"UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS"

Facultad 3

Solución para Alertas y Avisos en el Sistema de Planificación de Actividades SIPAC.

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

Autora: *Lianet Rodríguez Lorente.*

Tutor: *Ing. DionnyCardoso Carmona*

Co-tutoras: *Ing. Mairelys Fernández González.*

Ing. Ariadna Rendón Artola.

Ciudad de La Habana, Junio de 2013

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

Ernesto Guevara

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los_ días del mes de _____ del año ***2013****.*

Lianet Rodríguez Lorente
Autora

Ing. Dionny Cardoso Carmona
Tutor

Ing. Mairelys Fernández González
Co-tutor

Ing. Ariadna Rendón Artola
Co-tutor

AGRADECIMIENTOS

Deseo al concluir este trabajo de tesis agradecer de todo corazón:

A Dios.

A la Revolución Cubana.

A todos los profesores que me impartieron clases durante la carrera.

A mi tutor y cotutoras Dionny, Mairelys y Ariadna, por estar pendientes del avance de la solución y ayudarme en todo momento.

A mis padres y abuela, por el amor y cariño que me han dado sin condición, por confiar en mí, animarme e impulsarme a seguir aún cuando yo lo daba todo por perdido.

A mi esposo Yasmany, por su paciencia, palabras de aliento, confiar en mí y apoyarme en cada decisión.

A mi tía Olga y Jorge, por su constante preocupación, confianza y ser un hombro amigo para mis padres y abuela.

A mi primos Edisel y Yunelkis.

A María Teresa y Yordy, por su disposición y colaboración durante el proceso de desarrollo del componente.

A Jorgito, por su amistad incondicional, sus enseñanzas y ayuda en los momentos más difíciles de mi carrera.

A mis compañeros de estudio, en especial a: Nailet, Orlienis, Rosbel, Yaily, Claudia y Rogelio, por su compañía y cuidados durante estos años, principalmente durante mi embarazo y por demostrar afecto hacia ese pequeño ser que forma parte de mí.

A los miembros del tribunal, por sus recomendaciones para mejorar la calidad del trabajo.

Al oponente, que a pesar de no estar presente en las evaluaciones parciales de la investigación, colaboró en el aporte de ideas en los momentos de concebir la misma.

A Yuniel y Orlando, del departamento de Tecnología, por su disposición y ayuda durante el desarrollo de la solución.

A todos los que de una forma u otra han contribuido a mi formación.

DEDICATORIA

Dedico este Trabajo de Diploma:

A Dios por haberme permitido llevar a feliz término esta carrera.

A mi hijo por servirme de motor impulsor en los últimos momentos.

A esta Gloriosa Revolución Cubana, por haberme dado la oportunidad de formarme como profesional y a la que en pago dedicaré mis años de trabajo.

A mis padres y abuela sin el apoyo de los cuales no hubiera llegado al fin de esta carrera.

A mi esposo Yasmany por su apoyo y comprensión.

RESUMEN

El Sistema de Planificación de Actividades (SIPAC) constituye una herramienta para la gestión de las actividades a todos los niveles organizacionales, basado en la Instrucción no.1 del Presidente de los Consejos de Estado y de Ministros para la planificación en Cuba. Cuenta con varios módulos encargados de generar las configuraciones necesarias para el seguimiento de las tareas principales en función del cumplimiento de los objetivos de cada entidad, así como la gestión de los posibles involucrados, nomencladores y niveles de subordinación basados en reglas de la compartimentación de la información. Permite interrelacionar objetivos de trabajo y actividades en tiempo real, definir el flujo de información hacia los diferentes niveles a partir de la definición de estados y transiciones, así como la recuperación de la información de acuerdo al modelo de planificación actual. Independientemente de las funcionalidades que brinda el sistema, no da la posibilidad de notificar a los usuarios las acciones que se llevan a cabo sobre la información a la cual tienen acceso, así como la proximidad de determinadas tareas, dificultando la realización de los procesos de ejecución y control de la planificación a largo plazo (estratégica), así como la planificación a mediano y corto plazo (operativa). El presente trabajo comprende el desarrollo de la solución Alertas y Avisos para facilitar la realización de los procesos de Ejecución y Control de la Planeación Estratégica y Operativa en el Sistema de Planificación de Actividades (SIPAC).

Palabras claves: planificación de actividades, ejecución, control.

ÍNDICE

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

En los últimos años los países de América Latina y el Caribe registraron un significativo avance en el uso de las Tecnologías de la Información y las Comunicaciones (TIC) en los aspectos más diversos del desarrollo económico y social, que incluyó progresos en la digitalización de los procesos económicos para aumentar la productividad y la calidad de los mismos, así como la promoción de medios educativos, para la salud, el deporte y la cultura.

El uso de la tecnología de la comunicación permite el surgimiento de nuevas necesidades y deseos en los usuarios, aspectos que conducen al desarrollo acelerado de tecnologías de este tipo especialmente en los últimos años, las que a su vez crean nuevas posibilidades de intercambio de información. Por esta razón uno de los programas de la Revolución cubana es la informatización de la sociedad en todas sus ramas (1).

La Industria Cubana del Software está llamada a convertirse en una significativa fuente de ingresos nacional, como resultado del correcto aprovechamiento de las ventajas del considerable capital humano disponible. La Universidad de las Ciencias Informáticas y el sistema de empresas cubanas vinculadas a este trabajo, jugarán un papel importante en el desarrollo de la Industria Cubana del Software y en la materialización de los proyectos asociados al programa cubano de informatización.

Como parte de todo este proceso se desarrollan diversos sistemas informáticos para los distintos sectores del país, entre ellos, los sistemas encaminados a lograr la informatización de la planificación de actividades.

La Universidad de las Ciencias Informáticas, específicamente el Centro de Informatización de la Gestión de Entidades (CEIGE), de conjunto con la Secretaría del Comité Ejecutivo del Consejo de Ministros y la participación de especialistas del Ministerio de las Fuerzas Armadas Revolucionarias actualmente se encuentran vinculados en la realización del Sistema de Planificación de Actividades (SIPAC). Dicho sistema se encuentra 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 Popular. La solución está destinada 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. 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). Cuenta con varios módulos encargados de generar las configuraciones necesarias que otorgan al sistema y al cliente una simulación de los

procesos de organización del personal, así como los niveles de subordinación necesarios e indispensables para efectuar una planificación de actividades basadas en reglas estrictas de la compartimentación de la información, en otras palabras, para permitir que la información planificada sea accedida por la persona autorizada, en el momento indicado. Puede catalogarse como una solución integral para la gestión de elementos de la planeación estratégica y operativa basada en actividades, objetivos, y planes. Independientemente de las funcionalidades con las que cuenta el sistema actualmente no es posible:

1. Notificar al usuario activo sobre la alteración de la información creada por este, permisible para el resto de los usuarios.

2. Informar sobre la asignación de algún elemento de planificación.

3. Conocer concretamente cuáles atributos de los elementos de planificación fueron alterados así como el autor y la fecha de ejecución del cambio.

4. Saber si un documento está siendo objeto de modificación por otro usuario concurrentemente.

5. Indicar al usuario activo cuando la información que trata de introducir en el sistema colisiona.

Estos elementos provocan que:

1. Se dificulte el proceso de puntualización y seguimiento del plan; los usuarios que controlan la planificación en cada entidad deben acceder puntualmente a las tareas de los subordinados para conocer el estado de las mismas, independientemente de que hayan sido objeto o no de modificación. No se presentan en el sistema a modo de resumen las acciones realizadas sobre los elementos.

2. Los usuarios deban revisar constantemente la asignación de nuevas tareas, búsqueda que puede resultar engorrosa e innecesaria, y por otra parte de no hacerse en determinadas ocasiones puede comprometer el cumplimiento del plan, reflejándose en el Reporte del cumplimiento.

3. Los usuarios deban realizar un análisis detallado de cada elemento de la planificación que ha sido modificado para identificar los cambios, lo que atenta contra el aprovechamiento del tiempo.

4. La información asociada a cada elemento de la planificación debe ser única, si concurrentemente se trabaja sobre esta información cada usuario asume que los cambios que

se registran son los que él ha realizado. Es necesario notificar cuando esto ocurra para evitar inconsistencias.

5. El sistema permite a los usuarios introducir toda información referente a la planificación que considere necesaria; sin embargo no se indica cuando coincide el rango de fecha de alguna tarea que se pretende agregar en el sistema con otra(s) ya existente(s) en el período, por lo tanto cuando se lleva a cabo el proceso de Ejecución y Control por parte de los usuarios involucrados, las tareas pueden resultar ambiguas y desencadenar una re-planificación.

A partir de la problemática descrita con anterioridad surge el siguiente **problema a resolver**: En el Sistema de Planificación de Actividades (SIPAC) no se efectúan las Alertas y Avisos sobre eventos del sistema, lo que dificulta la realización de los procesos de Ejecución y Control de la Planeación Estratégica y Operativa.

El **objeto de estudio** lo constituyen los Sistemas de alertas y avisos y el **campo de acción** los Sistemas para las alertas y avisos en sistemas de planificación.

El **objetivo general** del presente Trabajo de Diploma es: Desarrollar la solución Alertas y Avisos para facilitar la realización de los procesos de Ejecución y Control de la Planeación Estratégica y Operativa en el Sistema de Planificación de Actividades (SIPAC).

Para encaminar la investigación en vista a resolver el problema planteado se propone la siguiente **idea a defender:** Si se desarrolla el componente Alertas y Avisos para la notificación de las acciones que se llevan a cabo sobre los diferentes elementos en el Sistema de Planificación de Actividades (SIPAC), se facilitará la realización de los procesos de Ejecución y Control de la Planeación Estratégica y Operativa.

Las **tareas** que se llevan a cabo para darle cumplimiento al objetivo trazado son:

1. Estudio del estado del arte referente a soluciones informáticas relacionadas con el campo de acción del presente trabajo.
2. Análisis de la Arquitectura del SIPAC para conocer las características fundamentales, marcos de trabajo, herramientas y tecnologías definidas para el desarrollo, así como la estrategia de integración entre los diferentes componentes.
3. Estudio del Modelo de Desarrollo del CEIGE.
4. Estudio de los patrones de Diseño.
5. Elaboración del Modelo conceptual y el Glosario de términos asociados a la solución.
6. Levantamiento y especificación de requisitos.

7. Diseño de los prototipos de interfaz de usuario.
8. Estudio y actualización del Modelo de Datos del SIPAC.
9. Elaboración del diseño de las clases asociado a la solución.
10. Implementación de los requisitos identificados.
11. Selección de los métodos y técnicas de pruebas para validar la solución.
12. Validación de la solución.

Para el desarrollo del presente trabajo de diploma se utilizaron los métodos científicos teóricos y empíricos, de modo que se pudieran estudiar las características del objeto de investigación que no son observables directamente y extraer de los fenómenos analizados las informaciones que se necesitan sobre ellos a través de observaciones y la propia experimentación.

Los métodos teóricos utilizados son:
Analítico - Sintético: el análisis permite el estudio de distintos sistemas y la bibliografía necesaria para la comprensión del objeto de estudio en su relativa independencia uno de otro, la síntesis permite descubrir las relaciones existentes entre un sistema y otro, tomar las características comunes para alcanzar el objetivo.

Análisis histórico lógico: permite estudiar la trayectoria de los sistemas de notificaciones, alertas y avisos, y así conocer cuáles son las tendencias de estos componentes.

Inducción – deducción: se utiliza en varios escenarios del trabajo, fundamentalmente en el Estado del Arte y en las Técnicas de captura de requisitos.

Modelación: se utiliza en la realización de diagramas encaminados al diseño del componente de Alertas y Avisos en el SIPAC.

Posibles resultados:
Desarrollo del componente Alertas y Avisos para facilitar las acciones que se han realizado sobre los diferentes tipos de elementos en el proceso de Planificación de actividades.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

En este capítulo se llevará a cabo una conceptualización de los términos y procesos más importantes referentes al problema a resolver. Se hará un análisis de sistemas desarrollados para el de envío de notificación de eventos tanto en el ámbito nacional como internacional, con el fin de identificar elementos que pudieran representar una alternativa en el desarrollo del componente informático que da solución al problema planteado en la presente investigación, de modo que esté a la altura de las exigencias del software de nuestros días. Además se explicará el Modelo de desarrollo, el cual está definido por la dirección del CEIGE. Se realizará un análisis de la arquitectura definida para el componente, así como las tecnologías y herramientas de desarrollo que se utilizarán durante la implementación.

1.1 Marco conceptual

El marco conceptual del presente trabajo de diploma está centrado en la definición de un conjunto de conceptos que tienen una estrecha relación con el problema en cuestión y su solución.

Alertas:

Situación de vigilancia o atención. (2)

Las alertas de un sistema informático son mensajes de correo electrónico (también se pueden usar otros medios) que se reciben cuando se encuentran nuevos resultados (por ejemplo: modificación de una actividad, eliminación de un objetivo, etcétera) que coinciden con sus consultas anteriores.(3)

Avisos:

Noticia o advertencia que se comunica a alguien. (2)

Escrito que advierte de algo.(2)

Indicio, señal. (2)

Notificación:

Acción y efecto de notificar.(2)

Documento en que consta la resolución comunicada. (2)

Socket

Es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. (4)

1.2 Sistemas de notificaciones existentes

En este apartado serán descritos sistemas de notificaciones que fueron estudiados para la solución del problema que se plantea. Serán expuestas sus principales características y similitudes con el componente que se desea desarrollar. Se valorará en cada caso su posible aplicación o uso durante la solución de alertas y avisos.

1.2.1 NetSupportNotify 2

Descripción

NetSupportNotify es una solución especialmente desarrollada para permitir la entrega fiable e inmediata de notificaciones y alertas en entornos LAN o WAN a equipos Windows, Mac y Linux. La filosofía de NetSupportNotify es ofrecer una solución sencilla y eficaz para que los administradores de una organización puedan enviar notificaciones importantes al personal y a los alumnos en un centro educativo sin importar el diseño de la red, además de ofrecer acuses de recibo de entrega y confirmación de los mensajes. Incluye avisos dirigidos a departamentos específicos. Presenta opciones de personalización de mensajes. (5)

Características(5)

- Soporte para Windows, Mac y Linux.
- Solución de aviso para equipos de escritorio.
- Envío de alertas a departamentos seleccionados o a todos los ordenadores.
- Integración con los sistemas existentes.
- Ofrece un sistema de notificación de bajo coste y alta velocidad que permite enviar de forma instantánea mensajes y alertas eliminando posibles retardos de los correos electrónicos en buzones de entrada de gran actividad.
- El mensaje se activa en primer plano en el ordenador del usuario.

