
Referencias Bibliográficas	67
ANEXO	67

Índices de tablas

Tabla 1: Métodos teóricos.....	3
Tabla 2: Métodos empíricos.....	4
Tabla 3: Listado de reglas del negocio.....	22
Tabla 4: Descripción del actor del negocio.....	23
Tabla 5: Descripción de los trabajadores del negocio.....	23
Tabla 6: Descripción del caso de uso del negocio.....	24
Tabla 7: Descripción de los actores del sistema.....	30
Tabla 8: Descripción del caso de uso Gestionar Viático.....	32
Tabla 9: Resultado de la complejidad ciclomática.....	54
Tabla 10: DCP – Camino básico 1.....	56
Tabla 11: DCP – Camino básico 2.....	56
Tabla 12: DCP – Camino básico 3.....	57
Tabla 13: DCP – Camino básico 4.....	57
Tabla 14: DCP – Camino básico 5.....	57
Tabla 15: Secciones a probar en el caso de uso Gestionar Viáticos.....	58
Tabla 16: Descripción de variables del caso de prueba.....	61
Tabla 17: Matriz de datos del escenario Crear Viáticos.....	62
Tabla 18: Matriz de datos del escenario Modificar Viático.....	63
Tabla 19: Matriz de datos del escenario Eliminar Viático.....	64
Tabla 20: Matriz de datos del escenario listar viático.....	64

Índices de Figuras

Fig. 1: Fases e iteraciones del flujo de trabajo de la metodología AUP-UCI.....	18
Fig. 2: Explicación gráfica del escenario 1 de la metodología AUP – UCI.....	18
Fig. 3: Propuesta de solución para el problema de investigación.....	21
Fig. 4: Diagrama de caso de uso del negocio.....	24
Fig. 5: Diagrama de actividades.....	27
Fig. 6: Diagrama de caso de uso del sistema.....	31
Fig. 7 - Prototipo de pantalla: Crear viático.....	35
Fig. 8 - Prototipo de pantalla: Listar viáticos.....	36
Fig. 9 - Prototipo de pantalla: Modificar viático.....	37
Fig. 10 - Prototipo de pantalla: Eliminar viático.....	38
Fig. 11: Arquitectura de Symphony2.....	39
Fig. 12: Diagrama de clase de diseño del caso de uso Gestionar Viático.....	41
Fig. 13: Diagrama Entidad-Relación.....	44
Fig. 14: Diagrama de despliegue.....	45
Fig. 15: Diagrama de componentes.....	48
Fig. 16: Código del método crearAction.....	50
Fig. 17: Ejemplo del código crearAction.....	53
Fig. 18: Flujo del método crearAction.....	53

Introducción

En la actualidad, uno de los principales problemas a los que se enfrentan la mayoría de los países es la superación económica, debido a la gran escasez de recursos y las crecientes necesidades a la cual se enfrenta el mundo. Uno de los aspectos más significativos en todas las empresas es el control de los costos de cualquier índole. Este tema es indispensable sin importar la magnitud que tengan, debido a esto los directivos deben tener un dominio de todas las características y pasos de este proceso para poder tener un buen control de los costos. A nivel mundial las instituciones se inclinan por informatizar los procesos que las soportan, optimizando la prestación de servicios, la dinámica de los cambios tecnológicos se acelera exponencialmente, lo cual permite realizar propuestas de nuevos sistemas de gestión, los que pueden trabajar de forma semiautomática o automática en procesos que anteriormente eran manuales.

Cuba no está exento de este proceso, por ende, trata de informatizar la sociedad. En la Universidad de las Ciencias Informáticas UCI, específicamente en el Vice-decanato administrativo de la Facultad de Ciencias y Tecnologías Computacionales CITEC la manera en que se gestiona los viáticos (dietas) engloba varios procedimientos dentro de este, relacionados con los temas económicos en general, entre estos se encuentra la Gestión de Viáticos, el cual se realiza de forma manual.

Teniendo en cuenta lo planteado en la UCI, los trabajadores realizan actividades correspondientes al flujo de trabajo de los proyectos tales como levantamiento de requisitos, actualización de soluciones informáticas y despliegue de aplicaciones. En las tareas anteriormente mencionadas, los profesionales desempeñan su función en instituciones fuera de la universidad, por lo que se les garantiza un viático para su alimentación o alojamiento (llamados gastos de viaje o dietas), estos pueden ser de tipo anticipo (es cuando la entidad cubre los gastos que se le solicita) o tipo liquidación (es cuando el trabajador cubre los gastos y después la entidad le reintegra el dinero al finalizar el proceso al que asistió).

Al no existir un procesamiento informatizado en la facultad CITEC, se hace engorroso el proceso de solicitud y aprobación de un viático, ya que el trabajador debe dedicar tiempo para acudir a su jefe de departamento, hacer una solicitud para participar en un evento o convención, éste a su vez al llenar la solicitud asiste al Vice-decano administrativo para la revisión de dicha solicitud, una vez aprobada el Vice-decano envía la misma a la secretaria administrativa para así crear el modelo del viático. Una vez terminado le hace entrega al Vice-decano administrativo para darle el primer nivel de aprobación y éste a su vez se lo entrega al

Decano para el segundo nivel de aprobación, todo esto conlleva a que aumente el tiempo de creación y aprobación de un viático, además de estar expenso a posibles errores ortográficos a la hora de llenar los campos del viático. Además, en las reuniones de balance se necesita la información de todos los viáticos otorgados ya sea por mes o por usuario, conlleva un mayor consumo de tiempo al organizar todos los viáticos generados en un determinado mes, año o de un usuario específico. También no es posible darle seguimiento al proceso de liquidación de los viáticos y se carece de mecanismos que permitan generar reportes sobre el comportamiento económico de la partida inherente al viático.

De la situación problemática se deriva el siguiente **problema de investigación** a resolver: ¿Cómo contribuir a la gestión de la información de la facultad de Ciencias y Tecnologías Computacionales para facilitar el proceso de Gestión de Viáticos?

Siendo el **objeto de estudio**: los sistemas de información para la gestión de viáticos, enmarcándose en el **campo de acción** la gestión de viáticos para la facultad CITEC.

Para darle solución al problema de investigación se propone como **objetivo de la investigación** desarrollar un sistema para la gestión de la información referente al proceso de viático de la Facultad de Ciencias y Tecnologías Computacionales.

Dicho objetivo general puede desglosarse en los siguientes **objetivos específicos**:

- Caracterizar los referentes teóricos para la elaboración del Sistema para la Gestión de Viáticos.
- Realizar el análisis del Sistema para la Gestión de Viáticos.
- Diseñar el Sistema para la Gestión de Viáticos.
- Implementar el Sistema para la Gestión de Viáticos.
- Validar el Sistema para la Gestión de Viáticos a través de pruebas.

Para dar cumplimiento al objetivo general, se trazan las siguientes **tareas de investigación**:

- Identificación de los referentes teóricos relacionados con los sistemas informáticos que permitan el control de los gastos por conceptos de viáticos para elaborar el marco teórico de la investigación.
- Fundamentación de las herramientas, la metodología y tecnologías a utilizar para el desarrollo del Sistema para la Gestión de Viáticos.
- Análisis del Sistema para la Gestión de Viáticos que permita determinar los requisitos funcionales y no funcionales que debe cumplir el sistema a implementar.

- Diseño del Sistema para la Gestión de Viáticos para determinar los detalles acerca de la arquitectura, las interfaces y los componentes del software, necesarios para implementar el sistema.
- Implementación del Sistema para la Gestión de Viáticos para la construcción del software a partir de los resultados obtenidos en las fases de análisis y diseño.
- Elaboración de los casos de pruebas que serán aplicados en el Sistema para la Gestión de Viáticos que permitan guiar la ejecución del proceso de prueba.
- Realización de las pruebas correspondientes para comprobar el correcto funcionamiento del Sistema para la Gestión de Viáticos.

En la investigación se empleó un conjunto de métodos empíricos y teóricos, tales como:

Del nivel teórico:

Tabla 1: Métodos teóricos

Método	Descripción	Utilización
Modelación	Este se utiliza para realizar la modelación de los procesos de negocio.	Permitir la modelación de los diferentes procesos del negocio, mediante diagramas y modelos de disímiles tipos.
Análisis-Síntesis	Permitirá hacer un análisis y estudio del objeto de la investigación posibilitando así procesar toda la información enfocada en el proceso de gestión de viáticos, lo cual lleva consigo organizar y simplificar el análisis de todo volumen de datos a recopilar en fragmentos más factibles.	Permitir determinar las diferentes aplicaciones derivadas de ello, lo que posibilita conocer sus características generales y las relaciones esenciales mediante el análisis.

Histórico-Lógico	Analiza la trayectoria completa del fenómeno, su condicionamiento a los diferentes periodos de la historia, revela etapas principales de su desenvolvimiento y las conexiones históricas fundamentales.	Establecer un estudio bibliográfico y una base sólida de fundamentos teóricos que permita relacionar el análisis documental y estado de los sistemas de gestión de viáticos.
-------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Del nivel empírico:

Tabla 2: Métodos empíricos

Método	Descripción	Utilización
Entrevista	<p>Técnica de licitación más utilizada, y de hecho es prácticamente inevitable en cualquier desarrollo, ya que es una de las formas de obtener información más natural entre las personas.</p> <p>Fases: preparación, realización y análisis.</p>	Se utilizará el proceso de comunicación verbal con los implicados en el centro y con otras personas que contribuyen en la investigación del objeto de estudio con vista a recopilar informaciones, en relación con una finalidad establecida en la investigación.

La trascendencia social, devenido el aporte práctico de este trabajo de diploma, radica en la posibilidad de mejorar la Gestión de Viáticos en la facultad CITEC, a su vez la solución que se propone puede ser implementada en otras facultades, pues se tuvo en cuenta las regularidades y reglamentos establecidos para este tipo de gestión en la UCI.

El trabajo investigativo consta de tres capítulos organizados de la siguiente forma:

Capítulo 1: Fundamentación Teórica del Sistema para la Gestión de Viáticos: en el presente capítulo se describen los elementos principales para la fundamentación del marco teórico de la investigación. También se abordan los conceptos fundamentales relacionados con los Sistemas para la Gestión de Viáticos; se realiza una investigación sobre las diversas soluciones existentes, de igual manera se muestran las características de la metodología, las herramientas y los lenguajes de programación que se van a utilizar para el desarrollo del Sistema para la Gestión de Viáticos de la facultad CITEC.

Capítulo 2: Modelado, análisis y diseño del Sistema para Gestión de Viáticos: en el presente capítulo se describe la propuesta de solución para el problema de investigación, se especifican los requisitos funcionales y no funcionales como características imprescindibles que debe cumplir el software. También se muestra el diagrama de caso de uso del negocio, las reglas, los trabajadores y actores del negocio, así como el diagrama de actividades. Todo esto para plasmar un mejor entendimiento del negocio. Se muestra el diagrama de caso de uso del sistema, diagrama de clase del diseño del caso de uso Gestionar Viáticos y el diagrama entidad-relación, este último para mostrar cómo estará conformada la base de datos.

Capítulo 3: Implementación y pruebas del Sistema para la Gestión de Viáticos: en este capítulo se describe el modelo de implementación como resultado del diseño anteriormente desarrollado, ejemplificado por el diagrama de componentes del caso de uso Gestionar Viáticos, los estándares de codificación. Se describen las pruebas realizadas, con el objetivo de comprobar las funcionalidades del software en los diferentes escenarios para, de esta forma, verificar en todos los casos que los resultados sean los esperados y, por tanto, el correcto funcionamiento del sistema.

Capítulo 1: Fundamentación Teórica del Sistema para la Gestión de Viáticos

En el presente capítulo se describen los elementos principales para la fundamentación del marco teórico de la investigación. También se abordan los conceptos fundamentales relacionados con los Sistemas para la Gestión de Viáticos; se realiza una investigación sobre las diversas soluciones existentes, de igual manera se muestran las características de la metodología, las herramientas y los lenguajes de programación que se van a utilizar para el desarrollo del Sistema para la Gestión de Viáticos de la facultad CITEC.

1.1 Conceptos asociados al dominio del problema

En el presente epígrafe se conceptualiza un conjunto de elementos relacionados con el objeto de estudio de la investigación, los cuales se describen a continuación para una mejor comprensión del trabajo.

Presupuesto

Es el cálculo y negociación anticipada de los ingresos y egreso de una actividad económica (personal, familiar, un negocio, una empresa, una oficina, un gobierno) durante un período, por lo general en forma anual. Es un plan de acción dirigido a cumplir un final previsto, expresado en valores y términos financieros que debe cumplirse en determinado tiempo y bajo ciertas condiciones previstas, este concepto se aplica a cada centro de responsabilidad de la organización. (CEIN, 2008).

Liquidación de Sueldo

La liquidación de sueldo es un documento que debe entregar al trabajador y este firmarlo, con el cual usted como empleador puede comprobar el pago de sueldo a su trabajador. En este se indican los montos pagados por sueldo base, cargas familiares y los descuentos legales de pensiones y salud. (CEIN, 2008).

Gastos financieros

Aquellos en los que incurre un sujeto económico para la obtención, uso o devolución de capitales financieros puestos a su disposición por terceras personas. Todos aquellos gastos originados como consecuencia de financiarse una empresa con recursos ajenos. (CEIGE, 2009).

Viático

Para los efectos de la legislación laboral, se entiende por viático la suma de dinero razonable y prudente que los empleadores pagan a los trabajadores a fin de que éstos solventen los gastos de alimentación, alojamiento o traslado en que incurran con motivo del desempeño de sus labores, siempre que para dicho efecto deban ausentarse del lugar de su residencia habitual. (BETSIME, 2011).

1.2 Análisis de Sistemas de Gestión de Viáticos

A continuación, se exponen algunas de las soluciones existentes que facilitan la gestión de la información referente a los viáticos en el ámbito nacional e internacional y que permitirá conocer las características de los sistemas y el posible aporte a la solución.

1.2.1 Sistemas Extranjeros

Viáticos-Portal

Portal electrónico para la administración eficiente de viáticos desarrollado por **EDICOM** compañía mexicana de software para aplicaciones de negocios con filiales en gran parte del mundo. Viáticos-Portal está compuesto por una serie de áreas funcionales o módulos que responden a los procesos operativos de las organizaciones.

La herramienta permite la administración inteligente de los documentos de viáticos a partir del establecimiento de un flujo de trabajo que se personaliza de acuerdo al organigrama de cada departamento afectado. Se definen diferentes roles de usuarios administradores que reciben notificaciones automáticas de revisión de los documentos de gastos para su validación. Posteriormente se someten a un tratamiento automático, que permite su integración con el sistema de gestión contable.

EDICOM incorpora a sus soluciones tecnológicas avanzados modelos de gestión que permiten a sus usuarios exterminar la administración y mantenimiento de grandes plataformas de comunicaciones, consiguiendo así un óptimo funcionamiento de todos sus sistemas. (edicom, 2013)

Edivolt

Edivolt (control de los viáticos y gastos de viaje de su empresa) control de viáticos es una solución en la nube, para administrar de forma eficiente los gastos de viaje y reembolso de viáticos de empresas de cualquier tamaño. (edivolt, 2013).

Tiene como características que:

- Las solicitudes se pueden centralizar, de esta manera recibir correo electrónico a manera de notificación.
- En el ámbito de autorización se puede configurar los flujos de aprobación de acuerdo a sus políticas internas.
- Los comprobantes de gastos se validan automáticamente.
- Sus reportes de gastos se generan automáticamente, evitando el papeleo y la captura manual.
- El mismo acelera la aprobación de sus reembolsos de gastos.
- Puede consultar reportes y gráficas para un mejor análisis.
- Tiene una completa integración vinculado el ERP (del inglés, *Enterprise Resource Planning*¹) para automatizar la captura de la información contable.

