

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 3



TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÁTICAS

Título: “Perfil de usuario para el Sistema de Planificación de Actividades
SIPAC”

Autor: Yuleidis Daudinot Hamiltón

Tutor(es):

Ing. Maria Teresa Rosales González

Ing. Yordi Chaveco Bustamante

Ing. Liliana Ramírez Zayas

Ciudad de La Habana, Junio 2015

“Año 57 de la Revolución”



*“La planificación a largo plazo
no es pensar en decisiones futuras,
sino en el futuro de las decisiones presentes”*

Peter Drucker

DECLARACIÓN DE AUTORÍA

DECLARACIÓN DE AUTORÍA

Declaro a Yuleidis Daudinot Hamiltón única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Yuleidis Daudinot Hamiltón

Ing. María Teresa Rosales González

Ing. Yordi Chaveco Bustamante

Ing. Liliana Ramírez Zayas

DATOS DE CONTACTO

Autor:

Yuleidis Daudinot Hamiltón

Correo electrónico: ydaudinot@estudiantes.uci.cu

Tutores:

Ing. Maria Teresa Rosales González

Correo electrónico: mtrosales@uci.cu

Ing. Yordi Chaveco Bustamante

Correo electrónico: ybustamante@uci.cu

Ing. Liliana Ramírez Zayas

Correo electrónico: iramirez@uci.cu

AGRADECIMIENTOS

Quiero agradecer

A mis amistades por compartir los buenos y malos momentos, por sus consejos y ayuda incondicional, en especial a Humberto, Martha, Juan David, Daniel, Sandy, Robin, Héctor Javier, Laritza, Rosa María, Yoandy y Geider.

A mis profesores por haberme ayudado en mi formación profesional, por sus consejos, por brindarme una mano amiga, en especial a Hilda, Mónica, Daílen y Hugo.

A mis tutores por toda su ayuda, tiempo y dedicación, en especial a una persona que me ha ayudado incondicionalmente, hoy me atrevo a llamarla amiga, hermana, compañera, a ti Tere gracias, siempre estaré en deuda contigo, solo voy a pedir por ti todo lo que quieras lograr en la vida y quiero que sepas que en mí tienes una persona que te quiere y te respeta.

A mi novio por su dedicación, su comprensión, su paciencia, su amor, por haberme apoyado y ayudado durante estos años, por haberme dado otra familia. Gracias mi amor.

A mi familia que es una de las cosas más grande que tengo en la vida, gracias por su apoyo, por sus palabras y sobre todo por haberme ayudado a llegar hasta aquí.

A mis hermanas y sobrina por ser mis mayores inspiraciones, por siempre esperar lo mejor de mí, por su confianza, por su amor.

A mis padres por darme la vida, por haberme educado, por siempre estar orgullosos de mí, por exigirme siempre lo que saben puedo dar, por sus consejos y sobre todo por mostrarme que la vida está llena de caminos y los caminos de pruebas, pero para llegar al fin del camino hay que saber escoger cuál podemos transitar y como ha de hacerlo.

DEDICATORIA

Quiero dedicar esta tesis a mí:

Abuela, mis hermanas, mi sobrina, mi papá, mi tutora Tere y sobre todo a lo más grande que tengo en esta vida, la mejor mujer que conozco y conoceré por el resto de mi vida, mi mamá.

Sí pudiera volver el tiempo atrás al menos un segundo, sólo te diría y quisiera una cosa, decirte Te Amo Mamí y verte sonreír.

RESUMEN

El presente trabajo tiene como objetivo desarrollar una solución informática para la personalización y configuración de las funcionalidades en el Sistema de Planificación de Actividades, de forma tal que se adapte a las preferencias de los usuarios y facilite el registro, seguimiento y control de la planificación a corto, mediano y largo plazo según establece la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros. Los principales resultados obtenidos referencian todos los artefactos de cada una de las disciplinas de la fase de ejecución de la metodología de desarrollo de la Universidad de las Ciencias Informáticas. La solución desarrollada posibilita la notificación a los usuarios de la ocurrencia de la variación en sus planes de trabajo, la definición de un sistema de trabajo que agiliza el proceso de elaboración del plan de los jefes y sus subordinados, así como la correspondencia de las preferencias de cada usuario en el sistema y sus funciones dentro de la organización.

Palabras clave: configuración, funcionalidades, personalización, preferencias, usuarios

ÍNDICE

INTRODUCCIÓN	2
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.....	5
1.1 Estudio de sistemas de planificación	5
1.1.1 SugarCRM	5
1.1.2 PlanningPME	5
1.1.3 SAP.....	6
1.2 Metodología de desarrollo.....	7
1.2.1 Fases	7
1.2.2 Disciplinas	8
1.3 Lenguajes, tecnologías y herramientas	8
1.3.1 Lenguajes de modelado	8
1.3.2 Lenguajes de desarrollo del lado del cliente	8
1.3.3 Lenguajes de desarrollo del lado del servidor.....	9
1.3.4 Tecnologías y herramientas de desarrollo.....	9
1.3.5 Técnicas de desarrollo	11
1.4 Conclusiones Parciales	12
CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL SISTEMA.....	14
2.1 Descripción de la solución.....	14
2.2 Modelo conceptual	14
2.3 Requisitos	17
2.3.1 Técnicas de captura de requisitos	17
2.3.2 Requisitos funcionales.....	17
2.3.3 Especificación de los requisitos funcionales	18
2.3.4 Validación de requisitos funcionales.....	20
2.3.5 Requisitos no funcionales.....	21
2.4 Modelado de la solución.....	22
2.4.1 Diseño de la solución en términos de componentes.....	22
2.4.2 Diagrama de clases del diseño.....	23
2.5 Patrones del diseño	25
2.5.1 Patrones GRASP	25
2.5.2 Patrones GoF (Gang of Four).....	26

ÍNDICE DE CONTENIDOS

2.6 Diagrama de secuencia.....	27
2.7 Modelo de datos.....	28
2.8 Conclusiones Parciales	29
CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA	30
3.1 Implementación de la solución	30
3.1.1 Estándares de código.....	30
3.1.2 Bibliotecas para envío de correo	31
3.1.3 Descripción de las clases y funcionalidades de la solución	33
3.2 Perfil de usuario para SIPAC.....	35
3.3 Impacto de la solución.....	37
3.4 Validación de la solución propuesta	38
3.4.1 Tamaño operacional de clases (TOC).....	38
3.4.2 Relación entre clases (RC).....	40
3.5 Pruebas de software	43
3.5.1 Descripción y aplicación de la Prueba de Caja Blanca o Estructural	44
3.5.2 Descripción y aplicación de la Prueba de Caja Negra o Funcional	50
3.6 Resultados de las pruebas aplicadas.....	51
3.7 Conclusiones Parciales	52
CONCLUSIONES GENERALES.....	53
RECOMENDACIONES	54
REFERENCIAS BIBLIOGRÁFICAS	55
ANEXOS.....	58

ÍNDICE DE TABLAS

Tabla 1. Descripción de requisito: Adicionar datos de los contactos de usuario.....	19
Tabla 2. Descripción de las principales clases del diagrama de clases del diseño.....	24
Tabla 3. Descripción de la clase GestionarPerfilUsuarioController.....	33
Tabla 4. Descripción de la clase GestionarPerfilUsuarioModel	34
Tabla 5. Descripción de la clase DatDatosUsuario.....	34
Tabla 6 Tamaño operacional de clase (TOC).....	38
Tabla 7. Relación entre clase (RC)	41
Tabla 8. Caso de prueba para el camino básico 1	48
Tabla 9. Caso de prueba para el camino básico 2	48
Tabla 10. Caso de prueba para el camino básico 3	49
Tabla 11. Caso de prueba de caja negra para validar el RF Adicionar datos de los contactos de usuario .	50
Tabla 12. Resultados de la aplicación de la métrica Tamaño Operacional de Clases (TOC)	59
Tabla 13. Resultados de la aplicación de la métrica Relación entre Clases (RC).....	60

ÍNDICE DE FIGURAS

Figura 1. Modelo conceptual del perfil de usuario para el SIPAC.....	16
Figura 2. Interfaz para adicionar los datos de contacto del usuario.....	20
Figura 3. Diagrama de la funcionalidad perfil de usuario para el SIPAC	23
Figura 4. Diagrama de clases de diseño del perfil de usuario	24
Figura 5. Diagrama de secuencia Adicionar datos de los contactos de usuario	27
Figura 6. Modelo de datos del perfil de usuario.....	29
Figura 7. Interfaz de usuario Información de contacto.....	35
Figura 8. Interfaz de usuario Personalización	36
Figura 9. Interfaz de usuario Patrones de actividades.....	37
Figura 10. Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad	39
Figura 11. Resultados de la evaluación de la métrica TOC para el atributo Complejidad.....	39
Figura 12. Resultados de la evaluación de la métrica TOC para el atributo Reutilización	40
Figura 13. Resultados de la evaluación de la métrica RC para el atributo Acoplamiento	41
Figura 14. Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento	42
Figura 15. Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.....	42
Figura 16. Resultados de la evaluación de la métrica RC para el atributo Reutilización.....	43
Figura 17. Funcionalidad EnviarCorreo ()	46
Figura 18. Grafo de flujo asociado al algoritmo EnviarCorreo ()	47
Figura 19. No conformidades detectadas durante las iteraciones de pruebas realizadas.....	52

INTRODUCCIÓN

Las organizaciones tradicionalmente han tenido la constante preocupación de incrementar su eficiencia organizacional. Para ello, la experiencia demuestra que el éxito y el desarrollo de las mismas, dependen en gran medida de una adecuada planificación. Esta consiste “en definir las metas de la organización, establecer una estrategia general para alcanzarlas y trazar planes exhaustivos para integrar y coordinar el trabajo de la organización. Se ocupa tanto de los fines como de los medios” [1].

Cuba no está exenta de los problemas que presenta el no realizar una adecuada planificación que garantice mantener un seguimiento y control eficaz sobre el cumplimiento de los objetivos trazados en los diferentes organismos e instituciones estatales. Para ello el Centro de Informatización de Entidades (CEIGE), de la Universidad de las Ciencias Informáticas (UCI) en conjunto con el Grupo de Planificación de la Secretaría del Consejo de Ministros, desarrolla el Sistema de Planificación de Actividades (SIPAC).

SIPAC está basado en la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros para la planificación de los objetivos y actividades en los Órganos, Organismos de la Administración Central de Estado, Entidades Nacionales y Administraciones locales del Poder Popular¹; está destinado a facilitar la gestión de las actividades a todos los niveles organizacionales, permite interrelacionar objetivos de trabajo y actividades en tiempo real, garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades. Éste informatiza los procesos de Ejecución y Control de la Planeación Estratégica (definición de los objetivos a largo plazo y estrategias para alcanzarlos) y Operativa (puntualización de las actividades que debe efectuar cada individuo a corto plazo) [2].

Independientemente de las funcionalidades con que cuenta SIPAC y las facilidades que brinda para llevar una adecuada planificación de las actividades, este:

- ✓ No le notifica a los usuarios si ha ocurrido alguna variación en sus planes de trabajo, salvo si acceden al sistema.
- ✓ No permite el establecimiento de patrones de actividades que en su conjunto conforman un sistema de trabajo, lo que ralentiza el proceso de elaboración de los planes.

¹ Instrucción No.1: Tiene como objetivo establecer el procedimiento para llevar a cabo el proceso de planificación del Gobierno, que permita dar cumplimiento a los acuerdos y resoluciones aprobadas en el VI Congreso del Partido Comunista de Cuba, las decisiones de la Asamblea Nacional del Poder Popular, el Consejo de Ministros y la actualización de los planes de la economía.

- ✓ Los usuarios no pueden adaptar el comportamiento del sistema de acuerdo a la visualización de los elementos y el envío a aprobación de ellos, lo que imposibilita que cada entorno de trabajo no esté en correspondencia a las preferencias de cada trabajador.

Teniendo en cuenta lo anteriormente planteado, se identificó el siguiente **problema a resolver**: ¿Cómo adaptar el comportamiento de SIPAC a las preferencias de los usuarios, de manera que facilite el registro, seguimiento y control de la planificación a corto, mediano y largo plazo?

Definiéndose como **objeto de estudio**: Perfiles de usuario en sistemas de planificación, enmarcado en el **campo de acción**: Perfil de usuario en el SIPAC.

Para dar respuesta al problema planteado se define como **objetivo general**: Desarrollar un perfil de usuario para el SIPAC, de manera que facilite el registro, seguimiento y control de la planificación a corto, mediano y largo plazo.

Desglosado en los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico para sustentar los conceptos, la propuesta de desarrollo de las funcionalidades y las herramientas y tecnologías a utilizar.
- ✓ Modelar el negocio asociado a la solución, para comprender los elementos significativos de la misma.
- ✓ Realizar el análisis y diseño de la solución a implementar teniendo en cuenta las necesidades del cliente.
- ✓ Implementar la solución que permita definir un perfil de usuario para el SIPAC y responda a las necesidades del cliente.
- ✓ Validar las funcionalidades desarrolladas mediante los métodos de pruebas: caja negra y caja blanca.

Una vez expuestos los elementos anteriores se define como **idea a defender**: Si se desarrolla una solución que permita adaptar el comportamiento de SIPAC a las preferencias de los usuarios, se facilitará el registro, seguimiento y control de la planificación a corto, mediano y largo plazo.

Para llevar a cabo esta investigación se utilizaron los siguientes **métodos científicos**:

Métodos teóricos

1. **Analítico-sintético:** Permitió analizar teorías y documentos relacionados con el tema de la investigación en cuestión y a partir de ello realizar un estudio previo acerca de las características y la necesidad de una solución, extrayendo de esta forma los datos necesarios.
2. **Histórico-lógico:** Se utilizó con el objetivo de realizar un estudio del estado del arte del tema a investigar. De esta manera se puede conocer acerca de la existencia y características de sistemas de este tipo que hayan sido desarrollados anteriormente.
3. **Modelación:** A través de este método se crearon modelos con vistas a representar los principales conceptos y elementos a tener en cuenta para resolver el problema. Así como el modelado de esquemas y diagramas que permiten obtener una mejor representación del objeto de estudio.

Métodos empíricos

1. **Entrevista:** Se optó por este método con la perspectiva de obtener una mayor información sobre el tema de la investigación y poder identificar las especificidades con las cuales debe cumplir la solución [3].

Estructura del documento

El presente documento se estructura en 3 capítulos desglosados de la siguiente forma:

Capítulo 1. Fundamentación teórica: En este capítulo se exhibe el estudio del estado del arte referente a perfiles de usuario en sistemas de planificación. Se identifican y describen herramientas y tecnologías que permiten el diseño e implementación de las funcionalidades que darán respuesta al problema planteado. Se describe la metodología por la cual se desarrolla la solución. Se definen conceptos básicos para un mejor entendimiento del tema en cuestión.

Capítulo 2. Análisis y diseño del sistema: En este capítulo se describen y especifican los requisitos funcionales y no funcionales que debe cumplir el perfil de usuario. Se desarrollan los artefactos que definen las fases y disciplinas de la metodología a utilizar. Se describen los patrones de diseño a emplear. Se obtiene el diseño del paquete de funcionalidades y la arquitectura que soporta la solución.

Capítulo 3. Implementación y prueba: En este capítulo se plantean los elementos relacionados con la implementación del perfil de usuario y los criterios que permitieron validar el cumplimiento de los objetivos propuestos. Además se describen las pruebas a las que estuvieron sometidas las funcionalidades implementadas, para determinar la calidad y validación de la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se realiza un estudio del estado del arte de los sistemas de planificación de actividades existentes, y para ello se tienen en cuenta las características asociados a cada uno de ellos. Además se expone la metodología por la cual se guía todo el proceso de desarrollo del software, así como las herramientas, tecnologías y métodos empleados en el desarrollo del perfil de usuario para el SIPAC.

1.1 Estudio de sistemas de planificación

Son diversos los sistemas que en su concepción incluyen funciones relacionadas con los perfiles de usuario. Muchos de estos sistemas cuentan con funciones las cuales forman parte de las incluidas o definidas para la creación del perfil de usuario a proponer, por tanto es necesario realizar un estudio para definir cuáles de ellos cuentan con esas funciones.

1.1.1 SugarCRM

Sistema para la administración de la relación con los clientes. Está diseñado para organizar y controlar cada una de las actividades de la organización, pero también para ayudar a cada uno de los colaboradores de la organización a obtener logros importantes.