1.2.2 WebSocket

Descripción

WebSockets es una tecnología que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Utilizando una conexión WebSocket, las aplicaciones web pueden realizar comunicaciones en tiempo real en lugar de tener que consultar a los cambios de ida y vuelta.(6)

Características(6)

- jWebSocket es una solución de Java/JavaScript de alta velocidad bidireccional de comunicación para la Web, segura, fiable y rápida.
- jWebSocket es una implementación de código abierto de Java y JavaScript del protocolo HTML5 WebSocket con un enorme conjunto de extensiones.
- En la actualidad los navegadores que soportan los Websockets son: Firefox 4, Google Chrome 4, Opera 11, Safari 5.
- La tecnología de Websockets se extiende por la mayoría de los servidores web más conocidos como Apache, Jetty, Tomcat, Tornado, Node.js, etcétera.
- Permite un sistema de notificaciones a determinados eventos. Es decir, un cartelito que aparece por encima del browser arriba a la derecha y te dice "tienes un nuevo email". Sólo que en este caso, los desarrolladores de sitios web son los encargados de crear las notificaciones.

1.2.3 Módulo de notificaciones y alertas del sistema de gestión universitaria

Descripción

Sistema desarrollado en la UCI en marzo de 2012, para administrar las distintas esferas de la vida universitaria.

Las notificaciones son generadas por la ocurrencia de eventos en el Sistema de gestión universitaria (SGU), mientras que las alertas son generadas a partir de una constante lectura y chequeo de variables sobre sus valores extremos.(7)

Características(7)

- Envía aviso al o los usuario(s) involucrado(s) sobre la alteración de la información concerniente a su trabajo.
- Envía mensaje a buzón de notificaciones y alertas del SGU.
- Anuncia la aproximación de determinados sucesos al usuario involucrado.
- Plataforma de software libre.
- Patrón arquitectónico MVC.

1.2.4 Sistema de Notificación de Eventos

Descripción

Sistema de video vigilancia desarrollado en la UCI en el año 2011 para el MINTUR. Su objetivo es enviar notificaciones por diferentes medios de comunicación para que pueda fluir la información entre el sistema y el usuario. Dicho componente soporta los medios de transmisión mediante plugins.(8)

Características(8)

- Envío de aviso al o los usuario(s) involucrado(s) sobre la alteración de la información concerniente a su trabajo.
- Envía notificación por correo electrónico
- Sistema operativo Windows y Linux
- Lenguaje de programación: C++

1.2.5 OpenFire

Descripción

Es un servicio de mensajería instantánea GPL, escrito en Java y se ha desarrollado con código abierto. Utiliza el protocolo XMPP. (9)

Características(9)

- Panel de administración web.
- Permite administrar usuarios y grupos.
- Mensajes offline.
- Contiene plugins gratuitos con diferentes funciones extras.
- Autenticación vía Certificados, Kerbeos, LDAP, PAM y Radius.
- Almacenamiento en Active Directory, LDAP, MS SQL, MySQL, Oracle y PostgreSQL.

1.2.6 Componente para la Gestión de notificaciones en el marco de trabajo Sauxe

Descripción

Es un componente de notificaciones en tiempo real para el marco de trabajo Sauxe, que permite dar a conocer a los usuarios según su rol, la ocurrencia de eventos concernientes a su trabajo específico, en el instante en que ocurren del lado del servidor, lo cual contribuye a economizar el tiempo y así beneficia el proceso de toma de decisiones.(10)

Características(10)

- Marco de trabajo Sauxe.
- Lenguaje de programación PHP.
- Envío de notificación en tiempo real.
- Integración con el servidor de mensajería OpenFire a través del protocolo XMPP.
- Notificación a los usuarios según su rol de las acciones realizadas sobre la información.
- Permite envío de notificaciones entre subsistemas.

- Especificación de los detalles de las modificaciones en la información.

1.3 Valoración del estado del arte

Después de realizar un estudio de las soluciones anteriormente mencionadas teniendo en cuenta los aspectos fundamentales desde el punto de vista del software y del producto, se arriba a que el 100% posee plataforma de software libre y envía notificaciones a los usuarios de las acciones realizadas sobre la información, el 60% anuncia sobre la aproximación de determinados eventos, el 25% utiliza el lenguaje de programación PHP y solo uno especifica los detalles de las modificaciones en la información.Presentan el envío de notificaciones a correo electrónico y la gestión de notificaciones en tiempo real, aspectos que deben ser tomados en cuenta a la hora de desarrollar la solución.

Por las características presentadas por el componente para la Gestión de notificaciones en el marco de trabajo Sauxe y las necesidades que presenta SIPAC, se realizará a consideración de la autora la integración del núcleo de la solución del Marco de trabajo Sauxe para la gestión de notificaciones en tiempo real, personalizando dicha solución de modo que las notificaciones sean generadas a nivel de usuarios, en lugar de roles y subsistemas como se realiza actualmente. Se realizará el envío notificaciones a los usuarios cuando un elemento de la planificación reciba alteraciones, sea modificado de forma concurrente, se asigne a un usuario, además de mostrar especificaciones de la modificación, anunciar la aproximación de determinada(s) tarea(s) de impacto para el (los) usuario(s) involucrado(s) y avisar al usuario activocuando la información que trata de introducir en el sistema colisiona.

1.4 Modelo de desarrollo de software

El modelo de desarrollo de software es un estándar que establece las distintas fases por las que debe transitar un proyecto durante su ciclo de vida, así como el conjunto de artefactos a generar en cada una de ellas.(11)

El presente Trabajo de Diploma se regirá por el Modelo de desarrollo del CEIGE, el cual consta de dos fases: Estudio preliminar; para realizar la planeación del proyecto, la evaluación de la factibilidad y el registro de éste y Desarrollo, que costa de 6 disciplinas, cada una con un impacto considerable en la obtención del producto final.

Las disciplinas que serán ejecutadas son:

Modelado del negocio: se comprende el funcionamiento del negocio que se desea automatizar, garantizando que el software cumpla su propósito. (11)

Requisitos: la tarea principal de esta fase es desarrollar un modelo del sistema que se va a construir. Se especifican los requisitos funcionales y no funcionales del sistema. (11)

Análisis y diseño: se modela el sistema de modo que soporte todos los requisitos(11). Debido a que se reutilizará el componente de alertas y avisos, se ajusta el modelado existente a los requisitos actuales.

Implementación: se implementa el sistema. Se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes. (11)

Pruebas internas: el grupo de calidad del centro realiza pruebas, verificando el resultado de la implementación. Permite saber si el software o la documentación presentan errores. (11)

Pruebas de liberación: se aplican pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación. (11)

1.5 Arquitectura

Según el documento de IEEE Std 1471-2000: la Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

1.5.1 Características de la Arquitectura Base

La arquitectura de software del SIPAC está organizada fundamentalmente en 3 vistas, las cuales a su vez se desagregan en 8 vistas. La organización utilizada para la representación de la arquitectura de software se muestra a continuación(12):

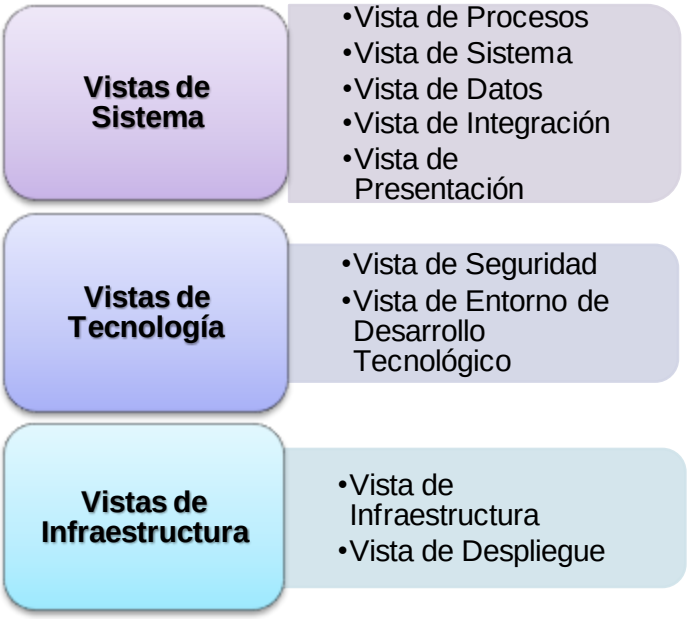

Figura 1 Vistas de la arquitectura.

- **Tecnología:** define la plataforma tecnológica sobre la que desarrolla el software, donde se especifican *frameworks* y herramientas base del sistema. (12)
- **Seguridad:** chequea e implementa todos los aspectos relacionados con el acceso a la aplicación, la modificación, lectura o eliminación de la información. (12)
- **Presentación:** define el aspecto visual del *software*, cuáles son los colores que lleva la aplicación, cómo son los botones, los vínculos y todos los elementos de alta significancia desde el punto de vista de la presentación. (12)
- **Sistema:** elabora las definiciones de los tipos de componentes que conforman el software, de la especificación de sus características, así como de la composición estructural interna de cada uno de estos componentes (vista vertical de la arquitectura). (12)
- **Integración:** se ocupa de los procesos de integración interna (entre componentes de un mismo proyecto) y externa (entre proyectos distintos), establece las definiciones, estándares, protocolos de comunicación y reglas de intercambio de información. (12)
- **Datos:** elabora todas las definiciones a nivel de datos, la integración de los distintos modelos, de los patrones, estándares y definiciones a este nivel. (12)
- **Desarrollo:** elabora la definición de la plataforma tecnológica a utilizar en la elaboración del

producto, así como de la definición y disponibilidad de los distintos servicios telemáticos necesarios en la confección del mismo. (12)

- **Despliegue:** define los requerimientos necesarios de *hardware* para la implantación del *software*, así como del diseño de los distintos escenarios de despliegue posibles. (12)

1.6 Herramientas y tecnologías

Las herramientas y tecnologías que se utilizarán están definidas por el CEIGE.

1.6.1 Marco de trabajo

Un framework o marco de trabajo es una estructura de apoyo definida a través de la cual un proyecto de software se puede organizar y desarrollar. Persigue acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo, como el uso de patrones.(13)

Se trabajará sobre el marco de trabajo Sauxe, el cual consta de 5 capas: las tres primeras están representadas por los frameworks: Ext js 2.2, Zendframework 1.5, Doctrine 1.0.

A continuación se muestra una descripción de cada una de estas herramientas y tecnologías.

Ext js 2.2: librería Javascript que permite construir aplicaciones web interactivas y complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java, lo que lo hace consistente sobre cualquier navegador sin tener que validar que el código funcione en cada uno de ellos. A pesar de esto, presenta algunos elementos negativos: el programador tiene que escribir más código para validar y enlazar los formularios, su sistema de licenciamiento no contempla la licencia LGPL, o el código es 100% GPL, o se debe pagar por su licencia de desarrollo(14). Esta librería incluye:

- Componentes para interfaz de usuarios (UI) del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open source y comerciales.

Zend 1.5: framework de código abierto para desarrollar aplicaciones web y servicios web con PHP. Su código es orientado a objetos y posee un bajo acoplamiento entre componentes, lo que permite a los desarrolladores utilizar los componentes por separado. Implementa el patrón Modelo Vista Controlador (MVC). Brinda un estándar de codificación, posee adapters para gran cantidad de tipos de bases de

datos diferentes; brinda componentes para la autenticación y autorización de usuarios, envío de correo electrónico y cache en varios formatos(13).

Doctrine 1.0: mapeador de objetos-relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un Sistema Gestor de Base de Datos (SGBD). Puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros frameworks. Posibilita escribir consultas de base de datos utilizando un dialecto de SQL denominado **DQL** (Doctrine QueryLanguage) que está inspirado en Hibernate (Java).(15)

1.6.2 Sistema Gestor de Base de Datos

Sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación.

PostgreSQL 8.3: Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos. Es un gestor de bases de datos de código abierto, brinda un control de concurrencia multi-versión que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación. Permite incluir las subconsultas, los valores por defecto, las restricciones a valores en los campos (constraints) y los disparadores (triggers). El código fuente se encuentra disponible para todos sin costo alguno. Está disponible para 34 plataformas con la última versión estable. Funciona en todos los sistemas operativos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows.
Debido a la liberación de la licencia, PostgreSQL se puede usar, modificar y distribuir de forma gratuita para cualquier fin, ya sea privado, comercial o académico. (16)

1.6.3 Herramienta CASE

Visual Paradigm 5.0 o superior: ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Se caracteriza por: su disponibilidad en múltiples plataformas (Windows, Linux), diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad, soporta aplicaciones Web,

compatibilidad entre ediciones, generación de bases de datos - transformación de diagramas de Entidad-Relación en tablas de base de datos, ingeniería inversa de bases de datos - desde SGBD existentes a diagramas de Entidad-Relación. (17)

1.6.4 Entorno Integrado de desarrollo (IDE)

NetBeans 7.1 o superior: entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Es un proyecto de código abierto de gran éxito. Usado como una estructura de integración para crear aplicaciones de escritorio grandes. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Entre las características de la plataforma están: administración de las interfaces de usuario (ej. menús y barras de herramientas), administración de las configuraciones del usuario, administración del almacenamiento (guardando y cargando cualquier tipo de dato), administración de ventanas, framework basado en asistentes (diálogos paso a paso). Permite crear aplicaciones Web con PHP.(18)

1.6.5 Servidor Web

Apache 2.0: servidor web HTTP de código abierto. Corre en diversos sistemas operativos, entre ellos, Linux y Windows. Es un servidor altamente configurable de diseño modular, extensible, popular (fácil conseguir ayuda/soporte). Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. (19)

1.6.6 Navegador web

Mozilla Firefox 17.0.1 o superior: navegador web libre y de código abierto desarrollado para Microsoft Windows, Mac OS X y GNU/Linux. Sus características incluyen navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas, navegación privada e integración del motor de búsqueda que desee el usuario. Proporciona un entorno para los desarrolladores web en el que se puede utilizar herramientas incorporadas, como la Consola de errores, el Inspector DOM, o extensiones como Firebug. Cuenta con una protección antiphishing, antimalware e integración con el antivirus. Implementa el sistema SSL/TLS para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo https. Es distribuido bajo triple licencia; Licencia Pública de Mozilla (MPL), Licencia pública general de GNU (GPL), o la Licencia pública general reducida de GNU (LGPL). Estas licencias permiten a cualquiera ver, modificar y/o redistribuir el código fuente.(20)

1.7 Lenguajes de modelado y desarrollo

1.7.1 Lenguaje de modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos.

- ✓ **UML 8.0**

Lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Ofrece nueve diagramas en los cuales modelar sistemas. Entre sus ventajas se encuentran: promueve la reutilización, permite a los ingenieros especificar o descubrir los requisitos para un sistema de propuesta o un sistema en evolución y revisar los requisitos para garantizar su exactitud e integridad, es útil en la industria y en los organismos de normalización, combina metas y escenarios, ayuda a reducir la brecha entre los conceptos formales e informales y entre los modelos de requisitos y modelos de diseño. Pero a su vez UML no define un formato de archivo estándar, lo que significa que cada vendedor de herramientas UML almacena la representación de su modelo UML en un formato propietario, está generalmente limitada a lo que el vendedor ofrece fuera de la caja, que es una cierta forma de generación de código. El código se genera una vez.(21)

1.7.2 Lenguajes de programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. (22)

Lenguaje del lado del servidor

En la tecnología cliente-servidor, es el lenguaje de programación que se ejecuta del lado del servidor y que brinda a los usuarios la respuesta del procesamiento de la información.