Infor Gestión de Gastos

Este software de Gestión de Gastos reduce los costos y aumenta la eficacia de los procesos de egresos e ingresos iniciados por los empleados, incluso los gastos de viajes y esparcimiento, la planificación y aprobación previa de los viajes, la captura de tiempos de recarga, y el comienzo del proceso de pago a los proveedores. Con diseño modular, la solución de Gestión de Gastos posibilita que las empresas sólo seleccionen las funciones que responden a sus requerimientos comerciales específicos, y la flexibilidad para el crecimiento. (infor, 2013).

La solución le provee:

- **Reporte de Gastos.**

¹ ERP – Empresa de Recursos Empresariales.

- **Planes de Viaje.**
- **Payment Requests².**
- **Asistencia de dispositivos móviles.**
- **Gestión de acuses de recibo digitales.**
- **Opciones de entregas flexibles y accesibles.**

Endalia

Gestiona todo el proceso de viajes, desde la solicitud inicial hasta la aprobación o rechazo de los gastos asociados. Reduce la carga administrativa del proceso controlando el seguimiento de todas las solicitudes. (endalia, 2014).

Funcionalidad básica

- Parametrización de los distintos conceptos y tipos de viaje, definiendo para cada tipo sus características (requerimiento de aprobación, tipo de gastos asociados, estructura de aprobadores)
- Gestionar el proceso completo de solicitud de viaje y aprobación de gastos: solicitud por parte del empleado, aprobación/rechazo por parte del responsable de aprobación, envío de los gastos, aprobación de los gastos.
- Posibilidad de definir diferentes estructuras de aprobación dependiendo del tipo de viaje
- Posibilidad de parametrización de diferentes políticas de límite de gastos según los distintos niveles de empleados de la organización
- Extracción de datos a Excel y generación de múltiples informes en diferentes formatos (PDF (siglās del inglés, *Portable Document Format*³), HTML (siglas del inglés, *HyperText Markup Language*⁴), Word). Generación de diversas estadísticas de gastos.

Conclusiones del estudio de las soluciones existentes

Después de haber realizado el estudio de las soluciones existentes antes mencionadas, donde la mayoría son sistemas propietarios y no permiten el acceso a su código, imposibilitando que éste pueda ser reutilizado, ninguna otorga la totalidad de funcionalidades que se necesita, añadiéndole a esto que no se encuentran alineados a las políticas de migración hacia el uso de software libre que presenta la UCI,

² Payment Request – Solicitud de Pago.

³ PDF – Formato de Documento Portátil.

⁴ HTML – Lenguaje de Marcas de Hipertexto.

teniendo en cuenta lo anterior supondrían trabas a la hora de establecer cualquiera de los sistemas anteriores, aunque sean soluciones viables.

1.3 Herramientas y tecnologías a utilizar

Se definen como herramientas y tecnologías de desarrollo las especificadas por la Estrategia Marcaria de los productos UCI, específicamente, Xedro. Este permite la gestión del sistema del lado del cliente sobre el cual trabaja dicho proyecto y las definidas para la versión anterior para lograr una mayor uniformidad de la información que se maneja. Además, también se tuvo en cuenta la característica de ser multiplataforma y estar desarrollada bajo licencia libre, siguiendo el principio de independencia tecnológica del país. A continuación, se explican cada una de estas.

1.3.1 Herramienta para el modelado

Las Herramientas CASE (siglas del inglés, *Computer Aided Software Engineering*⁵) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como: el proceso de realizar un diseño del proyecto, cálculo de costes, implementación automática de parte del código con el diseño dado, compilación automática y documentación o detección de errores.

Visual Paradigm para UML v8.0

Visual Paradigm para UML es una herramienta CASE que modela UML, la misma soporta, completamente el ciclo de vida del desarrollo de software: Análisis y Diseño Orientados a Objetos, Construcción, Pruebas y Despliegue. También permite la generación automática de reportes en formato PDF y HTML, el reconocimiento de artefactos de ingeniería a partir de reconocimiento de textos, implementa una actualización automática del modelo de diseño y código permitiendo mantener la documentación de ambos modelos actualizados con los cambios que ocurran en ambos sentidos, optimizando la descripción textual de elementos de código a partir de la descripción visual. (Paradigm, 2013).

⁵ CASE – Ingeniería de Software Asistida por Ordenador.

Dentro de sus principales características están:

- ✓ Diseño enfocado al negocio que generan un software de mayor calidad.
- ✓ Generación de bases de datos. Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Ingeniería inversa de bases de datos. Desde Sistemas Gestores de Base de Datos existentes a diagramas Entidad-Relación.

1.3.2 Lenguajes de Programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humano.

Está formado por un conjunto de símbolos sintácticos y semánticos que definen su estructura y el significado de sus elementos. Al proceso por el cual se describe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. (Lutz, 2010).

Pre-procesador de hipertexto (PHP)

Es un lenguaje de programación de uso general de código de lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación de lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. (Pérez, Julian y Gardey, Ana, 2012).

JavaScript

Es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web, no requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. Es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos

mucho más complejos. Con JavaScript se puede crear efectos e interactuar con los usuarios. (Pérez, Damian, 2007).

Lenguaje de marcas

HTML v5.0

El Lenguaje de Marcas de Hipertexto es un conjunto de etiquetas o comandos, complementados en la mayoría de los casos por extensiones que permiten dar formato a un archivo, con el objetivo básico de crear un documento que pueda ser visualizado en ambiente Internet en forma de Página Web y que esta, además, pueda, por medio de dichas etiquetas, tener la estructura o forma deseada por quien la diseñó. (Pérez, Julian y Gardey, Ana, 2012).

1.3.3 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. En algunos lenguajes, un IDE (siglas del inglés, *Integrated Development Environment*⁶) puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. (Maldonado, 2007).

PhpStorm v9.0

Para el desarrollo de la solución se seleccionó como IDE el PhpStorm v9.0. Este editor de código provee excelente soporte en diferentes tipos de lenguaje de programación como *JAVASCRIPT*. Tiene muchas características innovadoras y funciones que facilitan la realización del proyecto de manera ágil y eficiente.

Algunas de las principales características que cuenta *PhpStorm* son:

- ✓ Editor de código inteligente *PHP*.
- ✓ Calidad de análisis en el código.
- ✓ Entorno de desarrollo.

⁶ IDE – Entorno de Desarrollo Integrado

- ✓ Editor *HTML/CSS/JavaScript*.
- ✓ Depuración y pruebas.
- ✓ Experiencia multiplataforma.

1.3.4 Marco de trabajo

Los marcos de trabajo son un conjunto de conceptos, prácticas y criterios que sirven de base y facilitan el desarrollo de software. Para el desarrollo de aplicaciones web constituye buena práctica el uso de marcos de trabajo PHP que definen una filosofía de trabajo e implementan patrones de programación orientados a la Web.

Entre los marcos de trabajo a utilizar figuran

jQuery

Es un marco de trabajo basado en el lenguaje JavaScript, permite desarrollar la interfaz gráfica de usuario compatible con los estándares propios de los navegadores. Este marco de trabajo, ofrece una arquitectura con la que se tendrá mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery se obtiene ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Además, todas estas ventajas facilitan el trabajo, ya que el marco de trabajo tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. (Manzur, 2015).

Para la realización de este proyecto jQuery incluye un conjunto de funciones que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM (siglas del inglés, *Document Object Model*⁷), manejar eventos, además de facilitar la culminación con éxito del módulo web en cuestión. Con una combinación de versatilidad y capacidad de ampliación, jQuery ha cambiado la forma en que millones de personas escriben JavaScript, siendo hoy una de las más utilizada.

⁷ DOM – Modelos de Objetos del Documento.

Bootstrap v2.3.2

Es un marco de trabajo que permite crear interfaces web con CSS (del inglés⁸, *Cascading Style Sheets*) y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio se adapta automáticamente al tamaño de su PC, Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como diseño adaptativo. Es de código abierto, por lo que se puede utilizar de forma gratuita y sin restricciones. (Solis, 2014).

Symfony 2

Es un completo marco de trabajo diseñado para optimizar gracias a sus características el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y superior, por eso es uno de los marcos de trabajo PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajen en un proyecto. (Symfony.es, 2011).

1.3.5 Sistema Gestor de Base de Datos

PostgreSQL 9.4

Es un sistema objeto-relacional, este incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. No es un sistema de gestión de bases de datos puramente orientado a objetos. PostgreSQL está considerado como una de las bases de datos de código abierto más avanzadas del mundo. Proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. (Martínez, Rafael, 2010)

⁸ CSS – Hojas de Estilo en Cascada.

Además de las características mencionadas anteriormente PostgreSQL fue diseñado para ambientes de alto volumen y tiene soporte para vistas, procedimientos almacenados en el servidor, transacciones y almacenamiento de objetos de gran tamaño.

1.3.6 Tecnologías web

Servidor Web Apache v2.4

Apache es un servidor web multiplataforma, lo que lo hace prácticamente universal, es una tecnología de código fuente abierto, presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto. (Apache.org, 2015)

1.4 Marco de Trabajo Xedro

El marco de trabajo Xedro se estructura de manera tal que facilita la reutilización de los diferentes componentes y es de fácil entendimiento para quienes desarrollen sobre el mismo. Xedro utiliza jQuery para implementar la capa de presentación, se apoya en Symfony para el desarrollo de la lógica del negocio. Este utiliza como estilo arquitectónico Modelo Vista Controlador. También tiene como propósito insertar la programación orientada a objeto. (UCI, Estrategia Mercaria de los productos de la UCI, 2013)

1.5 Metodología de Desarrollo de Software

Las metodologías para el desarrollo del software imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Una metodología de desarrollo de software tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos.) exigiéndose así que el proceso sea configurable. (Sánchez Rodríguez, Tamara, 2014)

1.5.1 Metodología de desarrollo para la actividad productiva de la UCI

Variación de AUP

Con el objetivo de guiar el proceso de desarrollo del software para la Gestión de Viáticos se escoge la metodología AUP en su variación AUP-UCI, siendo una metodología ágil que mantiene los conceptos de RUP (del inglés, *Rational Unified Process*⁹), y de manera fácil describe como desarrollar proyectos de software.

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP) en inglés, es una versión simplificada del Proceso Unificado de Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio utilizando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. (Sánchez Rodríguez, Tamara, 2014).

Fases de AUP para la UCI

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, además de decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elabora la arquitectura, el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

La metodología AUP-UCI se adapta a la configuración y ciclo de vida de la actividad productiva de la UCI, con esta se da cumplimiento a las buenas prácticas que define CMMI-DEV v1.3. El lenguaje de trabajo es totalmente entendible logrando con su ejecución un producto de trabajo de software con la calidad requerida. Consta de los principales aspectos positivos de la mayoría de las metodologías ágiles adaptándose

⁹ RUP – Proceso Racional Unificado.

correctamente a equipos de desarrollo pequeños; es por esto que se define a AUP-UCI como la metodología a utilizar.

Teniendo en cuenta el escenario No. 1 de dicha metodología

Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado, obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio o actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Los casos de uso del negocio (CUN) muestran como los procesos son llevados a cabo por personas y los activos de la organización.

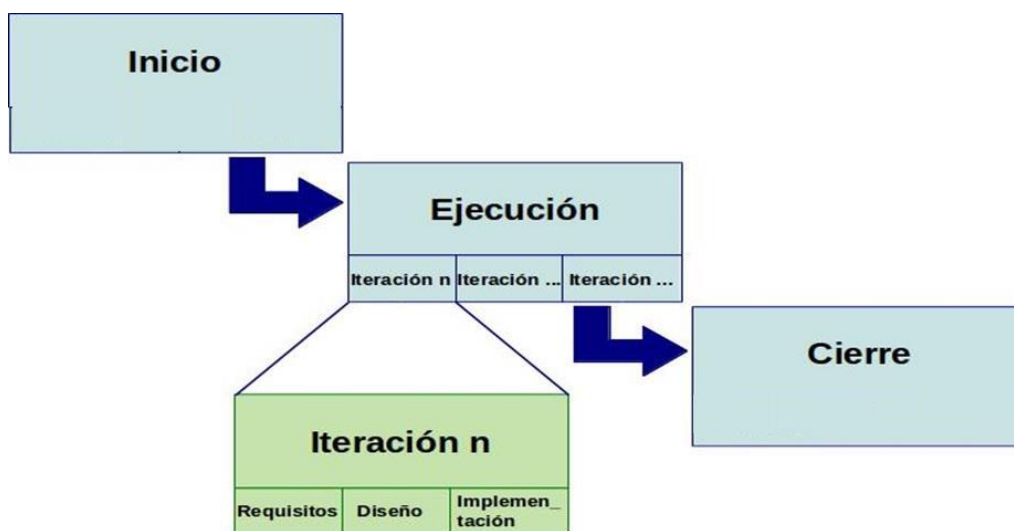


Fig. 1: Fases e iteraciones del flujo de trabajo de la metodología AUP-UCI

Escenario No 1

Proyectos que no modelen negocio solo pueden modelar el sistema con CUS.



10

Fig. 2: Explicación gráfica del escenario 1 de la metodología AUP – UCI

¹⁰ CUN – Caso de Uso del Negocio.
 MC – Modelo Conceptual. CUS – Caso de Uso del Sistema

Resultados esperados

Sistema para la gestión de información que soporte el flujo de trabajo del proceso de viático y contribuya con:

- El control de los gastos de transportación.
- El control de los gastos de alimentación.
- La generación de reportes sobre la ejecución presupuestaria.
- El control contable del presupuesto de viático.

Conclusiones Parciales

El estudio de la bibliografía e investigación proporcionó elementos suficientes y necesarios para confirmar la necesidad de implementar un sistema que sea capaz de acoplar las necesidades nutrientes de las ventajas que presentan las aplicaciones analizadas. La utilización de las herramientas y tecnologías propuestas favorece eficiencia y rapidez a lo largo del ciclo de vida del desarrollo de la solución y soberanía tecnológica. Teniendo en cuenta el análisis de la bibliografía consultada y en correspondencia con las características la solución a realizar se plantea la utilización de la metodología AUP-UCI. Se utilizará el Visual Paradigm para UML v8.0; como lenguaje de modelado UML 2.0 para el diseño de los artefactos. Como Gestor de Bases de Datos se utilizará PostgreSQL v9.4, como marco de trabajo Symfony v2.8, utilizando como lenguaje de programación PHP 5.3.5 y como servidor web Xampp v3.2.1 además de la biblioteca jQuery v1.9.1.

Capítulo 2: Modelado análisis y diseño del Sistema para la Gestión de Viáticos

En el presente capítulo se describe la propuesta de solución para el problema de investigación, se especifican los requisitos funcionales y no funcionales como características imprescindibles que debe cumplir el software, y su diagramación a través de casos de uso contribuyendo al cumplimiento del objetivo general de la investigación, empleando para eso patrones de casos de uso. También se muestra el diagrama de caso de uso del negocio con su descripción, las reglas, trabajadores y actores del negocio, así como el diagrama de actividades. Todo esto para plasmar un mejor entendimiento del negocio. Se muestra el diagrama de clase del diseño del caso de uso Gestionar Viáticos y el modelo de datos o diagrama entidad-relación, este último para mostrar cómo estará conformada la base de datos.

Breve descripción del proceso

La Gestión de los Viáticos (dietas) engloba varios procesos dentro de éste, relacionados con los temas de costos, y económicos en general, ya sea en una empresa o un centro de trabajo, y dentro de estos muchos procesos se encuentra la Gestión de Viáticos, que no es más que el presupuesto asignado para cubrir los gastos en término de alimentación y hospedaje que solicita un trabajador a su empresa para poder asistir a algún evento o convención. Éste puede ser de tipo anticipo (es cuando la empresa cubre los gastos que se le solicita) o tipo liquidación (es cuando el trabajador cubre los gastos y después la empresa le reintegra el dinero al finalizar el evento al que asistió).