Cuenta con funcionalidades que permiten al usuario definir preferencias como:

- Nombre de usuario, el nombre usado como usuario para autenticarse y la contraseña, y el estatus del usuario.
- Especificar sucesos como si se desea enviar un correo electrónico a un usuario en cada ocasión que algo le sea asignado, debe especificar el nombre que desea que aparezca en los correos electrónicos, también puede crear o seleccionar una firma para dicho usuario.
- Especificar los valores personalizados (hora, fecha), de lo contrario podrá dejar que los ajustes generales se apliquen a todos los usuarios.
- Especificar el estado del empleado, título, departamento, números de teléfonos.
- Introducir la dirección del usuario.
- Introducir una clave si lo desea para evitar el acceso de otros usuarios al calendario del usuario que está creando, de lo contrario dicho calendario será accesible por el resto de usuarios [4] [5].

1.1.2 PlanningPME

Sistema de planificación para administrar los cambios, proyectos, tareas o citas de una empresa. Brinda a sus usuarios una serie de funcionalidades de forma tal que les facilita el trabajo, permitiendo:

- Crear, modificar, planificar y controlar los planes de trabajos de las empresas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

- Permite la personalización de los planes de trabajos según los tipos de usuarios que interviene en el sistema.
- Se pueden definir usuarios y departamentos para controlar el acceso a la planificación.
- Permite el seguimiento de todas las modificaciones de horario y control en todo momento de los cambios que se realizan.
- Mejora su comunicación con las notificaciones de correo electrónico y la información del tiempo que le resta para cumplir con las actividades de su plan de trabajo [6] [7].

1.1.3 SAP

Sistema de Aplicaciones y Productos es una solución que maneja los procesos de negocios de las empresas que lo usan, gestionado sus recursos humanos, sus finanzas, el control de sus costos, su logística y sus materiales. Define datos de los usuarios, entre ellos:

- Datos personales como el nombre, apellidos, departamento, teléfono.
- El usuario puede tener por defecto un menú inicial que no sea el general del sistema. Incluso se puede crear menús personalizados y asignárselos a los usuarios que van a trabajar en un determinado ámbito.
- Se pueden guardar configuraciones predefinidas de cada área o sector con el cual esté trabajando el usuario [8].

Luego de haber realizado un estudio de los sistemas Edisa, Isis, Epica y Abanq que aun así no aportaron datos a la investigación pues no cuentan con perfiles de usuarios definidos y los descritos anteriormente con el objetivo de analizar cómo gestionan las preferencias de sus usuarios, se puede señalar que éstos permiten una mejor comprensión del funcionamiento o definición de lo que puede representar un perfil de usuario. En su composición definen una serie de funciones que serán de utilidad en la confección de las funcionalidades que se proponen como respuesta al problema existente, entre ellas, el uso de datos más detallados sobre la persona o usuario que está vinculada al uso sistémico con el sistema, las notificaciones vía correo electrónico, las modificaciones que puedan existir en los planes de trabajo de los usuarios y la personalización de estos planes en correspondencia a las preferencias de los usuarios.

Aun cuando las características y funciones de estos sistemas son semejantes a lo que se desea lograr, no se pueden usar como solución al problema existente, debido a que no se pueden integrar al núcleo de SIPAC, pues no cuentan con las mismas tecnologías de desarrollo sobre las que está creado el mismo. Estos sistemas presentan como principal desventaja que son de licencia privativa y su utilización no está

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

en correspondencia con las políticas de soberanía tecnológica del país. Teniendo en cuenta todo lo expresado anteriormente se decide desarrollar un perfil de usuario para el SIPAC que dé respuesta al problema planteado.

1.2 Metodología de desarrollo

En la Universidad de las Ciencias Informáticas para la creación de nuevas aplicaciones y garantizar el desarrollo de sistemas con mayor calidad y precisión en lo que se desea alcanzar con sus objetivos, se opta por la metodología AUP_ Agile Unified Process² (Proceso Unificado Ágil) en su variación para su uso en la UCI, teniendo en cuenta el modelo CMMI-DEV v1.3_Capability Maturity Model Integration for Development³ (Modelo de Integración de Capacidad y Madurez para el Desarrollo) [9].

AUP define tres fases (Inicio, Ejecución, Cierre), ocho disciplinas para el ciclo de vida de los proyectos (Modelado del negocio, Requisitos, Análisis y diseño, Implementación, Pruebas interna, Pruebas de liberación, Pruebas de aceptación, Despliegue) y 11 roles (Jefe de proyecto, Planificador, Analista, Arquitecto de información, Desarrollador, Administrador de la configuración, Stakeholder, Administrador de calidad, Probador, Arquitecto de software), de estas fases y disciplinas se transita por las siguientes para el desarrollo del perfil de usuario:

1.2.1 Fases

- ✓ **Inicio:** en esta fase se realizan actividades relacionadas con la planeación del proyecto, se efectúa un estudio de la organización y los clientes de forma tal que se obtenga información relacionada con el alcance del proyecto, si se realiza o no, además establecer estimaciones de tiempo, esfuerzo y costos.
- ✓ **Ejecución:** se ejecutan las actividades requeridas para el desarrollo de la solución, dígase el modelo de negocio, obtención de requisitos, la arquitectura y diseño, la implementación y por último la liberación del producto; donde se entrega el producto al cliente final y se capacitan para su uso [10].

² AUP describe de forma fácil de entender el proceso de desarrollo de aplicaciones de software del negocio usando técnicas ágiles y conceptos que aún se mantiene válidos en RUP.

³ CMMI-DEV proporciona un enfoque disciplinado para mejorar los procesos de una organización. Constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora, estas prácticas se centran en el desarrollo de productos y servicios de calidad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.2.2 Disciplinas

- ✓ **Modelado de negocio:** esta disciplina puede ser opcional, pero está destinada a comprender el funcionamiento del negocio que se desea informatizar, garantizando de esta forma que el producto final cumpla con los objetivos o propósitos trazados.
- ✓ **Requisitos:** se procede a desarrollar un modelo de la solución que se desea obtener, además se administran y gestionan los requisitos funcionales y no funcionales especificados por el cliente.
- ✓ **Análisis y diseño:** se modela la solución y su forma, incluyendo la arquitectura, de manera tal que soporte los requisitos funcionales y no funcionales, se puede además, perfeccionar los requisitos de forma tal que se tenga una mejor claridad o entendimiento de lo que se debe desarrollar para conseguir el producto deseado.
- ✓ **Implementación:** se construye la solución a partir de los resultados obtenidos en la fase de análisis y diseño.
- ✓ **Pruebas internas:** después de implementadas todas las funcionalidades que conforman la solución final, en esta fase se verifica el resultado de la implementación desarrollada probando cada funcionalidad; se generan ciertos artefactos de pruebas entre ellos, los diseños de casos de prueba, entre otros [10].

1.3 Lenguajes, tecnologías y herramientas

Para el desarrollo de toda aplicación es necesario tener en cuenta los lenguajes, herramientas y tecnologías a utilizar, de forma tal que brinde la posibilidad de crear un sistema bajo las especificaciones requeridas asegurando su correcto funcionamiento. Las herramientas, tecnologías y lenguajes a usar para desarrollar la solución de la cual es objeto este trabajo de diploma, son las definidas por el equipo de desarrollo del proyecto SIPAC, estas se describen a continuación:

1.3.1 Lenguajes de modelado

UML v2.0 Unified Modeling Language (Lenguaje de Modelado Unificado) fue concebido para modelar los elementos de un sistema de software de una manera estandarizada que incluya conceptos del proceso de negocio y funciones del sistema. Este modelado es de fácil comprensión para cualquier desarrollador con conocimientos sobre UML, y puede ser utilizado en cualquier tipo de desarrollo [11].

1.3.2 Lenguajes de desarrollo del lado del cliente

HTML Hyper Text Markup Language (Lenguaje de Marcas de Hipertexto) es un lenguaje sencillo escrito mediante etiquetas (Tags) que permiten definir y ubicar los distintos elementos que componen una página

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

web. Se integra con lenguajes como JavaScript y PHP y no todos los navegadores interpretan este lenguaje de la misma forma, lo cual puede ser visto como una de sus desventajas [12].

CSS Cascading Style Sheets (Estilo en Cascada) constituye un estándar que permite establecer el estilo (tamaños, íconos, imágenes, colores, espacios y bordes) de documentos estructurados, dígame la capa de presentación de una página web. Este lenguaje permite definir el aspecto visual del documento, mientras que separa la parte semántica (HTML) de la presentacional (style sheets) [13].

JavaScript v1.6 es un lenguaje que no requiere compilación ya que es interpretado por todos los navegadores y es utilizado para la creación de páginas web dinámicas. Dicho lenguaje permite crear diferentes efectos e interactuar con los usuarios. Además interactúa, perfectamente con códigos basados en HTML, lo cual constituye una de sus principales ventajas [14].

1.3.3 Lenguajes de desarrollo del lado del servidor

PHP v5.4 Hypertext Preprocessor es el lenguaje que se empleó para programar del lado del servidor. Es interpretado y completamente orientado al desarrollo de aplicaciones web dinámicas. Es gratuito, fácil de usar y aprender, portable, de código abierto y multiplataforma. Presenta interfaces para una gran cantidad de sistemas de base de datos diferentes, así como bibliotecas incorporadas para muchas tareas web habituales [15].

1.3.4 Tecnologías y herramientas de desarrollo

Un **framework** (marco de trabajo) es una estructura de soportes de programas, bibliotecas y lenguajes de scripting⁴. Es considerado una arquitectura de software que modela las relaciones generales de los componentes del proyecto que lo implementa; provee una estructura y manera de trabajo la cual utilizan las aplicaciones del proyecto. La finalidad de los marcos de trabajos es facilitar el desarrollo de software, permitiéndoles a diseñadores y programadores concentrarse en los requerimientos del proyecto, reduciendo los posibles problemas con las tecnologías utilizadas, así como facilitando ciertas funcionalidades básicas y comunes [16].

Saaxe v1.3.4 es un marco de trabajo, fusionado bajo tecnologías totalmente libres (entre ellas PHP, Postgresql) que posee el desarrollo de tecnologías propias basadas en otros marcos de trabajo como ZendFramework para el manejo de la lógica de negocio, Doctrine para el acceso a datos y ExtJS para la

⁴ Un lenguaje scripting es un tipo de lenguaje de programación que es generalmente interpretado (proceso intermedio entre el texto escrito y su significado).

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

capa de presentación. Cuenta con una arquitectura en capas que a su vez presenta en su capa superior un MVC (patrón arquitectónico Modelo Vista Controlador). Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo [17].

ZendFramework v1.9.7

ZendFramework2 o ZF2 es un marco de trabajo para el manejo de la lógica de negocio. Es una implementación que usó código 100% orientado a objetos. Es un marco de trabajo de código abierto para desarrollar aplicaciones web y servicios web con PHP5. Brinda facilidades de uso y poderosas funcionalidades, posee buenas capacidades de ampliación y proporciona un sistema de caché de forma que se puedan almacenar diferentes datos, así como los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador. Consta de mecanismos de filtrado y validación de entradas de datos. Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX y provee capacidades de búsqueda sobre documentos y contenidos [18].

Doctrine v1.2.2 es empleado para la capa de acceso a datos. Es un sistema ORM (en inglés Object Relational Mapper) para PHP 5.2 o superior que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL, manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Además, exporta una base de datos existente a sus clases correspondientes y convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos [19].

ExtJS v2.2 es el marco de trabajo empleado para el desarrollo de la capa de presentación. Está basado completamente en la programación orientada a objeto. Cada objeto contiene lo típico: propiedades, métodos y eventos. Basa toda su funcionalidad en JavaScript a través de bibliotecas. Así, en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso de Document Object Model (DOM). Los datos son obtenidos con AJAX. Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos. Además permite que exista un balance entre el Cliente-Servidor, posibilitando que la carga de procesamiento se distribuye, permitiendo que el servidor al tener menor carga, pueda manejar más clientes al mismo tiempo [20].

1.3.5 Técnicas de desarrollo

Ajax por sus siglas en inglés Asynchronous JavaScript And XML (JavaScript asíncrono y XML) es la técnica de desarrollo web que se usa para poder hacer consultas asíncronas al servidor sin necesidad de recargar la página. Esta surge de la combinación de tres tecnologías ya existentes: HTML (o XHTML y Hojas de Estilo en Cascada (CSS) para presentar la información, DOM y JavaScript, para interactuar dinámicamente con los datos, además de XML y XSLT, para intercambiar y manipular datos de manera sincronizada con un servidor web [21].

CASE: **Visual Paradigm v8.0**

Se emplea Visual Paradigm for UML v8.0 como herramienta CASE. Utiliza UML v2.1 como lenguaje de modelado, con soporte multiplataforma y proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Ayudando a construir aplicaciones de calidad más rápido, mejor y a bajo costo [22].

Sistema de control de versiones: **Subversión**

Subversión (SVN) 1.6 es la herramienta de entorno colaborativo que se utiliza para el control de versiones. Se encuentra preparado para funcionar en red y se distribuye bajo licencia libre. Mantiene versiones no sólo de archivos, sino también de directorios y versiones de los metadatos asociados a esos directorios. Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre. Brinda facilidades de soporte tanto de ficheros de texto como binarios [23].

Entorno integrado de desarrollo (IDE): **NetBeans**

La presente solución se desarrolla sobre el IDE de programación multiplataforma NetBeans 8.0 el cual tiene soporte para la versión 8.2 de PhpStorm, JavaScript, el diseño de Hojas de Estilo (CSS) y HTML. Se integra con varias herramientas como el Subversión y servidores web. Es un producto de código abierto, con todos los beneficios del programa disponible en forma gratuita. Hace uso de plugins para ampliar sus funcionalidades, lo que le da una gran facilidad de uso [24].

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Servidor web: **Apache**

Se utiliza como servidor web Apache v2.2 pues es una tecnología gratuita de código abierto compatible con muchos Sistemas Operativos. Tiene todo el soporte que se necesita para tener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Tiene una alta configuración en la creación y gestión de registros de actividad. Apache permite la creación de ficheros de registro a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor [25].

Sistema gestor de bases de datos: **PostgreSQL**

Como Sistema Gestor de Base de Datos (SGBD) se emplea la versión 9.1 de PostgreSQL. Constituye un sistema de gestión de bases de datos relacional. Es una herramienta de código abierto, de bajo coste y multiplataforma. Se destaca en ejecutar consultas complejas, subconsultas y uniones de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Soporta transacciones, claves ajenas con comprobaciones de integridad referencial y almacenamiento de objetos de gran tamaño. Cuenta con varias herramientas gráficas de diseño y administración de bases de datos como el pgAdmin, que no es más que, una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración [26].

Navegador web: **Mozilla Firefox v24**

Mozilla Firefox es un navegador web de código abierto y software libre, entre sus funciones está la de mejorar la velocidad de carga de las páginas. Entre sus características generales se incluyen la navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas y navegación privada; además es compatible con varios estándares web, incluyendo HTML, XHTML y CSS [27].

1.4 Conclusiones Parciales

Luego de finalizado este capítulo, se concluye que se realizó un estudio sobre los sistemas de planificación, lo que permitió deducir la necesidad de desarrollar de un perfil de usuario que dé respuesta o facilite el registro, seguimiento y control de la planificación a corto, mediano y largo plazo en el SIPAC. La metodología seleccionada: AUP, comprende todas las fases de la construcción de un sistema de gestión y ofrece varias ventajas que facilitan su desarrollo. Se realizó un estudio de las tecnologías,

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

lenguajes y herramientas para el desarrollo de aplicaciones, lo que ayudó a seleccionar las herramientas para el desarrollo del perfil de usuario, teniendo como resultado:

- ✓ Sauxe: como marco de trabajo.
- ✓ PostgreSQL: como servidor de base de datos.
- ✓ Apache: como servidor web.
- ✓ NetBeans: como entorno de desarrollo.
- ✓ pgAdmin: como cliente para la administración de la base de datos.
- ✓ Subversion: para el control de versiones.
- ✓ Visual Paradigm: como herramienta CASE.
- ✓ Mozilla Firefox: como navegador web.
- ✓ PHP y JavaScript: como lenguajes de programación.
- ✓ UML: como lenguaje de modelado.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL SISTEMA

A lo largo de este capítulo se definen las principales características que debe cumplir la solución a desarrollar. Se definen y especifican los requisitos funcionales garantizando las funcionalidades que debe proporcionar la misma. De igual forma se definen los requisitos no funcionales a utilizar para el desarrollo de la solución, los cuales especifican las restricciones de las funcionalidades ofrecidas. Además se describen los patrones y estilos arquitectónicos a utilizar para la solución propuesta; permitiendo fundamentar el uso de los patrones empleados durante el modelado de los diagramas, que describen la relación entre las clases existentes del diseño, así como la presentación del modelo de datos de la solución.