- ✓ **PHP 5.2**

Es un lenguaje de programación de alto nivel, utilizado para la creación de páginas web dinámicas. Es una tecnología de código abierto utilizada para diseñar aplicaciones web dirigidas a bases de datos. Es multiplataforma. Algunas ventajas son: su gratuidad, independencia de plataforma, rapidez y seguridad, capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL, el código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador; esto hace que la programación en PHP sea segura y

confiable, permite aplicar técnicas de programación orientada a objetos; incluso aplicaciones como Zendframework, empresa que desarrolla PHP, están totalmente desarrolladas mediante esta metodología. A pesar de esto posee algunos inconvenientes, debido a que es un lenguaje interpretado, un script en PHP suele funcionar considerablemente más lento que su equivalente en un lenguaje de bajo nivel.(23)

Lenguajes del lado del cliente

Un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Es asimilado directamente por el navegador y no necesita pre-tratamiento. (24)

- ✓ **JavaScript**

Es un lenguaje de programación orientado a objetos, basado en prototipos, dinámico, puede ser interpretado por todos los navegadores. Es un lenguaje de alto nivel, multiplataforma y no necesita compilación. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). Algunas de sus ventajas y desventajas son: es seguro y fiable, es ligero de carga, fácil de integrar, se ejecuta en el cliente por lo que el servidor no es solicitado más de lo debido; un problema importante es que el código es visible y puede ser leído por cualquiera, incluso si está protegido con las leyes del copyright, los script tienen capacidades limitadas por razones de seguridad, por lo cual no es posible hacer todo con Javascript, sino que es necesario usarlo conjuntamente con otros lenguajes evolucionados, posiblemente más seguros, como Java. (24)

1.8 Técnicas de captura de requisitos

A continuación se muestran diferentes técnicas para recoger los requisitos.

Extracción de requisitos desde la información proporcionada por el cliente: el cliente proporciona algo de información desde su punto de vista y necesidades. Dicha información puede tener un carácter heterogéneo, mezclando datos actuales de la empresa con documentos que recogen la idea de lo que necesitan.

Entrevistas con el cliente: una vez analizada la documentación inicial, se pasa a realizar entrevistas al cliente, haciendo preguntas con el objetivo de obtener información del entrevistado sobre sus necesidades, lo que permite al equipo de desarrollo definir los requisitos del sistema.

Tormenta de ideas: se realizan reuniones con el equipo de desarrollo y demás implicados en la solución. Consiste en la acumulación de ideas acerca de las necesidades de la solución.

Reuniones: son más formales y estructuradas, realizadas por un experto. Participa el equipo de desarrollo. Se detallan los requisitos, sirviendo de base para la especificación de los mismos.

1.9 Técnicas de validación de requisitos

Una vez definidos los requisitos, deben ser validados. Tiene como misión demostrar que los requisitos definen realmente el sistema que el usuario necesita o el cliente desea. El proceso de validación de requisitos debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva.

Reviews o Walk-throughs: consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida.

Auditorías: la revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.

Matrices de trazabilidad: consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

Prototipos: algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

1.10 Patrones

Un patrón es una regla que consta de tres partes, y expresa una relación entre un contexto, un problema y una solución. Por lo general, sigue el siguiente esquema(21):

- ***Contexto*:** situación de diseño en la que aparece un problema de diseño.
- ***Problema*:** conjunto de fuerzas que aparecen repetidamente en el contexto.
- ***Solución*:** configuración que equilibra estas fuerzas. Abarca:
  - Estructura con componentes y relaciones.

- Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos, etcétera.

Patrón arquitectónico

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular, el cual es recurrente dentro de un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades y relaciones(21).

El patrón Modelo-Vista Controlador MVC (del inglés *Model-View Controller*) es un patrón de arquitectura utilizado en sistemas web para separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios. Plantea la necesidad de tener una clase controladora frontal que reciba las peticiones de los usuarios y ésta delegue así las responsabilidades a otras clases controladoras aplicando el patrón experto. Esta case controladora es la responsable de comunicar los elementos de la vista con el modelo y mantener actualizada ambas partes según los eventos que se desarrollen en el sistema.(25)

Entre sus ventajas se pueden encontrar: soporte de vistas múltiples, ya que la vista se encuentra separada del modelo y no existe una dependencia directa entre ellos, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente y posee una amplia adaptación al cambio; en ocasiones los requisitos de interfaz de usuario pueden cambiar y al no haber una dependencia entre la vista y el modelo, se pueden agregar nuevas opciones de presentación sin afectar el modelo.(25)

El **Modelo** representa las estructuras de datos, reglas de negocio y las funcionalidades correspondientes para la comunicación con el persistidor de datos Doctrine.

La **Vista** es la información presentada al usuario. Esto es posible mediante el marco de trabajo de la capa de presentación EXTJS.

El **Controlador** actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página, es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, utilizando el marco de trabajo ZendFramework.

Patrón de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software(21). Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Su uso pretende establecer un lenguaje común entre los programadores, contribuir a la reutilización, ahorrar tiempo en la implementación y obtener un producto con calidad. Es muy importante tener presente: su nombre, el problema (cuándo aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios)(26).

Patrones GOF

Los patrones Gang Of Four (GoF), son soluciones concretas, técnicas, que se utilizan en situaciones frecuentes, favorecen la reutilización de código, su uso no se refleja en el código(21).

Patrones GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones(21).

1.11 Conclusiones parciales

El capítulo que recién concluye fue esencial para valorar el estado actual de los sistemas de alertas y avisos tanto nacionales como internacionales. Se analizaron diferentes sistemas que llevan a cabo la gestión de notificaciones y alertas, lo que evidencia la necesidad de desarrollar el componente Alertas y Avisos en el SIPAC, integrando el núcleo de la solución del Marco de trabajo Sauxe para la gestión de notificaciones en tiempo real. Se empleará el modelo de desarrollo definido por el CEIGE, el cual define las tareas, actividades y herramientas necesarias que garanticen, como resultado final, un software de calidad.

CAPÍTULO 2: ANÁLISIS Y DISEÑO.

En este capítulo se presenta la propuesta de solución para Alertas y Avisos en el SIPAC. Inicialmente se definen los principales conceptos del negocio, se identifican y describen los requisitos funcionales y no funcionales, se muestran además algunos artefactos generados por el modelo de desarrollo de CEIGE, realizados durante las disciplinas de Requisitos, y Análisis y Diseño. Se describen también los patrones de diseño y arquitectónicos utilizados en la solución.

2.1 Modelo Conceptual.

La siguiente figura representa el modelo conceptual de la solución para alertas y avisos en el SIPAC, cuyo flujo se centra en enviar notificaciones y alertas a los usuarios involucrados cuando se realice una acción (cambiar de estado, modificar, eliminar, involucrar) sobre un elemento primario de la planificación (planes, objetivos y actividades). Los conceptos fundamentales con los que se va a trabajar se encuentran descritos en el artefacto CIG-SPA-N-i2340 del expediente de proyecto.

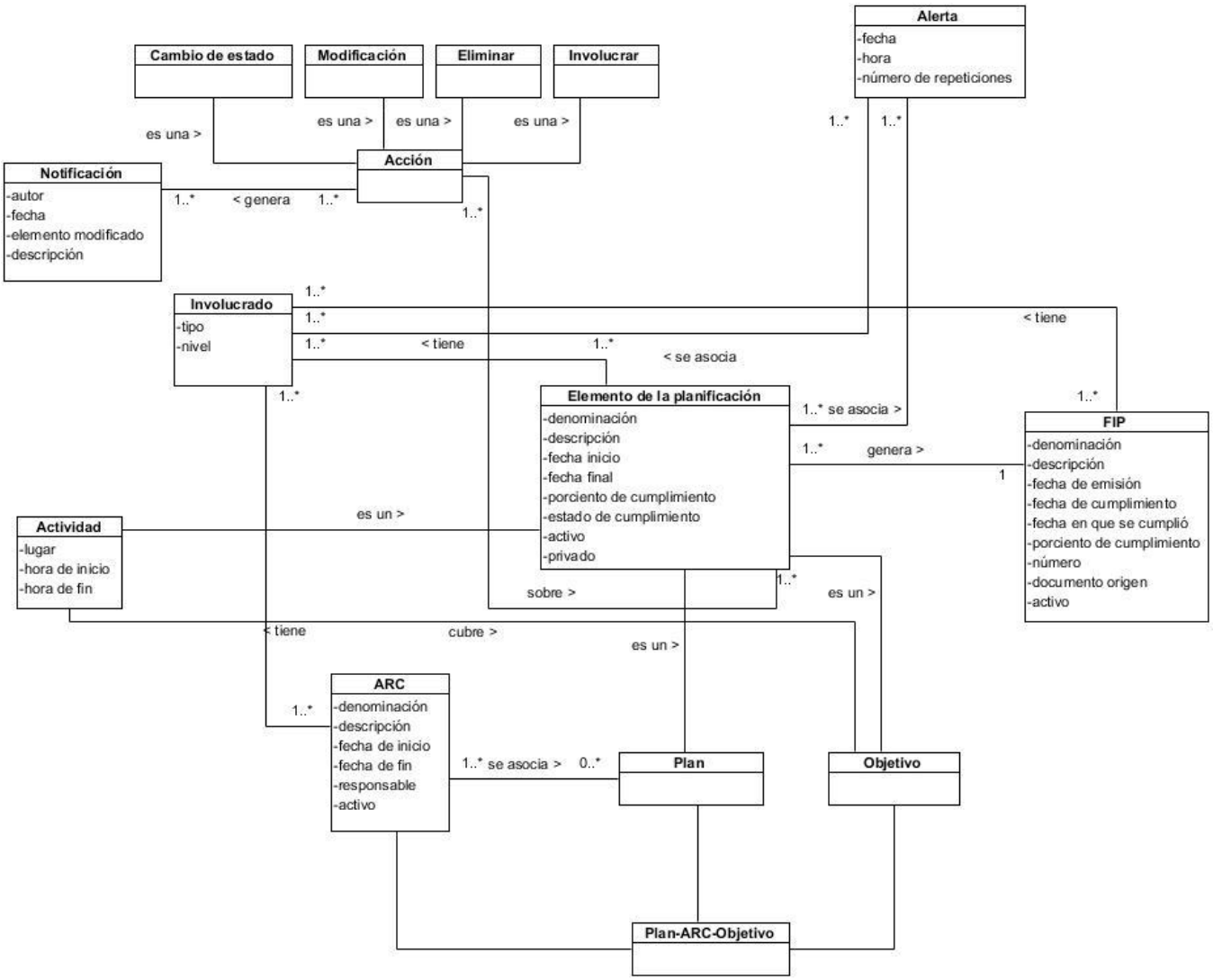

Figura 2 Modelo conceptual.

2.2 Requisitos de software

2.2.1 Requisitos funcionales

Los requisitos funcionales son definidos por el equipo de desarrollo teniendo en cuenta la información brindada por el cliente. Partiendo de esta información se elaboran los documentos de especificación de requisitos y finalmente se valora y valida la información en busca de errores, inconsistencias o faltas para evitar omitir algún requerimiento del cliente.(27)

A través de las técnicas de captura de requisitos: entrevistas con el cliente, tormenta de ideas y reuniones, se obtuvieron 8 **requisitos funcionales**:

RF1 Notificar al usuario activo sobre la modificación de un elemento primario de la planificación.

RF2 Notificar al usuario activo sobre la asignación de un elemento primario de la planificación.
RF3 Notificar al usuario activo sobre la eliminación de un elemento primario de la planificación.
RF4 Visualizar los atributos de los elementos primarios de la planificación que fueron modificados.
RF5 Alertar sobre la aproximación de determinadas tareas a los usuarios involucrados.
RF6 Configurar la notificación de la alerta.
RF7 Notificar trabajo concurrente.
RF8 Identificar la superposición de la información asociada a los elementos primarios de la planificación.

Especificación de requisitos

Según el modelo de desarrollo de CEIGE se debe realizar la especificación de los requisitos capturados con el objetivo de detallar un conjunto de aspectos necesarios para lograr una mejor comprensión de los requisitos.

A continuación se muestra la especificación del requisito funcional Notificar al usuario activo sobre la modificación de un elemento primario de la planificación, la especificación de los restantes requisitos funcionales se encuentra en el expediente de proyecto en el artefacto CIG-SPA-N-i2340.

RF_ 1: Notificar al usuario activo sobre la modificación de un elemento primario de la planificación.

Precondiciones	El usuario debe estar autenticado en el sistema. El elemento de la planificación debe existir.
Flujo de eventos	
Flujo básico << Notificar al usuario activo la modificación de un elemento primario de la planificación >>	
1	El usuario modifica un elemento primario de la planificación.
2	El sistema muestra la notificación a los involucrados de que ese elemento ha sido modificado.
3	Se concluye el requisito.
Pos-condiciones	
1	Se notifica al usuario la modificación del elemento

	de la planificación.	
Flujos alternativos		
Flujo alternativo N/A		
Pos-condiciones		
Validaciones		
1	N/A	
Conceptos	N/A	N/A
Requisitos especiales	RNF1-RNF10	
Asuntos pendientes	N/A	

Tabla 1 Descripción del requisito funcional 1.

Para validar los requisitos especificados se empleó la técnica de **Prototipos**, además de aplicar los C**riterios para validar los requisitos del cliente** y realizar el **Acta de aceptación de los requisitos**, según el modelo de desarrollo de CEIGE.

2.2.3 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Permiten que clientes y usuarios valoren las características no funcionales del producto, pues aunque cumpla con las funcionalidades requeridas, las características no funcionales pueden hacer una gran diferencia entre un producto que goce de la aceptación de todos y uno con poca aceptación. (27)

Existen requisitos no funcionales generales para el producto. Los que se listan a continuación impactan directamente en la Solución para Alertas y Avisos:

Eficiencia

Rendimiento

Breve descripción: 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.

1. 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).
2. 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).