El trabajador que desea participar en un evento, expresa a su jefe de departamento el motivo de su viaje, además de otorgarle los datos de contacto del mismo junto con las fechas de dicho evento. El jefe de departamento realiza la solicitud y se la envía al Vice-decano administrativo para su aprobación, una vez aprobada se la envía a la secretaria administrativa para la creación del modelo del viático. La secretaria administrativa crea el modelo relleno los campos Nombre, Apellidos, Solapín, Motivo de Viaje, Fecha de salida estimada, Fecha de salida, Fecha de regreso, Fecha de liquidación, Cantidad solicitada, Gasto en Hospedaje, Alimentación y Transporte, aparte de hacer un desglose de gastos por día teniendo en cuenta los anteriores parámetros, esta envía dicho modelo al Vice-decano administrativo para otorgar el primer nivel de aprobación (revisión). El Vice-decano administrativo envía el modelo al decano para que esta le otorgue el segundo nivel de aprobación (firma), una vez firmado lo reenvía al Vice-decanato administrativo para notificarle al trabajador que puede pasar a recoger el modelo del viático. Ya realizado esto, el trabajador

debe recoger el modelo de dieta y presentarlo en el departamento de economía, para que el dinero sea ingresado a su cuenta.

Propuesta de solución

La solución más adecuada, es la realización de un Sistema de Gestión web para la facultad CITEC que facilite todo el trabajo en la etapa del proceso de solicitud de viático y que controle todos los procesos asociados al mismo, como son: la creación de solicitudes para un viático, aprobación de solicitudes, creación de viáticos, modificación de viáticos, eliminación de viáticos, impresión de viáticos, exportación de reportes, y otros. El sistema web es mucho más factible para sistemas de gestión y control de información donde intervienen clientes que se encuentran interrelacionados y comparten información relevante en el proceso. La inmediatez de la información debe ser un principio básico, por lo que una plataforma web viabiliza el flujo continuo de la información necesaria para la gestión de un proceso viático.

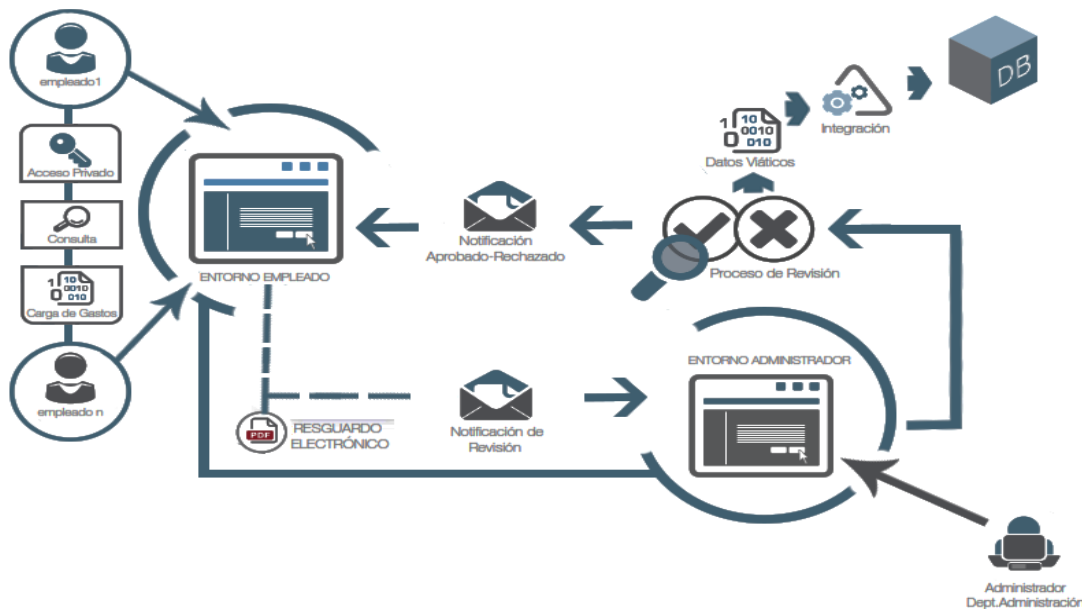


Fig. 3: Propuesta de solución para el problema de investigación

2.1 Reglas del negocio

Según Business Rules Group, que es una organización cuyo propósito es fomentar el entendimiento y estandarización del concepto de reglas de negocio, se basa en dos perspectivas para definir una regla de negocio:

- Desde la perspectiva del negocio, es una orientación en la cual hay una organización conducida por una acción, práctica o proceso, dentro de una particular actividad o giro.
- Desde la perspectiva de sistemas de información, es una declaración que define o restringe algunos aspectos del negocio. Intenta hacer valer la estructura del negocio, o controlar o influir la conducta del negocio. (Dsc. Becrril, Jorge, 2010).

Tabla 3: Listado de reglas del negocio

No	Tipo	Nombre	Descripción
1	Relación	Primer nivel de aprobación de un viático.	El único que está autorizado a otorgar el primer nivel de aprobación a un viático es el Vice-decano administrativo.
2	Relación.	Segundo nivel de aprobación de un viático.	El único autorizado a otorgar el segundo nivel de aprobación a un viático es el decano.
3	Textual.	Confección de solicitud.	Para esta concesión es necesario que el jefe de departamento del trabajador introduzca el motivo por el cual solicita el viático.
4	Relación.	Creación de un viático.	Las únicas personas autorizadas a crear un viático son la secretaria administrativa o el Vice-decano administrativo.
5	Modelo de datos.	Fecha y hora.	La fecha estará dada en el formato de dd/mm/aa y la hora estará dada en formato hh:mm (horario de verano).
6	Modelo de datos.	Viático de tipo anticipo.	Se rellena el campo "entregado" y se precisa el campo "salida estimada".
7	Modelo de datos.	Viáticos de tipo liquidación.	Se rellenan los campos "utilizado", "regreso" y "salida", no se rellena el campo "salida estimada".
8	Modelo de datos.	Desglose de alimentación.	Los campos de la tabla de "desglose de alimentación" tienen que estar llenos en su totalidad.
9	Derivación.	Fecha.	La fecha de salida tiene que ser menor a la fecha de regreso.
10	Derivación.	Hora en "desglose de alimentación".	En los casos en que la hora de salida o regreso, sea mayor de 7:00am, 2:00pm y 8:00pm no se efectuará el pago a desayuno, almuerzo o comida respectivamente.

2.2 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. (Dra. Hernández González, Anaisa, 2012).

Tabla 4: Descripción del actor del negocio

Actor	Descripción
Trabajador	Profesor o especialista de la facultad que necesita viajar.

2.3 Trabajadores del negocio

Un trabajador del negocio representa un rol que juega una persona o grupo de personas, una máquina o un sistema automatizado; actuando en el negocio. Son los que realizan las actividades, interactuando con otros trabajadores del negocio y manipulando entidades. (Dra. Hernández González, Anaisa, 2012).

Tabla 5: Descripción de los trabajadores del negocio

Trabajador	Descripción
Secretaria administrativa	Persona encargada de recibir la solicitud del viático, rellenar los campos restantes del mismo y enviarlo al Vice-decano administrativo.
Jefe de departamento	Persona encargada de crear la solicitud y entregarla al Vice-decano administrativo.
Vice-decano Administrativo	Persona encargada de revisar el modelo, otorgarle el primer nivel de aprobación y enviárselo al Decano.
Decano	Persona encargada de revisar el modelo del viático, otorgarle el segundo nivel de aprobación del viático y reenviarlo al Vice-decanato administrativo.

2.4 Diagrama de Caso de Uso del Negocio

Constituye una representación gráfica entre los actores y los casos de uso, así como las relaciones y dependencias que se establecen entre ellos. (Dra. Hernández González, Anaisa, 2012).

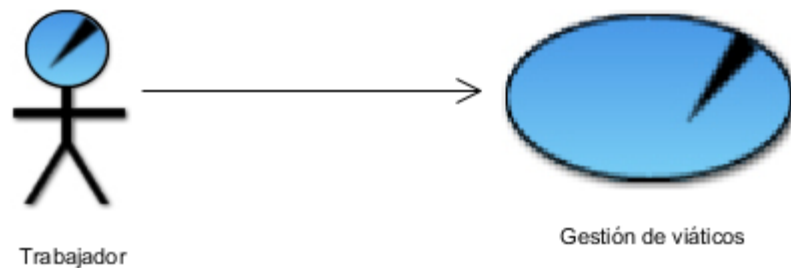


Fig. 4: Diagrama de caso de uso del negocio

2.5 Casos de uso del negocio

Es una secuencia de acciones, realizadas en el negocio, que producen un resultado de valor observable para los actores del negocio. Desde la perspectiva de un actor individual, define un flujo de trabajo completo que produce resultados deseados (Dra. Hernández González, Anaisa, 2012).

Tabla 6: Descripción del caso de uso del negocio

Caso de uso del negocio	Gestión de Viáticos.
Actores	Trabajador.
Resumen	El caso de uso comienza cuando el trabajador solicita a su jefe de departamento una carta que le permita abalar la necesidad del viaje. Una vez redactada la solicitud, el jefe de departamento envía al Vice-decano administrativo dicha carta para su aprobación, este se lo envía a la secretaria administrativa para la creación del modelo del viático, una vez creado esta envía el modelo al Vice-decano administrativo para su aprobación y este a su vez lo envía a la decana para que lo firme. El caso de uso termina cuando la Decana reenvía dicho modelo al Vice-decano administrativo para que se le notifique al trabajador que solicitó el viático.
Casos de uso relacionados	No procede

Acción del actor	Respuesta de la organización
1. El trabajador realiza la solicitud del viático.	
	<p>2. El jefe de departamento redacta la solicitud y se la entrega al Vice-decano administrativo para su revisión.</p> <p>3. El Vice-decano administrativo le envía la solicitud aprobada la secretaria administrativa para que cree el modelo del viático.</p> <p>4. La secretaria administrativa crea el modelo del viático y lo envía al Vice-decano administrativo para su aprobación.</p> <p>5. El Vice-decano administrativo aprueba el modelo del viático y se lo envía al Decano para que lo firme.</p> <p>6. El Decano firma el modelo del viático y lo envía al Vice-decanato administrativo.</p> <p>7. El trabajador es notificado de que su solicitud ha sido aprobada.</p>
Flujos alternativos	
Rechazo del modelo	
	<p>1. El Vice-decano administrativo rechaza el modelo del viático por las siguientes razones:</p> <ul style="list-style-type: none"> • Campos mal redactados. • La facultad no cuenta con el presupuesto necesario para cubrir el evento. <p>2. En el caso que los campos estén mal redactados envía el modelo del viático a la secretaria administrativa para su modificación (cualquier otro caso se rechaza el</p>

	<p>viático y termina el flujo).</p> <p>3. Una vez modificado el modelo del viático se lo reenvía al Vice-decano administrativo para su aprobación.</p> <p>4. Continúa el flujo normal a partir del paso 5.</p>
Mejoras propuestas	No procede.

2.6 Diagrama de actividades

Es un diagrama UML que muestra el flujo de control entre actividades, es decir es aquel que muestra las operaciones que se pasan entre los objetos en su interacción.

Este diagrama es usado para mostrar la secuencia de actividades, ya que muestran el flujo de trabajo desde el punto de inicio hasta el punto final detallando muchas de las rutas de decisiones que existen en el progreso de eventos contenidos en la actividad. La versatilidad de estos diagramas permite detallar situaciones donde el proceso paralelo puede ocurrir en la ejecución de algunas las actividades. (Dra. Hernández González, Anaisa, 2012)

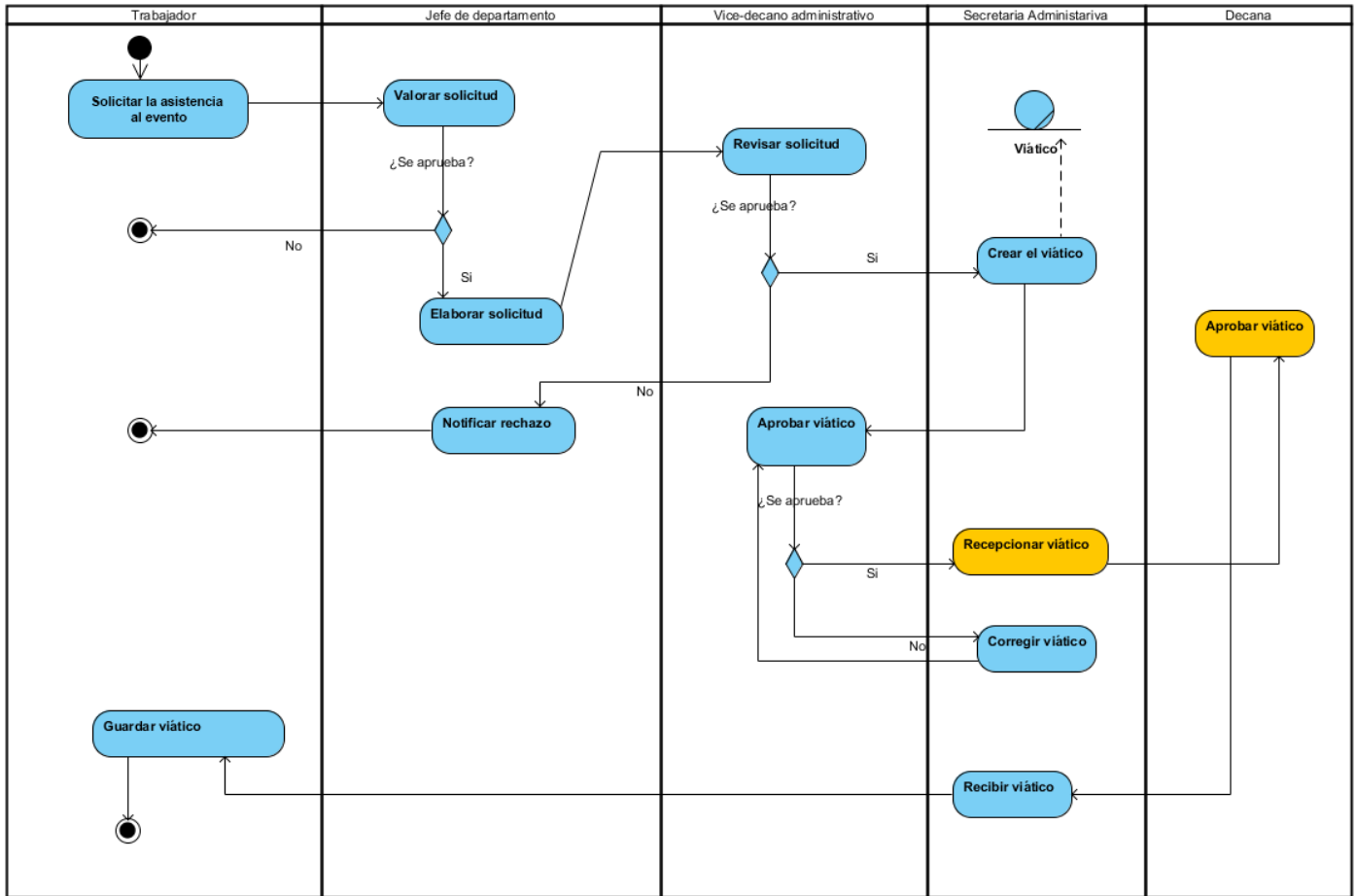


Fig. 5: Diagrama de actividades

2.7 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, sin tomar en consideración ningún tipo de restricción física. Por tanto, los requisitos funcionales especifican el comportamiento de entrada y salida del sistema. La calidad con que se realice la captura de los requisitos influye en todo el proceso de desarrollo del software, repercutiendo en el resto de las fases de desarrollo del mismo. De acuerdo con los objetivos planteados se definen los siguientes requisitos. (JACOBSON, 2004)

2.7.1 Requisitos funcionales

Para la descripción de los requisitos Ver: [SGV_1.0_Especificacion_de_requisitos_de_software.doc](#)

- RF1 Asignar Rol.
- RF2 Modificar Rol.
- RF3 Eliminar Rol.
- RF4 Listar Rol.
- RF5 Crear Solicitud.
- RF6 Modificar Solicitud.
- RF7 Listar Solicitud.
- RF8 Crear Viático.
- RF9 Modificar Viático.
- RF10 Eliminar Viático.
- RF11 Listar Viático.
- RF12 Filtrar Viático.
- RF13 Crear Desglose de Viático.
- RF14 Modificar Desglose de Viático.
- RF15 Eliminar Desglose de Viático.
- RF16 Listar Desglose de Viático.
- RF17 Declarar Presupuesto anual.
- RF18 Validar Viáticos.
- RF19 Crear reporte de viáticos.
- RF20 Autenticar usuario.