2.1 Descripción de la solución

Actualmente en SIPAC se hace necesaria la implementación de una solución que permita al usuario personalizar sus preferencias centrado en dos elementos fundamentalmente:

- 1- La visualización del sistema, teniendo en cuenta la forma en que se muestran algunos elementos.
- 2- Las funcionalidades del sistema con el objetivo de adaptar lo mejor posible SIPAC a las preferencias del usuario.

De esta forma permite registrar los datos de contacto del usuario, definir patrones de actividades genéricas asociadas a los tipos de actividades, recibir notificaciones al correo del usuario, entre otras funcionalidades que facilitan el registro, seguimiento y control de la planificación a corto, mediano y largo plazo.

Teniendo en cuenta los elementos antes descritos se propone desarrollar un perfil de usuario como solución al problema existente, el cuál puede definirse como “el conjunto de opciones de configuración que hacen que un sistema tenga las características o preferencias que el usuario desea”, haciendo de esta forma los entornos de trabajo adaptables a las necesidades de los usuarios [28]. Teniendo más claridad de lo que constituye un perfil de usuario, se representa seguidamente en el modelo conceptual los conceptos básicos que maneja.

2.2 Modelo conceptual

El modelo conceptual constituye una representación de los principales conceptos que se manejan en el desarrollo de una solución, este describe las clases más significativas dentro del contexto o campo que se esté analizando [29].

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

Conceptos asociados al perfil de usuario

Usuario: son las personas que utilizan o trabajan directamente con un sistema, o algún objeto.

Personalizaciones: son las modificaciones que los usuarios pueden realizar en el sistema de acuerdo a sus preferencias.

Visualización: es la forma en que se muestran las personalizaciones realizadas por el usuario en el sistema.

Funcionalidad: son las funciones implementadas o modificadas en el sistema, para poder desarrollar el perfil de usuario.

Notificación: se refieren al hecho de informar a los usuarios cuando ha ocurrido una acción sobre algún elemento de la planificación.

Correo electrónico: es un servicio consumido para poder notificar a los usuarios, de las variaciones ocurridas.

Patrones de actividad: son los modelos por los cuales se planifican las actividades dentro de la planificación.

Los restantes conceptos se encuentran especificados en el artefacto Modelo conceptual del expediente de proyecto SIPAC 2.1.

La **figura 1** representa el modelo conceptual asociado a los conceptos que se manejan para el dominio del perfil de usuario a desarrollar.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

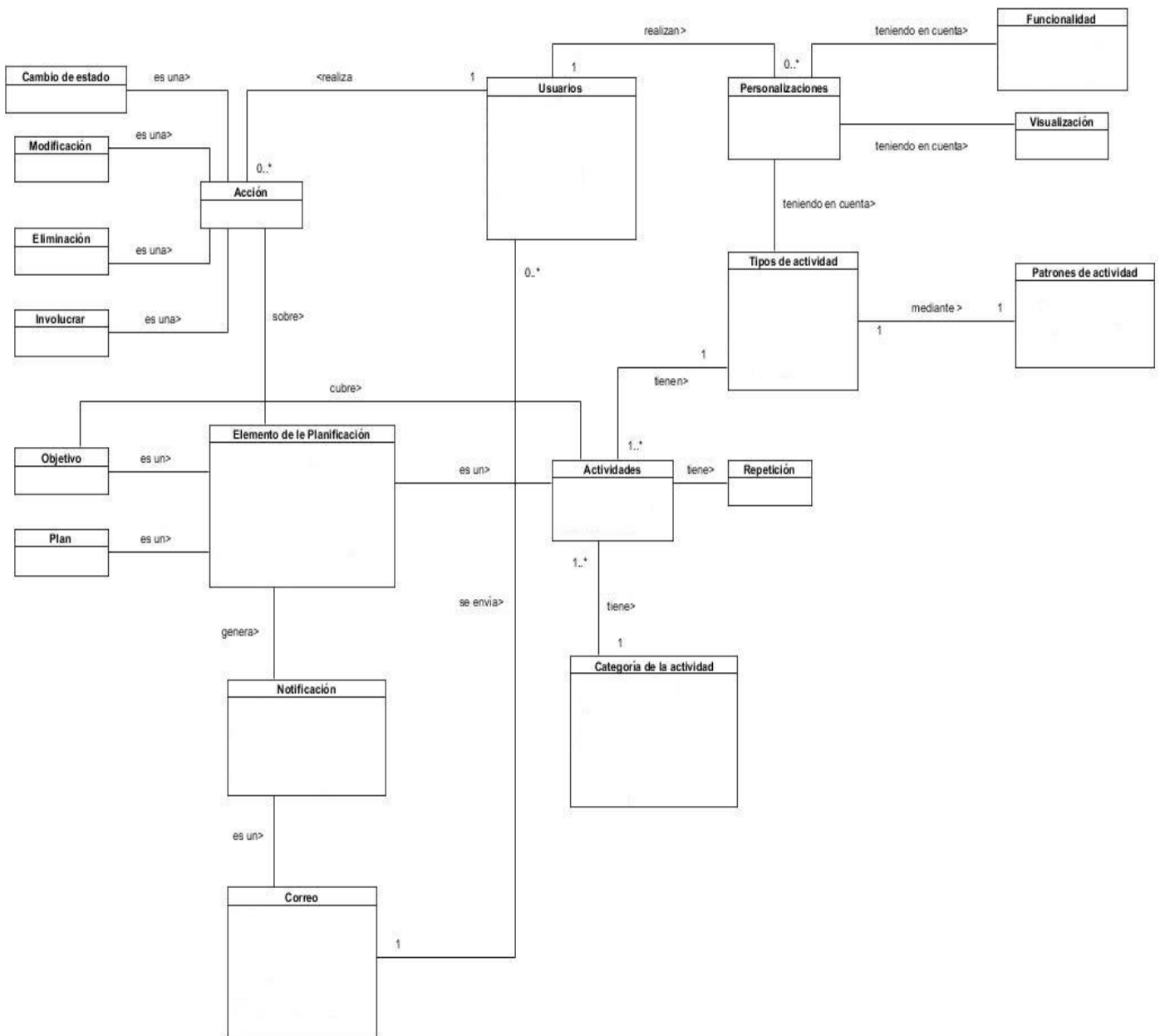


Figura 1. Modelo conceptual del perfil de usuario para el SIPAC

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.3 Requisitos

Un requisito constituye una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste; es una definición detallada y formal de una función del sistema [30].

En este acápite se hace alusión a la descripción de los requisitos que debe cumplir la solución desarrollada para lograr resolver el problema existente, garantizando así que éste cuente con la calidad requerida y cumpla las expectativas del cliente.

2.3.1 Técnicas de captura de requisitos

La calidad con la que se capturen los requisitos, garantiza el desarrollo exitoso de una solución que cuente con las preferencias especificadas por el cliente, además facilita el trabajo para el equipo de desarrollo [31]. Las técnicas empleadas para la obtención de los requisitos que debe cumplir la solución se describen a continuación:

- ✓ **Entrevista:** permitió entender con mayor claridad el problema existente por el cual es necesario el desarrollo de dicha solución, siendo entrevistado el cliente principal del sistema.
- ✓ **Talleres:** se intercambiaron ideas entre miembros del equipo de desarrollo del SIPAC y el cliente, de manera tal que se obtuvieron los requisitos específicos a desarrollar, para lograr una solución con mayor calidad dada las necesidades existentes.

2.3.2 Requisitos funcionales

Los requisitos funcionales (RF) se pueden definir como las declaraciones para determinar qué hará el software, definiendo las relaciones de su operación y su implementación, estos deben ser explícitos, no ambiguos y claros, para lograr un mejor entendimiento [30].

Los RF identificados para el desarrollo del perfil de usuario para SIPAC fueron 22 en su totalidad, de los cuales 2 son de complejidad Baja, 17 de complejidad Media y 3 de complejidad Alta, dicha complejidad fue determinada mediante el artefacto Evaluación de requisitos, a continuación se listan:

RF1 Datos de contacto de usuario.

RF1.1 Adicionar datos de los contactos de usuario.

RF1.2 Modificar datos de los contactos de usuario.

RF1.3 Listar datos de los contactos de usuario.

RF1.4 Cargar imagen del contacto.

RF1.5 Guardar imagen de contacto.

RF2 Configurar visualización del sistema.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

RF2.1 Adicionar configuración de la visualización del sistema.

RF2.2 Modificar configuración de la visualización del sistema.

RF2.3 Cargar configuración de la visualización del sistema.

RF3 Enviar notificaciones vía correo electrónico.

RF4 Gestionar patrones para las actividades con frecuencia periódicas.

RF4.1 Adicionar patrón de actividad con frecuencia periódica.

RF4.2 Modificar patrón de actividad con frecuencia periódica.

RF4.3 Eliminar patrón de actividad con frecuencia periódica.

RF4.4 Cargar listado de patrones de actividad con frecuencia periódica.

RF5 Visualizar elementos.

RF5.1 Visualizar calendario por defecto.

RF5.2 Visualizar elementos según la preferencia de los usuarios, antes, durante y después del mes.

RF6 Insertar relación automática de las actividades asignadas por el superior en el PTI⁵ del mes.

RF7 Gestionar envío a aprobación.

RF7.1 Adicionar configuración de envío a aprobación.

RF7.2 Modificar configuración de envío a aprobación.

RF7.3 Cargar configuración de envío de aprobación.

RF7.4 Enviar a aprobación los elementos según el período establecido.

RF8 Gestión de actividades.

RF8.1 Adicionar actividad partiendo de un patrón de actividades configurado.

RF8.2 Modificar actividad partiendo de un patrón de actividades configurado.

2.3.3 Especificación de los requisitos funcionales

Al realizar la especificación de requisitos se tiene como resultado la descripción del comportamiento de las funcionalidades que se van a desarrollar. Es necesario que en su descripción se emplee un lenguaje sencillo de forma tal que sea fácilmente entendible por el equipo de desarrollo [32].

La tabla que se muestra a continuación, responde a la descripción del requisito funcional: Adicionar datos de los contactos de usuario; las restantes descripciones de los requisitos se encuentran explícitamente descritos en el expediente de proyecto SIPAC 2.1 en el artefacto Descripción de requisitos.

⁵ PTI se refiere al Plan de Trabajo Individual.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

Tabla 1. Descripción de requisito: Adicionar datos de los contactos de usuario

Precondiciones		El usuario debe estar autenticado en el sistema.
Flujo de eventos		
Flujo básico Registrar datos del usuario		
1.	El usuario accede al perfil de usuario, desde el menú Inicio/Configuración/Perfil de usuario.	
2.	El sistema muestra una interfaz con los datos de contacto del usuario, ver figura2 , el usuario selecciona la opción Editar , para registrar sus datos de contacto.	
3.	El sistema muestra una interfaz que permite al usuario introducir los siguientes datos: Nombre, Apellidos, Cargo, Teléfono, Nombre del jefe, Cargo del jefe, Usuario, Contraseña, Servidor, Puerto, Seguridad	
4.	El sistema valida que los datos introducidos por el usuario sean correctos y los registra.	
5.	Concluye el requisito.	
Pos-condiciones		
1.	Se registra en el sistema un nuevo contacto.	
Flujos alternativos		
Flujo alternativo 4a Campo vacío		
1	El sistema señala el campo vacío y permite corregirlo.	
2	El usuario ingresa el dato del campo vacío y selecciona la opción Aceptar .	
3	Volver al paso 4 del flujo básico.	
Pos-condiciones		
1	Se registra en el sistema un nuevo contacto.	
Validaciones		
1	Se validan los datos según lo establecido en el documento Modelo_conceptual.doc	
Conceptos	Usuarios	Visibles en la interfaz: Utilizados internamente:
Requisitos especiales		

Prototipo de interfaz de usuario

Inicio Perfil de...

Perfil de usuario

Información de contacto Personalización Patrones de actividades

Acerca de:

Nombre(s): No definido
Apellidos: No definido
Cargo: No definido
Teléfono: No definido
Correo electrónico: No definido

Editar

Información de contacto:

Datos del usuario

Nombre: Apellidos:
Cargo: Teléfono:

Datos del jefe

Nombre:
Cargo:

Configuración del correo

Usuario: Contraseña:
Servidor: Seguridad:
Puerto:

Cambiar imagen

Cancelar Aceptar

Activar Windows
Ir a Configuración de PC para activar Windows.

Figura 2. Interfaz para adicionar los datos de contacto del usuario

2.3.4 Validación de requisitos funcionales

Una vez definidos los requisitos del sistema, estos deben ser validados para asegurar que el análisis de los mismos y los resultados obtenidos son correctos. La validación de los requisitos tiene como misión demostrar que su definición especifica realmente el sistema que el cliente desea. Durante este proceso se examinan las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones; que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajuste a los estándares establecidos para el proceso, el proyecto y el producto [33].

Para la validación de los requisitos funcionales antes definidos, se utilizó la técnica Revisión de requisitos mediante los artefactos: Criterios para validar requisitos del cliente y Criterios para validar requisitos del producto y la técnica Construcción de prototipos de interfaz de usuario con el uso de la herramienta Visual Paradigm. En la revisión se localizaron errores de ambigüedad los cuales fueron corregidos para garantizar que los requisitos cuenten con la calidad requerida. Mediante los prototipos se le mostró al cliente un modelo ejecutable del perfil de usuario, que le permitió tener una visión preliminar de cómo será el mismo, permitiendo así comprobar si satisface sus necesidades y de esta forma dar su aprobación mostrada mediante un Acta de aceptación de requisitos (ver anexo 5).

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.3.5 Requisitos no funcionales

Los requisitos no funcionales están orientados a las restricciones de los servicios o funciones ofrecidos por el sistema, incluyen además las restricciones sobre el proceso de desarrollo, estándares, usabilidad, portabilidad, entre otros [31]. Los requisitos no funcionales relacionados con la solución se describen a continuación:

Precisión

Capacidad del software para proporcionar efectos o resultados correctos o convenidos con el grado de exactitud necesario.

- ✓ Las notificaciones generadas por el sistema al usuario activo, deberán especificar siempre la fecha de emisión de la misma así como al elemento de planificación al que se encuentra relacionada y el autor de la operación que generó dicha notificación.

Usabilidad

Comprensibilidad

Capacidad del producto de software para permitirle al usuario entender si el software es idóneo, y cómo puede usarse para las tareas y condiciones de uso particulares.

- ✓ Todos los mensajes de error del sistema deberán incluir una descripción textual del error.
- ✓ Las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.

Cognoscibilidad

Capacidad del producto de software para permitirle al usuario aprender su aplicación.

- ✓ El idioma de todas las interfaces de la aplicación será el español.
- ✓ La iconografía utilizada será única para cada operación, permitiendo representar todos los conceptos del dominio de la aplicación con un ícono distintivo.
- ✓ En cada formulario del sistema los campos obligatorios serán señalados en rojo.
- ✓ El sistema mostrará las opciones desactivadas siempre que no se hayan cumplido las condiciones previas para su activación.

Atracción

Capacidad del producto de software de ser atractivo o amigable para el usuario.

- ✓ En el sistema no existirá una cadena de más de 3 interfaces de usuario para lograr una funcionalidad completa.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

- ✓ El sistema diferenciará los mensajes de información de los mensajes de error y de advertencia valiéndose de distintos íconos para cada tipo.
- ✓ Se colocará un máximo de 10 campos en los formularios del sistema.
- ✓ El sistema presentará por cada interfaz, los términos capitalizados, es decir, la primera palabra tendrá su primera letra en mayúsculas.
- ✓ La tipografía y colores serán estándares en toda la aplicación.
- ✓ En cada interfaz de usuario, los campos de texto tendrán un tamaño estándar y su alto dependerá del espacio con que cuente el panel contenedor.

Eficiencia

Rendimiento

Capacidad del producto de software para proporcionar apropiados tiempos de respuesta y procesamiento, así como tasas de producción de resultados, al realizar su función bajo condiciones establecidas.

- ✓ El sistema no excede los 5 s de respuesta al efectuar acciones de cargar un registro con gran cantidad de datos dispersos y los 2 s para 20 registros con datos poco dispersos (esta cifra no incluye los retardos por concepto de tráfico de red).
- ✓ El sistema no excede los 1.5 s para efectuar acciones de salvar información (esta cifra no incluye los retardos por concepto de tráfico de red).

Flexibilidad

Capacidad del producto de software para permitir la aplicación de una modificación especificada.

- ✓ El sistema permitirá agregar nuevas funcionalidades o modificar alguna existente sin romper la estructura y consistencia de los componentes.

El resto de los RNF se encuentran descritos en el expediente de proyecto SIPAC 2.1 en el artefacto CIG-SPA-N-i3514-RNF [34].

2.4 Modelado de la solución

En este epígrafe se procede a modelar los artefactos y conceptos necesarios para obtener una mejor comprensión de la funcionalidad a desarrollar.