2.3 Diagrama de clases del diseño web

El diagrama de clases del diseño especifica la estructura de clases del sistema con relaciones entre clases y estructuras de herencia. Es desarrollado buscando una solución ideal(21). A continuación se muestra un fragmento del diagrama de clases del diseño Obtener_notificación, basado en estereotipos web, que se realiza durante el proceso de desarrollo (para obtener una vista más ampliada ver Anexo 1). Está representado por las clases: cmpgestionarX.phtml; donde "elemento" presenta el mismo comportamiento para planes, objetivos y actividades, cmpnotificaciones.js, cmpnotificacionesController, cmpnotificacionesModel y las clases del dominio: dat_notificacion, dat_usuario_notificacion, dat_elementos, nom_accion, dat_historial y dat_alerta. Durante el diseño de las clases se aplicó el patrón Modelo Vista Controlador. (Ver epígrafe 2.5.1)

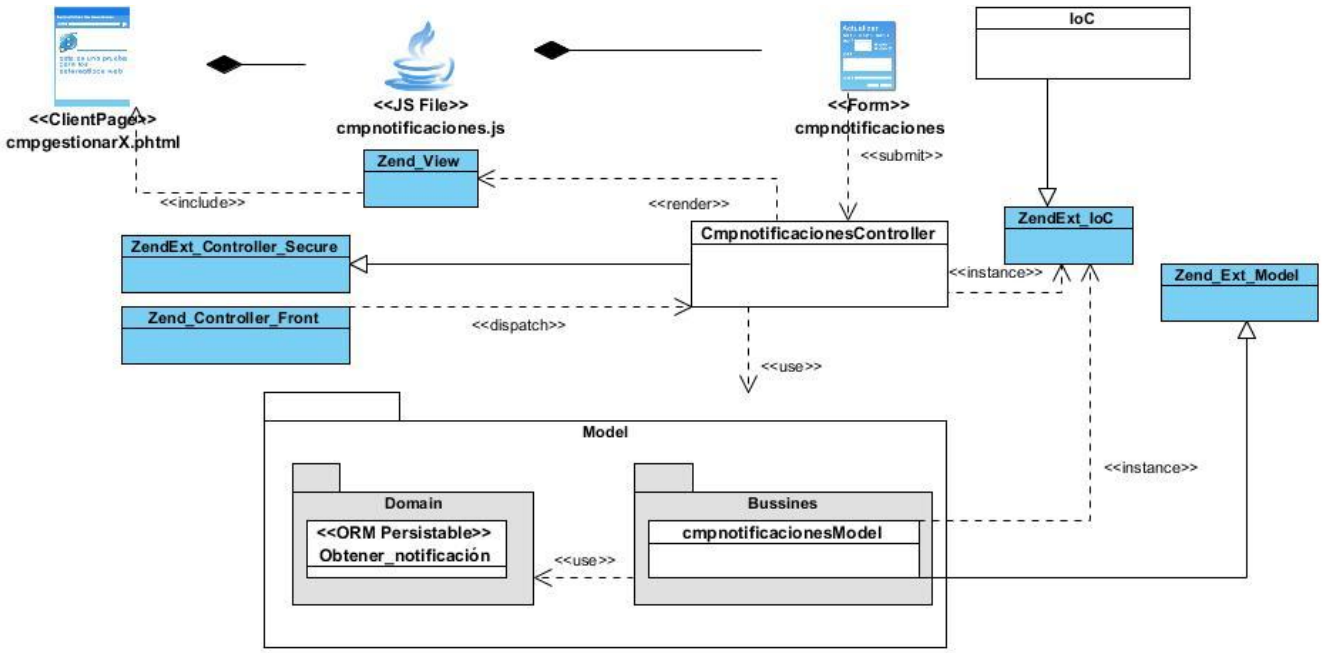

Figura 3 Diagrama de clases del diseño Obtener_notificación

Clases	Descripción
Extjs	Contiene los componentes generados a través de la librería JavaScript Extjs.
cmpgestionarX.phtml	Página encargada de visualizar, a través de los js que debe incluir, la información necesaria para generar la notificación.
cmpnotificaciones.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería Extjs los componentes que poseen la información de la notificación. Debe enviar y recibir los datos de la controladora utilizando tecnología AJAX.
CmpnotificacionesController	Clase controladora responsable de realizar las diferentes funcionalidades sobre la notificación, según las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
ZendExt_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos.

Tabla 2 Descripción del diseño de clases Obtener_notificación.

2.4 Diagrama de secuencias

Los diagramas de secuencia muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo. Se modelan para cada requisito funcional y contienen 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. A continuación se muestra el diagrama de secuencia del requisito funcional Notificar al usuario activo sobre la modificación de un elemento primario de la planificación. Para obtener una vista más ampliada ver Anexo 2.

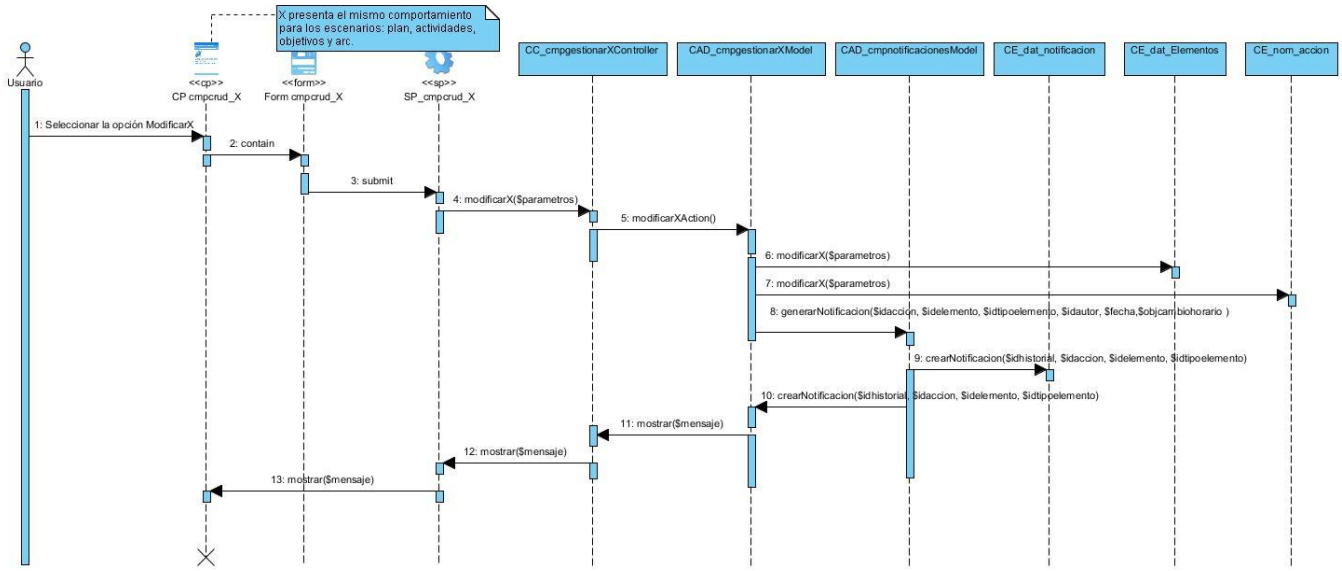

Figura 4 Diagrama de secuencias GenerarNotificacion_Modificar

En el diagrama se muestra la secuencia de pasos realizados por el usuario que generan una notificación al modificar un elemento primario de la planificación.

Inicialmente si el usuario desea modificar un elemento primario de la planificación, selecciona en la interfaz correspondiente al elemento la opción Modificar, a través de la cual se muestra un formulario con los datos del elemento. A continuación se introducen los nuevos datos y se presiona el botón Aceptar para enviarlos. La clase controladora cmpgestionarXController envía la petición a la modelo cmpgestionarXModel quien realiza la consulta a la clase de acceso a datos dat_elementos actualizando la tabla correspondiente al elemento de la planificación modificado. Al mismo tiempo, se recepciona en la tabla nom_accion la acción Modificar, realizada sobre el elemento de la planificación con los datos de la modificación. Al ser esta una acción generadora de notificación, se generan las notificaciones a través de la comunicación entre las clases cmpgestionarXModel, quien mediante un objeto de cmpnotificacionesModel, genera una nueva notificación en la tabla dat_notificación, con el contenido de la misma y los usuarios que la recibirán, estos son los involucrados al elemento. Finalmente se les muestra la notificación a los usuarios involucrados.

2.5 Patrones utilizados

2.5.1 Patrón arquitectónico

- **Modelo-Vista-Controlador (MVC)**

Al diseñar las clases del componente se aplicó el patrón arquitectónico MVC. Éste es empleado en la capa de control del marco de trabajo Sauxe y provee a la solución de flexibilidad y facilidad a la hora de hacer futuros cambios.

El modelo está representado por las clases: cmpnotificacionesModel, dat_notificacion, dat_elementos, dat_usuario_notificacion, nom_accion, dat_historial y dat_alerta.

La vista es la información presentada al cliente, a través de las páginascmpnotificaciones.phtml y cmpnotificaciones.js.

El controlador está representado por la clase cmpnotificacionesController que es la responsable de realizar las diferentes funcionalidades sobre la notificación, según las peticiones del usuario.

2.5.2 Patrones de diseño

Patrones GOF

De estos patrones se aplicaron:

- **Fachada:** un ejemplo de su uso es en los servicios, donde la relación existente entre las clases controladoras y éstos, permite acceder a métodos de otros componentes que se encuentran tanto dentro como fuera del subsistema Planificación. Evidenciado en los servicios que consume el componente a través del IOC.
- **Chain of Responsibility (Cadena de responsabilidad):** se evidencia al distribuir las responsabilidades, ya que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo es manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error.

Patrones GRASP

Para el diseño de la solución se tuvieron en cuenta los 5 patrones GRASP: Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador. El marco de trabajo busca un máximo rendimiento y flexibilidad en sus soluciones y pone en práctica estos patrones para lograr un sistema reusable y flexible.

- **Experto**

Las clases pertenecientes al Domain poseen la información necesaria para cumplir con su responsabilidad. Específicamente las clases dat_notificacion, dat_usuario_notificacion, dat_alerta, y nom_accion que contienen la información referente a las notificaciones.

- **Creador**

Este patrón es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.

- **Controlador**

La clase controladora definida: CmpnotificacionesController, es un ejemplo de la aplicación de este patrón, la misma tendrá a cargo la responsabilidad de manejar los eventos dentro del componente.

- **Alta cohesión**

Este patrón fue utilizado en el diseño del componente de manera general; agrupando las clases en dependencia de los requerimientos, según la premisa de quela información almacenada en una clase debe ser coherente y estar relacionada con ésta en mayor medida, enfocada en sus responsabilidades.

- **Bajo acoplamiento**

Las clases se encuentran lo menos relacionadas entre sí, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás, potenciando la reutilización y disminuyendo la dependencia entre las clases.

En el modelo de datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

2.6 Modelo de Datos

Luego del estudio realizado al modelo de datos de SIPAC se evidencia la necesidad de crear nuevas tablas para almacenar los datos referentes a las notificaciones. Para el funcionamiento del componente se utilizan, del modelo SIPAC, la tabla dat_elementos que se relaciona con las tablas: nom_tipoelemento, dat_plan_pdo, dat_objetivo y dat_actividades. A este modelo se le agregan las tablas: dat_alerta, que está relacionada con dat_actividades, dat_notificacion, dat_usuario_notificacion, dat_historial y nom_accion.

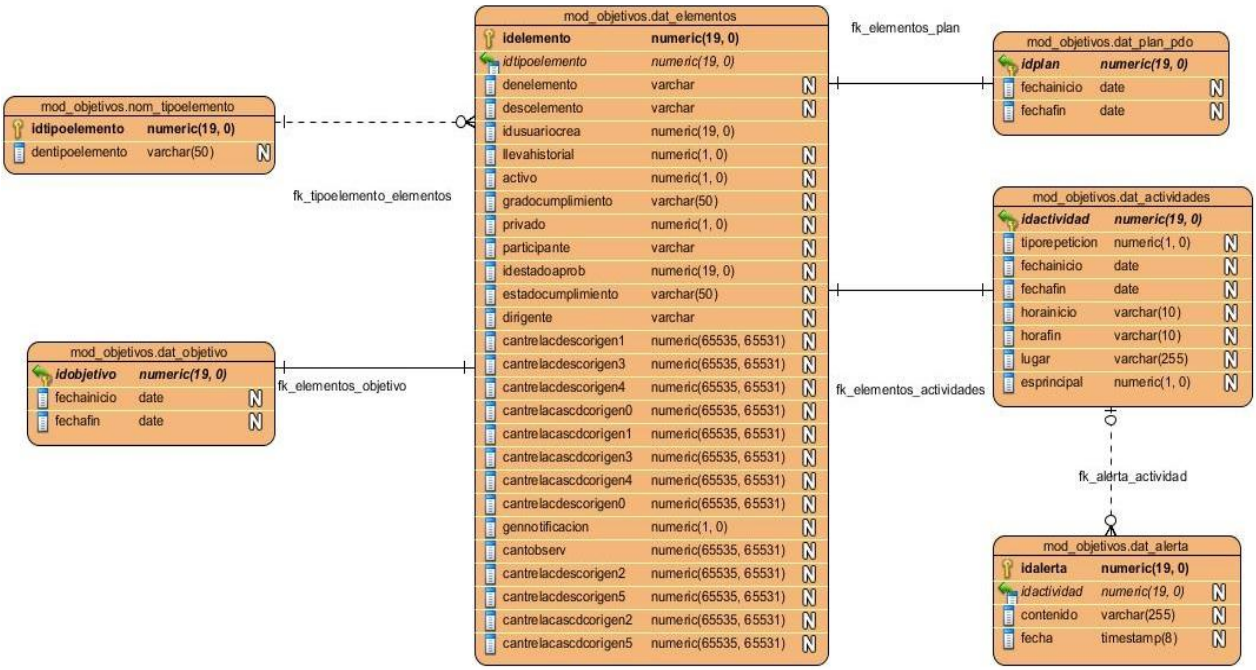

Figura 5 Modelo de datos utilizado por el componente

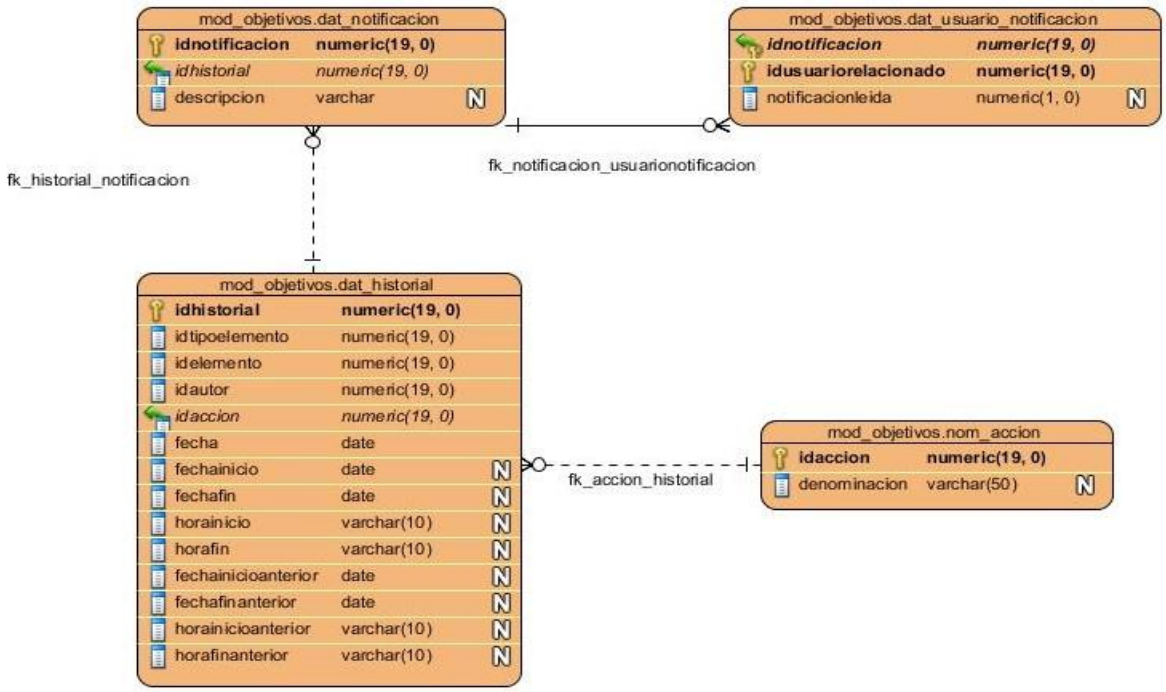

Figura 6 Modelo de datos creado para el componente.