2.8 Requisitos no funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. (JACOBSON, 2004).

2.8.1 Requisitos no funcionales del sistema

Para lograr una buena calidad en el sistema y de esta manera satisfacer al cliente se listaron las siguientes propiedades o cualidades.

Usabilidad: El sistema podrá ser usado por personas que posean un nivel medio o alto en conocimientos de computación, aunque el trabajo de la aplicación es sencillo, permitiendo la fácil comprensión por el usuario. El software tendrá siempre visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento de sus funcionalidades por parte de los usuarios.

Confiabilidad: Deben establecerse los mecanismos necesarios para el restablecimiento del sistema ante fallos de comunicación u otros. Deben montarse sistemas de respaldo eléctrico en los locales de los servidores para mantener la vitalidad de los servicios.

Rendimiento: Teniendo en cuenta que el producto se debe diseñar sobre una arquitectura Modelo-Vista-Controlador, los tiempos de respuestas del sistema deben ser rápidos, al igual que la velocidad de procesamiento de la información para lograr respuestas rápidas del mismo.

Soporte: Se debe utilizar el sistema gestor de base de datos PostgreSQL.

Portabilidad: El sistema deberá funcionar en los sistemas operativos Windows (7 o superior) y las distribuciones de GNU/Linux, como son: Ubuntu, Debian, Xubuntu, y otros. El servidor Web y el servidor de Base de Datos pueden estar en la misma PC sin ocasionar problema alguno.

Seguridad: El usuario debe autenticarse antes de entrar al sistema.

- **Confiabilidad:** La información que se maneje en el sistema estará protegida de acceso no autorizado y divulgación, a partir de los diferentes roles de los usuarios que empleen el sistema.
- **Integridad:** La información tratada por el sistema será objeto de cuidadosa protección contra corrupción y estados inconsistentes, de igual manera el origen y autoridad de los datos.
- **Disponibilidad:** La información se encontrará disponible en todo momento para aquellos usuarios autorizados a acceder al sistema, ya que la aplicación deberá estar online las 24 horas.

Software:

Para las PC clientes:

- Navegador Mozilla Firefox 30.0 o superior.

Para los Servidores:

- Sistema operativo: Windows (7 o superior) o las distribuciones de GNU/Linux, como son: Ubuntu, Debian, Xubuntu, y otros.

- Gestor de bases de datos: PostgreSQL.

Hardware:

Para las PC clientes:

- RAM: 512 MB o superior.
- Tarjeta de Red: 512 MB o superior.
- Disco duro: 150 GB o superior.

Para los servidores:

- Procesador: Doble Micro a 3.0 GHz o superior.
- RAM: 2 GB o superior.
- Disco duro: 100 GB o superior.
- Tarjeta de Red: 1 GB o superior.

2.9 Modelo de casos de uso del sistema

El modelo de caso de uso del sistema (MCUS) describe las funcionalidades propuestas del nuevo sistema, sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios. El modelo de caso de uso (CU) está compuesto por actores, casos de uso y la relación que existe entre ellos. (Schmuller, 2010).

Casos de uso

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones, posibilidades (requisitos) que debe cumplir el sistema.

Actores del sistema

Cada trabajador que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor va a interactuar con el sistema, entonces también será un actor del sistema.

Tabla 7: Descripción de los actores del sistema

Actor	Objetivos
Trabajador	Es el encargado de listar los viáticos que le han sido aprobados.
Jefe de departamento	Es el encargado de crear la solicitud.

Vice-decano administrativo	Es el encargado de gestionar los roles, insertar el presupuesto y de generar reportes de viáticos.
Secretaria administrativa	Es el encargado de gestionar los viáticos.

2.9.1 Diagrama de caso de uso del sistema

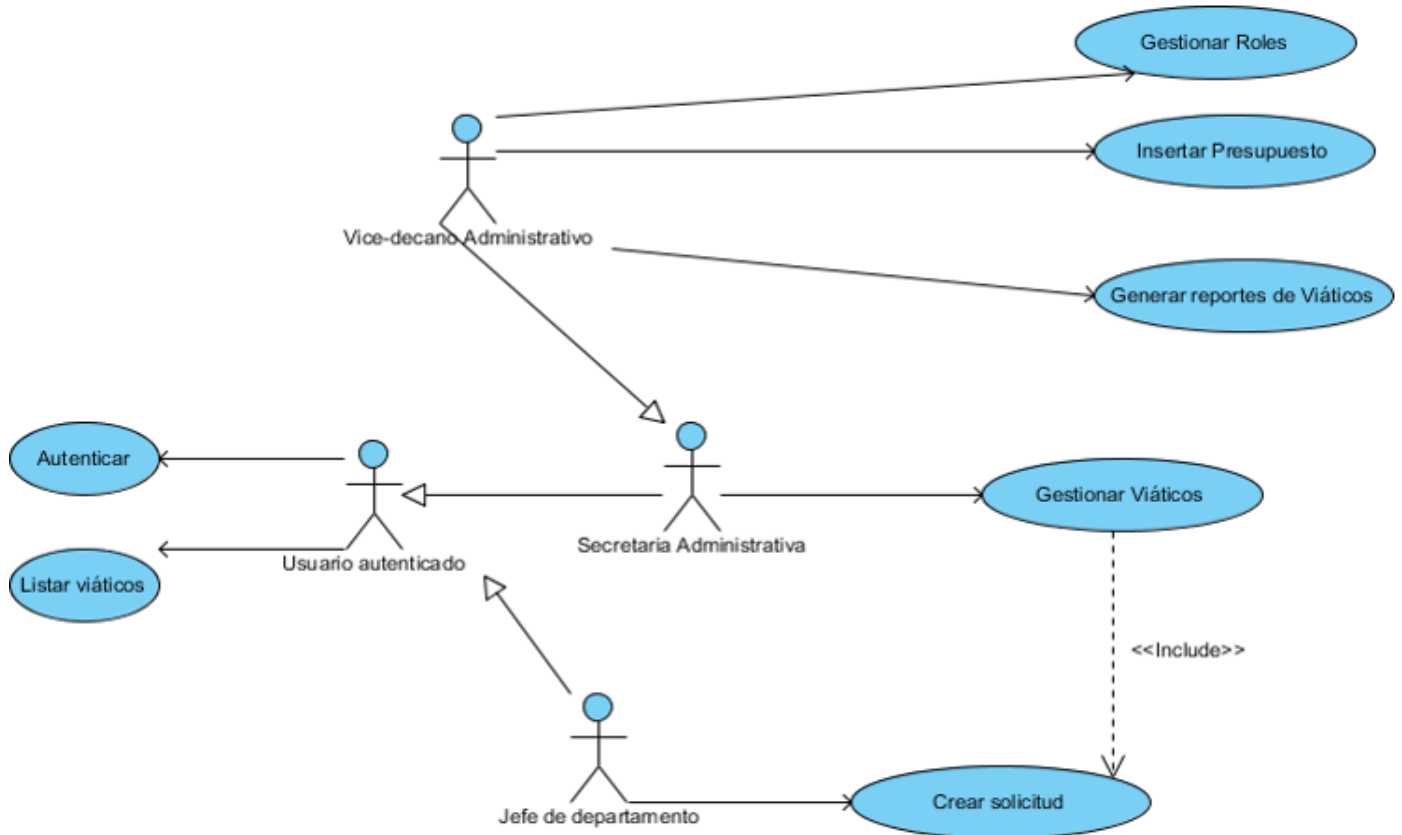


Fig. 6: Diagrama de caso de uso del sistema.

2.10 Patrones de caso de uso

En la arquitectura de software, se utiliza la palabra arquitectura, en contraste con el diseño, para evocar nociones de codificación, de abstracción, de las normas, de formación (de los arquitectos de software), y de estilo. La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema y establece los fundamentos para que analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. (JACOBSON, 2004).

Para la elaboración del diagrama de caso de uso del sistema se utilizaron los siguientes patrones de casos de uso:

CRUD Completo

CRUD (del inglés, Create, Read, Update, Delete¹¹), consiste en un caso de uso que permite modelar las diferentes operaciones para gestionar una entidad de información, tales como crear, leer, modificar y eliminar. Este patrón es usado cuando todas las operaciones contribuyen al mismo valor de negocio y todas son cortas y simples. Ejemplo de esto se evidencia en el caso de uso Gestionar Rol representado en la Fig. 6.

CRUD Parcial

Este patrón es una versión del CRUD Completo con la diferencia de que puede no incluir una o varias operaciones presentes en el CRUD Completo. Ejemplo de esto se evidencia en el caso de uso Gestionar Viático representado en la Fig. 6.

Tabla 8: Descripción del caso de uso Gestionar Viático

Objetivo	Gestionar viáticos.
Actor	Secretaria administrativa
Complejidad	Alta
Prioridad	Crítico
Precondiciones	La secretaria administrativa debe estar previamente autenticada en el sistema.
Pos condiciones	En dependencia de la acción de la secretaria: Se adiciona un nuevo viático. Se modifica un viático existente. Se elimina un viático existente. Se muestra un listado de los viáticos existentes.
Flujo de eventos	

¹¹ CRUD – Crear, Leer, Actualizar, Modificar.

	Actor	Sistema
1.	Selecciona la opción “Gestionar viático” .	
2.		Se muestran las opciones donde la secretaria administrativa puede: a. Crear viático. b. Listar viático: Los viáticos se listan automáticamente al iniciar la opción. Dentro de la opción Listar viáticos se muestran las opciones: b.1 Modificar viáticos. b. 2 Eliminar viáticos.
Sección a: <Crear viático>		
Flujo básico < Crear viático >		
		Sistema
1.	Selecciona la opción “Crear viático”	
2.		Muestra la interfaz para crear un nuevo viático, solicitando los siguientes datos: Usuario Solapín Motivo del viaje Tipo de viático (anticipo y liquidación) Importe Salida estimada Fecha de salida Fecha de regreso

		Fecha de liquidación
3.	Introduce los datos correspondientes y selecciona la opción “Crear” .	
4.		Valida los datos introducidos.
5.		Crea el nuevo viático, notificándole al Vicedecano administrativo mediante un correo, y dirige hacia la sección “Listar viáticos” mostrando un mensaje “Su viático ha sido creado”.
Prototipo		

Fig. 7 - Prototipo de pantalla: Crear viático

Sección b: <Listar viáticos>

Flujo Básico <Listar viáticos>

	Actor	Sistema
1.	Selecciona la opción “ listar viáticos ”.	
2.		Muestra la interfaz con todos los viáticos creados dando la opción de “ modificar viáticos ” y “ eliminar viáticos ”.

Prototipo

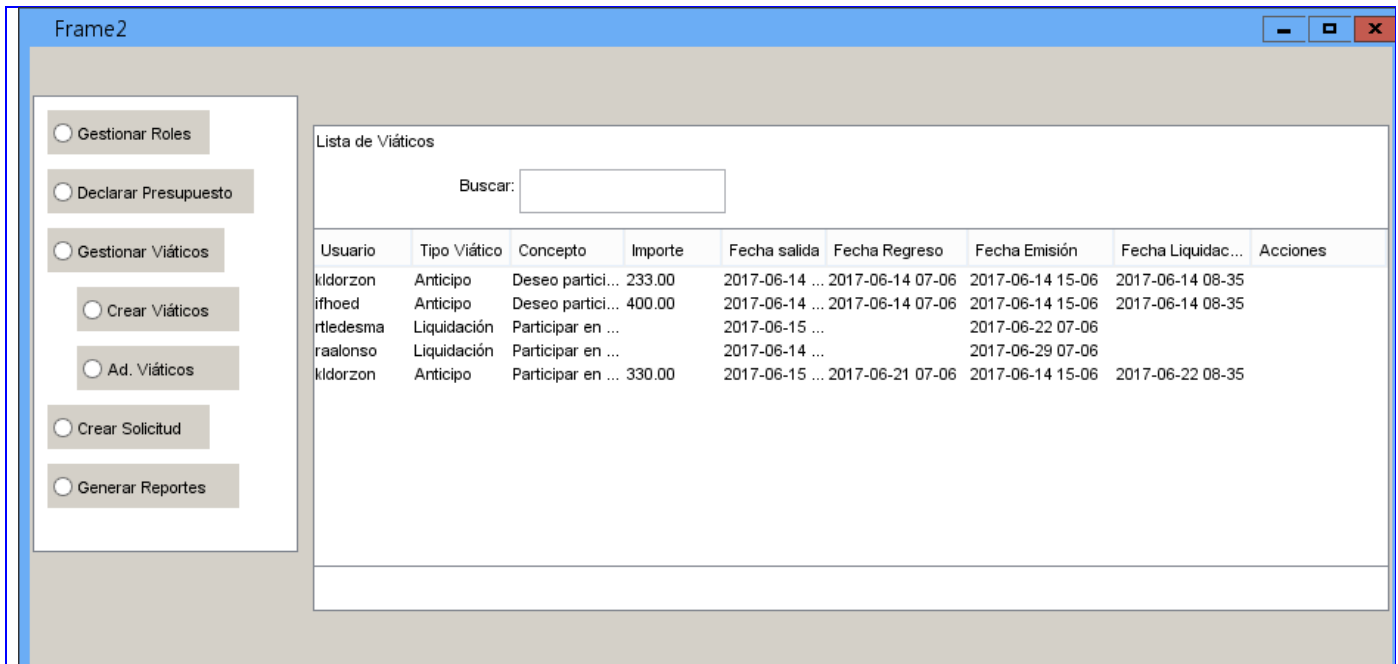


Fig. 8 - Prototipo de pantalla: Listar viáticos

Sección b.1

Flujo básico “Modificar viáticos”

	Actor	Sistema
1.	La secretaria selecciona en el viático la opción “ editar viático ”.	
2.		Se muestra la interfaz del viático que necesita modificar.
3.	Modifica los campos pertinentes y selecciona el botón “ Editar ”.	
4.		El sistema re direcciona hacia la interfaz donde se encuentran todos los viáticos.