2.4.1 Diseño de la solución en términos de componentes

La arquitectura de SIPAC presenta una estructura que responde a diferentes niveles de empaquetamiento, dígame subsistema, componente, funcionalidad y requisito.

De esta forma el subsistema Planificación está constituido por los siguientes componentes:

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

- ✓ **Configuración:** permite configurar y gestionar los usuarios y grupos de usuarios, permisos y nomencladores del sistema, además de la funcionalidad perfil de usuario.
- ✓ **Planeación:** permite gestionar los planes, áreas de resultados clave, objetivos y criterios de medida, así como actividades y factores que influyen en la planificación.
- ✓ **Notificaciones:** permite gestionar las acciones que generarán una notificación así como notificar al usuario activo sobre la asignación o variación de algún elemento de la planificación.

En el componente Configuración la solución perfil de usuario se encuentra relacionado con la funcionalidad GestionarNomencladores. A dicha funcionalidad se le realizaron modificaciones en correspondencia a la solución.

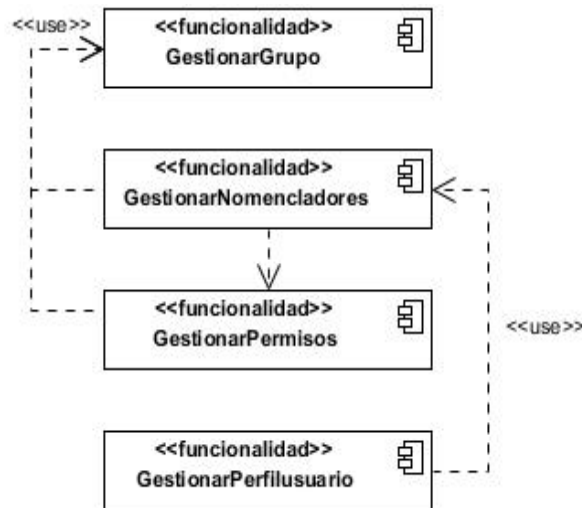


Figura 3. Diagrama de la funcionalidad perfil de usuario para el SIPAC

2.4.2 Diagrama de clases del diseño

El diagrama de clases del diseño, describe las especificaciones de las clases y las interfaces en una aplicación [35]. El diagrama de clases del diseño referente a la solución propuesta, está basado en estereotipos web y en el patrón arquitectónico Modelo-Vista-Controlador (MVC). Para obtener una imagen ampliada de las partes que componen dicho patrón, ver anexos 1, 2 y 3.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

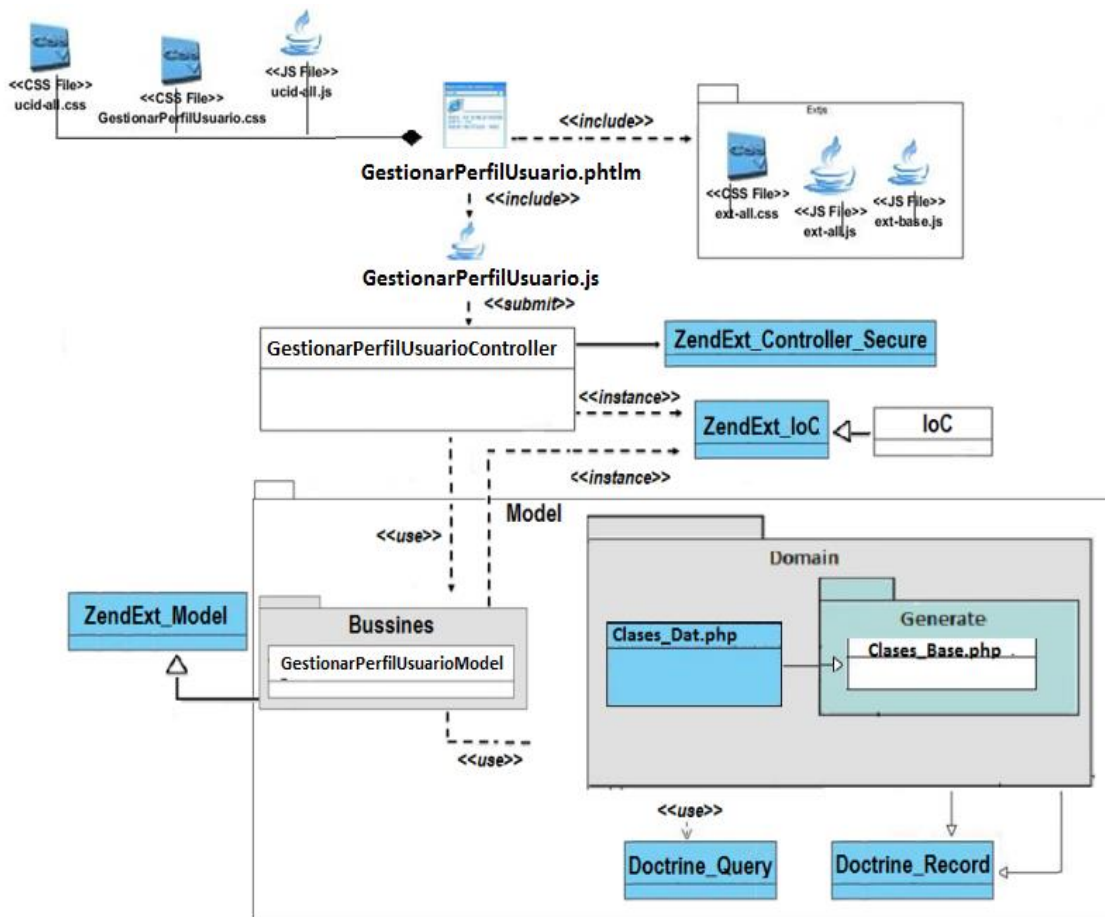


Figura 4. Diagrama de clases de diseño del perfil de usuario

Tabla 2. Descripción de las principales clases del diagrama de clases del diseño

Clases	Descripción
GestionarPerfilUsuario.phtml	Página cliente de la funcionalidad, encargada de incluir las clases js y css a visualizar.
GestionarPerfilUsuario.css	Contiene las clases para darle los estilos gráficos a la solución, separando de esta forma los estilos del contenido.
GestionarPerfilUsuario.js	Contiene las funciones y variables globales que se ejecutan llamando a sus funciones desde cualquier clase sin tener que incluirlas en cada una de ellas.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

GestionarPerfilUsuarioController	Clase encargada de controlar la comunicación entre la vista y el modelo.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
Zend_Ext_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos.
Extjs	Contiene los componentes generados a través de la biblioteca JavaScript Extjs.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
Paquete Bussines	Encargado de manejar toda la lógica de negocio.
Paquete Domain	Contiene los métodos para manejar el negocio.
IoC	Archivo xml donde se publican los servicios que brindan cada uno de los componentes de SIPAC mediante un lenguaje de descripción de servicios web (WSDL).
Paquete Services	Paquete que contiene las clases donde se implementa la lógica de los servicios que brinda el componente Configuración.

2.5 Patrones del diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Permiten tener una estructura de código común, por lo que ahorran gran cantidad de tiempo en la construcción de software [36].

2.5.1 Patrones GRASP

Los patrones GRASP General Responsibility Assignment Software Patterns son patrones generales de software para asignación de responsabilidades [37].

Con el objetivo de desarrollar una solución flexible y reusable, se pusieron en práctica los siguientes patrones GRASP que a continuación se describen:

- ✓ **Experto:** determina la clase que posee mayor jerarquía para asignarle una responsabilidad según la información que posee, garantizando el encapsulamiento de la información y facilitando el bajo

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

acoplamiento en las aplicaciones. El uso de este patrón se evidencia en las clases `DatDatosUsuario`, `DatVisualizacion`, `DatPatronActividad`, `DatNomenclador`, `DatAprobacion`.

- ✓ **Creador:** este patrón es usado cuando se quiere a partir de una clase con alta jerarquía obtener clases descendientes o instancias a partir de las clases obtenidas, se evidencia en la clase `GestionarPerfilUsuarioController` la cual se encarga de crear las instancias de la clase `GestionarPerfilUsuarioModel`, para usar las funcionalidades de ésta.
- ✓ **Bajo Acoplamiento:** define que las clases deben implementar la menor cantidad de métodos para garantizar independencia entre ellas. Garantizando así, que exista la menos repercusión en las clases en caso de existir alguna modificación en alguna de ellas, de esta forma se potencia la reutilización.
- ✓ **Alta Cohesión:** el uso de este patrón indica que la información almacenada en las clases debe ser coherente y relacionada a lo que se maneja en dicha clase. Su uso se evidencia en todas las clases, ejemplo de ellas `GestionarPerfilUsuarioController`.
- ✓ **Controlador:** las diferentes clases controladoras se encargan de llevar el control de todos los eventos relacionados con el negocio. Implementan las funcionalidades que dan respuesta a las peticiones del usuario. En la solución desarrollada se refleja este patrón en la clase `GestionarPerfilUsuarioController`.

2.5.2 Patrones GoF (Gang of Four)

Los patrones GoF bien conocidos como el grupo de los cuatro, se clasifican en tres grupos dependiendo de sus propósitos, Creacional, Estructural y Comportamiento [38]. Estos describen soluciones simples a problemas específicos en el diseño de software orientado a objetos y otros ámbitos referentes al diseño de interfaces.

En la solución desarrollada fueron usados los siguientes patrones GoF:

- ✓ **Fachada (Estructural):** proporciona una interfaz unificada de alto nivel para un subsistema que oculta las interfaces de bajo nivel de las clases que lo implementan simplificando así la interacción con el subsistema. Se utiliza para proporcionar un fácil acceso a subsistemas complejos. En el diseño de la solución que se propone, la clase que utiliza como fachada es `Zend_Ext_IoC`.
- ✓ **Cadena de responsabilidad (Comportamiento):** permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada. Se evidencia al distribuir las responsabilidades.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.6 Diagrama de secuencia

Los diagramas de secuencia muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo. Estos se modelan para cada requisito funcional y contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el mismo, además de los mensajes intercambiados entre los objetos [39]. La siguiente figura muestra el diagrama de secuencia correspondiente al requisito funcional Adicionar datos de los contactos de usuario:

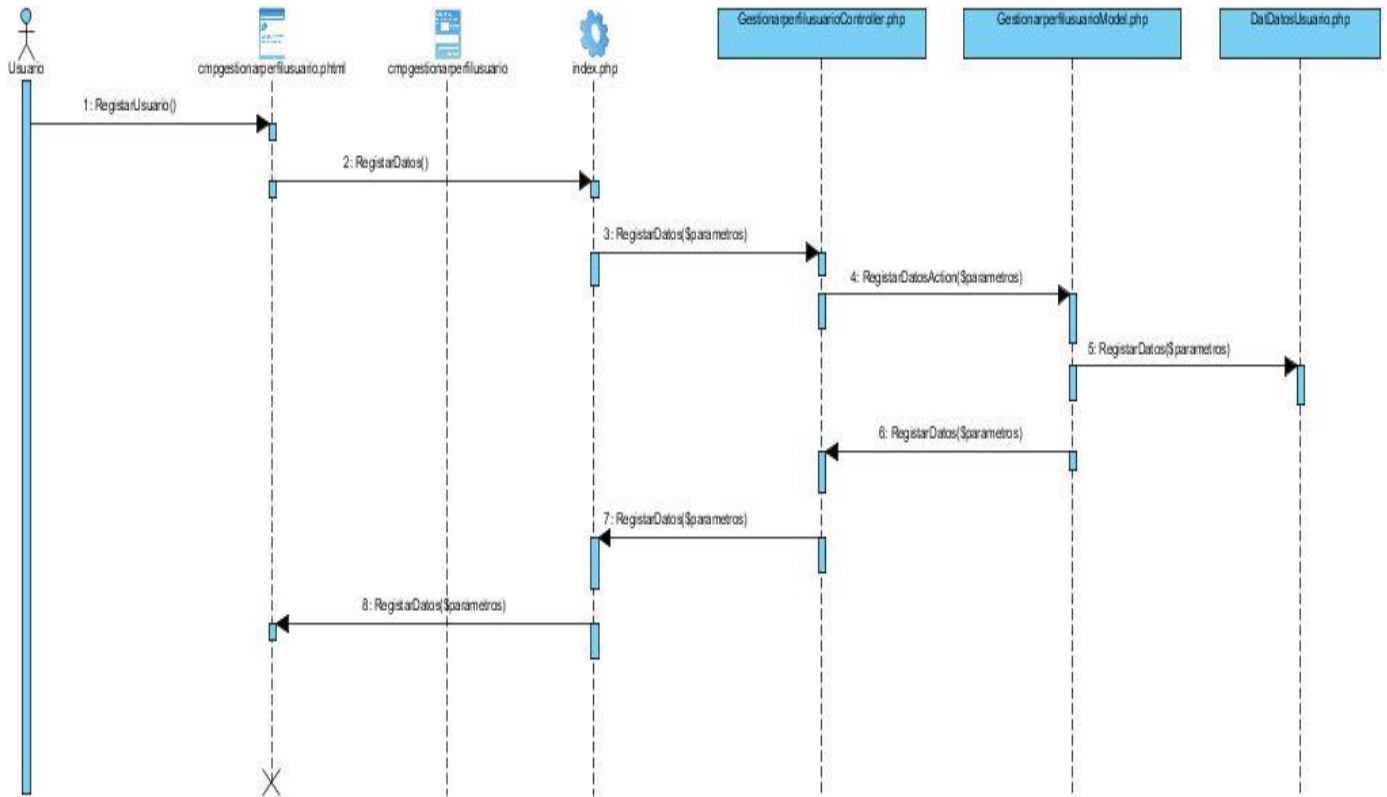


Figura 5. Diagrama de secuencia Adicionar datos de los contactos de usuario

El flujo representado en este diagrama comienza cuando el usuario da clic en la opción Editar para registrar los datos de contacto del usuario. Esta solicitud es atendida por la server page *index.php* que construye el formulario donde el usuario debe registrar los datos. Luego la clase controladora *GestionarPerfilUsuarioController.php* envía la petición a la clase modelo *GestionarPerfilUsuarioModel.php* la cual realiza la consulta a la clase de acceso a datos *DatDatosUsuario.php*. Esta a su vez actualiza la tabla *datos_usuario* donde se insertan los datos del usuario adicionado.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.7 Modelo de datos

Un modelo de datos es una especificación de conceptos usados para describir la estructura de una base de datos, estos están compuestos por entidades, atributos y relaciones. Sobre estos se pueden realizar un conjunto de operaciones que permiten especificar consultas y actualizaciones de la base de datos [40].