Las notificaciones generadas serán almacenadas en la tabla dat_notificacion, en la misma se guarda su contenido, que no es más que el mensaje mostrado. Éstas al ser enviadas a los usuarios involucrados serán almacenadas en la tabla dat_usuario_notificacion, que se relaciona con la anterior, registrando si el usuario leyó la notificación o no.

La tabla dat_notificacion se relaciona además con dat_historial, donde se almacena: la fecha de la notificación, su autor, la acción que la generó y el elemento sobre el que se realizó la acción.

Al realizar una acción sobre un elemento de la planificación será almacenada su denominación, que son los distintos estados por los que pasan los elementos de la planificación, en la tabla nom_accion. Ésta se relaciona con dat_historial.

Al crear o modificar una actividad se puede generar una alerta si el usuario lo desea, por lo que se creó la tabla dat_alerta, que se relaciona con dat_actividades, en ella se almacenan su contenido y la fecha en la que debe ser lanzada.

2.7 Conclusiones parciales

Se realizó el análisis y diseño del componente, el mismo permitió pasar a la fase de implementación al tener los principales artefactos para el desarrollo.

Se obtuvo como resultado el diagrama de clases del diseño para una mejor comprensión de cómo están estructuradas las clases que conforman la solución y la relación entre ellas, así como los patrones arquitectónicos y de diseño propuestos. Esto posibilita conocer las clases principales del negocio para luego diseñar el modelo de datos correspondiente a la solución propuesta.

La aplicación del patrón Modelo-Vista Controlador permite un mejor flujo de información entre las capas de presentación, negocio y datos, lo que permitió separar cada una de estas capas para que cumplan con sus funcionalidades independientemente.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

En el presente capítulo se mostrarán los posibles escenarios en los cuales puede ser desplegado el sistema a través del diagrama de despliegue, se realizará la implementación del componente de Alertas y Avisos para el SIPAC, se validará además el diseño propuesto en el capítulo anterior mediante la aplicación de las métricas Tamaño Operacional de Clase (TOC) y Relación entre Clases (RC), para verificar el cumplimiento de los atributos de calidad definidos por las propias métricas seleccionadas, finalmente se describen las pruebas efectuadas al software.

3.1 Implementación del componente.

A continuación se presenta la estructura física de la solución propuesta en concordancia con el marco de trabajo Sauxe y el diagrama de despliegue asociado a la misma.

3.1.1 Estructura física del sistema.

El Departamento de Tecnología de CEIGE define una estructura contenedora donde van a estar ubicados cada uno de los subsistemas implementados dentro de la aplicación, de manera que facilite la organización y claridad durante el desarrollo. La especificación que a continuación se presenta está enfocada al componente Alertas y avisos.

Dentro de la carpeta raíz del sistema *Cedrux* se encuentran definidas las carpetas *apps*y *web*, que contienen la lógica de negocio y las vistas de los componentes de cada subsistema respectivamente.

En la siguiente figura se muestra el contenido del subsistema planificación.

Figura 7 Estructura carpeta apps.

comun: Comprende la configuración concreta para el subsistema. Dentro de ella se encuentra la carpeta **recursos** quien contiene la carpeta denominada **xml** que incluye los ficheros *ioc* y *posiciones_modelo_intercambio_sipac,* en el primero se publican los servicios que brindan cada uno de los componentes de SIPAC mediante un lenguaje de descripción de servicios web (WSDL) y el segundo contiene las posiciones exactas donde se encuentran los atributos de los elementos de la planificación en el fichero con extensión.

temp: Contiene ficheros temporales que se utilizan para el proceso de intercambio y actualización de la información, estos ficheros a su vez poseen las estructuras físicas de la base de datos heredada de SQLite.

pdo: Contiene al Subsistema de Planificación de Actividades que incluye el componente de alertas y avisos, el cual está estructurado por un conjunto de paquetes que se especifican a continuación:

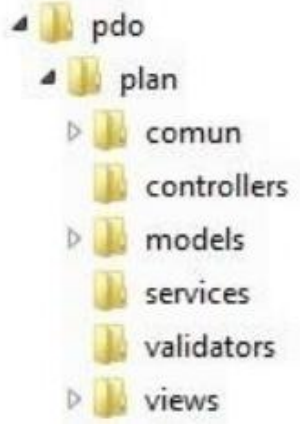

Figura 8 Estructura carpeta pdo.

comun: almacena la configuración utilizada por el componente, incluye la carpeta **recursos** y dentro de esta una denominada *xml* que contiene los ficheros: *validation, el cual chequea* las precondiciones antes de ejecutar una determinada función en el servidor según el tipo de parámetros, la acción y el usuario que la realice y *exception, donde se* encuentran las descripciones de los mensajes de las excepciones que pueden ser lanzadas por el servidor.

controllers: este fichero contiene las clases controladoras encargadas de gestionar las funcionalidades del sistema.

models: esta carpeta contiene los paquete bussines y domain, los cuales a su vez agrupan clases y otras subcarpetas. El paquete *bussines* contiene las clases necesarias para acceder a los datos que persisten en la base de datos. El paquete *domain* debe contener las clases generadas por Doctrine a partir de cada una de las tablas existentes en la base de datos. Cada una de estas clases heredará de clases bases que contienen los atributos y dependencias entre las tablas y se encuentran en el paquete llamado *generated* y son generadas igualmente por Doctrine.

Figura 9 Estructura carpeta models.

services: este paquete incluye todas las clases y funcionalidades de los servicios que va a ofrecer el componente.

validators: agrupa las clases y funcionalidades correspondientes a las validaciones que realiza el servidor antes de ejecutar una determinada función.

views: contiene los paquetes idioma y scripts, que contiene el idioma en que se va a mostrar la aplicación y las páginas clientes.

Figura 10 Estructura carpeta views.

La carpeta ***web*** contiene un fichero *index.php* que almacena la dirección del archivo de configuración y a través de éste inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento. Su código es el mismo para todos los componentes.

Figura 11 Estructura carpeta web.

pdo: contiene todas las vistas del Subsistema de Planificación de Actividades, incluye el componente de alertas y avisos, que igualmente incorpora el fichero *index.php*. También contiene el paquete *views* que se especifica a continuación.

Figura 12 Estructura carpeta plan.

views: ésta contiene los css y js necesarios para visualizar todo el contenido de la presentación del componente de alertas y avisos.

Figura 13 Estructura carpeta views del componente.

css: contiene las clases necesarias para darle la estructura gráfica al componente, separando así el estilo del contenido.
js: comprende los ficheros JavaScript necesarios para que el usuario interactúe con el sistema y obtenga los resultados esperados. Está compuesto por paquetes con los nombres de las funcionalidades que contiene.

3.2 Estructura de datos a utilizar

Se define como estructura de datos en programación, a la forma de organizar un conjunto de datos elementales con el fin de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema. Existen diversas estructuras que en sí poseen ventajas y desventajas según la simplicidad y eficiencia para la realización de cada operación. Sin embargo a la hora de elegir la estructura de datos más conveniente es preciso evaluar factores básicos como la frecuencia y el orden en que se realiza cada operación sobre los datos(28).

La sintaxis de código en PHP; lenguaje definido para la implementación de los diferentes componentes de la aplicación debido a las facilidades que brinda; contiene variables que tienen información similar y que se procesan de forma semejante, concretamente, se está hablando de los arrays como estructura de datos.

Un array o como también se le conoce: arreglo, es un conjunto de variables (elementos) agrupadas bajo un único nombre en distintas casillas(28).

Existen dos tipos de arreglos: los índices numéricos y los índices asociativos. Poseen una serie de funciones creadas para ordenarlos que facilitan su utilización. A partir de lo antes descrito y por decisión del equipo de arquitectura esta vendría siendo la estructura de datos a utilizar de forma general para el desarrollo en todos los subsistemas.

3.3 Descripción de las clases y funcionalidades del componente.

Clases Controladoras:

Las clases de controladoras coordinan las actividades de los objetos que implementan las funcionalidades, definen el flujo de control y las transacciones entre los objetos.

Nombre: CmpnotificacionesController
Tipo de clase: Controladora
Para cada responsabilidad:

Nombre:	**Descripción**:
cargarTipoNotificacionesAction()	La función carga los tipos de notificaciones.
cargarNotificacionesAction()	Permite cargar dado una estructura todas las notificaciones.
eliminarNotificacionAction()	La función elimina de la base de datos la notificación seleccionada por el usuario.
notificacionLeidaAction()	La notificación se le marca como leída al usuario.

Tabla 3 Descripción de la clasecmpnotificacionescontroller.

ClasesModel:

Estas clases manejan la información persistente que poseen una larga vida, conceptos y sucesos que ocurren en el mundo real.

Nombre: CmpnotificacionesModel	
Tipo de clase: Model	
Para cada responsabilidad:	
Nombre:	**Descripción:**
cargarTipoNotificaciones()	Carga los tipos de notificaciones.
obtenerArrayUsuarios($idelemento, $idautorelem, $idtipoelemento,$idaccion)	Permite obtener el array de los usuarios que pueden ver la notificación generada.
cargarNotificaciones($idusuario, $idusuariorelacionado, $asunto, $idaccion, $arraytipoelem, $fecha, $autores, $limit, $start)	Permite cargar los datos de una notificación.
crearNotificacionesPorUsuario($idnotificacion, $arrayusuarios)	Permite asociar una notificación a varios usuarios.
crearHistorial($idaccion, $idelemento, $idtipoelemento, $idautor, $objcambiohorario)	Permite crear un historial cuando se ejecute una acción.
crearNotificacion($idhistorial, $idaccion, $idelemento, $idtipoelemento)	Permite crear notificación cuando se ejecute una acción.
generarNotificacion($idaccion, $idelemento,	Permite crear notificación cuando se ejecute una acción.

$idtipoelemento, $idautor, $fecha,$objcambiohorario)	
construirAsunto($idaccion, $idelemento, $idtipoelemento)	Permite crear el contenido de una notificación cuando se ejecute una acción.
eliminarNotificacion($idusuario, $idusuariorelacionado, $arrayidnotificaciones)	Permite eliminar una notificación.
actualizarNotificacion($idusuario, $idusuariorelacionado, $idnotificacion)	Permite actualizar los datos de una notificación.
adicionarAlerta($parametrosAlarma)	Permite adicionar y configurar una alerta al adicionar la actividad, seleccionando la opción de alertar su proximidad.
modificarAlerta($parametrosAlarma)	Permite modificar la alerta si se modifica la actividad.
eliminarAlerta($arrayidactividades)	Elimina una alerta al eliminar una actividad.
buscarAlarma($datos)	Busca determinada alarma.

Tabla4 Descripción de la clase cmpnotificacionesModel.

Clases Domain:

Nombre:dat_notificación	
Tipo de clase: Domain	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Buscar($idnotificacion)	Devuelve la notificación dada una estructura determinada.
GetTodos()	Muestra todas las notificaciones.
cargarNotificaciones($idusuario, $idusuariorelacionado,$asunto,$idaccion,$arraytipoelem,$fecha,$autores,$limit,$start)	Carga las notificaciones del usuario en sesión.
Insertar($obj)	Inserta una nueva notificación.
eliminarNotificacionDB($idnotificacion)	Elimina una notificación.

Tabla 5 Descripción de la clase dat_notificacion.

Nombre:dat_usuario_notificacion	
Tipo de clase: Domain	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
eliminarUserNotificacion($idusuario, $idusuariorelacionado,$idnotificacion)	Elimina de la bandeja de entrada la notificación correspondiente a un usuario.
Insertar($obj)	Relaciona un usuario a una notificación.
actualizarUserNotificacion($idusuario, $idusuariorelacionado, $idnotificacion)	Actualiza los usuarios relacionados con una notificación.

Tabla 6 Descripción de la clase dat_usuario_notificacion.

Nombre:dat_historial	
Tipo de clase: Domain	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Buscar($idhistorial)	Devuelve el historial dada una estructura determinada.
Insertar($obj)	Inserta un historial con la fecha de la notificación, su autor, la acción que la generó y el elemento sobre el que se realizó la acción.

Tabla 7 Descripción de la clase dat_historial.

Nombre:nom_accion	
Tipo de clase: Domain	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Buscar($idaccion)	Devuelve la acción dada una estructura determinada.

Tabla 8 Descripción de la clase nom_accion.

Nombre:dat_alerta	
Tipo de clase: Domain	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Buscar($idalerta)	Devuelve la alerta dada una estructura determinada.
ExisteAlerta($idactividad)	Permite conocer si existe una alerta para esa actividad.
EliminarAlerta($idalerta)	Elimina una alerta.
ActualizarAlerta($idactividad,$fecha,$contenido)	Actualiza la fecha y contenido de la alerta correspondiente a una actividad dada.
BuscarAlertas()	Busca periódicamente si hay alertas pendientes.

Tabla 9 Descripción de la clase dat_alerta.

3.4 Prototipos de interfaz de usuario

Prototipos de interfaz de usuarios asociados al componente Alertas y Avisos.

Figura 14 Prototipo de interfaz de usuario para la funcionalidad Bandeja de entrada.

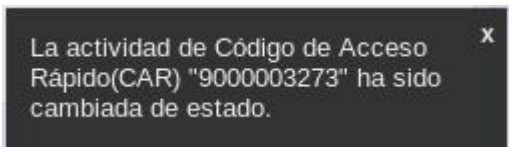

Figura 15 Prototipo de interfaz de usuario para la funcionalidad Notificación en tiempo real.

3.5 Estrategia de integración

En cada uno de los componentes del sistema el flujo de datos que va desde la vista hacia el modelo y viceversa, responde completamente a una estrategia de integración vertical concebida sobre 4 nodos y a partir de cada uno de los elementos arquitectónicos definidos.

El primer nodo se sitúa entre la vista y el controlador, el segundo está entre el controlador y el modelo, el tercero vincula el modelo con el framework doctrine y el último se encuentra entre la base de datos y el doctrine.

Vista – Controlador: los datos recogidos en un formulario son enviados al Controlador haciendo uso del protocolo de comunicación HTTP a través del método “post” para ser procesados y los resultados son enviados por el controlador a la vista en un JSON a través del método “echo”.

Controlador – Modelo: el Controlador toma los datos recibidos desde la vista, instancia una determinada clase del modelo y llama a uno de sus métodos, pasándole como parámetros los datos recibidos.

Modelo – Doctrine: el Modelo utiliza llamadas a métodos de Doctrine que le permitan crear, modificar, eliminar o actualizar los datos almacenados en las tuplas de la base de datos.

Doctrine – Base de Datos: doctrine ejecuta las consultas a la Base de Datos utilizando programación orientada a objetos.