Prototipo

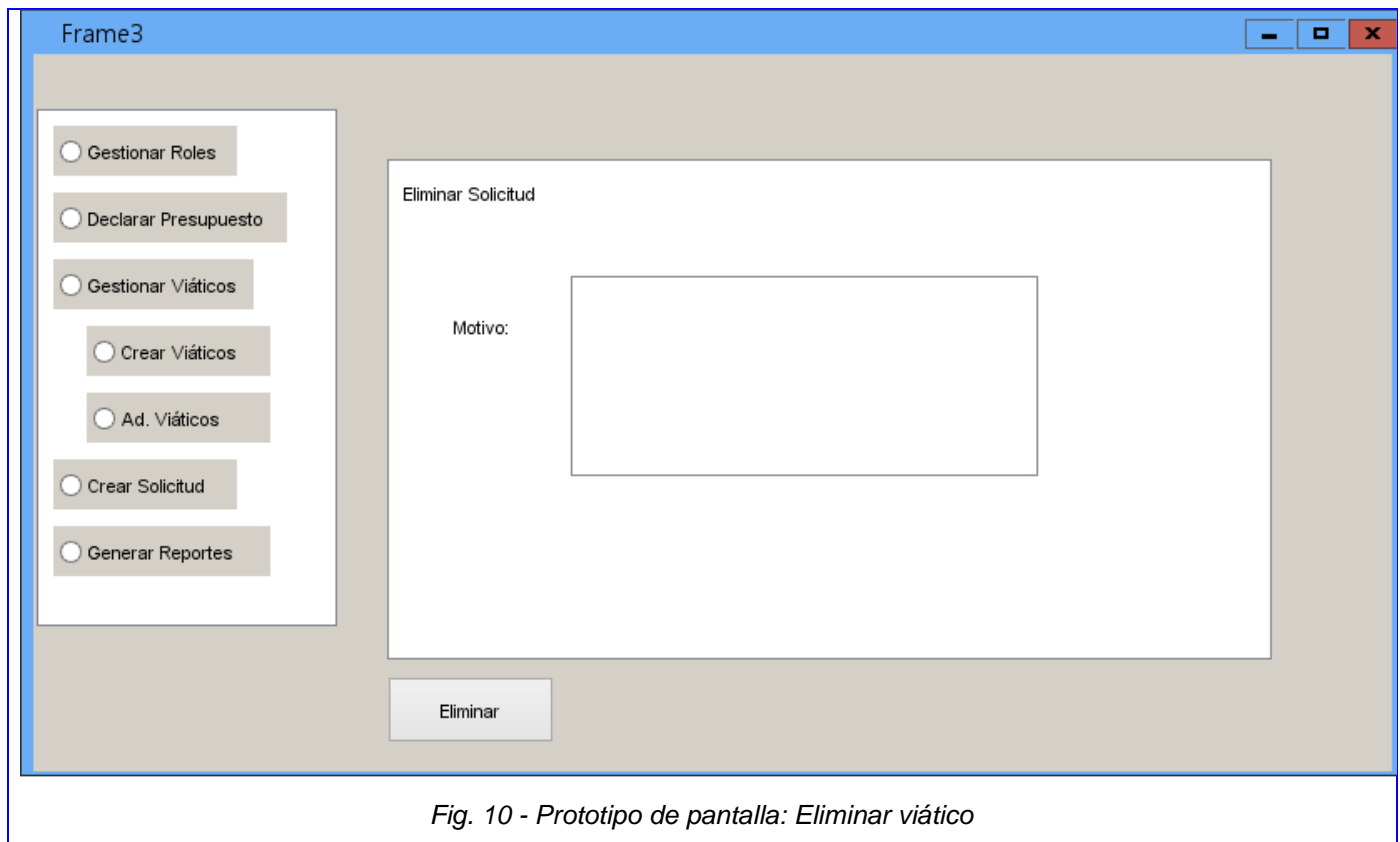
Fig. 9 - Prototipo de pantalla: Modificar viático

Sección b.2

Flujo básico “Eliminar viático”

1.	Selecciona en el viático la opción “eliminar viático”.	
2.		Se muestra una página donde se debe especificar el motivo por el cual se elimina este viático
3.	Elige la opción “Eliminar”.	
4.		Se elimina el viático y se envía una notificación mediante el correo al usuario relacionado con este.

Prototipo



2.11 Modelo de diseño

El modelo de diseño es un modelo de objeto que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen un impacto en el módulo, siendo la principal vía de acceso en la actividad de implementación. (Rumbaugh, 2011).

Patrón arquitectónico Modelo-Vista-Controlador

El patrón de diseño es una solución estándar para un problema común de programación, un proyecto o estructura de implementación que logra una finalidad determinada. También se considera como una manera más práctica de describir ciertos aspectos de la organización de un programa representados por conexiones entre componentes de programas. (Larman, 2010).

El Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El estilo de llamada y

retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista. Los beneficios de la utilización de dicho patrón van desde la imposición de decisiones tempranas en el desarrollo hasta la reutilización.

Teniendo en cuenta las características del sistema y la utilización de Symfony2, es la arquitectura más adecuada para la representación sistema de gestión ya que el marco de trabajo basa su funcionamiento interno en la arquitectura MVC Fig. 11:

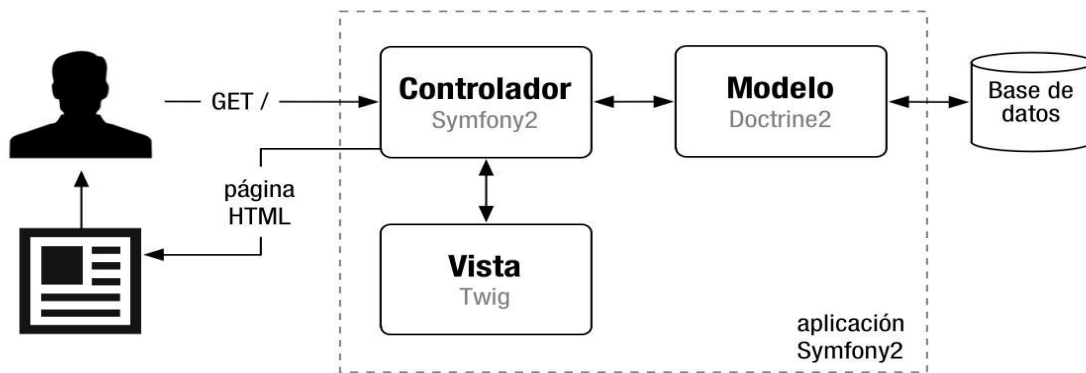


Fig. 11: Arquitectura de Symfony2

Symfony toma lo mejor de la arquitectura MVC y la realiza de modo que el desarrollo de aplicaciones sea rápido y sencillo.

Modelo: este contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia. Este es el responsable de acceder a la capa de almacenamiento de datos, define las reglas de negocio (la funcionalidad del sistema).

Vista: o interfaz de usuario son aquellas que componen la información que se envía al cliente y los mecanismos de interacción con éste. Son las responsables de recibir los datos del modelo y mostrarlos al usuario.

Controlador: actúa como intermediario entre el Modelo y la vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno. Es el responsable de recibir los eventos de entrada.

Diagrama de clases del diseño

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas y describen gráficamente las especificaciones de las clases de software y las interfaces en una aplicación. (JACOBSON, 2004)

Un diagrama de clases del diseño está compuesto por los siguientes elementos:

- ✓ Clases: atributos, métodos y visibilidad.
- ✓ Relaciones: herencia, composición, agregación, asociación y uso.

A continuación, en la Fig. 12, se muestra el diagrama de clases del diseño para el caso de uso Gestionar Viáticos.

2.11.1 Diagrama de clases del diseño del caso de uso Gestionar Viáticos

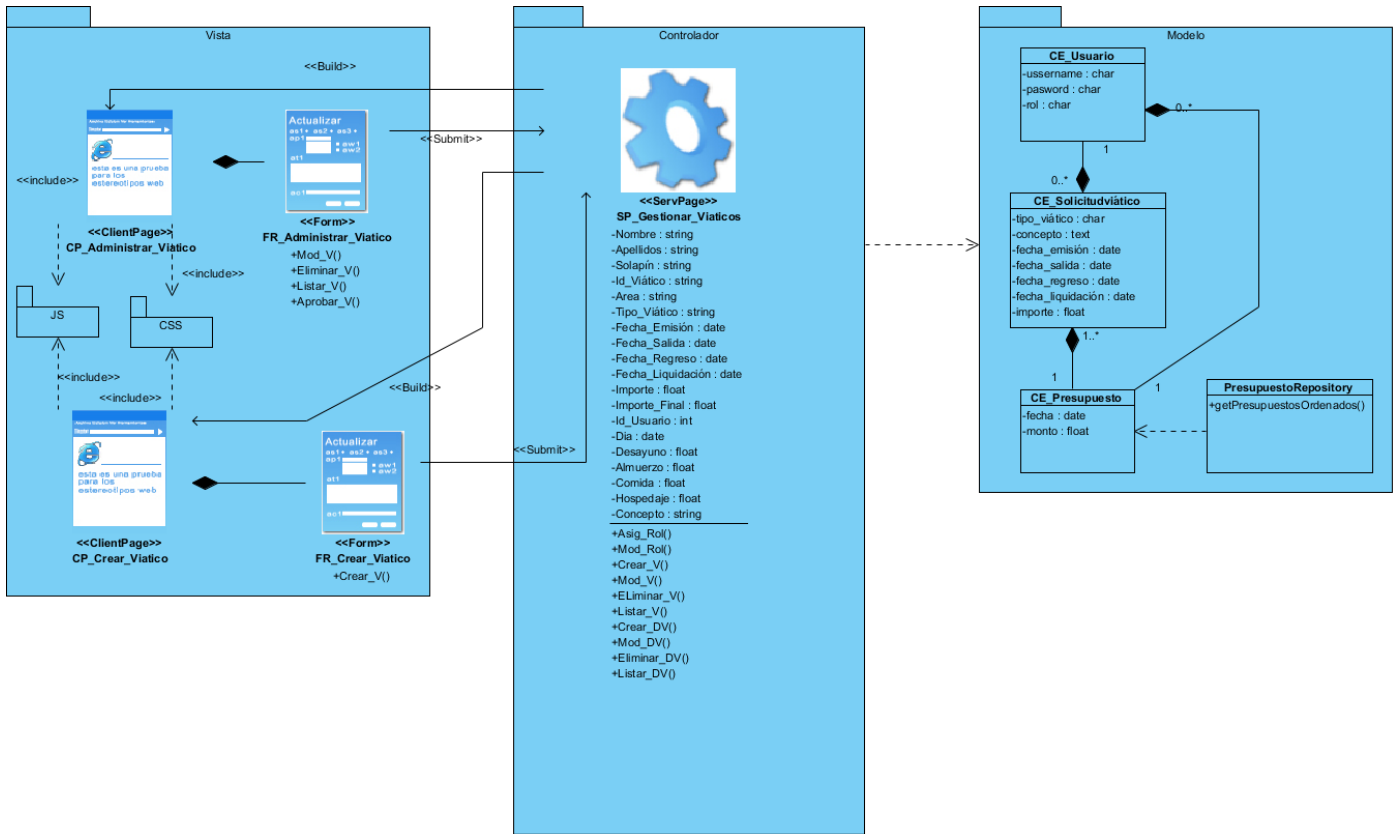


Fig. 12: Diagrama de clase de diseño del caso de uso Gestionar Viático

En la Fig. 12 se representan todas las clases con los métodos y atributos correspondientes al caso de uso Gestionar Viáticos. En el diseño se utilizó la arquitectura que trabaja con Symfony: Modelo-Vista-Controlador, donde en el controlador se alojan todas las clases que gestionen las respuestas del servidor, además de las acciones propias del mismo, estos están implementados en lenguaje PHP y dependen del subsistema Symfony. En la vista se encuentran todos los elementos observables por el cliente, encontrándose las páginas cliente CP_Administrar_Viático y CP_Crear_Viatico seguida de sus respectivos formularios FR_Administrar_Viático y FR_Crear_Viático, las cuales incluye los CSS y JS que contienen el Bootstrap y el jQuery. Los elementos del modelo corresponden a la clase generada por el ORM Doctrine que responde a las necesidades del sistema, consta de cuatro tablas; la entidad de PresupuestoRepository, la CE_Usuario, la CE_Solicitudviático y CE_Presupuesto, estas con sus respectivos atributos.

2.12 Patrones de diseño

Un patrón de diseño se caracteriza como una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución. Para el diseño de software, el contexto permite al lector entender el ambiente en el que reside el problema y qué solución sería apropiada en dicho ambiente. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas que influyen en la manera en la que puede interpretarse el problema en este contexto y en cómo podría aplicarse con eficiencia la solución. (Larman, 2010).

Los Patrones Generales de Software para Asignar Responsabilidades o General Responsibility Assignment Software Patterns (GRASP) son guías o principios que sirven para asignar responsabilidades a las clases. Teniendo en cuenta el patrón arquitectónico seleccionado, los patrones de diseño que se aplican son los siguientes:

Patrones de principios generales para asignar responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. A continuación, se detalla el uso de estos patrones en la solución:

Controlador: Todas las peticiones web son manipuladas por un solo controlador frontal, que es el punto de entrada único de toda la aplicación en un entorno determinado. Este patrón se evidencia en la clase controladora SP_Gestionar_Viáticos.php.

Alta Cohesión: Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme, es decir que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el resto de las clases. Un ejemplo se evidencia en las clases SP_solicitudController y CE_Solicitudviático, las cuales están formada por varias funcionalidades que están estrechamente relacionadas, siendo las mismas la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

Bajo acoplamiento: Determina el nivel de dependencia de una clase con respecto a otras. Su uso potencia la reutilización, el mantenimiento y la mitigación de efectos a producirse en una clase al hacer cambios en otra. En la presente solución se pone de ejemplo la existencia de una baja interdependencia entre la clase

controladora y el modelo o la vista. Estas características evidencian la posibilidad de efectuar cambios en estas sin que las otras sufran grandes afectaciones.

Patrones de la banda de los cuatro (GoF)

Los patrones GoF (del inglés, The Gand of Four), se utilizan para diseñar objetos y solucionar problemas de creación de instancias, ya que ayudan a encapsular y abstraer dicha creación. En la propuesta de solución se manifiestan los patrones:

Decorador: es un patrón de tipo estructural, el cual permite agregar dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa a la herencia para extender funcionalidades. Dicho patrón está presente a través de las plantillas twig para las vistas, las cuales heredan de la plantilla "layout.html.twig", que, a su vez, ésta hereda de "base.html.twig", lo cual permite declarar bloques editables dentro de los cuales se realizan modificaciones específicas manteniendo una apariencia homogénea.

Agente Remoto: Este patrón se utiliza cuando se requiere que el sistema se comunique con un servicio externo y no es posible acceder directamente a éste (Larman C. , 2005). La utilización de este patrón se evidencia en la autenticación mediante el servicio web disponible para la UCI. Permite la autenticación y obtención de los datos referentes a los usuarios de la universidad utilizando también clase mediadora defaultController.

2.13 Modelo de datos

Es la descripción de la organización de una base de datos, constituye una representación gráfica orientada a la obtención de la estructura de datos mediante métodos. En un enfoque más amplio, un Modelo de Datos permite describir los elementos que intervienen en una realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí. El Modelo de Datos está formado por dos componentes: componente estática, relacionada con el lenguaje de definición de datos (LDD) y dinámica, relacionada con el lenguaje de manipulación de datos (LMD). (Sánchez, 2004).