Con el análisis del modelo de datos de SIPAC se demostró que los datos referentes al perfil de usuario no pueden ser almacenados en las tablas ya existentes, por tanto es necesaria la creación de nuevas tablas y el uso de otras ya definidas. En las tablas creadas se almacenan los datos referentes a la configuración de los datos de usuarios, la visualización, los patrones de actividades y la aprobación. Las relaciones y datos distintivos entre ellas se describen a continuación:

- ✓ La relación existente entre la tabla *nom_nomencladores* y *dat_patron_actividad* es debido a que el perfil de usuario y los Nomencladores se encuentran en el componente Configuración del subsistema Planificación. Lo que representa que la comunicación entre ellos es mediante instancias de las clases y no de servicios.
- ✓ En las tablas *data_conf_datos_usuario* y *dat_conf_visualizacion* el idusuario se guarda como un campo y no como una llave foránea debido a que la tabla que almacena los datos de usuario (*seg_usuario*) se encuentra en el esquema *mod_seguridad* y para acceder a ella se debe realizar mediante los servicios definidos en el *loc.xml* externo.
- ✓ En la tabla *dat_conf_aprobacion* se guarda el idestado y el idusuario como campos y no como llaves foráneas debido a que la tabla que almacena los datos de usuario (*seg_usuario*) se encuentra en el esquema *mod_seguridad* y la que almacena los datos del estado (*dat_estado*) se encuentra en *mod_flujotrabajo* y para acceder a ellas se debe realizar mediante los servicios definidos en el *loc.xml* externo.
- ✓ La relación de las actividades con el nomenclador tipo de actividad se realiza mediante servicios en el *loc.xml* interno del subsistema Planificación.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

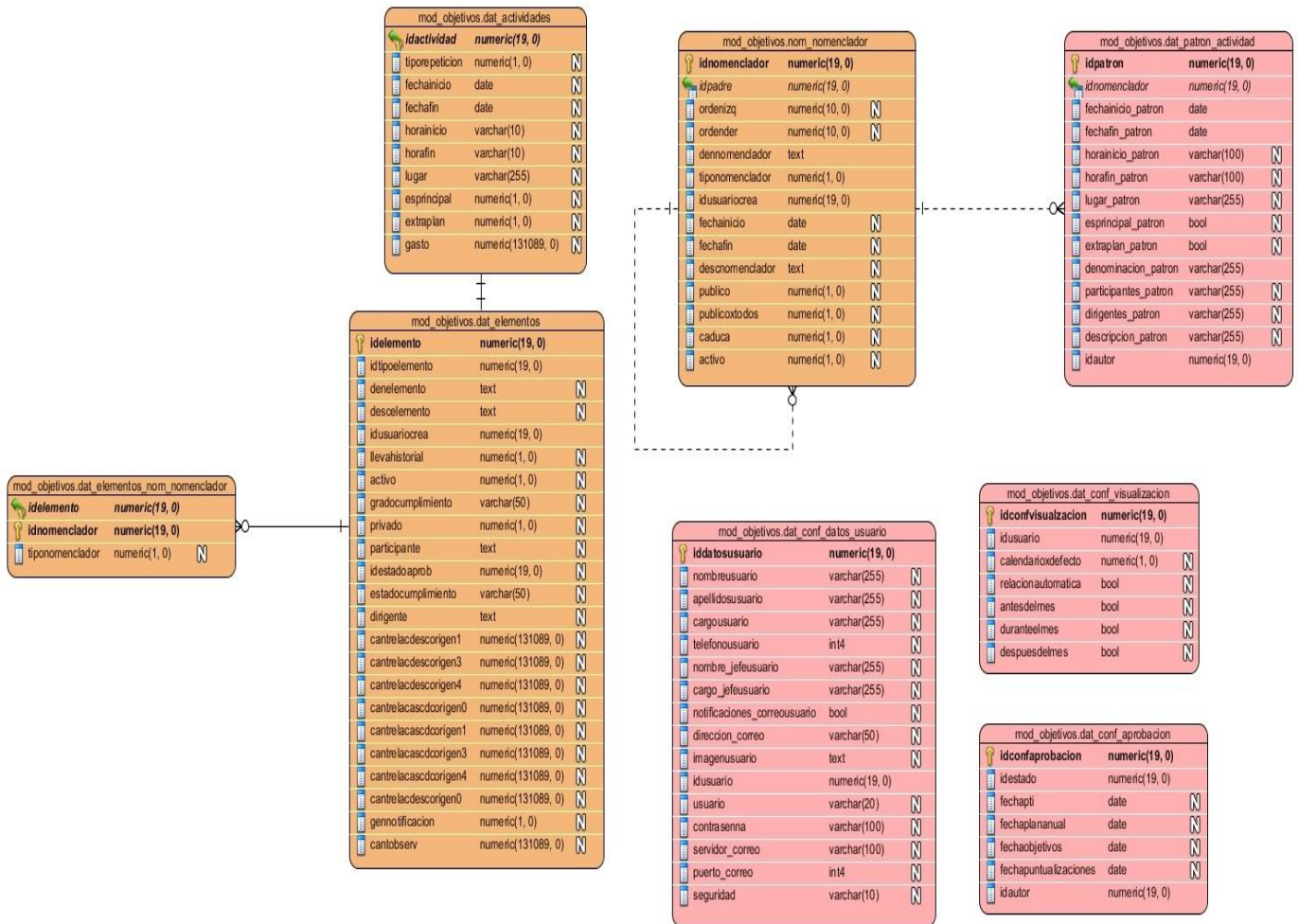


Figura 6. Modelo de datos del perfil de usuario

2.8 Conclusiones Parciales

El desarrollo de este capítulo permitió concluir lo siguiente:

- ✓ Se definieron los principales conceptos asociados al perfil de usuario, mediante el modelado del negocio, lográndose de esta forma un mejor entendimiento de la solución.
- ✓ Se identificaron 22 RF como principales preferencias del cliente y 16 RNF como cualidades fundamentales que debe cumplir el producto final mediante el proceso de captura, descripción y validación los requisitos funcionales y no funcionales.
- ✓ Se obtuvo un diseño de la solución que soporta lo especificado en las necesidades del cliente, mediante la utilización de patrones de diseño, lo que permitió un punto de partida hacia la implementación de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se describe la arquitectura de la solución desarrollada, los estándares de codificación por los que se rige el código fuente y la estrategia de integración a seguir. Además son detalladas las pruebas de caja blanca y caja negra realizadas al perfil de usuario para garantizar que cumpla con las métricas de calidad establecidas y corregir posibles errores existentes.

3.1 Implementación de la solución

La implementación del perfil de usuario como respuesta al problema existente, proporciona un producto que cumple con las especificidades del cliente y las necesidades existentes en SIPAC.

3.1.1 Estándares de código

Un estándar de código comprende la organización y aspecto físico del software con el objetivo de poseer un código fácil de entender, de mantener y que pueda ser reutilizable a lo largo del proceso de desarrollo [41].

Usar estándares de codificación y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para garantizar la calidad del software, dado que no solo define la nomenclatura de las variables, objetos, métodos y funciones sino que además incluye las reglas para el orden y legibilidad del código escrito.

Por lo anteriormente expuesto se definen 3 partes principales en un estándar de programación:

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se emplea la notación *PascalCasing*⁶, lo que posibilita que con solo leerlo se entienda el propósito de la misma.

Ejemplo: GestionarPerfilUsuario, en este caso el nombre de la clase está compuesto por dos nombres e inicialmente comienzan con letra mayúscula.

Nomenclatura según el tipo de clases

Clases controladoras: estas clases después del nombre le sigue la palabra *Controller*.

Ejemplo: GestionarPerfilUsuarioController.

⁶ Esta especifica que los indicadores y nombres de variables, métodos y funciones que están compuestos por múltiples palabras juntas, deben iniciar cada palabra con letra mayúscula.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Clases de los modelos:

Business (Negocio): las clases que se encuentran dentro del negocio después del nombre llevan la palabra *Model*.

Ejemplo: GestionarPerfilUsuarioModel.

Domain (Dominio): las clases que se encuentran dentro del dominio el nombre que reciben es el de la tabla en la base de datos.

Ejemplo: DatDatosUsuario.

Generated (Dominio base): las clases que se encuentran dentro del dominio base el nombre comienza con la palabra *Base* y seguido el nombre de la tabla en la base de datos.

Ejemplo: BaseDatDatosUsuario.

Nomenclatura de las funcionalidades y atributos

El nombre a emplear para las funciones y los atributos se escriben con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se emplea la notación *PascalCasing* similar a la antes mencionada, con la excepción de la primera letra.

Ejemplo de método:

Las funcionalidades de las clases controladoras se les asigna el nombre y seguidamente la palabra *Action*.

Ejemplo: adicionarDatosUsuarioAction ().

Ejemplo de atributo: \$parametros.

Nomenclatura de los comentarios

Los comentarios deben ser lo suficientemente claros y concisos de forma tal que se entienda el propósito de los que se está desarrollando. En caso de ser una función complicada se debe comentar su interior para lograr una mejor comprensión del código.

3.1.2 Bibliotecas para envío de correo

Como parte del estudio del estado del arte se tuvo en cuenta el estudio de bibliotecas que permiten el envío de correo electrónico, debido a que esto constituye un aspecto fundamental a tener en cuenta en la presente investigación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

PHPMailer

Es una clase php para enviar correos. Permite enviar mensajes de correo con ficheros adjuntos y mensajes en formato HTML. Con PHPMailer se pueden enviar correos vía `sendmail`⁷, `php mail ()`⁸, o con SMTP Simple Mail Transfer Protocol⁹. Se pueden usar varios servidores SMPT, lo que permite enviar un mayor número de mensajes en un tiempo menor y el envío de mensajes con varios destinatarios [42].

XPertMailer

Esta biblioteca de código abierto contiene las clases para el envío de correos electrónicos, ya sea en texto plano o en HTML. Además permite adjuntar archivos y añadir imágenes dentro del correo. Ofrece otras funciones como leer correos, encriptación, autenticación, lo cual permite que la aplicación pueda enviar y recibir correos en la forma que necesite el cliente y sin tener que depender de una plataforma en particular [43].

SwiftMailer

Biblioteca para el envío de correo de aplicaciones PHP. Al crear los mensajes los divide en dos partes una cabecera y el cuerpo del mensaje lo que facilita el envío, dado que toma la cabecera que contiene la los parámetros de configuración del correo y el cuerpo y los envía como un único mensaje. Permite establecer la prioridad de los mensajes y solicitar confirmaciones de lectura. Se puede enviar un correo a través de cualquier servidor que utilice autenticación mediante los protocolos TSL Transport Layer Security o SSL Secure Sockets Layer¹⁰ [44].

Con el estudio de las bibliotecas PHPMailer, XPertMailer y SwiftMailer se llegó a la conclusión que no se recomiendan las dos primeras para su uso en esta investigación debido a que una de sus principales desventajas es que requiere de la instalación de paquetes para la gestión de imágenes y archivos

⁷ Sendmail: es una agente de transporte de correo, soporta varios tipos de métodos de transferencia de correo y entrega, incluyendo el SMPT.

⁸ Php mail (): es una función que permite enviar correos electrónicos directamente desde un script o conjunto de sentencias.

⁹ SMPT (Protocolo para la Transferencia Simple de Correo Electrónico): en un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre dispositivos.

¹⁰ TSL (Seguridad de la Capa de Transporte) y SSL (Capa de Conexión Segura): es un protocolo mediante el cual se establece una conexión segura por medio de un canal cifrado entre el cliente y el servidor. Así el intercambio de información se realiza en un entorno seguro y libre de ataques.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

adjuntos, lo que limita su utilización en múltiples plataformas. Además PHPMailer no se ha desarrollado ni dado soporte desde 2013 y en el caso de XPertMailer su última versión 4.0.5 data desde el 2007, por lo que pueden catalogarse como tecnologías atrasadas en comparación con SwiftMailer.

En cambio SwiftMailer no necesita el uso de paquetes adicionales para su uso por lo que no está ligado a un sistema operativo o versión específica. Además cuenta con amplia documentación, lo que facilita su conocimiento y su última actualización fue generada en el año actual.

3.1.2.1 Integración de SwiftMailer

Para realizar la integración de la biblioteca SwiftMailer en el núcleo de SIPAC, primeramente se incluye el paquete `compuci.phar` que trabaja con un gestor de dependencia de `php`, dentro del paquete de la biblioteca. Luego se instalan las dependencias que necesita el SwiftMailer escribiendo el comando `php compuci.phar update` en la consola, de esta forma se prepara la biblioteca para ser usada e incluirla en la aplicación. Por último se incluye la dirección donde se encuentra el paquete de SwiftMailer dentro de la clase `CmpgestionarelementoModel` que implementa la lógica común para todos los elementos.

3.1.3 Descripción de las clases y funcionalidades de la solución

Clase controladora: clase encargada de controlar la comunicación entre la vista y el modelo. Es la responsable de realizar las diferentes funcionalidades sobre el perfil de usuario.

Tabla 3. Descripción de la clase GestionarPerfilUsuarioController

Nombre: GestionarPerfilUsuarioController	
Tipo de clase: Controladora	
Nombre	Descripción
AdicionarPatronAction ()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de controlar los datos de usuario registrados.
AdicionarConfiguracionVisualizacion ()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de controlar los datos de la configuración para la visualización de los elementos.
EnviarCorreoAction ()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de controlar

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

	la información referente al efectuarse el envío del correo.
--	---

Clase Model: clase encargada de manejar los datos persistentes dentro del perfil. Contiene el Bussines y el Domain.

Tabla 4. Descripción de la clase GestionarPerfilUsuarioModel

Nombre: GestionarPerfilUsuarioModel	
Tipo de clase: Model	
Nombre	Descripción
AdicionarPatron()	Funcionalidad encargada de adicionar los patrones de actividad, obteniéndose como resultado los patrones para el registro de actividades.
AdicionarConfiguracionVisualizacion ()	Funcionalidad encargada de guardar la configuración de la visualización especificada, obteniéndose como resultado una configuración según la preferencia del usuario.
EnviarCorreo()	Funcionalidad encargada de efectuar el envío de correo a los usuarios, obteniéndose como resultado una notificación al correo del mismo cuando ocurre una variación en un elemento de la planificación.

Clases Domain: contiene los métodos para manejar el negocio.

Tabla 5. Descripción de la clase DatDatosUsuario

Nombre: DatDatosUsuario	
Tipo de clase: Domain	
Nombre	Descripción
ExistePatron ()	Funcionalidad encargada de verificar si existe el patrón de actividad especificado para el usuario en sesión.
ExisteConfVisualizacion ()	Funcionalidad encargada de comprobar si existe una configuración de la visualización de los

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

	elementos para el usuario en sesión.
ExisteConfAprobacion ()	Funcionalidad encargada de comprobar si existe una configuración de envío a aprobación de los elementos para el usuario en sesión.
ExisteDatos ()	Funcionalidad encargada de comprobar si existen los datos de contacto para el usuario en sesión.

3.2 Perfil de usuario para SIPAC

A continuación se muestran las principales vistas con las que puede interactuar el usuario al hacer uso de la solución implementada:

Cada usuario del sistema tiene la posibilidad de visualizar su perfil donde puede editar sus datos de contacto, además de los referentes a la configuración para el envío de correo electrónico.

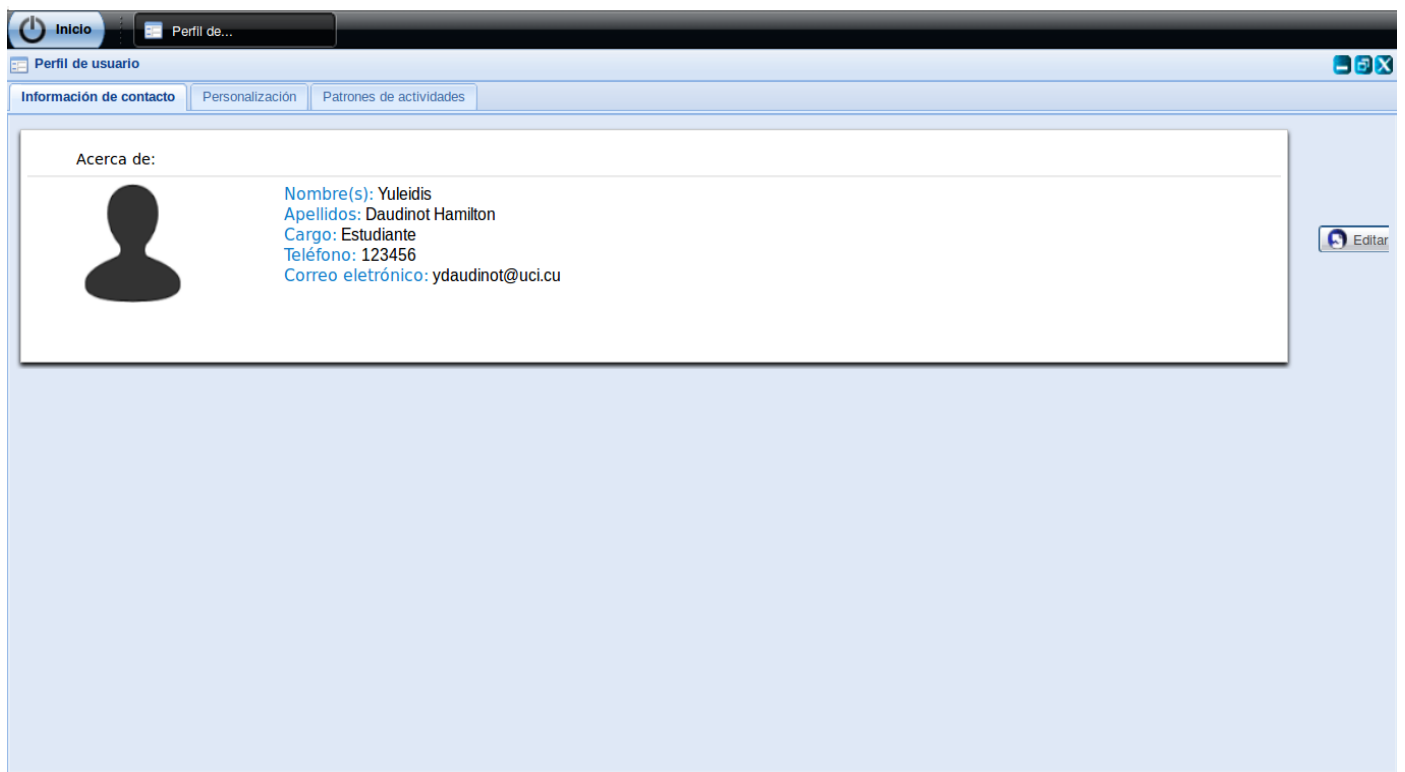


Figura 7. Interfaz de usuario Información de contacto

El usuario puede personalizar el sistema teniendo en cuenta la visualización de los elementos según el período deseado, el calendario por defecto según las vistas y la relación automática de las actividades.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Además puede enviar a aprobación según el período de tiempo definido, los elementos que estén en un estado que tenga una transición hacia el estado de envío seleccionado.