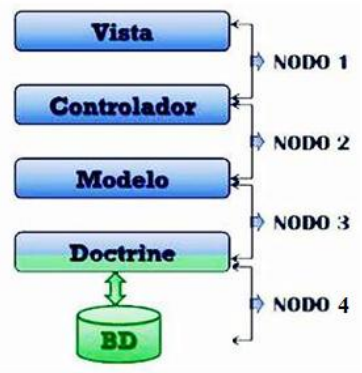

Figura 16 Nodos involucrados en la integración

La comunicación dentro de un mismo componente se ejecuta de forma directa, sin embargo, la que se establece entre los diferentes componentes y subsistemas de la aplicación va mas allá de un simple llamado a un servicio, esta se basa en el empleo de un registro de datos de los subsistemas contenidos en un fichero xml mapeado por el framework para el funcionamiento del componente llamado: inversión de control (IoC). El IoC registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema, especifica respuestas deseadas a sucesos o solicitudes de datos concretas, orden necesario y el conjunto de sucesos que tienen que ocurrir según los parámetros requeridos para poder hacer uso de un determinado servicio.

Servicios que recibe el componente como parte de la integración:

Servicios externos

- **nombreUsuario:** servicio brindado por el subsistema Seguridad que le permite al componente Alertas y avisos obtener el nombre del usuario en sesión el cual es el autor de la notificación y los usuarios que recibirán la misma.
- **obtenerConfDocEstado:** servicio brindado por el subsistema Configuración para obtener el estado de un elemento de la planificación.

Servicios internos

- **dameUsuariosDeGrupo:** servicio brindado por el componente Configuración del subsistema Planificación que muestra los usuarios pertenecientes a un grupo determinado.
- **dameInvolucradosDadoIdUsuario:** servicio brindado por el componente Planeación del subsistema Planificación que devuelve los usuarios que han sido involucrados por un usuario dado a un elemento de la planificación.
- **obteneridusuariosDadoInvol:** servicio brindado por el componente Planeación del subsistema Planificación que permite conocer el identificador de los usuarios involucrados.
- **obtenerCriterioMedidaPorId:** servicio brindado por el componente Planeación del subsistema Planificación que permite obtener el criterio de medida de un elemento de la planificación.
- **obtenerElementoDadoId:** servicio brindado por el componente Planeación del subsistema Planificación que permite obtener un elemento dado su identifircador.

Servicio que brinda el componente como parte de la integración:

- **generarNotificacion:** servicio que brinda el componente Alertas y avisos que permite generar un conjunto de notificaciones.
- **eliminarArrayNotificaciones:** servicio que brinda el componente Alertas y avisos que permite eliminar un conjunto de notificaciones.

3.6 Diagrama de componentes

A continuación se presenta el Mapa de componentes que responde al funcionamiento del SIPAC (12), el cual está estructurado de acuerdo a los diferentes niveles de empaquetamiento de la Arquitectura Base: subsistema, componente, funcionalidad y requisito; punto de partida de la solución para alertas y avisos.

La solución propuesta para el SIPAC está enmarcada en el componente Notificación, el cual permitirá gestionar las acciones que generarán una notificación. El mismo se encarga del manejo de las notificaciones generadas en tiempo real, lo cual que permitirá notificar al usuario activo sobre la asignación o alteración de algún elemento de planificación, aun cuando no se encuentre activo en el sistema (o sea, cuando no haya iniciado sección). Se relaciona a través de interfaces de comunicación con los componentes Configuración y Planeación que se encuentran dentro del subsistema de Planificación y el subsistema Seguridad que se encuentra fuera del mismo.

Especificación de los niveles de empaquetamiento subsistema y componente:
Subsistema Planificación: permite el registro, seguimiento y control de la Planificación Estratégica y Operativa para todos los niveles organizacionales. Dicho subsistema contiene los componentes:

- *Planeación*: permite la gestión de los elementos de la planificación: Planes, Actividades, Objetivos, Factores que Influyen en Plan (FIP) y Áreas de Resultados Clave (ARC).
- *Configuración*: brinda las funcionalidades para crear grupos de usuarios a partir de los usuarios sobre los cuales se puede planificar, teniendo en cuenta el dominio de acceso y la estructura definida configurados en los subsistemas Seguridad y Estructura respectivamente.
- *Notificaciones*: permite la generación de notificaciones a partir de las acciones que se realizan sobre los diferentes elementos primarios de la planificación.

*Subsistema Configuración: b*rinda una serie de funcionalidades que deben ser ejecutadas antes de comenzar a utilizar el resto de los subsistemas, específicamente para el funcionamiento del sistema SIPAC es necesario definir el flujo de información de los elementos de la planificación a partir de estados de los documentos y las transiciones el cual es posible que a través del componente: WorkFlow.

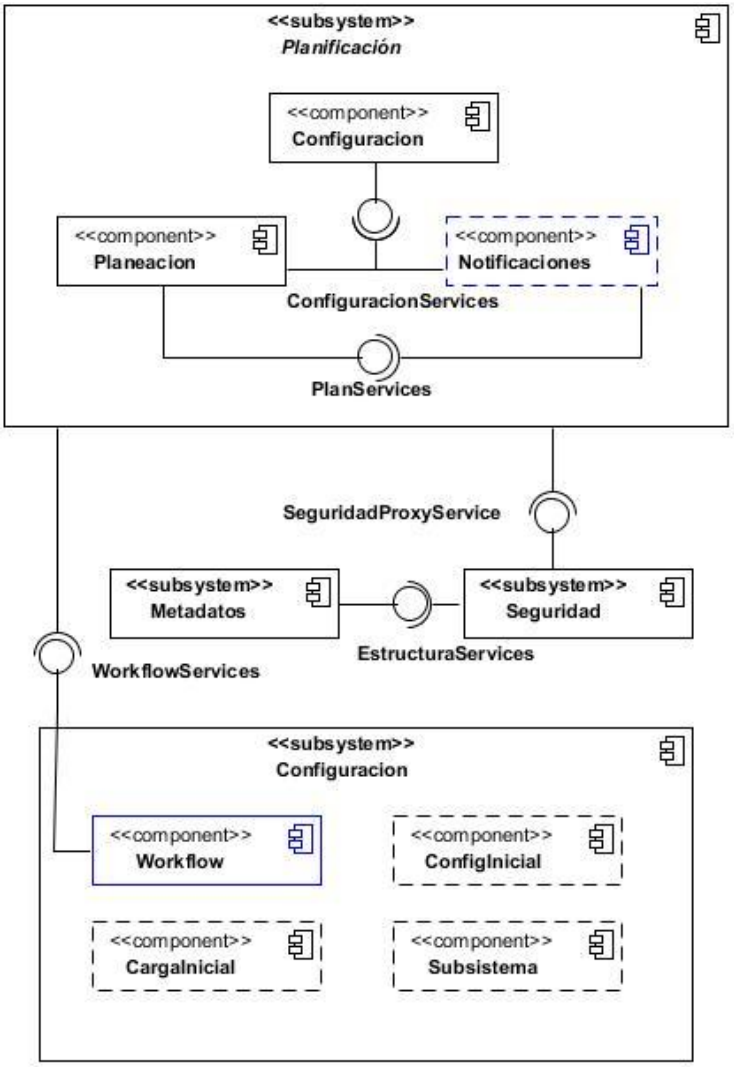

Figura 17 Diagrama de componentes del subsistema SIPAC

3.7 Diagrama de despliegue

El componente se adecúa al despliegue de SIPAC. Existen dos posibles escenarios para el despliegue de la aplicación.

- Diagrama de despliegue de escenario para PC Cliente con disco(12).

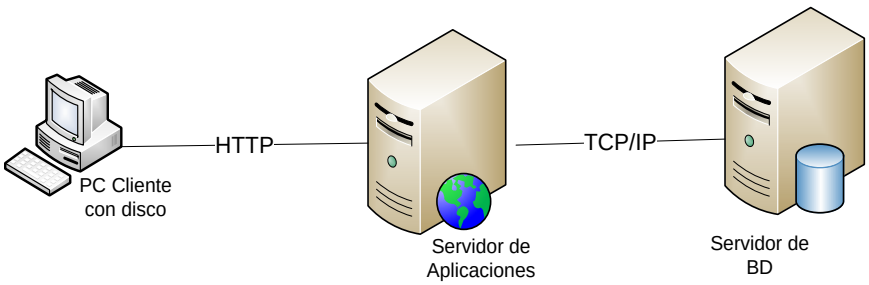

Figura 18 Esquema estructural para PC Cliente con disco.

- Diagrama de despliegue de escenario para PC cliente sin disco (condiciones básicas de hardware)(12).

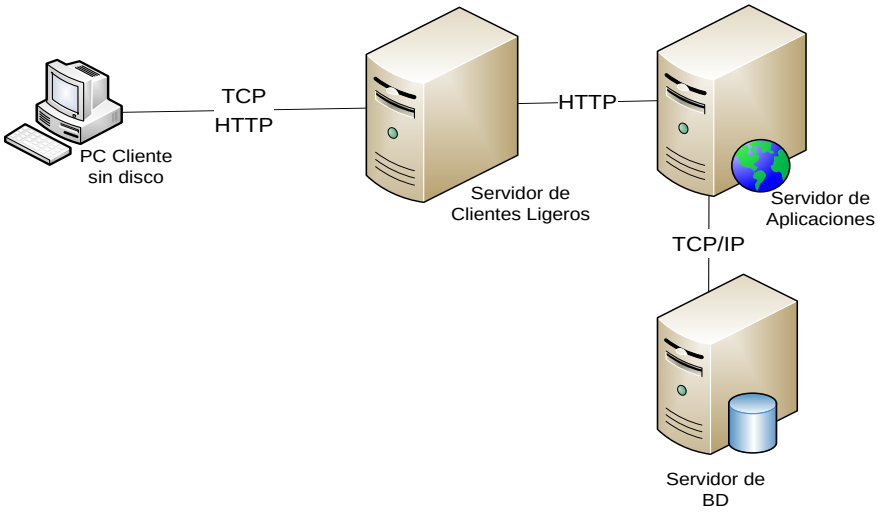

Figura 19 Esquema estructural para PC Cliente sin disco.

3.8 Validación del modelo de diseño propuesto

Con el objetivo de comprobar cuán bien están definidas las clases, se emplean las métricas TOC y RC, diseñadas para evaluar los siguientes atributos de calidad:

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.
- **Complejidad de implementación**: grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización**: grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento**: grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento**: grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.
- **Cantidad de pruebas**: número o grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

3.8.1 Métrica Tamaño Operacional de Clase

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase.

Al aplicar la métrica se tuvieron en cuenta las clases Controladoras, Modelos y del Domain reflejadas en el diseño de acuerdo a la propuesta planteada, así como la cantidad de procedimientos (métodos) por cada una de ellas.

Para determinar el valor de los atributos, se calcula el promedio de la columna cantidad de procedimientos y este promedio es el que se emplea en la columna criterio (Ver artefacto Validación del diseño en el expediente de proyecto).

La tabla que se muestra a continuación ofrece las clases del sistema a las que se le aplicó la métrica y los resultados obtenidos para cada atributo evaluado.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
CmpnotificacionesController	6	Baja	Baja	Alta
CmpnotificacionesModel	17	Alta	Alta	Baja
NomAccion	3	Baja	Baja	Alta
DatUsuarioNotificacion	6	Baja	Baja	Alta
DatNotificacion	7	Baja	Baja	Alta
DatAlerta	7	Baja	Baja	Alta

Figura 20 Resultado obtenido por clases.

Como resultado de la evaluación de la métrica TOC se obtuvo lo siguiente:

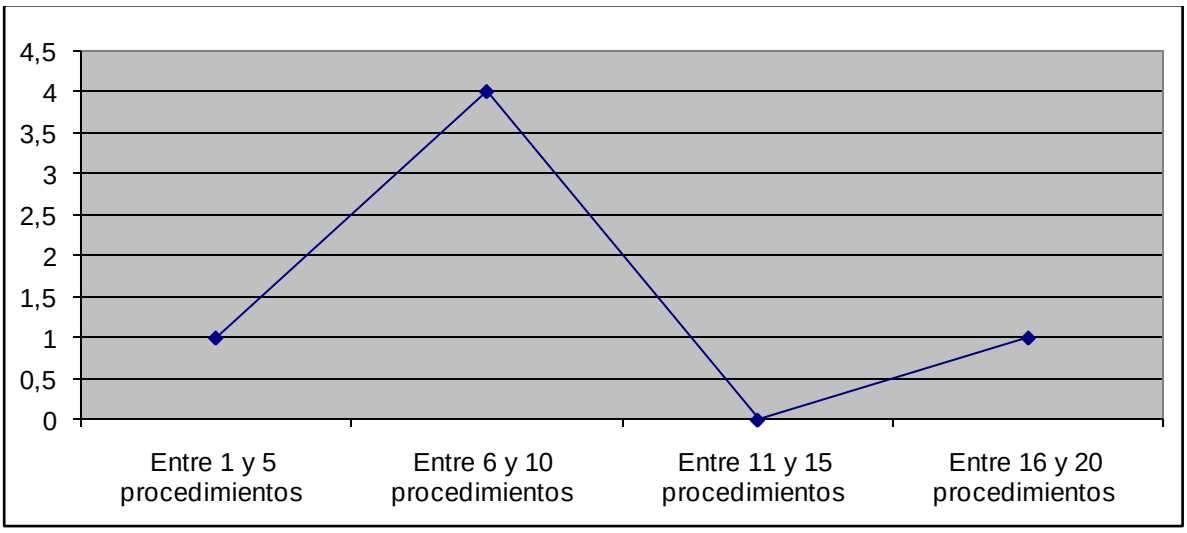

Figura 21 Representación de la cantidad de clases y el número de procedimientos que contienen.

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos:

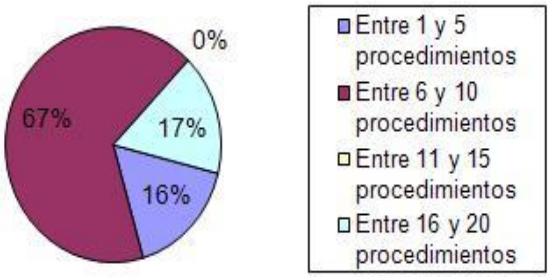

Figura 22 Representación en % de la cantidad de clase y el número procedimientos que contienen.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Responsabilidad**:

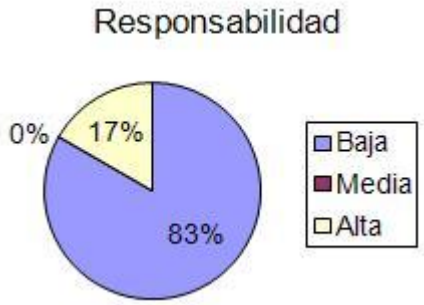

Figura 23 Representación del valor en % del atributo responsabilidad.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Complejidad de Implementación.**

Figura 24 Representación del valor en % del atributo Complejidad de Implementación.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Reutilización.**

Figura 25 Representación del valor en % del atributo reutilización.