2.13.1 Diagrama Entidad-Relación

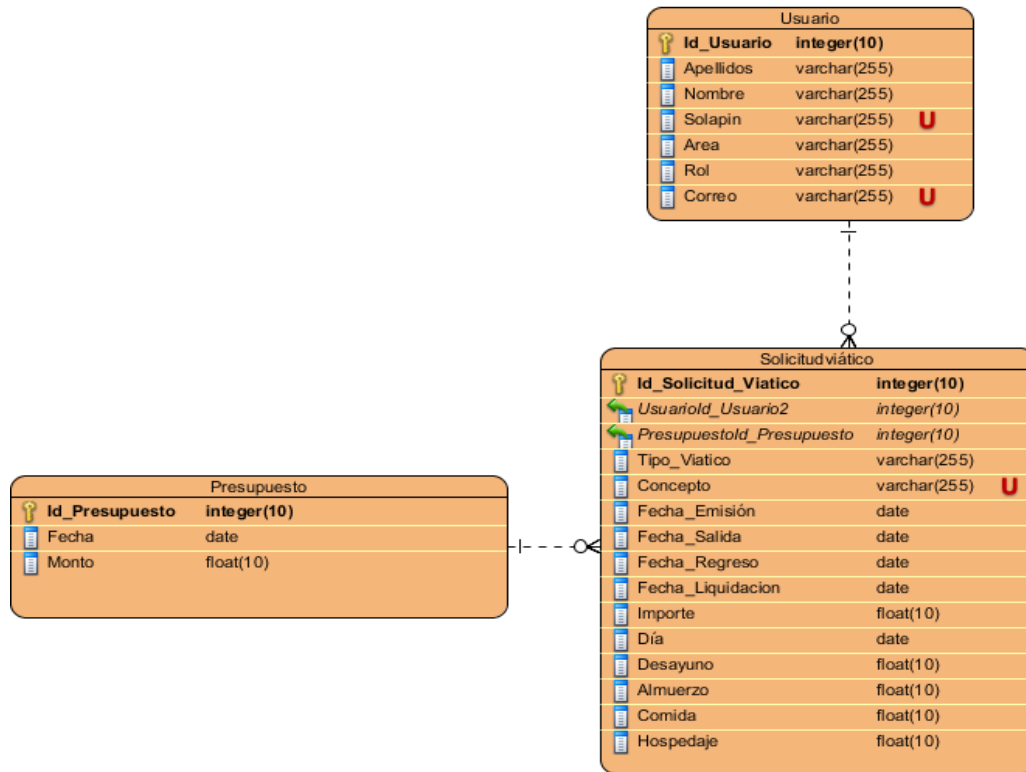


Fig. 13: Diagrama Entidad-Relación

En la Fig. 13 se muestra el Diagrama Entidad-Relación del sistema para la Gestión de Viáticos en el cual se evidencia la relación de las entidades Usuario, Solicitudviático y Presupuesto. Donde la entidad Usuario contiene los datos referentes a los usuarios que se autentican en el sistema, datos como: Nombre, Apellido, Solapín, Área, Rol y Correo, además del identificador de cada usuario. En la entidad Presupuesto se muestran todos los datos correspondientes al presupuesto asignado; como son: la fecha y el monto, además del identificador de dicho presupuesto. En la entidad Solicitudviático se muestran los datos referentes a los viáticos; como: Tipo viático, concepto, fecha emisión, fecha salida, fecha regreso, fecha liquidación, importe, día, desayuno, almuerzo, comida y hospedaje, además de tener los identificadores de las entidades Usuario y Presupuesto.

2.14 Diagrama de despliegue

Describe el ambiente dentro del cual el sistema será instalado. Establece una correspondencia entre la arquitectura de software del sistema, por lo que el rol encargado de realizarlo es el Arquitecto de Software. Muestran a los nodos procesadores de la distribución de los procesos y de los componentes. (Larman, 2010)

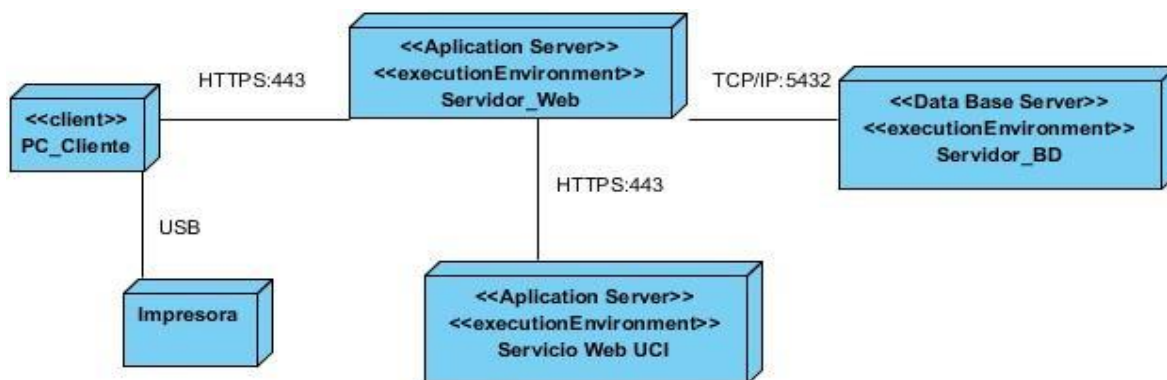


Fig. 14: Diagrama de despliegue

En la figura Fig. 14 se aprecia el Diagrama de despliegue referente a la solución, el cual representa un modelo de objetos que muestra cómo se distribuirá físicamente el sistema, donde:

- **PC-Cliente:** se refiere a las estaciones de trabajo que el usuario utilizará para acceder a la aplicación web y transcribir sus datos.
- **Impresora:** es un dispositivo periférico que permite producir una gama permanente de textos o gráficos de documentos almacenados en un formato electrónico. Es la encargada de imprimir los elementos enviados por parte de los usuarios desde la PC-Cliente.
- **Servidor de Aplicaciones:** es el servidor utilizado para la publicación de la aplicación; y para lograr la conexión del sistema con la PC-Cliente se utiliza HTTPS (del inglés *Hypertext Transfer Protocol Securted*) como protocolo de comunicación. Es la herramienta principal para ejecutar la lógica de negocio en el lado del servidor. Por otra parte, se utiliza el servicio LDAP-UCI para la autenticación de los usuarios.
- **Servidor de Bases de Datos:** se refiere a un servidor que radica en cada nodo regional en el cual el cliente define que sean guardados los datos. En el servidor central estarán almacenados todos los datos recopilados por todos los nodos regionales. El servidor de bases de datos elegido fue *PostgreSQL*, el cual está disponible para Linux y Windows.

- **Conexión USB:** es un cable para establecer la comunicación eléctrica entre ordenadores y periféricos. Este fue utilizado para la comunicación entre la PC-Cliente y la Impresora.
- **Conexión HTTPS:** es el protocolo utilizado entre el navegador de los clientes y el servidor web por el puerto 443. Este elemento de la arquitectura representa un tipo de comunicación no orientado a la conexión entre cliente y servidor.
- **Conexión TCP/IP:** es la base del Internet que sirve para enlazar computadoras. El protocolo TCP/IP es utilizado para establecer para establecer la conexión entre el servidor de aplicación y el servidor de bases de datos por el puerto 5432.

Conclusiones Parciales

La investigación realizada mediante entrevistas, permitieron un mejor entendimiento del negocio, definiendo así las reglas del negocio por las cuales se rige el proceso de Gestión de Viáticos, además de los actores y trabajadores del negocio, se identificaron 20 requisitos funcionales agrupados en 7 casos de uso utilizando patrones de casos de uso donde estos fueron relacionados entre sí mediante el diagrama de caso de uso del sistema, para una mejor comprensión de la funcionalidad del sistema. Mediante el estudio de los patrones arquitectónicos se especificó la arquitectura a utilizar Modelo-Vista-Controlador, ya que se encuentra incluida en el marco de trabajo Symphony2. Posteriormente se realizó el diagrama de despliegue para mostrar la representación física del sistema una vez que se termine.

Capítulo 3: Implementación y pruebas del Sistema para la Gestión de Viáticos

En el presente capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, ejemplificado por los estándares de codificación empleados en el código. Se describen las pruebas realizadas, con el objetivo de comprobar las funcionalidades del software en los diferentes escenarios para, de esta forma verificar en todos los casos que los resultados de las pruebas sean los esperados, y, por tanto, el correcto funcionamiento del sistema.

3.1 Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (Hernandez, 2013)

3.1.1 Diagrama de Componentes

Un diagrama de componentes permite visualizar la estructura de alto nivel del sistema y el comportamiento del servicio que estos componentes proporcionan y usan a través de interfaces. Además, se utilizan para describir un diseño que está implementado en cualquier idioma o estilo. Solo es necesario identificar los elementos del diseño que interactúan con los otros elementos del mismo a través de un conjunto de entradas y salidas. (Larman C. , 2005)

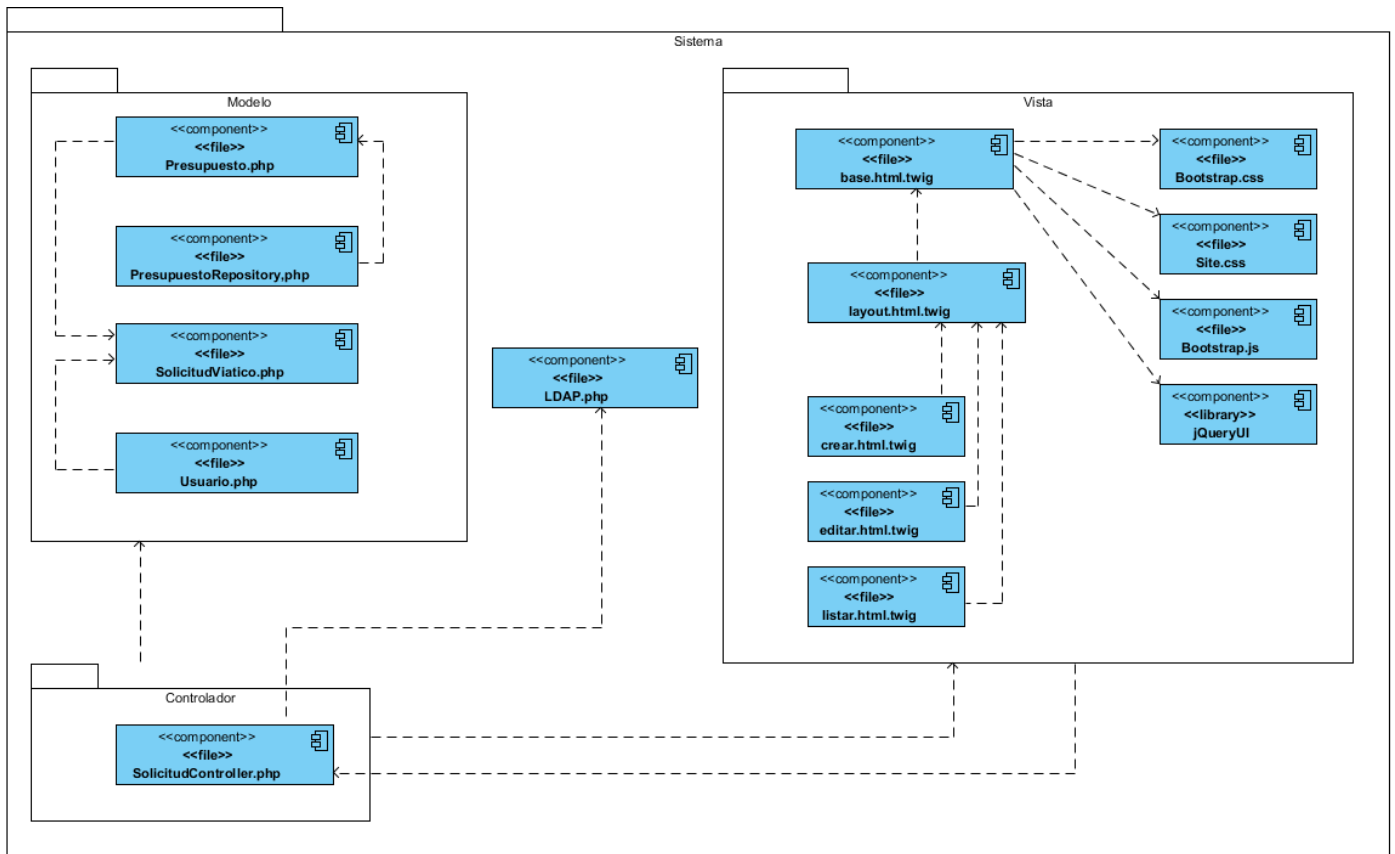


Fig. 15: Diagrama de componentes

El diagrama de componentes muestra las principales dependencias entre los principales elementos que conforman la estructura del sistema ubicados en el paquete "Sistema", por su parte el paquete Symfony2 muestra la estructura del marco de trabajo. El paquete "Vista" se ocupa de agrupar las vistas correspondientes a las acciones que ejecuta el usuario. El paquete "Modelo" se encarga de los componentes encargados de manejar los elementos referentes al negocio, y en el paquete "Controlador" se muestran los elementos que se encargan de la lógica del negocio. En el paquete "Recursos" se muestran los elementos encargados de proporcionar estilo al sistema.

3.2 Código Fuente

El código fuente es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar un programa informático. Por tanto, en el código fuente de un programa está escrito por completo su funcionamiento. Estas líneas de textos están escritas en un lenguaje de programación específico y que puede ser leído por un programador. Debe traducirse a lenguaje máquina para que pueda

ser ejecutado por la computadora, este proceso se denomina compilación. (Bonet, 2010).

3.2.1 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programas deben implementar un estándar de forma prudente, este debe tener siempre a lo práctico. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada, permitiendo que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea legible. De esta forma se pueden minimizar riesgos de incumplimiento de fechas de actividades importantes, de gastos excesivos en relación a los costos estimados e insatisfacciones por parte de los usuarios de los sistemas. (Bruegge, 2008).

Teniendo en cuenta que la propuesta de solución se desarrolla en Symfony2, se define el uso de los estándares de codificación PSR-1 (PHP Standards Recommendation 1) y PSR-2 (PHP Standards Recommendation 2). Estos estándares comprenden un grupo de elementos a tener en cuenta a la hora de codificar y que contiene como característica una uniformidad de estilos y reglas a cumplir.

- Se debe utilizar solamente las etiquetas `<? php` y `<? =`.
- Se debe emplear solamente la codificación UTF-8 para el código PHP.
- El código debe usar 4 espacios como indentación, no tabuladores.
- Debe haber una línea en blanco después de la declaración del *“namespace”* y otra después del bloque de declaraciones *“use”*.
- Las llaves de apertura en la estructura de control no deben estar en la misma línea, y las de cierre deben ir en la línea siguiente al cuerpo.
- Los paréntesis de apertura en las estructuras de control no deben tener espacio después de ellos, y los paréntesis de cierre no deben tener espacio antes de estos.

Se muestra un fragmento del código en la siguiente imagen.

```

public function crearAction() {
    $peticion = $this->getRequest();
    $solicitud = new SolicitudViatico();
    $formulario = $this->createForm(new SolicitudViaticoType(), $solicitud);
    $formulario->handleRequest($peticion);

    $fechaE = $this->get('request')->request->get('fecha_emision');
    $fechaS = $this->get('request')->request->get('fecha_salida');
    $horaS = $this->get("request")->request->get("hora_salida");
    $fechaR = $this->get('request')->request->get('fecha_regreso');
    $fechaL = $this->get('request')->request->get('fecha_liquidacion');
    if ($formulario->isValid()) {
        $em = $this->getDoctrine()->getManager();
        if (!$this->compararFecha($fechaS, $fechaR)) {
            $this->get('session')->getFlashBag()->add('alert', 'Error!! La fecha de salida tiene que ser menor a la fecha de entrada.');
```

Fig. 16: Código del método crearAction

3.3 Pruebas del software

Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Una estrategia de prueba del software debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código fuente se han implementado correctamente, así como pruebas de alto nivel que validen las principales funciones del sistema frente a los requisitos del cliente. De esta forma, el proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales para la detección de errores. Además de ser utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. (Pressman, 2005).

3.3.1. Niveles de pruebas

Las pruebas son aplicadas para diferentes tipos de objetivos, en determinados momentos del ciclo de vida

del software. Existen diferentes tipos de pruebas como: pruebas de desarrollador, independiente, integración, sistema y aceptación. En el desarrollo de la fase de pruebas del Sistema para la Gestión de Viáticos se aplicarán las pruebas a nivel de desarrollador y de sistema para probar que el sistema cumpla con los requisitos previamente definidos en la fase de análisis.