The screenshot shows a web application interface for user customization. At the top, there is a navigation bar with 'Inicio' and 'Perfil de...'. Below this, a breadcrumb trail shows 'Perfil de usuario' with sub-tabs for 'Información de contacto', 'Personalización', and 'Patrones de actividades'. The 'Personalización' tab is active. The interface is divided into two main sections: 'Visualización' and 'Envío a aprobación'. The 'Visualización' section includes a 'Calendario' dropdown set to 'Vista semanal' and a checked checkbox for 'Relación automática de las actividades'. Below this is a 'Carga de los elementos' section with three checkboxes: 'Antes del mes' (unchecked), 'Durante el mes' (checked), and 'Después del mes' (unchecked). The 'Envío a aprobación' section features a dropdown menu for 'Estado' set to 'Aprobado por el superior'. Below this is a 'Fecha de envío a aprobación' section with three date pickers: 'Objetivos' (13/05/2015), 'Plan anual' (20/05/2015), and 'Plan individual' (27/05/2015). An 'Aceptar' button is located at the bottom right of the form.

Figura 8. Interfaz de usuario Personalización

La vista de patrones de actividades le permite a los usuarios crear patrones para las actividades con frecuencia periódicas, antes de ello necesita crear un nomenclador para el tipo de actividad y luego se procede a la creación de las actividades teniendo en cuenta el patrón definido.

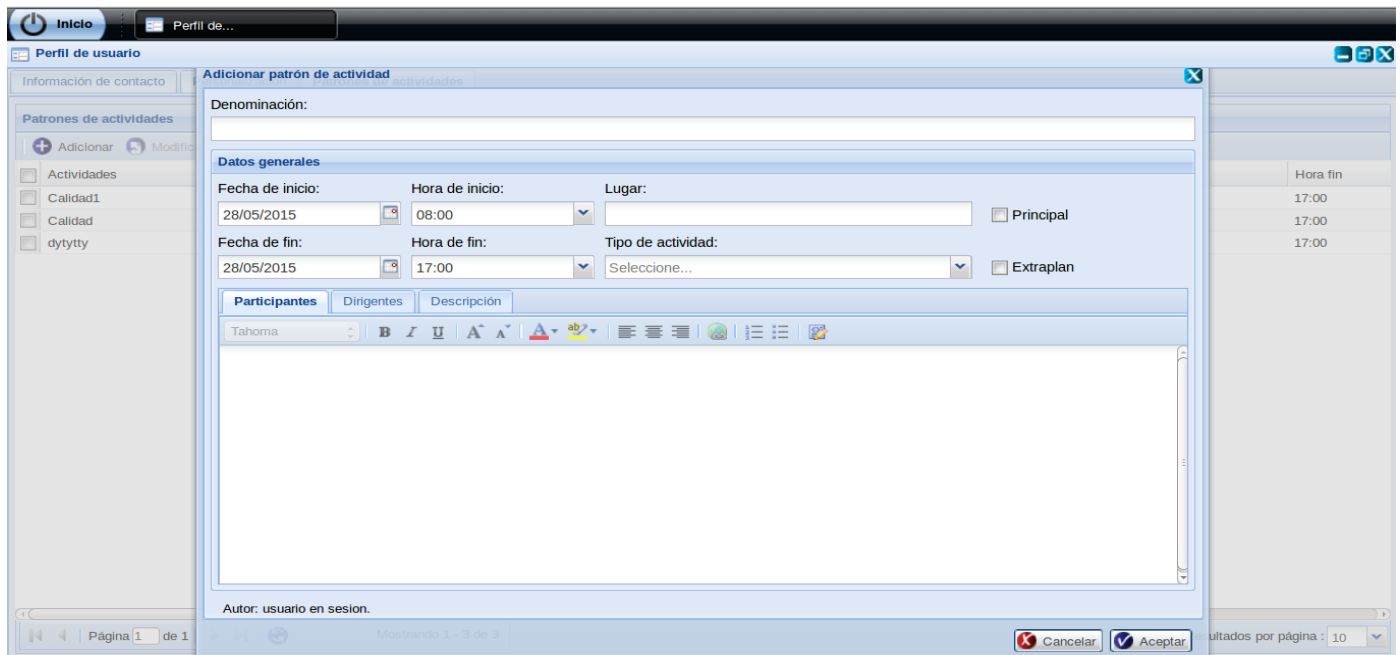


Figura 9. Interfaz de usuario Patrones de actividades

3.3 Impacto de la solución

De manera general la incorporación en el SIPAC del perfil de usuario desarrollado facilita el registro, seguimiento y control de la planificación a corto, mediano y largo plazo según establece la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros. Teniendo en cuenta que posibilita la notificación a los usuarios sobre la ocurrencia de alguna variación en sus planes de trabajo mediante el envío de correo en tiempo real, lo que posibilita el seguimiento y control del cumplimiento de los elementos de la planificación; proceso que anteriormente se controlaba accediendo directamente al sistema. Otra de las funcionalidades que tributan al seguimiento y control es el caso del envío a aprobación, dado que se personaliza el envío de los elementos según el período deseado por el usuario. Esta funcionalidad permite la configuración del sistema según la preferencia de los usuarios haciendo más flexible este proceso. Para facilitar el registro, el usuario puede definir un sistema de trabajo mediante los patrones de actividades que agilizan el proceso de elaboración del plan de los jefes y sus subordinados, haciendo flexible y adaptable el funcionamiento del sistema según el tipo de actividad. Dicho sistema de trabajo se compone de actividades que luego conforman los planes de trabajo.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.4 Validación de la solución propuesta

Para detectar y corregir los posibles errores existentes en la solución implementada se detalla a continuación la validación de la solución propuesta así como las métricas para la validación del diseño propuesto y se describen las pruebas de aplicación realizadas [30].

3.4.1 Tamaño operacional de clases (TOC)

Está dado por el número de métodos asignados a una clase [45]. Se encarga de medir la calidad de acuerdo a los siguientes atributos:

- ✓ **Responsabilidad:** es la responsabilidad asignada a una clase en un marco de modelado de un concepto, de la problemática propuesta.
- ✓ **Complejidad de implementación:** consiste en el grado de dificultad que tiene implementar un diseño de clases.
- ✓ **Reutilización:** es el grado de reutilización presente en una clase o estructura de clases, dentro de un diseño de software.

La relación entre esta métrica y los atributos de calidad antes mencionados, se distinguen en la siguiente tabla:

Tabla 6 Tamaño operacional de clase (TOC)

Atributo	Categorías	Criterios
Responsabilidad	Baja	\leq Prom
	Media	Entre Prom. y 2^* Prom
	Alta	$> 2^*$ Prom
Complejidad de implementación	Baja	\leq Prom
	Media	Entre Prom. y 2^* Prom
	Alta	$> 2^*$ Prom
Reutilización	Baja	$> 2^*$ Prom
	Media	Entre Prom. y 2^* Prom
	Alta	\leq Prom

La evaluación técnica y su influencia en los parámetros de calidad Responsabilidad, Complejidad de Implementación y Reutilización son mostradas en el anexo 4.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

En la figura 10 se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad. Este gráfico muestra un resultado satisfactorio pues el 55% de las clases poseen una baja responsabilidad. Esta característica permite que en caso de fallos, como la responsabilidad está distribuida de forma equilibrada ninguna clase es demasiado crítica como para dejar al sistema fuera de servicio.



Figura 10. Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad

En la figura 11 se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo complejidad de implementación. Este gráfico muestra un resultado satisfactorio pues el 55% de las clases poseen una baja complejidad de implementación. Esta característica permite mejorar el mantenimiento y soporte de estas clases.

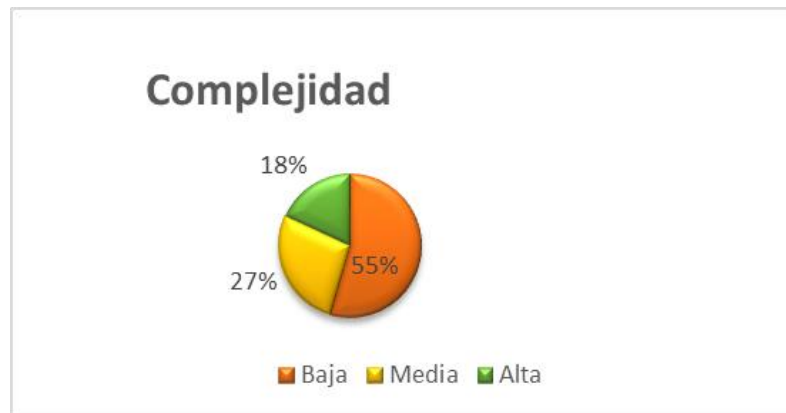


Figura 11. Resultados de la evaluación de la métrica TOC para el atributo Complejidad

La figura 12 muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización. Queda demostrado que el diseño de la solución es eficiente pues solamente el 18% del total de las clases poseen una baja reutilización.



Figura 12. Resultados de la evaluación de la métrica TOC para el atributo Reutilización

Luego de realizado un análisis de los resultados obtenidos para los atributos de la métrica TOC, se observa que la mayoría de las clases que conforman la solución perfil de usuario para los atributos responsabilidad y complejidad están dentro de la categoría media y baja para un 45% del total. Esto demuestra que las clases cuentan con un nivel medio de dependencias y pocos procedimientos en su implementación, mientras que el atributo reutilización cuenta con igual por ciento en las categorías alta y media mostrando así que el módulo cuenta con una elevada reutilización. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

3.4.2 Relación entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otra [45]. Esta se encarga de medir la calidad de acuerdo a los siguientes atributos:

- ✓ **Acoplamiento:** representa el grado de dependencia o interconexión de una clase o estructura de clases, con otras.
- ✓ **Complejidad del mantenimiento:** consiste en el grado de esfuerzo a realizar para desarrollar un arreglo, una mejora o una rectificación de un error del software. Este puede influir indirectamente, pero fuertemente en los costos y planificación del proyecto.
- ✓ **Cantidad de pruebas:** es el número o grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.
- ✓ **Reutilización:** es el grado de reutilización presente en una clase o estructura de clases, dentro de un diseño de software.

La relación entre esta métrica y los atributos de calidad antes mencionados, se distinguen en la siguiente tabla:

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Tabla 7. Relación entre clase (RC)

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

La evaluación técnica y su influencia en los parámetros de calidad Acoplamiento, Complejidad de mantenimiento, Cantidad de pruebas y Reutilización son mostradas en el anexo 4.

La figura 13 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo acoplamiento. Se evidencia un bajo acoplamiento entre las clases pues el 73% de las clases presentan una dependencia con otra. Este resultado es muy favorable para el diseño de la funcionalidad pues al existir poca dependencia entre las clases aumenta el grado de reutilización de la misma.

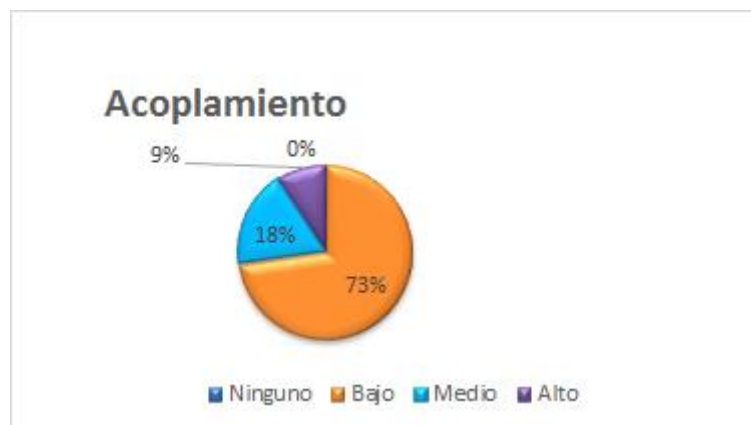


Figura 13. Resultados de la evaluación de la métrica RC para el atributo Acoplamiento

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

La figura 14 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento. El gráfico refleja un resultado aceptable del atributo pues el 91% de las clases presentan una baja complejidad de mantenimiento.



Figura 14. Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento

La figura 15 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas.



Figura 15. Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas

La figura 16 muestra la representación de la incidencia de los resultados de la evaluación del atributo reutilización. Esto evidencia que el 91% de las clases poseen una alta reutilización lo que es un factor fundamental que debe ser tenido en cuenta en el desarrollo de software.



Figura 16. Resultados de la evaluación de la métrica RC para el atributo Reutilización

Al analizar los resultados obtenidos de la métrica RC, se puede concluir que el diseño de la solución perfil de usuario tiene una calidad aceptable. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 73% de las clases el nivel de acoplamiento es bajo. La complejidad de mantenimiento y la cantidad de pruebas son bajas a un 91%, mientras que, el atributo reutilización cuenta con igual por ciento en la categoría alta, lo que representa valores favorables para el diseño realizado.

3.5 Pruebas de software

Las pruebas de software se pueden definir como el proceso de verificar el comportamiento de una programa ante un conjunto de casos de prueba, contra el comportamiento esperado [46].

Por tanto las pruebas de software son un conjunto de actividades dentro del desarrollo de software que dependiendo del tipo de pruebas, estas actividades pueden ser ejecutadas en cualquier momento en el proceso de desarrollo. Son usadas con el fin de detectar posibles errores de implementación, calidad o usabilidad de un programa.

Los niveles de pruebas que se pueden aplicar a un software son descritos a continuación:

- ✓ **Pruebas de unitarias:** son usadas para comprobar el correcto funcionamiento de un módulo de código. Con esta prueba se comprueba que cada módulo funcione correctamente por separado.
- ✓ **Pruebas de integración:** en estas pruebas se combinan módulos individuales de forma tal que sean probados como un grupo. Se usan para detectar errores de interfaces y relaciones entre componentes.
- ✓ **Pruebas de sistema:** con estas pruebas se trata de asegurar la correcta navegación dentro del sistema, ingreso, procesamiento y recuperación de datos. Son usadas para detectar fallas en el cubrimiento de los requisitos.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

- ✓ **Pruebas de aceptación:** son desarrolladas y ejecutadas con el cliente para determinar si el producto satisface sus criterios o requisitos. Estas son las únicas pruebas que se realizan directamente con el cliente, a diferencia de las restantes.

Métodos de prueba

Un método de prueba no es más que un procedimiento definitivo que produce un resultado de una prueba. Al aplicarle pruebas a un determinado producto se pueden detectar diversos tipos de errores al usar conjuntos de casos de pruebas¹¹.

Las pruebas usadas para asegurar la calidad de la solución y determinar cómo y en qué sentido el producto final cumple con los intereses del cliente, son las pruebas de caja blanca y caja negra, para ello se trazan como objetivos los siguientes:

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

3.5.1 Descripción y aplicación de la Prueba de Caja Blanca o Estructural

Descripción

Se denomina pruebas de caja blanca a las pruebas de software que se realizan sobre las funciones internas de un módulo. Se realiza un examen minucioso de los detalles de procedimientos, comprobando los caminos lógicos del programa, los bucles y condiciones y examinando el estado del programa.

Las técnicas usadas para ejecutar dichas pruebas son las descritas a continuación:

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

¹¹ Casos de prueba: especifican una forma de probar la solución, incluye: la entrada, las condiciones bajo las cuales ha de probarse y los resultados esperados.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

- ✓ **Prueba de flujo de datos:** se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.
- ✓ **Prueba de bucles:** se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.

Aplicación

Con el objetivo de valorar la calidad con la que se llevó a cabo la implementación de la solución perfil de usuario, fue necesario aplicar una de las técnicas descritas anteriormente: **la prueba de camino básico**.