Análisis de los resultados obtenidos en la evaluación de la métrica TOC

Considerando que el 83% de las clases contienen un número de funcionalidades inferior a la media de procedimientos por clases (cuyo valor asciende a 7.8) se puede afirmar que el diseño de clases elaborado se encuentra dentro de los límites de calidad. El 83% de las clases que intervienen en el diseño tienen un nivel alto de Reutilización.

3.8.2 Métrica Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad (Ver artefacto Validación del diseño en el expediente de proyecto):
La siguiente tabla muestra las clases a las cuales se le aplicó la métrica RC en la columna **Clase**, la cantidad de relaciones que tiene cada una respecto a las demás clases en la columna **Cantidad de Relaciones de Uso**, junto a los atributos de calidad de Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas en sus columnas correspondientes. Al promediar la cantidad de relaciones que tiene cada clase se obtuvo un resultado de relaciones de dependencia por clase de 3.6 el cual se utiliza en la tabla anterior para hallar el valor de la columna criterio.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
CmpnotificacionesController	2	Medio	Baja	Alta	Baja
CmpnotificacionesModel	12	Alto	Alta	Baja	Alta
NomAccion	1	Bajo	Baja	Alta	Baja
DatUsuarioNotificacion	1	Bajo	Baja	Alta	Baja
DatNotificacion	2	Medio	Baja	Alta	Baja
DatAlerta	2	Medio	Baja	Alta	Baja

Figura 26 Resultados de la aplicación de la métrica RC para cada clase del sistema.

Como resultado de la evaluación de la métrica TOC se obtuvo lo siguiente:

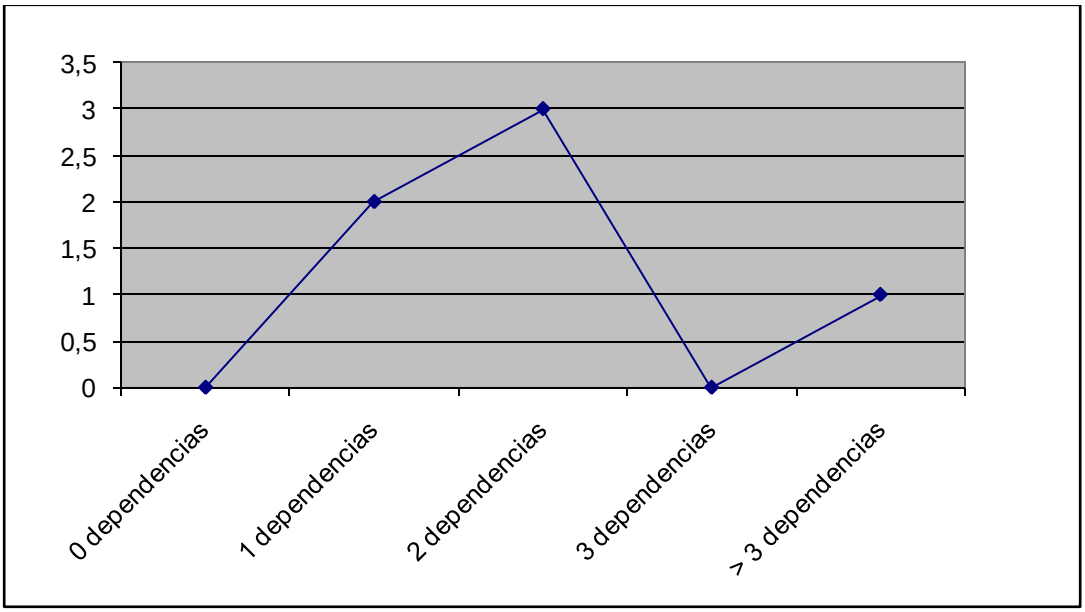

Figura 27 Representación de la cantidad de clases y el número de dependencias que contienen.

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos:

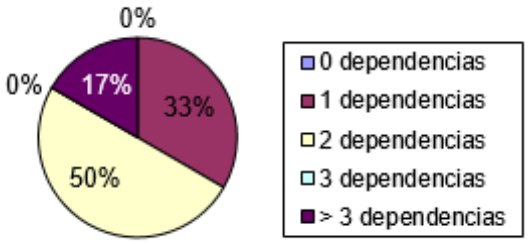

Figura 28 Representación en % de la cantidad de clase y el número procedimientos que contienen.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Acoplamiento**.

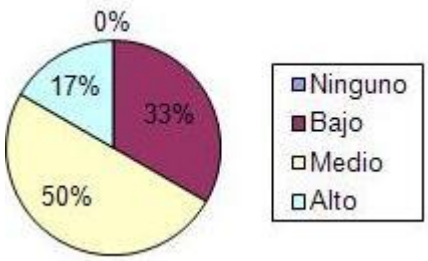

Figura 29 Representación del valor en % del atributo acoplamiento.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Complejidad de mantenimiento.**

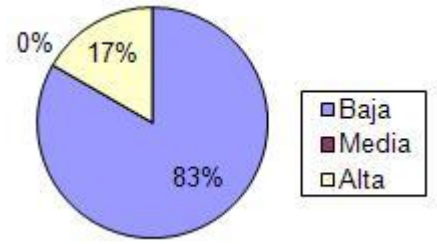

Figura 30 Representación del valor en % del atributo Complejidad de mantenimiento.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Cantidad de pruebas.**

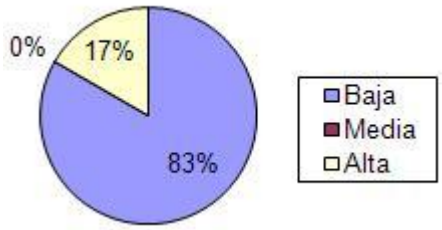

Figura 31 Representación del valor en % del atributo cantidad de pruebas.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Reutilización.**

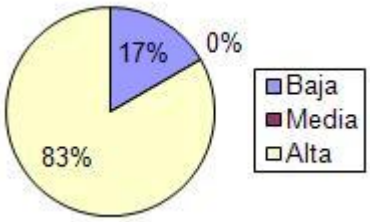

Figura 32 Representación del valor en % del atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el componente Alertas y Avisos está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (64%) poseen menos de 3 dependencias respecto a otras.

Los atributos de calidad se encuentran en un nivel satisfactorio; en el 47% de las clases el grado de dependencia o acoplamiento es medio, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 75% de las clases.

3.9 Pruebas de software

Una etapa muy importante durante el desarrollo de la aplicación la constituyen las pruebas. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad del componente; probando el comportamiento del mismo.

Entre sus objetivos están(29):

1. Detectar defectos en el software.
2. Verificar la integración adecuada de los componentes.
3. Verificar que todos los requisitos se han implementado correctamente.
4. Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
5. Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

Se realizarán pruebas de unidad por parte del equipo de desarrollo para separar y probar el correcto funcionamiento de las partes individuales de un programa y asegurar que cada uno de estos se ejecuta correctamente por separado.

3.9.1 Pruebas de caja blanca.

En la prueba de caja blanca se comprueban los componentes internos. Se examina el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. El empleo de este tipo de pruebas permitirá diseñar casos de prueba que comprueben que todas las sentencias del sistema y las condiciones, tanto verdaderas como falsas se ejecuten al menos una vez. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

Con el objetivo de valorar qué tan certera ha sido la implementación del componente Alertas y Avisos se aplica la técnica del camino básico. Para ello es necesario conocer el número de caminos independientes de un determinado algoritmo mediante el cálculo de la complejidad ciclomática. Se debe comenzar por un análisis del código, posteriormente son enumeradas cada una de las instrucciones, se construye el grafo de flujo asociado y según las fórmulas pertinentes se calcula dicha complejidad:
A continuación se analizan y enumeran las sentencias de código de uno de los procedimientos contenidos en la clase cmpnotificacionesModel, específicamente: generarNotificación, este por su

parte es el encargado de crear una notificación cuando se ejecute una acción sobre un elemento primario de la planificación.

```
public function generarNotificacion($idaccion, $idelemento, $idtipoelemento, $idautor, $fecha,$objcambiohorario) {
    if($idtipoelemento != 2 && $idtipoelemento != 5){ //1
    if(is_array($idautor)) //2
      $idautor = $idautor[0]; //3
    if($idtipoelemento == 4) { //4
        $objcriterio = $this->pIntegrator->planpdo->obtenerCriterioMedidaPorId($idelemento);   //5
        $objelemento = $this->pIntegrator->planpdo->obtenerElementoDadoId($objcriterio->idobjetivo);  //5
        $idautorelem = $objcriterio->idusuariocrea; //5
    } else { //6
        $objelemento = $this->pIntegrator->planpdo->obtenerElementoDadoId($idelemento); //6
        $idautorelem = $objelemento->idusuariocrea; //6
    } //7
    $notificacion =" ";   //8
    if ($objelemento->gennotificacion) {    //9
        $idhistorial = $this->crearHistorial($idaccion,$idelemento,$idtipoelemento,$idautor,$objcambiohorario); //10
        $notificacion = $this->crearNotificacion($idhistorial, $idaccion, $idelemento, $idtipoelemento); //10
        $arrayusuarios = $this->obtenerArrayUsuarios($idelemento, $idautorelem, $idtipoelemento,$idaccion); //10
        $this->crearNotificacionesPorUsuario($notificacion, $arrayusuarios);  //10
    } else { //11
      if(!empty($objelemento->idestadoaprob)){ //11
        $objconfdocestado = $this->integrator->configuracion->ObtenerConfDocEstado($objelemento->idtipoelemento,
        $objelemento->idestadoaprob); //12
        if (!empty($objconfdocestado[0]->notificacion)) { //13
            $objelemento->gennotificacion = 1; //14
            $objelemento->ActualizarInfElemento(); //14
            $idhistorial = $this->crearHistorial($idaccion, $idelemento, $idtipoelemento, $idautor,
            $objcambiohorario); //14
            $notificacion = $this->crearNotificacion($idhistorial, $idaccion, $idelemento, $idtipoelemento); //14
            $arrayusuarios = $this->obtenerArrayUsuarios($idelemento, $idautorelem, $idtipoelemento,$idaccion); //14
            $this->crearNotificacionesPorUsuario($notificacion, $arrayusuarios); //14
        } //15
      }
    }
    }
    return $notificacion->idnotificacion; //16
}
```

Figura 33 Fragmento de código del algoritmo generarNotificación.

Después de este paso, es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones.

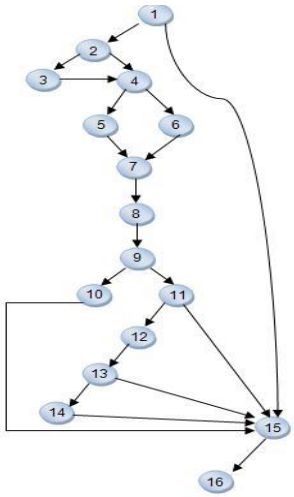

Figura 34 Grafo de flujo asociado al algoritmo generarNotificación.

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el 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 total de nodos.

V (G) = (21 - 16) + 2

V (G) = 7.

2. V (G) = P + 1

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

V (G) = 6 + 1

V (G) = 7.

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) = 7

A través del cálculo efectuado mediante las fórmulas antes presentadas se obtiene una complejidad ciclomática de valor 7, de manera que existen siete posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Posteriormente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

Camino básico #1: 1-15-16.
Camino básico #2:1-2-3-4-5-7-8-9-10-15-16.
Camino básico #3: 1-2-3-4-6-7-8-9-11-15-16.
Camino básico #4: 1-2-4-5-7-8-9-10-15-16.
Camino básico #5: 1-2-4-6-7-8-9-11-15-16.
Camino básico #6: 1-2-4-5-7-8-9-11-12-13-15-16.
Camino básico #7: 1-2-4-6-7-8-9-11-12-13-14-15-16.

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

Los elementos primarios de la planificación (planes, objetivos y actividades) tienen como idtipoelemento: 1, 3, 4 respectivamente.

1. Caso de prueba para el camino básico # 1.

Descripción: se realiza una acción sobre un elemento no primario de la planificación.
Condición de ejecución: para ello es necesario tener el identificador del elemento (idtipoelemento).
Entrada:
$idtipoelemento= 3
Resultados esperados: teniendo en cuenta que el elemento de la planificación sobre el que se realiza la acción no es primario, no se genera la notificación.
Resultado: satisfactorio.

2. Caso de prueba para el camino básico # 2.

Descripción: se realiza una acción sobre un elemento primario de la planificación y el objeto posee un estado generador de notificación.
Condición de ejecución: el elemento es un criterio de medida (idtipoelemento = 4) y la acción es generadora de notificación.
Entrada:
$idtipoelemento = 4
$objetoelemento = 900000435

Resultados esperados: se genera la notificación.
Resultado: satisfactorio.

3. Caso de prueba para el camino básico # 3.

Descripción: se realiza una acción sobre un elemento primario de la planificación, el objeto no posee un estado generador de notificación y no es cambiado de estado de aprobación.
Condición de ejecución: el elemento no es un criterio de medida, no posee un estado generador de notificación, no se realiza un cambio de estado.
Entrada:
$idtipoelemento = 3
$objetoelemento = 900000461
$idestadoaprob = ´´
Resultados esperados: no se genera la notificación.
Resultado: satisfactorio.

4. Caso de prueba para el camino básico # 4.

Descripción: se realiza una acción sobre un elemento primario de la planificación, el objeto posee un estado generador de notificación.
Condición de ejecución: la acción tiene un autor, el elemento es un criterio de medida y posee un estado generador de notificación.
Entrada:
$idtipoelemento = 4
$objetoelemento = 900000435
Resultados esperados: se genera la notificación.
Resultado: satisfactorio.

5. Caso de prueba para el camino básico # 5.

Descripción: se realiza una acción sobre un elemento primario de la planificación, el objeto no posee un estado generador de notificación y no es cambiado de estado de aprobación.
Condición de ejecución: la acción tiene un autor, el elemento no es un criterio de medida, no posee un estado generador de notificación y no es cambiado de estado de aprobación.
Entrada:
$idtipoelemento = 3
$objetoelemento = 900000475

Resultados esperados: no se genera la notificación.
Resultado: satisfactorio.

6. Caso de prueba para el camino básico # 6.

Descripción: se realiza una acción sobre un elemento primario de la planificación, el objeto no posee un estado generador de notificación y es cambiado de estado de aprobación.
Condición de ejecución: la acción tiene un autor, el elemento es un criterio de medida, no posee un estado generador de notificación y es cambiado de estado de aprobación pero el nuevo estado no es generador de notificación.
Entrada:
$idtipoelemento = 4
$objetoelemento = 900000575
Resultados esperados: no se genera la notificación.
Resultado: satisfactorio.

7. Caso de prueba para el camino básico # 7.

Descripción: se realiza una acción sobre un elemento primario de la planificación, el objeto no posee un estado generador de notificación y es cambiado de estado de aprobación a un estado generador de notificación.
Condición de ejecución: la acción tiene un autor, el elemento no es un criterio de medida, no posee un estado generador de notificación y es cambiado de estado de aprobación a un estado generador de notificación.
Entrada:
$idtipoelemento = 3
$objetoelemento = 900000578
Resultados esperados: se genera la notificación.
Resultado: satisfactorio.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función está correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.9.2 Pruebas de caja negra

Estas pruebas se realizan sobre la interfaz del software, comprobando sus funcionalidades sin tener en cuenta su estructura interna.