3.3.2 Tipos de pruebas

Existen diferentes tipos de pruebas que se puede aplicar para comprobar el correcto funcionamiento del módulo de seguridad, dentro de los que se seleccionaron los que a continuación se muestran:

- **Pruebas funcionales:** prueban una funcionalidad completa, donde pueden estar implicadas una o varias clases y la propia interfaz de usuario.
- **Pruebas unitarias:** Son pruebas que realiza el equipo de desarrollo que verifiquen los componentes unitarios codificados bajo condiciones de robustez, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. (Pressman, 2010).
- **Pruebas de aceptación:** Representa aquella fase de ciclo de vida de desarrollo del software en el equipo de trabajo y el área usuaria de un sistema de información tiene que garantizar que el sistema de desarrollo se corresponde con los requerimientos definidos (Ponce, 2014).
- **Pruebas de rendimiento:** mediante estas pruebas es posible hallar tendencias y comportamientos para los elementos de la aplicación, los cuales generen bajo rendimiento, permiten identificar cuellos de botella, capacidad de concurrencia de usuarios, tiempos de respuesta de operaciones a nivel de sistema, establecer un marco de referencia para pruebas futuras, determinar el cumplimiento de los objetivos de rendimiento y requerimientos no funcionales, entre otros. (Quality, 2016).

3.4 Métodos de pruebas

Actualmente hacer un software sin fallos y que satisfaga totalmente las necesidades de los clientes es la principal preocupación de los equipos de desarrollo. Aquí surge el término de calidad de software, el cual está definido por *“la totalidad de características de un producto, proceso o servicio que cuenta con la habilidad de satisfacer necesidades explícitas o implícitas”* (eumet.net, 2010). Para garantizar la misma se realizan las distintas pruebas de software. A continuación, se exponen las pruebas realizadas al sistema para la Gestión de Viáticos.

Pruebas de Caja Blanca: También suelen ser llamadas estructurales o de cobertura lógica. En ellas se pretende investigar sobre la estructura interna del código, exceptuando los detalles referidos a datos de entrada o salida, para probar la lógica del programa desde el punto de vista algorítmico. Realiza un seguimiento del código fuente según se va ejecutando los casos de pruebas, determinándose de manera concreta las instrucciones, bloques, etc. que han sido ejecutados por los casos de prueba. (Solazar Martínez, 2015).

En las pruebas de Caja Blanca se desarrollan casos de prueba que produzcan la ejecución de cada posible ruta del programa o módulo, considerándose una ruta como una combinación específica de condiciones manejadas por un programa.

Mediante las pruebas de Caja Blanca el ingeniero de software puede obtener Casos de Pruebas que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdaderas y falsas.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Como parte de las pruebas unitarias se utilizó el método de camino básico, permitiendo este: obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico, es decir, derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Los pasos a seguir para aplicar esta técnica son:

1. Representar el programa en un grafo de flujo.
2. Calcular la complejidad ciclomática.
3. Determinar el conjunto básico de caminos independientes.
4. Derivar los casos de pruebas que fuerzan la ejecución de cada camino.

La complejidad ciclomática se calcula de tres formas:

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- La complejidad ciclomática $V(G)$ de un grafo de flujo G se define como: $V(G)=A-N+2$ donde A es el número de aristas del grafo del flujo y N es el número de nodos del mismo.
- $V(G)$ también se calcula como el resultado de $P+1$ donde P es el número de nodos predicados (nodos de los cuales parten dos o más aristas) que tiene contenido el grafo de flujo G .