Para ello fue necesario seguir los siguientes pasos básicos:

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

Para dar cumplimiento a los anteriores pasos básicos se enumeran cada una de las sentencias de código de uno de los procedimientos de la clase *CmpgestionarelementoModel*, específicamente la funcionalidad *EnviarCorreo ()*.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

```
///  
function EnviarCorreo($datos_usuario_correo,$notificacion)  
{  
    $usuario_correo = $datos_usuario_correo[0]['usuario_correo']; 1  
    $contrasenna_correo = $datos_usuario_correo[0]['contrasenna_correo']; 1  
    $servidor_correo = $datos_usuario_correo[0]['servidor_correo']; 1  
    $puerto_correo = $datos_usuario_correo[0]['puerto_correo']; 1  
    $seguridad_correo = $datos_usuario_correo[0]['seguridad_correo']; 1  
    $correocompleto = $datos_usuario_correo[0]['correocompleto']; 1  
    $seguridad_correo=$this->setearSeguridadCorreo($seguridad_correo); 1  
    $cuerpo_correo=$notificacion[0]['descripcion']; 1  
    $punto='.'; 1  
    $cantidad = substr_count($servidor_correo, $punto); 1  
    if($cantidad==2){ 2  
        $correo=explode($punto, $servidor_correo); 3  
        $dominio=$correo[1].'.'.$correo[2]; 3  
        $correocompleto = $usuario_correo.'@'.$dominio; 3  
        $transport = Swift_SmtpTransport::newInstance($servidor_correo, $puerto_correo, $seguridad_correo) 3  
        ->setUsername($correocompleto) 3  
        ->setPassword($contrasenna_correo); 3  
        $mailer = Swift_Mailer::newInstance($transport); 3  
        $message = Swift_Message::newInstance('Notificación SIPAC') 3  
        ->setFrom(array($correocompleto => $usuario_correo)) 3  
        ->setTo(array($correocompleto => $usuario_correo)) 3  
        ->setBody($cuerpo_correo); 3  
        if ($recipients = $mailer->send($message, $failures)) { 4  
            $exito=1; 5  
        } 4 } else { 6  
            $exito=0; 7  
        } 6  
    } 2  
    else{ 8  
        $exito=2; 9  
    } 8  
    return $exito; 10  
}
```

Figura 17. Funcionalidad EnviarCorreo ()

Luego del paso anterior, es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, en ese caso:

- ✓ **Nodo:** círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, un nodo en sí puede representar un proceso, una secuencia de procesos o una sentencia de decisión. Los nodos que no están asociados se utiliza al inicio y final del grafo.
- ✓ **Aristas:** saetas a través de las cuales se unen los nodos y constituyen el flujo de control del procedimiento.
- ✓ **Regiones:** son las áreas delimitadas por las aristas y nodos.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

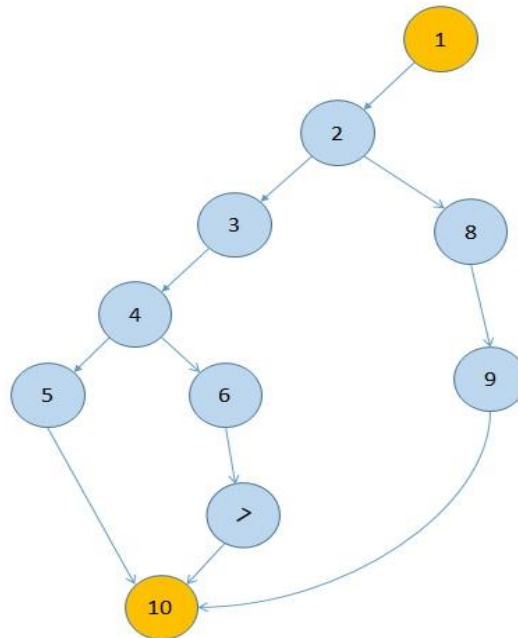


Figura 18. Grafo de flujo asociado al algoritmo EnviarCorreo ()

Una vez construido el grafo de flujo asociado al procedimiento anterior se procede a determinar la complejidad ciclomática, dicho cálculo es necesario efectuarlo mediante tres vías o fórmulas de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

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

Siendo "A" la cantidad total de aristas y "N" la cantidad de nodos.

$$V(G) = (11 - 10) + 2$$

$$V(G) = 3.$$

2. $V(G) = P + 1$

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

$$V(G) = 2 + 1$$

$$V(G) = 3.$$

3. $V(G) = R$

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

$$V(G) = 3.$$

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

El cálculo efectuado mediante las fórmulas antes presentadas muestra una complejidad ciclomática de valor 3, de manera que existen tres posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Seguidamente se especifican los caminos básicos que puede tomar el algoritmo durante su ejecución:

Camino básico 1: 1-2-3-4-5-10.

Camino básico 2: 1-2-3-4-6-7-10.

Camino básico 3: 1-2-8-9-10.

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo.

Tabla 8. Caso de prueba para el camino básico 1

Camino básico 1: 1-2-3-4-5-10	
Descripción	Se envía cuando ocurre una acción sobre algún elemento de la planificación.
Condición de ejecución	Se ejecuta cuando se genera una notificación.
Entrada	<code>\$usuario_correo = mtrosales</code> <code>\$contrasenna_correo = encriptado en base a 64 bits</code> <code>\$servidor_correo = smtp.uci.cu</code> <code>\$puerto_correo = 25</code> <code>\$seguridad_correo = tls</code> <code>\$correocompleto = mtrosales@uci.cu</code> <code>\$cuerpo_correo = La actividad de Código de Acceso Rápido (CAR): "9000003406 y denominación: Reunión" ha sido cambiada de estado.</code>
Resultado esperado	El usuario recibe un correo notificando la acción acometida sobre el elemento de la planificación.

Tabla 9. Caso de prueba para el camino básico 2

Camino básico 2: 1-2-4-6-7-10	
Descripción	Se envía cuando ocurre una acción sobre algún

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

	elemento de la planificación.
Condición de ejecución	Se ejecuta cuando se genera una notificación.
Entrada	<pre> \$usuario_correo = ydaudinot \$contrasenna_correo = encriptado \$servidor_correo = smtp.uci.cu \$puerto_correo = 25 \$seguridad_correo = tcp \$correocompleto = ydaudinot@.uci.cu \$cuerpo_correo = La actividad de Código de Acceso Rápido (CAR): "9000003368 y denominación: Chequeo de proyecto" ha sido modificada. </pre>
Resultado esperado	El usuario recibe un correo notificando la acción acometida sobre el elemento de la planificación.

Tabla 10. Caso de prueba para el camino básico 3

Camino básico 3: 1-2-8-9-10	
Descripción	Se envía cuando ocurre una acción sobre algún elemento de la planificación.
Condición de ejecución	Se ejecuta cuando se genera una notificación.
Entrada	<pre> \$usuario_correo = mtrosales \$contrasenna_correo = encriptado \$servidor_correo = smtp.uci.cu \$puerto_correo = 25 \$seguridad_correo = tcp \$correocompleto = mtrosales@uci.cu \$cuerpo_correo = ∅ </pre>
Resultado esperado	El usuario no recibe el correo, el sistema muestra un mensaje de error.

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

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.5.2 Descripción y aplicación de la Prueba de Caja Negra o Funcional

Descripción

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

En esencia permite encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a la base de datos externa.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y fin.

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

- ✓ **Partición de equivalencia:** esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ **Análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ **Grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Aplicación

A continuación se aplica la prueba de partición de equivalencia como parte de las pruebas de caja negra realizadas al RF Adicionar datos de los contactos de usuario. Para ello se definieron un conjunto de estados válidos y no válidos para las condiciones de entrada del sistema.

Tabla 11. Caso de prueba de caja negra para validar el RF Adicionar datos de los contactos de usuario

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Adicionar datos de los contactos de	1. El sistema debe permitir que el usuario al seleccionar	E.P 1.1 Flujo básico de eventos.	1. El usuario accede desde el menú Inicio/Configuración/Perfil de usuario/Información de contacto.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

usuario.	la opción Editar, adicione los datos de usuario.		<ol style="list-style-type: none"> 2. El usuario presiona el botón Editar. 3. Registra los datos de contacto y da clic en el botón Aceptar. 4. Se adicionan los datos.
		E.P 1.2 Se introducen datos incorrectos.	<ol style="list-style-type: none"> 1. El usuario accede desde el menú Inicio/Configuración/Perfil de usuario/Información de contacto. 2. El usuario presiona el botón Editar y registra caracteres numéricos en el campo de Nombre, da clic en el botón Aceptar. 3. El sistema no permite introducir caracteres diferentes de letras.
		E.P 1.3 Existen campos vacíos.	<ol style="list-style-type: none"> 1. El usuario accede desde el menú Inicio/Configuración/Perfil de usuario/Información de contacto. 2. El usuario presiona el botón Editar, deja campos vacíos y da clic en el botón Aceptar. 3. El sistema muestra un mensaje indicando que existen campos vacíos y debe permitir corregirlos.

Los juegos de datos a probar están descritos en el artefacto Diseño de casos de prueba del expediente de proyecto SIPAC 2.1.

3.6 Resultados de las pruebas aplicadas

Luego de aplicados los métodos de pruebas a las funcionalidades implementadas, se pudo concluir que los resultados obtenidos han sido satisfactorios desde el punto de vista funcional. En una primera iteración las no conformidades detectadas fueron un total de 3, de ellas 1 significativa y 2 no significativas, las cuales fueron debidamente atendidas para una segunda iteración, donde no se detectaron no conformidades (ver figura 19) en aras de lograr el correcto comportamiento de la solución desarrollada ante diferentes situaciones. Emitiéndose de esta forma el Acta de aceptación del cliente (ver anexo 6).

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

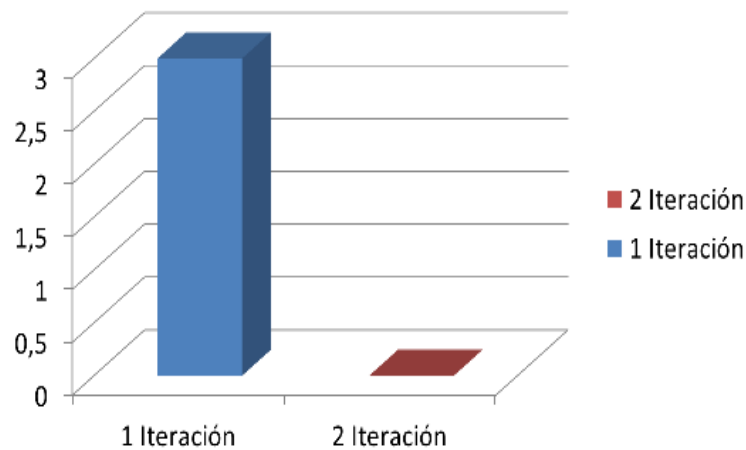


Figura 19. No conformidades detectadas durante las iteraciones de pruebas realizadas

3.7 Conclusiones Parciales

El desarrollo del presente capítulo permitió concluir lo siguiente:

- ✓ Se realizó un correcto diseño de la solución teniendo en cuenta aspectos como: complejidad de implementación, responsabilidad y reutilización de las clases, comprobado mediante la aplicación de las métricas Tamaño operacional de clase y Relaciones entre clases.
- ✓ Se desarrolló una solución correcta desde el punto de vista funcional, validado mediante la aplicación de pruebas de caja blanca y caja negra como se expresa en el Acta de liberación (ver anexo 7) y que cumpla con los requisitos del cliente como se refleja en el Acta de aceptación (ver anexo 6).

CONCLUSIONES GENERALES

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

- ✓ Se evidenció la necesidad de desarrollar el perfil de usuario para el SIPAC con el uso de la biblioteca SwiftMailer mediante la elaboración del marco teórico y el estudio del estado del arte.
- ✓ Se identificó y describió la solución a implementar teniendo en cuenta las necesidades del cliente mediante la elaboración de los artefactos que define la metodología AUP.
- ✓ Se obtuvo la solución perfil de usuario que cuenta con un correcto funcionamiento, validado mediante la aplicación de pruebas de caja blanca y caja negra como se expresa en el Acta de liberación y que cumple con los requisitos del cliente como se refleja en el Acta de aceptación.

RECOMENDACIONES

Al haber concluido el presente trabajo de diploma y el haber cumplido los objetivos trazados con el mismo, se recomienda:

- ✓ Encapsular la funcionalidad para el envío de correo en una solución independiente donde solo el administrador del sistema sea el encargado de gestionar estos datos, abstrayendo al usuario final de configuraciones que le son poco familiares.
- ✓ Ampliar la gestión del sistema de trabajo incluyendo funcionalidades para la caracterización de: norma de gastos por actividades y planes por categoría del plan.

REFERENCIAS BIBLIOGRÁFICAS

1. Robbins, S.P.y.C., M, *Administración*, ed. 8. 2005, México: Pearson Educación.
2. *Instrucción No. 1 del Presidente de los Consejos de Estado y de Ministros*. septiembre 2011.
3. Zayas, C.A., *Métodos de la investigación*. 1995, Santiago de Cuba
4. SugarCRM, *Información tecnológica*, in *Manual de Administración de SugarCRM*.
5. SugarCRM. [cited 2014 8 de diciembre]; Available from: <http://www.ecbloguer.com/marketingdigital/?p=130>.
6. PlanningPM, *Información tecnológica*, in *PlanningPME-Guía de usuario*.
7. *PlanningPME*. [cited 2014 2 de diciembre]; Available from: <http://www.planningpme.es/>.
8. SAP, *Información tecnológica*, in *Manual de usuarioSAP*.
9. Campos, O.A.M., *CMMI-Development V1.2 Seminario de CMMI*. 046_CMMI02-Seminario General CMMI 0801020 v2.3.pdf. Junio, 2008.
10. *Metodología*, in *Metodología de desarrollo para la actividad productiva de la UCI*.
11. *Lenguaje de Modelado*. [cited 2014 11 de diciembre]; Available from: <http://www.cientec.com/analisis/ana-uml.html> , <http://www.docirs.cl/uml.htm>.
12. *Introducción a HTML*. [cited 2014 11 de diciembre]; Available from: <http://eusalud.uninet.edu/Cursos/doc99/INTROHTML.html>.
13. *Manual de CSS, hojas de estilo*. [cited 2014 11 de diciembre]; Available from: <http://www.desarrolloweb.com/manuales/manual-css-hojas-de-estilo.html>.
14. *Definición de JavaScript*. [cited 2014 11 de diciembre]; Available from: <http://www.pergaminovirtual.com.ar/definicion/JavaScript.html>.
15. *PHP*. [cited 2014 11 de diciembre]; Available from: <http://www.php.net/>.
16. Chaux, H.R., *Marco de trabajo para aplicaciones web de código abierto en instituciones universitarias*.
17. CEIGE, *Vista Entorno de Desarrollo Tecnológico*, in *Proyecto Marco de trabajo para el desarrollo de aplicaciones web de gestión: SAUXE*.
18. *Introducción a Zend Framework*. [cited 2014 11 de diciembre]; Available from: <http://alemohamad.com/tutorial-zend-framework/>.
19. *Doctrine 1.2 ORM Manual*. [cited 2014 11 de diciembre]; Available from: <http://docs.doctrine-project.org/projects/doctrine1/en/latest/en/manual/introduction.html#aboutthis-version>.
20. *Curso ExtJS*. Buenas tareas [cited 2014 11 de diciembre]; Available from: <http://www.buenastareas.com/ensayos/Curso-Extjs/3156528.html>.

REFERENCIAS BIBLIOGRÁFICAS

21. *Introducción a AJAX.* [cited 2014 11 de diciembre]; Available from: http://librosweb.es/ajax/capitulo_1.html.
22. *Visual Paradigm.* [cited 2014 11 de diciembre]; Available from: <http://www.visual-paradigm.com/whats-new/>.
23. *Apache Subversion* [cited 2014 11 de diciembre]; Available from: <http://subversion.apache.org/>.
24. *NetBeans IDE.* [cited 2014 11 de diciembre]; Available from: <https://netbeans.org/features/index.html>.
25. *Apache - Servidor HTTP.* [cited 2014 11 de diciembre]; Available from: <http://httpd.apache.org/docs/2.0/es/invoking.html>.
26. Artola, A.R., *Vista Entorno de Desarrollo Tecnológico.*
27. *Mozilla Firefox.* [cited 2015 9 de febrero]; Available from: http://www.ecured.cu/index.php/Mozilla_Firefox , <http://firefoxmania.uci.cu/acerca-de-firefoxmania/>.
28. Hernández, D.A., *Perfil de usuario. Facultad de Humanidades, Universidad Nacional de Mar del Plata.*
29. *Definición de modelo conceptual, in ModeloConceptual_BN.*
30. Pressman, R.S., *Ingeniería del Software.* Quinta Edición. McGraw Hil ed. Un enfoque práctico. s.l. 2005.
31. *Capítulo 2 Técnicas para identificar requerimientos* 3. Metodología Gestión de Requerimientos.
32. *Especificación de requisitos según el estándar de IEEE 830.* IEEE Std. 830-1998.
33. *Capítulo 3 Definición de Requerimientos.* Metodología Gestión de Requerimientos.
34. *CEIGE, Requisitos No Funcionales, in Requisitos No Funcionales.*
35. *Modelo de clases.* [cited 2015 10 de abril]; Available from: <http://users.dcc.uchile.cl/~psalinas/uml/modelo.html>.
36. *EcuRed.* [cited 2015 30 de marzo]; Available from: http://www.ecured.cu/index.php/Patr%C3%B3n_de_dise%C3%B1o_de_software.
37. Hernán, M.V.A.y., *Fundamentos de Ingeniería de Software.* Fundamentos de Ingeniería de Software.pdf. Departamento de Informática, Universidad Técnica Federico Santa María.
38. *Clasificación de patrones.*
39. *Diagrama de secuencia UML.*
40. *Modelos de datos ER-UML-relacional.* [cited 2015 16 de marzo]; Available from: <http://personales.unican.es/zorrillm/PDFs/Docencia/Master/02%20-%20Modelos%20de%20datos%20ER-UML-relacional.pdf>.

REFERENCIAS BIBLIOGRÁFICAS

41. Calleja, M.A., *Estándares de codificación*.
42. *Uso de la clase phpMailer*. Programación en castellano.
43. *XPertMailer Advanced PHP Mail Engine*. [cited 2015 25 de mayo]; Available from: <http://www.xpertmailer.com/>.
44. Paniagua, A. *Conexiones Razonables*. 7 de diciembre de 2007 [cited 2015 25 de mayo]; Available from: <http://conexionesrazonables.blogspot.com/2007/12/enviar-correos-desde-php-mediante-gmail.html>.
45. Baryolo, O.G., *Solución informática de autorización en entornos multientidad y multisistema*. 2010, Universidad de las Ciencias Informáticas. La Habana.
46. S, J.Z.P.d.S., *Ingeniería de software con énfasis en pruebas*.

ANEXOS

Anexo 1. Figura ampliada de la *Vista* del patrón Modelo-Vista-Controlador del diagrama de clases del diseño.

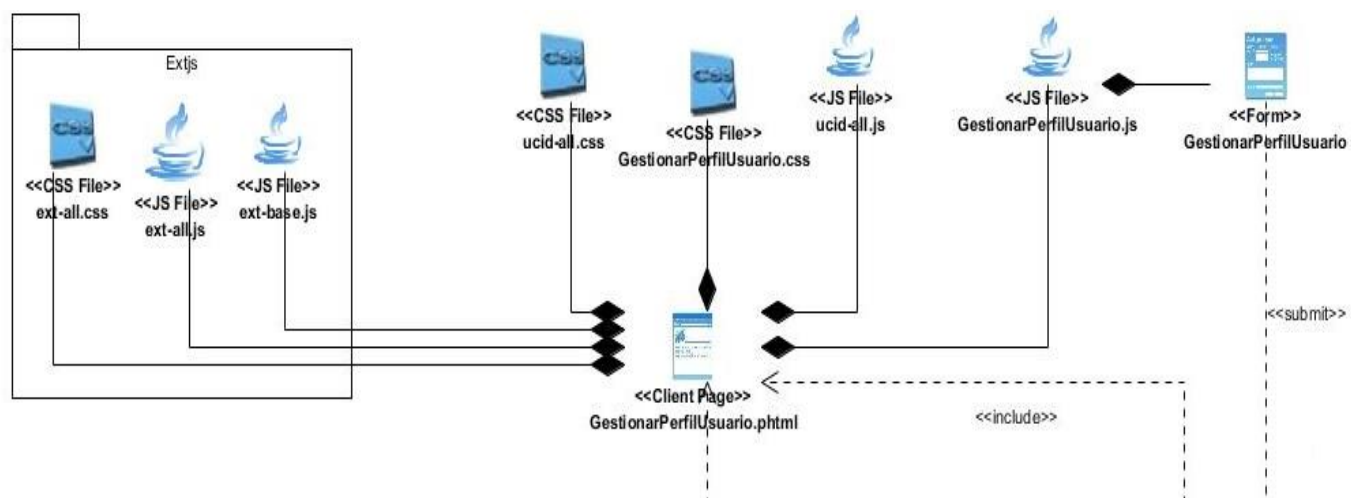


Figura 4.1 Vista del patrón arquitectónico Modelo-Vista-Controlador del diseño de clases

Anexo 2. Figura ampliada del *Controlador* del patrón arquitectónico Modelo-Vista-Controlador del diagrama de clases del diseño.

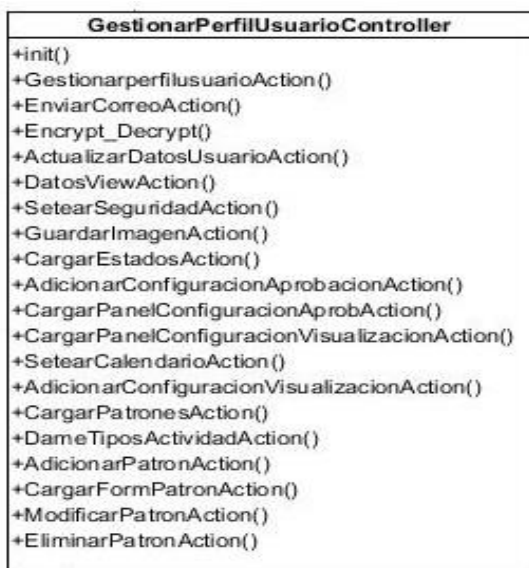


Figura 4.2 Controlador del patrón arquitectónico Modelo-Vista-Controlador del diseño de clases

Anexo 3. Figura ampliada del *Modelo* del patrón arquitectónico Modelo-Vista-Controlador del diagrama de clases del diseño.

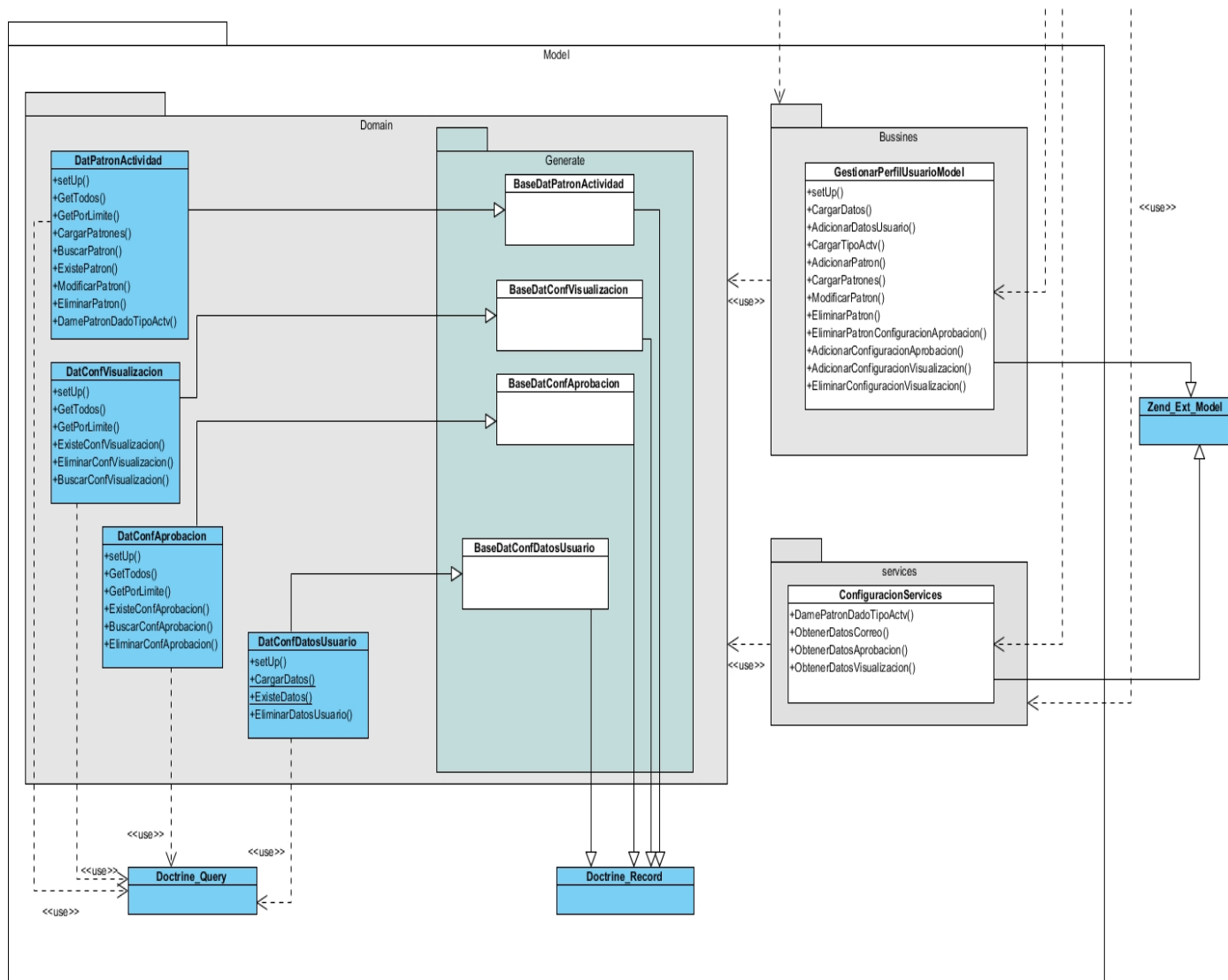


Figura 4.3 Modelo del patrón arquitectónico Modelo-Vista-Controlador del diseño de clases

Anexo 4. Resultados de las métricas (TOC y RC) para la validación de diseño.

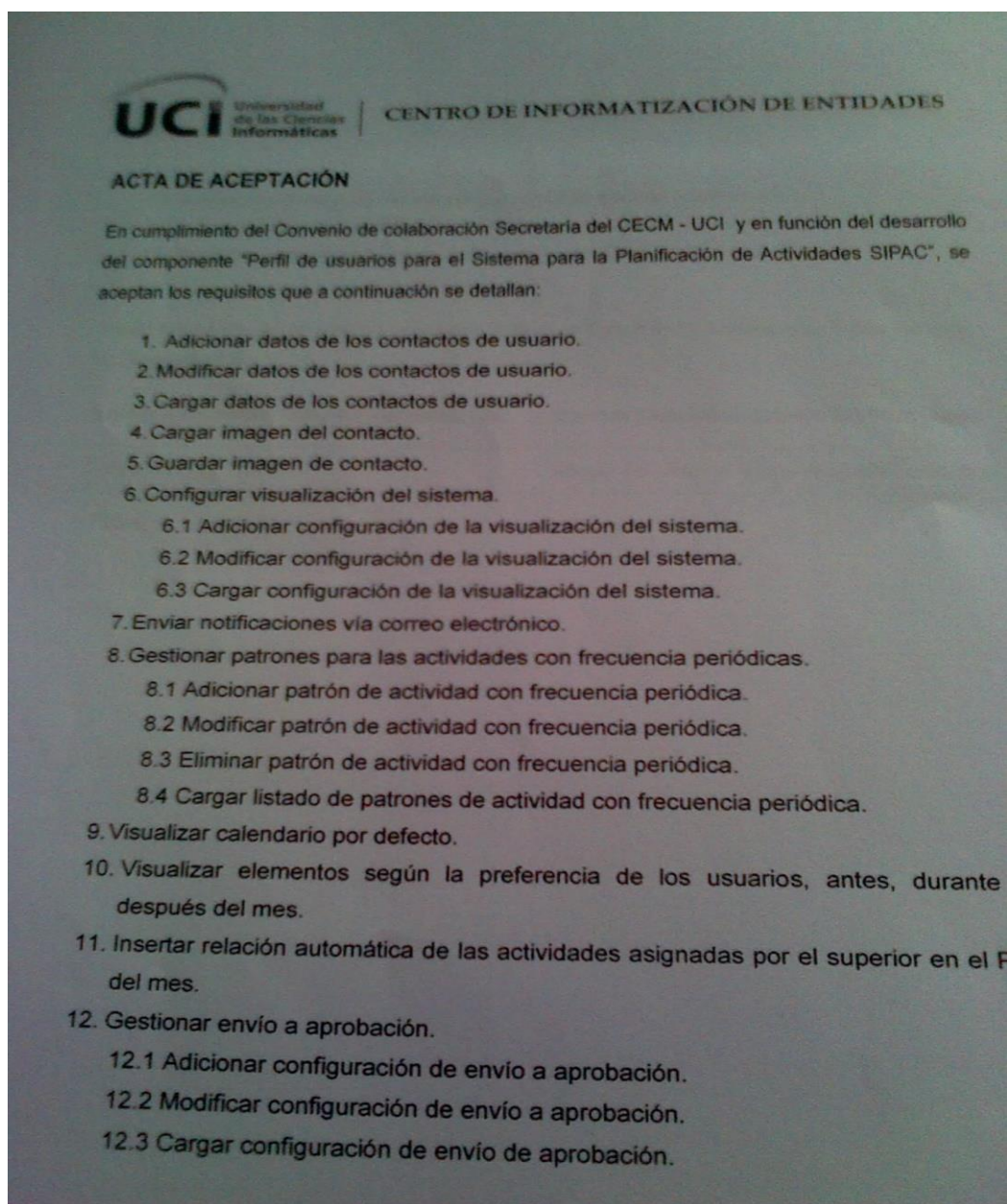
Tabla 12. Resultados de la aplicación de la métrica Tamaño Operacional de Clases (TOC)

Clases	Cantidad de procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
GestionarPerfilUsuarioController	20	Alta	Alta	Baja
GestionarPerfilUsuarioModel	12	Media	Media	Media

DatConfDatosUsuario	4	Baja	Baja	Alta
DatConfVisualizacion	6	Baja	Baja	Alta
DatPatronActividad	9	Media	Media	Media
DatConfAprobacion	6	Baja	Baja	Alta
BaseDatConfDatosUsuario	4	Baja	Baja	Alta
BaseDatConfVisualizacion	6	Baja	Baja	Alta
BaseDatPatronActividad	9	Media	Media	Media
BaseDatConfAprobacion	6	Baja	Baja	Alta
ConfiguracionServices	24	Alta	Alta	Baja

Tabla 13. Resultados de la aplicación de la métrica Relación entre Clases (RC)

Clases	Cantidad de relaciones	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
GestionarPerfilUsuarioController	3	Alto	Media	Media	Media
GestionarPerfilUsuarioModel	2	Medio	Baja	Alta	Baja
DatConfDatosUsuario	1	Bajo	Baja	Alta	Baja
DatConfVisualizacion	1	Bajo	Baja	Alta	Baja
DatPatronActividad	1	Bajo	Baja	Alta	Baja
DatConfAprobacion	1	Bajo	Baja	Alta	Baja
BaseDatConfDatosUsuario	1	Bajo	Baja	Alta	Baja
BaseDatConfVisualizacion	1	Bajo	Baja	Alta	Baja
BaseDatPatronActividad	1	Bajo	Baja	Alta	Baja
BaseDatConfAprobacion	1	Bajo	Baja	Alta	Baja
ConfiguracionServices	2	Medio	Baja	Alta	Baja

Anexo 5. Acta de aceptación de requisitos.

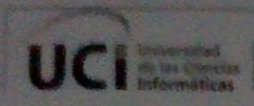
UCI Universidad de las Ciencias Informáticas | CENTRO DE INFORMATIZACIÓN DE ENTIDAD

13. Enviar aprobación de los elementos según el período establecido.
14. Adicionar actividad partiendo de un patrón de actividades configurado.
15. Modificar actividad partiendo de un patrón de actividades configurado.

Y para que así conste se suscribe la presente Acta en la Ciudad de La Habana a los 5 días d
de Marzo de 2015.

Nombre y Apellidos: Mairelys Fernández Glez.	Nombre y Apellidos: Orlando Martine
Cargo: Jefe de Proyecto.	Cargo: 2do Jefe del Grupo de Planific
Firma:	Firma: 

Anexo 6. Acta de aceptación del cliente.

 **CENTRO DE INFORMATIZACIÓN DE ENTIDADES**

ACTA DE ACEPTACIÓN

En cumplimiento del **Convenio de colaboración** Secretaría del CECM - UCI y en función del desarrollo del componente "Perfil de usuario para el Sistema para la Planificación de Actividades SIPAC", se considera que:

De manera general la incorporación en el SIPAC del perfil de usuario desarrollado facilita el registro, seguimiento y control de la planificación a corto, mediano y largo plazo según establece la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros; teniendo en cuenta que posibilita la notificación a los usuarios sobre la ocurrencia de alguna variación en sus planes de trabajo mediante el envío de correo en tiempo real, así como la definición de un sistema de trabajo que agiliza el proceso de elaboración del plan de los jefes y sus subordinados. Se especifican además patrones de actividades con frecuencia periódicas, haciendo flexible y adaptable el funcionamiento del sistema, según el tipo de actividad y se establecen las preferencias para cada usuario en el sistema en correspondencia con sus funciones dentro de la organización.

Y para que así conste se suscribe la presente Acta en Ciudad de la Habana en los 29 de mayo de 2015.

Nombre y Apellidos	Cargo
Oriando Martínez Obeso	2do jefe Grupo de Planificación de la Secretaría del CECM.

Anexo 7. Acta de liberación.