Ejemplo del código de la función crearAction.

```

public function crearAction() {
    $peticion = $this->getRequest();
    $solicitud = new SolicitudViatico();
    $formulario = $this->createForm(new SolicitudViaticoType(), $solicitud);
    $formulario->handleRequest($peticion);

    $fechaE = $this->get('request')->request->get('fecha_emision');
    $fechaS = $this->get('request')->request->get('fecha_salida');
    $horaS = $this->get('request')->request->get("hora_salida");
    $fechaR = $this->get('request')->request->get('fecha_regreso');
    $fechaL = $this->get('request')->request->get('fecha_liquidacion');
    if ($formulario->isValid()) {
        $em = $this->getDoctrine()->getManager();
        if (!$this->compararFecha($fechaS, $fechaR)) {
            $this->get('session')->getFlashBag()->add('alert', 'Error!! La fecha de salida tiene que ser menor a la fecha de entrada.');
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

Fig. 17: Ejemplo del código crearAction

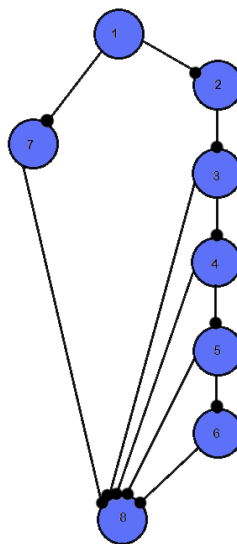


Fig. 18: Flujo del método crearAction

Resultados del cálculo de la complejidad ciclométrica

$$V(G) = A - N + 2 = 11 - 8 + 2 = 5$$

$$V(G) = P + 1 = 4 + 1 = 5$$

A partir del resultado obtenido se determina que la funcionalidad presenta una complejidad ciclométrica de 5, lo que deriva que existen a lo sumo 5 caminos lógicos por donde ejecutarse dicha funcionalidad.

Tabla 9: Resultado de la complejidad ciclométrica

No.	Camino básico
1	1-2-3-4-5-6-8
2	1-2-3-8
3	1-2-3-4-8
4	1-2-3-4-5-8
5	1-7-8

En la sección de Diseño de casos de pruebas se muestran los diseños de casos de pruebas correspondientes a los caminos básicos obtenidos en la tabla anterior.

Pruebas de Caja Negra: También pueden ser llamadas funcionales y basadas en especificaciones. En ellas se pretende examinar el programa en busca de que cuente con las funcionalidades que debe tener y como lleva a cabo las mismas, analizando siempre los resultados que devuelve y probando todas las entradas en sus valores e inválidos.

Al ejecutar las pruebas de Caja Negra se desarrollan casos de pruebas reales para cada condición o combinación de condiciones y se analizan los resultados que arroja el sistema para cada uno de los casos. En esta estrategia se verifica el programa considerándolo una caja negra. Las pruebas no se hacen en base al código, sino a la interfaz. No importa que se cuban todas las rutas dentro del programa, lo importante es probar todas las entradas en sus valores válidos e inválidos y lograr que el sistema tenga una interfaz amigable. (Solazar Martínez, 2015).

Las pruebas de caja negra buscan encontrar errores en cinco categorías:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a bases de datos externas.

4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Para este método se va a aplicar la técnica de Partición Equivalente, que según (Pressman, 2008) permite examinar los valores válidos e inválidos de las entradas existentes en el software. Además, esta técnica se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de pruebas a desarrollar. Para su implementación se divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software a partir de las cuales pueden derivarse casos de pruebas.

Para el desarrollo de las pruebas de aceptación se utilizaron las de tipo alfa, que son las que realiza el cliente, una vez recibido el producto final y su documentación, de conjunto con los desarrolladores del sistema. Este proceso se realizó en el Vicedecanato de Administración de la facultad CITEC en presencia de los desarrolladores y los especialistas del área quienes se encargaron de comprobar las funcionalidades y aprobar el producto. La realización de las pruebas de aceptación tuvo como salida la liberación de un acta de aceptación, la cual se anexa a este documento, con firma de los clientes de la solución.

Para el desarrollo de las pruebas de rendimiento se pueden aplicar pruebas de carga y de tensión. En la presente solución solamente se realizan pruebas de carga debido a que el sistema no estará expuesto a grandes niveles de concurrencia significativa de usuarios. Mediante las pruebas de carga se es posible identificar la capacidad de recuperación de un sistema cuando es sometido a cargas variables, tanto de usuarios como de procesos. (Quality, 2016).

Para la realización de esta prueba el sistema fue instalado en un entorno de prueba con las siguientes características en el servidor:

- Memoria RAM: 2GB
- Disco Duro: 100GB
- Sistema Operativo: Windows 7

3.5 Diseño de casos de prueba

Los casos de prueba son un conjunto de condiciones o variables bajo las cuales el analista determinará si los requisitos de la aplicación son parcial o temporalmente satisfactorios. Ayudan a validar que el sistema desarrollado realice las funciones para las que ha sido creado en base a los requerimientos del cliente.

A continuación, se muestran los Diseño de Casos de Prueba (DCP) correspondiente a los caminos básicos generados a través de las pruebas de caja blanca.

Tabla 10: DCP – Camino básico 1

Caso de prueba	
Camino 1-2-3-4-5-6-8	
Descripción:	Adicionar un Viático
Condiciones de ejecución	Se debe estar autenticado
Entrada	Se introducen todos los datos referentes a los viáticos
Resultados esperados	Se adiciona un nuevo viático

Tabla 11: DCP – Camino básico 2

Caso de prueba	
Camino 1-2-3-8	
Descripción:	Adicionar un Viático
Condiciones de ejecución	Se debe estar autenticado
Entrada	Se introduce la fecha de salida mayor a la fecha de regreso
Resultados esperados	Se muestra en mensaje indicando “La fecha de salida tiene que ser menor a la fecha de regreso”

Tabla 12: DCP – Camino básico 3

Caso de prueba	
Camino 1-2-3-4-8	
Descripción:	Adicionar un Viático
Condiciones de ejecución	Se debe estar autenticado
Entrada	Se introduce la fecha de emisión mayor a la fecha de liquidación
Resultados esperados	Se muestra en mensaje indicando “la fecha de emisión tiene que ser menor a la fecha de liquidación“

Tabla 13: DCP – Camino básico 4

Caso de prueba	
Camino 1-2-3-4-5-8	
Descripción:	Adicionar un Viático
Condiciones de ejecución	Se debe estar autenticado
Entrada	Se introduce la fecha salida menor que la fecha liquidación
Resultados esperados	Se muestra en mensaje indicando “la fecha de salida tiene que ser mayor a la fecha de liquidación”.

Tabla 14: DCP – Camino básico 5

Caso de prueba	
Camino 1-7-8	
Descripción:	Adicionar un Viático

Condiciones de ejecución	Se debe estar autenticado
Entrada	Los campos están vacíos
Resultados esperados	Se muestra en mensaje de indicando “todos los campos deben estar llenos”

A continuación, se muestra, como parte de las pruebas de caja negra, los diseños de casos de prueba correspondiente al caso de uso Gestionar Viáticos.

DCP: CU Gestionar Viáticos

Descripción General

El CU se inicia cuando la secretaria administrativa necesita realizar alguna de las siguientes acciones: Crear un nuevo viático, Modificar un viático, Eliminar un viático, Listar un viático existente. Finaliza cuando la secretaria administrativa realiza alguna de las acciones antes mencionadas.

Condiciones de ejecución

La secretaria administrativa debe estar previamente autenticada en el sistema, para posteriormente realizar las acciones anteriormente mencionadas.

Secciones a probar en el CU Gestionar Viáticos

Tabla 15: Secciones a probar en el caso de uso Gestionar Viáticos.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
	EC 1.1: La secretaria administrativa agrega todos los datos referentes a los viáticos.	El sistema guarda todos los datos introducidos por el usuario.	<ol style="list-style-type: none"> 1. Clic en el botón “Crear viático”. 2. Se agregan los datos indicados. 3. Clic en el botón “Crear”.

SC 1: “Crear viático ”	EC 1.2 La secretaria administrativa introduce la fecha de salida mayor a la fecha de regreso.	El sistema muestra un mensaje de error indicando que la fecha de salida debe ser menor a la fecha de regreso.	1. Clic en el botón “Crear viático”. 2. Se agregan los datos indicados. 3. Clic en el botón “Crear”.
	EC 1.3 La secretaria administrativa introduce la fecha de emisión mayor a la fecha de liquidación.	El sistema muestra un mensaje de error indicando que la fecha de emisión tiene que ser menor a la fecha de liquidación.	1. Clic en el botón “Crear viático”. 2. Se agregan los datos indicados. 3. Clic en el botón “Crear”.
	EC 1.4 La secretaria administrativa introduce la fecha de salida menor a la fecha de liquidación.	El sistema muestra un mensaje de error indicando que la fecha de salida tiene que ser mayor a la fecha de liquidación.	1. Clic en el botón “Crear viático”. 2. Se agregan los datos indicados. 3. Clic en el botón “Crear”.
	EC 1.5 La secretaria administrativa deja todos los campos vacíos.	El sistema muestra un mensaje de error indicando que todos los campos deben estar llenos.	1. Clic en el botón “Crear viático”. 2. Se agregan los datos indicados. 3. Clic en el botón “Crear”.
	EC 2.1 La secretaria administrativa selecciona modificar un viático.	El sistema muestra todos los viáticos listados.	1. Clic en el botón “Listar viático”. 2. Clic en la opción “Editar viático”. 3. Se modifican los campos pertinentes. 4. Clic en el botón “Aceptar”.
	EC 2.2 La secretaria administrativa deja la	El sistema muestra un mensaje de error	1. Clic en el botón “Listar viático”. 2. Clic en la opción “Editar viático”.

SC 2: “Modificar viático”	fecha de salida mayor a la fecha de regreso.	indicando que la fecha de salida debe ser menor a la fecha de regreso.	3. Se modifican los campos pertinentes. 4. Clic en el botón “Aceptar”.
	EC 2.3 La secretaria administrativa deja la fecha de emisión mayor a la fecha de liquidación.	El sistema muestra un mensaje de error indicando que la fecha de emisión tiene que ser menor a la fecha de liquidación.	1. Clic en el botón “Listar viático”. 2. Clic en la opción “Editar viático”. 3. Se modifican los campos pertinentes. 4. Clic en el botón “Aceptar”.
	EC 2.4 La secretaria administrativa deja la fecha de salida menor que la fecha de liquidación.	El sistema muestra un mensaje de error indicando que la fecha de salida tiene que ser mayor a la fecha de liquidación.	1. Clic en el botón “Listar viático”. 2. Clic en la opción “Editar viático”. 3. Se modifican los campos pertinentes. 4. Clic en el botón “Aceptar”.
	EC 2.5 La secretaria administrativa deja todos los campos vacíos.	El sistema muestra un mensaje de error indicando que todos los campos deben estar llenos.	1. Clic en el botón “Listar viático”. 2. Clic en la opción “Editar viático”. 3. Se modifican los campos pertinentes. 4. Clic en el botón “Aceptar”.
SC 3: “Eliminar viático”	EC 3.1 La secretaria administrativa selecciona el viático que desea eliminar.	El sistema elimina el viático seleccionado.	1. Clic en el botón “Listar viático”. 2. Clic en la opción “Eliminar viático”. 3. Mensaje que indica si está seguro eliminar ese viático. 4. Clic en el botón “Aceptar”. 5. Mensaje indicando que el viático ha sido eliminado satisfactoriamente.

SC 4: "Listar viáticos"	EC 4.1 La secretaria administrativa lista todos los viáticos	El sistema lista todos los viáticos.	1. Clic en el botón "Listar viáticos".
-----------------------------------	---------------------------------------------------------------------	--------------------------------------	----------------------------------------

Descripción de las variables

Tabla 16: Descripción de variables del caso de prueba

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Usuario	Campo de texto	No	Debe ser una combinación de caracteres perteneciente a un usuario del sistema UCI LDAP.
2	Tipo de viático	Campo de texto	No	Debe ser una combinación de caracteres que describa al viático.
3	Concepto	Campo de texto	No	Debe ser una combinación de caracteres que explique la razón del viaje.
4	Fecha de emisión	Campo de elección que trae asociado un rango de hora que puede ser modificado.	No	Debe seleccionarse la fecha deseada y escribir la hora.
5	Fecha de salida	Campo de elección que trae asociado un rango de hora que puede ser modificado.	No	Debe seleccionarse la fecha deseada y escribir la hora.
6	Fecha de regreso	Campo de elección que trae asociado un rango de hora que puede ser modificado.	No	Debe seleccionarse la fecha deseada y escribir la hora.

7	Fecha de liquidación	Campo de elección que trae asociado un rango de hora que puede ser modificado.	No	Debe seleccionarse la fecha deseada y escribir la hora.
8	Importe	Campo de texto	No	Debe ser una combinación de números reales positivos.

Matriz de datos: para el CU Gestionar Viáticos

Tabla 17: Matriz de datos del escenario Crear Viáticos

Escenario	Descripción	Variables									Respuesta del sistema	Resultado de la prueba	
		1	2	3	4	5	6	7	8	9			
EC 1.1	La secretaria administrativa agrega todos los datos referentes a los viáticos.	v	v	v	v	v	v	v	v	v	v	El sistema guarda todos los datos añadidos.	Satisfactorio
EC 1.2	La secretaria administrativa introduce la fecha de salida mayor a la fecha de regreso.	V	V	V	V	I	I	V	V	V	Muestra un mensaje de error indicando que la fecha que la de salida debe ser menor a la fecha de regreso.	Satisfactorio	
EC 1.3	La secretaria administrativa introduce la fecha de emisión mayor a la fecha de liquidación.	V	V	V	I	V	V	I	V	V	El sistema muestra un mensaje de error indicando que la fecha de emisión tiene que ser menor a la fecha de liquidación.	Satisfactorio	
EC 1.4	La secretaria administrativa introduce la fecha de	V	V	V	V	I	V	I	V	V	El sistema muestra un mensaje de error indicando que	Satisfactorio	

	salida menor que la fecha de liquidación.											la fecha de salida tiene que ser mayor a la fecha de liquidación.	
EC 1.5	La secretaria administrativa deja todos los campos vacíos.	I	I	I	I	I	I	I	I	I	I	El sistema muestra un mensaje de error indicando que todos los campos deben estar llenos.	Satisfactorio

Tabla 18: Matriz de datos del escenario Modificar Viático.

Escenario	Descripción	Variables									Respuesta del sistema	Resultado de la prueba
		1	2	3	4	5	6	7	8	9		
EC 2.1	La secretaria administrativa modifica todos los datos referentes a los viáticos.	v	v	v	v	v	v	v	v	v	El sistema guarda todos los datos añadidos.	Satisfactorio
EC 2.2	La secretaria administrativa deja la fecha de salida mayor a la fecha de regreso.	V	V	V	V	I	I	V	V	V	Muestra un mensaje de error indicando que la fecha que la de salida debe ser menor a la fecha de regreso.	Satisfactorio
EC 2.3	La secretaria administrativa deja la fecha de emisión mayor a la fecha de liquidación.	V	V	V	I	V	V	I	V	V	El sistema muestra un mensaje de error indicando que la fecha de emisión tiene que ser menor a la fecha de liquidación.	Satisfactorio
EC 2.4	La secretaria administrativa deja	V	V	V	V	I	V	I	V	V	El sistema muestra un mensaje de	Satisfactorio

	la fecha de salida menor que la fecha de liquidación.												error indicando que la fecha de salida tiene que ser mayor a la fecha de liquidación.	
EC 2.5	La secretaria administrativa deja todos los campos vacíos												El sistema muestra un mensaje de error indicando que todos los campos deben estar llenos.	Satisfactorio

Tabla 19: Matriz de datos del escenario Eliminar Viático

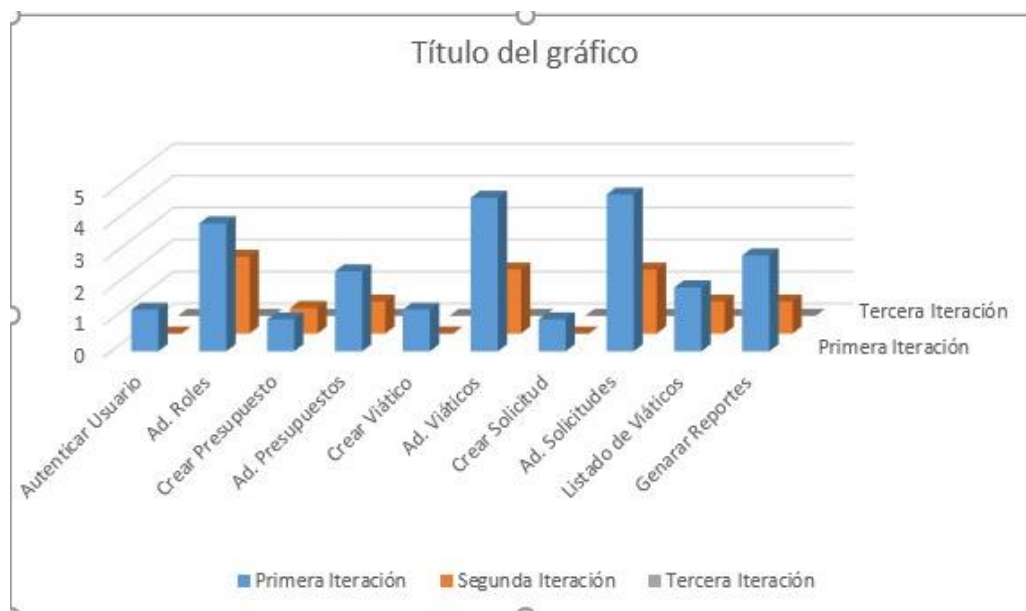
Escenario	Descripción	Variables									Respuesta del sistema	Resultado de la prueba
		1	2	3	4	5	6	7	8	9		
EC 3.1	La secretaria administrativa elimina el viático seleccionado.	v	v	v	v	v	v	v	v	v	El sistema muestra un mensaje indicando que el viático ha sido eliminado satisfactoriamente.	Satisfactorio

Tabla 20: Matriz de datos del escenario listar viático

Escenario	Descripción	Variables									Respuesta del sistema	Resultado de la prueba
		1	2	3	4	5	6	7	8	9		
EC 4.1	La secretaria administrativa lista todos los viáticos.	v	v	v	v	v	v	v	v	v	El sistema muestra la interfaz con todos los viáticos listados.	Satisfactorio

3.6 Resultado de las pruebas

Como resultado de las pruebas aplicadas teniendo en cuenta el objetivo principal de identificar en qué medida satisface la aplicación las funcionalidades implementadas, se realizó una primera iteración de pruebas, donde fueron aplicados los diseños de casos de pruebas realizados. Esta primera iteración mostró como resultado un total de 26 no conformidades, clasificadas en 11 no significativas y 5 significativas. Una vez corregidas se procedió a realizar una segunda iteración de pruebas donde se identificaron 10 nuevas no conformidades no significativas. Una vez corregidas se procedió a la realización de una tercera iteración de pruebas, dando resultados satisfactorios, por lo que se determinó no realizar más iteraciones dando por concluidas las pruebas.



Conclusiones parciales

En el transcurso de la implementación de la solución se quedó definido el modelo de implementación mediante el diagrama de componentes. El estudio del marco de trabajo Symfony realizado en el capítulo 2 ayudó a identificar los estándares de codificación y los estilos de programación que posteriormente fueron utilizados y descritos para un mejor entendimiento del código, además para un futuro mantenimiento de la aplicación. Mediante el desarrollo del proceso de pruebas se pudo identificar y resolver los errores en la implementación aumentando la calidad del producto obtenido. Se comprobó que el mismo satisface las funcionalidades requeridas y cumple con los requisitos funcionales y no funcionales planteados.

Conclusiones generales

Con la realización del trabajo de diploma se logró cumplir con los objetivos planteados, desarrollándose un Sistema para la Gestión de Viáticos para la Facultad de las Ciencias y Tecnologías Computacionales. Por lo que se concluye que:

- El estudio de los fundamentos teóricos de la investigación, permitió seleccionar una metodología para organizar de manera estructurada el proceso de desarrollo de software, así como las herramientas que cuentan con las características necesarias para desarrollar la solución propuesta.
- El Análisis y diseño del Sistema para la Gestión de Viáticos, posibilitó un mejor entendimiento del proceso de negocio, permitiendo la implantación de una solución que responda a las necesidades del cliente.
- La implementación de las funcionalidades viabilizó el cumplimiento de los requisitos funcionales definidos para la implementación, obteniendo como resultado un sistema funcional acorde a los requerimientos especificados.
- La validación de la solución mediante las pruebas realizadas y las correcciones de las deficiencias en cada una de las iteraciones, posibilitó la culminación exitosa de un producto de calidad.

Referencias Bibliográficas

- AMPUERO, M. A. (2015).** Análisis comparativo de modelos y estándares para evaluar la calidad del producto de software. *Revista Cubana de Ingeniería*. Recuperado el 16 de Enero de 2016, de El Modelo de Caso de Uso: http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html
- Apache.org. (2015).** *Apache*. Obtenido de <http://httpd.apache.org>
- BETSIME. (2011).** La revista dle empresario cubano. *BETSIME*. Obtenido de http://www.betsime.disaic.cu/secciones/eco_enemar_07.htm#2.
- Bonet, E. (2010).** Recuperado el 25 de 5 de 2016, de http://www.carlospes.com/minidiccionario/codigo_fuente.php.
- Bruegge, B. y. (2008).** *Ingeniería de Software orientado a objetos*.
- Carmen Garcia, A. M. (2014).** *Ingeniería de software QA at BQ*. Obtenido de <http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html>
- CEIGE. (2009).** *Definición del ciclo de vida de desarrollo de software*.
- CEIN. (2008).** *Centro Europeo de innovación en Navarra*. Obtenido de http://www.navactiva.com/es/asesoria/centros-de-costos-en-una-empresa_26173.
- Dra. Hernández González, Anaisa. (2012).** *Diagramas de Casos de Uso del Negocio y del Sistema*.
- Dsc. Becrril, Jorge. (2010).** Obtenido de www.businessrulesgroup.org/home-brg.shtml
- edicom. (2013).** Viaticos-Portal, edicom Integration Data Tool. *Portal electrónico para la administración eficiente de viáticos*.
- edivolt. (2013).** *edivolt*. Obtenido de Emisión y recepción de facturación electrónica, control de viáticos y gastos de viaje para ERP's: <http://www.edivolt.com>
- endalia. (2014).** *Endalia HR*. Obtenido de <http://www.endalia.com>
- eumet.net. (2010).** *Enciclopedia Virtual*. Obtenido de MEJORES PRÁCTICAS PARA EL ESTABLECIMIENTO Y ASEGURAMIENTO DE LA CALIDAD DE SOFTWARE.: <http://www.eumed.net/libros-gratis/2008a/351/Calidad%20de%20Software.htm>
- Feldt, K. (2007).** *Programando Firefox*. O'Reilly.
- GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J. (2003).** *Patrones de Diseño*. Madrid: PEARSON EDUCACION.
- Hernandez, L. (2013).** *Modelo de Implementación*. Obtenido de <http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html>

- infor. (2013).** *infor*. Obtenido de <http://www.infor.com>
- JACOBSON, I. y. (2004).** *El Proceso Unificado*.
- Larman, C. (2005). *UML y Patrones*. California: Prentice Hall
- Larman, C. (2010).** *UML y Patrones. 2a Edición*.
- Lutz, M. (2010).** *Learning Python, Fourth Edition*. O Reilly. Obtenido de <http://www.oreillynet.com/pub/au/446>
- Maldonado, D. (3 de septiembre de 2007).** *El CoDiGo K. Qué son los IDE de programación*. Recuperado el 24 de septiembre de 2012, de <https://elcodigok.blogspot.com/2007/09/que-son-los-ide-de-programacin.html>
- Manzur, S. (2015).** Obtenido de <http://www.mexired.com>
- Martínez, Rafael. (2010).** *Página oficial de postgresql*. Obtenido de <http://www.postgresql.org.es>
- Paradigm, V. (2013).** *Visual Paradigm for UML-UML tool for Software*. Obtenido de Visual Paradigm for UML: <https://www.visual-paradigm.com/features/>
- Pérez, Damian. (2007).** Recuperado el 3 de julio de 2010, de Maestros del Web: <http://maestrosdelweb.com/que-es-javascript/>
- Pérez, Julian y Gardey, Ana. (2012).** *Definición.de*. Obtenido de <http://definición.de/php/>
- Ponce, D. G. (2014).** Pruebas de aceptación orientadas al usuario. *Ibersid: Revista de sistemas de información y comunicación*.
- Pressman. (2005).** Ingeniería de software, Sixth Edition.
- Pressman. (2008).** Ingeniería de Software.
- Pressman. (2010).** Ingeniería de Software. Un enfoque práctico, 7ma edición.
- Pressman, R. S. (2010).** *Ingeniería del software*. New York: Higher Education.
- Quality. (2016).** *Quality permormance, security and process solutions*. Obtenido de <http://vyvquality.com/pruebas-rendimiento/>
- Rumbaugh, J. &. (2011).** *UML Reference Manual*.
- Sánchez Rodríguez, Tamara. (2014).** *Metodología de desarrollo para la actividad de la UCI*.
- Sánchez, J. (2004).** Recuperado el 18 de marzo de 20016
- Sánchez, J. (2004).** Recuperado el 18 de Marzo de 2016, de <https://books.google.com.cu/books?id=udFECQAAQBAJ&pg=PR7&lpg=PR7&dq=concepto+oficial+de+modelo+de+datos+relacional&source=bl&ots=lgQd1rhmbR&sig=iWymVd2oGlnQrCx9YRL7G>

mL_3tQ&hl=es&sa=X&ved=0ahUKEwj67dX4pdbMAhVIWCYKHWPiBcoQ6AEILzAE#v=onepage&q=concepto%20ofi

Schmuller. (2010). *Aprendiendo UML en 24 horas.* .

Solazar Martínez, E. (2015). Obtenido de <http://www.informatica-juridica.com/trabajos/procedimiento-realizar-pruebas-caja-blanca/>

Solis, J. (2014). Obtenido de <http://www.arweb.com>

Symfony.es. (2011). Obtenido de <http://symfony.es>

UCI. (2013). *Estrategia Marcaria de los productos de la UCI.* Obtenido de <http://iux.prod.uci.cu>


UCI. (2017). Programa de Mejora, Metodología de desarrollo para la Actividad productiva de la UCI.

ANEXO

Anexo1: Acta de Aceptación de la solución propuesta.

Acta de aceptación emitida y aprobada por el MSc. Omar Mar Cornelio y el Ing. Carlos Luis Hernández Hernández. Fecha 16-06-2017.

Vicedecanato de Administración Facultad CITEC



La Habana, 8 de junio del 2017
"Año 59 de la Revolución".

ACTA DE ACEPTACIÓN

De una parte, el Vicedecano de Administración de la facultad CITEC, representado en este acto por: Ing. Carlos Luis Hernández Hernández y por **otra parte** el tutor y Director de Servicios Generales, representado en este acto por: MSc. Omar Mar Cornelio y por último los estudiantes: Raimel Torres Ledesma y Karel Luis Dorzón Fonfecilla.

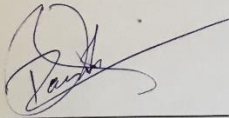
Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.


CONSIDERANDO: Que los hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.


CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requerimientos establecidos.


POR TANTO: **Las Partes** acuerdan formalizar mediante la presente Acta, la aceptación del producto: Sistemas para la Gestión de Viáticos de la Facultad de Ciencias y Tecnologías Computacionales (CITEC).

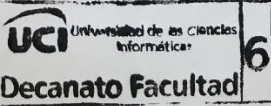
Y para que así conste, se extiende la presente Acta en 3 ejemplares, rubricados por **Las Partes**.









Entregan  Reciben **6**