Este tipo de pruebas permite encontrar(29):

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Se aplica la técnica Partición de equivalencia, la cual divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

Caso de prueba de caja negra para validar el requisito funcional Notificar al usuario activo sobre la modificación de un elemento primario de la planificación.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Notificar al usuario activo sobre la modificación de un elemento primario de la planificación.	El sistema debe enviar una notificación al usuario en el momento en que un elemento de la planificación sea modificado, cambiado de estado o eliminado.	EP 1.1: Notificar cuando se modifique un elemento primario de la planificación.	1. El usuario modifica el elemento primario de la planificación. 2. El sistema muestra una notificación en bandeja de entrada y en tiempo real a los involucrados sobre la modificación de este elemento de la planificación. “La/El (id del elemento) ha sido modificado”
		EP 1.2: Notificar cuando se cambie de estado un elemento primario de la planificación.	3. El usuario cambia de estado el elemento primario de la planificación. 4. El sistema muestra una notificación en bandeja de entrada y en tiempo real a los involucrados informando el cambio de estado del elemento de la planificación. “La/El (id del elemento) ha cambiado de estado”
		EP 1.3: Notificar cuando se elimine un elemento primario de la planificación.	5. El usuario elimina el elemento primario de la planificación. 6. El sistema muestra una notificación en bandeja de entrada y en tiempo real a los involucrados informando la

	eliminación del elemento de la planificación. "La/El (id del elemento) ha sido eliminado"

Tabla 10 Caso de prueba de caja negra para validar el requisito funcional 1.

3.10 Impacto de la solución

La solución desarrollada ha sido validada mediante la realización de pruebas de caja blanca y caja negra para el nivel de unidad, a través de pruebas de liberación por el Dpto. Calidad interna del CEIGE (ver anexo: Acta de liberación) y mediante pruebas de aceptación por parte del cliente (ver anexo 3: Acta de aceptación del Cliente). Se considera que los procesos de ejecución y control de la planeación estratégica y operativa en el sistema se ven favorecidos con la incorporación de la solución para Alertas y Avisos, teniendo en cuenta que:

1. La gestión de la planificación se realiza de una manera más interactiva hacia a todos los niveles organizacionales; los usuarios son notificados sobre la alteración de los elementos primarios de la planificación, como autor o involucrado, a través de una bandeja de entrada o mediante la notificación en tiempo real si el usuario se encuentra en sesión, haciendo uso de un servidor de mensajería XMPP.
2. Permite el aprovechamiento del tiempo en cuanto a la búsqueda de la asignación de nuevas tareas, que en determinados escenarios resultaba engorrosa e innecesaria, así como para los usuarios del sistema encargados del seguimiento del desarrollo y cumplimiento de los objetivos de cada entidad.
3. Evita inconsistencias en la información debido a que se notifica si esta está siendo objeto de modificación por otro usuario concurrentemente.
4. Da la posibilidad de conciliar la fecha de alguna tarea que se pretende agregar en el sistema que coincida con otra(s) ya existente(s) en el período, de manera que el usuario pueda hacer las modificaciones pertinentes antes de la aprobación del plan.

3.11 Conclusiones parciales

La calidad del componente desarrollado fue el elemento clave del capítulo que recién concluye. En ese sentido se efectuaron pruebas de software en el nivel de unidad mediante casos de pruebas, para los cuales se tuvieron en cuenta las entradas, las salidas, los resultados esperados y el tratamiento de errores en caso de anomalías; se aplicaron además las métricas: Relaciones entre Clases y Tamaño Operacional de la Clase para validar y evaluar el diseño, las cuales arrojaron valores satisfactorios para cada uno de los indicadores correspondientes.

Capítulo 3: Implementación y validación de la solución propuesta

El componente Alertas y Avisos desde el punto de vista funcional cumple con los requerimientos capturados y especificados en las primeras etapas de desarrollo a partir de las expectativas del cliente.

CONCLUSIONES GENERALES

Se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para esto:

- Se evidencia la necesidad de realizar una solución informática capaz de ejecutar las funcionalidades referentes al envío de notificaciones a los usuarios involucrados, principalmente cuando se ejerce una acción de forma concurrente sobre un mismo elemento o se solapa la información; a partir del estudio de diferentes sistemas informáticos vinculados al envío de notificaciones en tiempo real.

- Con el objetivo de erradicar los problemas de los sistemas existentes y fusionar sus mejores prácticas, se realizó el diseño y la implementación del componente Alertas y Avisos correspondiente al Subsistema Planificación de Actividades del Sistema Integral de Gestión de Entidades CEDRUX.

- Las funcionalidades previstas para el componente se cumplieron, las mismas fueron comprobadas a través de pruebas de software efectuadas al componente, las cuales arrojaron resultados favorables.

- La solución desarrollada facilita los procesos de ejecución y control en la planeación estratégica y operativa, mediante la comunicación a los usuarios de las acciones que se realizan sobre la información a la cual tienen acceso en el sistema, así como la proximidad de determinadas tareas.

RECOMENDACIONES

Al concluir el presente trabajo de diploma, considerando cumplidos los objetivos trazados en el mismo, se recomienda:

- Continuar realizando pruebas de calidad a la solución.
- Optimizar los algoritmos para un mejor funcionamiento interno de la solución.
- Definir alertas en función de la fecha fin y el porciento de cumplimento de las tareas.
- Incorporar el envío de notificaciones vía correo.

REFERENCIAS BIBLIOGRÁFICAS

1. Internet es vital para el desarrollo de Cuba. *Juventud Rebelde.* Edición digital, 2009.

2. Diccionario de la Real Academia de la Lengua Española. [En línea] 2010. [Citado el: 20 de 11 de 2012.] http://lema.rae.es/drae.

3. Alertas de google. [En línea] 2012. [Citado el: 20 de 11 de 2012.] http://www.google.com.cu/alerts?hl=es.

4. **Rauch, Gullermo.** Socket.IO. [En línea] 2012. [Citado el: 20 de 11 de 2012.] http://socket.io.

5. NetSupport Notify. [En línea] 12 de 03 de 2009. [Citado el: 22 de 11 de 2012.] http://www.argentinawarez.com/programas-gratis/18347-netsupport-notify-v2-0-a.html.

6. **Furukawa, Y.** WebSocket.org. [En línea] 2012. [Citado el: 22 de 11 de 2012.] http://websocket.org/.

7. **Diana Rosa Pérez Santiesteban, Irelys Díaz Tellez, Irina Fuentes Viña.***Módulo de notificaciones y alertas del sistema de gestión universitaria.* UCI. La Habana : s.n., 2011.

8. **Barroso, Javier Pérez.***Sistema de notificación de eventos.* UCI, La Habana : s.n., 2011.

9. **Kinderen, Guss der.** Ignite realtime. [En línea] 2011. [Citado el: 22 de 11 de 2012.] http://www.igniterealtime.org .

10. **Díaz, Ariel Trujillo.***Componente para la gestión de notificaciones en el marco de trabajo Sauxe.* UCI, La Habana : s.n., 2012.

11. **Subdirección de producción, CEIGE.***Modelo de desarrollo de software v1.1.* UCI, La Habana : s.n., 2012.

12. **Artola, Ing. Ariadna Rendón.***CEIGE-SPA-N-i3401, Arquitectura de Software.* CEIGE : s.n., 2011.

13. Zend Framework. [En línea] 2006. [Citado el: 23 de 11 de 2012.] http://framework.zend.com/.

14. **Bullock, Christian.** ExtJS. [En línea] 2007. [Citado el: 25 de 11 de 2012.] http://extjses.com/ .

15. Doctrine. [En línea] 02 de 02 de 2008. [Citado el: 28 de 11 de 2012.] http://www.doctrine-project.org.

16. **Martínez, Rafael.** PostgreSQL-es. [En línea] 2009. [Citado el: 28 de 11 de 2012.] http://www.postgresql.org.es/.

17. Ecured. [En línea] 2012. [Citado el: 28 de 11 de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm .

18. NetBeans IDE Features. [En línea] 2011. [Citado el: 29 de 11 de 2012.] https://netbeans.org/features/index.html.

19. The Apache Software Foundation. [En línea] 2012. [Citado el: 29 de 11 de 2012.] http://www.apache.org/.

20. Mozilla Firefox. [En línea] 2012. [Citado el: 01 de 12 de 2012.] http://www.mozilla.org/es-ES/firefox/features/.

21. **Larman, C.***UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Madrid : Pearson Educación S.A, 2003. 2da edición.

22. Ecured. [En línea] 2012. [Citado el: 01 de 12 de 2012.] http://www.ecured.cu/index.php/Lenguaje_de_Programaci%C3%B3n.

23. **Gómez, Ángel Cobo y Patricia.***PHP y MySQL- tecnologias para el desarrollo de aplicaciones web.* s.l. : Ediciones Díaz Santos, 2005.

24. **Quintero, Carlos J.***JavaScript.* Panamá : s.n., 2011.

25. **Kicillof, Carlos Billy Reinoso y Nicolás.***Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires : s.n., 2004.

26. **Gamma, Erich.***Design Patterns: Elements of Reusable Object-Oriented Software. .* s.l. : Addison-Wesley, 1999.

27. **Pressman, Roger.***Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill Companies, 2002. Quinta edición.

28. Introducción a la programación. [En línea] 2009. [Citado el: 05 de 02 de 2013.] http://alexgordillo.blogspot.com/2008/02/tipos-de-estructuras-de-datos.html.

29. Ecured. [En línea] 2013. [Citado el: 10 de 05 de 2013.] http://www.ecured.cu/index.php/Pruebas_de_software.

BIBLIOGRAFÍA

1. Internet es vital para el desarrollo de Cuba. *Juventud Rebelde.* Edición digital, 2009.

2. Diccionario de la Real Academia de la Lengua Española. [En línea] 2010. [Citado el: 20 de 11 de 2012.] http://lema.rae.es/drae.

3. Alertas de google. [En línea] 2012. [Citado el: 20 de 11 de 2012.] http://www.google.com.cu/alerts?hl=es.

4. **Rauch, Gullermo.** Socket.IO. [En línea] 2012. [Citado el: 20 de 11 de 2012.] http://socket.io.

5. NetSupport Notify. [En línea] 12 de 03 de 2009. [Citado el: 22 de 11 de 2012.] http://www.argentinawarez.com/programas-gratis/18347-netsupport-notify-v2-0-a.html.

6. **Furukawa, Y.** WebSocket.org. [En línea] 2012. [Citado el: 22 de 11 de 2012.] http://websocket.org/.

7. **Diana Rosa Pérez Santiesteban, Irelys Díaz Tellez, Irina Fuentes Viña.***Módulo de notificaciones y alertas del sistema de gestión universitaria.* UCI. La Habana : s.n., 2011.

8. **Barroso, Javier Pérez.***Sistema de notificación de eventos.* UCI, La Habana : s.n., 2011.

9. **Kinderen, Guss der.** Ignite realtime. [En línea] 2011. [Citado el: 22 de 11 de 2012.] http://www.igniterealtime.org .

10. **Díaz, Ariel Trujillo.***Componente para la gestión de notificaciones en el marco de trabajo Sauxe.* UCI, La Habana : s.n., 2012.

11. **Subdirección de producción, CEIGE.***Modelo de desarrollo de software v1.1.* UCI, La Habana : s.n., 2012.

12. **Artola, Ing. Ariadna Rendón.***CEIGE-SPA-N-i3401, Arquitectura de Software.* CEIGE : s.n., 2011.

13. Zend Framework. [En línea] 2006. [Citado el: 23 de 11 de 2012.] http://framework.zend.com/.

14. **Bullock, Christian.** ExtJS. [En línea] 2007. [Citado el: 25 de 11 de 2012.] http://extjses.com/ .

15. Doctrine. [En línea] 02 de 02 de 2008. [Citado el: 28 de 11 de 2012.] http://www.doctrine-project.org.

16. **Martínez, Rafael.** PostgreSQL-es. [En línea] 2009. [Citado el: 28 de 11 de 2012.] http://www.postgresql.org.es/.

17. Ecured. [En línea] 2012. [Citado el: 28 de 11 de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm .

18. NetBeans IDE Features. [En línea] 2011. [Citado el: 29 de 11 de 2012.] https://netbeans.org/features/index.html.

19. The Apache Software Foundation. [En línea] 2012. [Citado el: 29 de 11 de 2012.] http://www.apache.org/.

20. Mozilla Firefox. [En línea] 2012. [Citado el: 01 de 12 de 2012.] http://www.mozilla.org/es-ES/firefox/features/.

21. **Larman, C.***UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Madrid : Pearson Educación S.A, 2003. 2da edición.

22. Ecured. [En línea] 2012. [Citado el: 01 de 12 de 2012.] http://www.ecured.cu/index.php/Lenguaje_de_Programaci%C3%B3n.

23. **Gómez, Ángel Cobo y Patricia.***PHP y MySQL- tecnologias para el desarrollo de aplicaciones web.* s.l. : Ediciones Díaz Santos, 2005.

24. **Quintero, Carlos J.***JavaScript.* Panamá : s.n., 2011.

25. **Kicillof, Carlos Billy Reinoso y Nicolás.***Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires : s.n., 2004.

26. **Gamma, Erich.***Design Patterns: Elements of Reusable Object-Oriented Software. .* s.l. : Addison-Wesley, 1999.

27. **Pressman, Roger.***Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill Companies, 2002. Quinta edición.

28. Introducción a la programación. [En línea] 2009. [Citado el: 05 de 02 de 2013.] http://alexgordillo.blogspot.com/2008/02/tipos-de-estructuras-de-datos.html.

29. Ecured. [En línea] 2013. [Citado el: 10 de 05 de 2013.] http://www.ecured.cu/index.php/Pruebas_de_software .

30. Postgres SQL Cuba. [En línea] UCI, 2011. [Citado el: 20 de diciembre de 2012.] http://postgresql.uci.cu.

31. PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 21 de diciembre de 2012.] http://www.postgresql.org.es/node/655.

32. Visual-paradigm. [En línea] [Citado el: 22 de diciembre de 2012.] http://www.visual-paradigm.com.

33. **Álvarez González, Larisa y Roberkys Martín Cruañes.***Arquitectura de software del Sistema de Gestión de Información de los Recursos de la Facultad 3.* Universidad de las Ciencias Informáticas : s.n., 2013.

34. Prácticas de Software. [En línea] 2011. [Citado el: 12 de 04 de 2013.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp.

35. **Hernández González, Dra. Anaisa.***Un método para el diseño de la base de datos a partir del modelo orientado a objetos.* México DF : Computación y Sistemas, 2004.

36. **Pérez, Wilson y Hilbert, Martin.***La sociedad de la información en América Latina y el Caribe. Desarrollo de las tecnologías y tecnologías para el desarrollo. .* Santiago de Chile : Impreso en Naciones Unidas, 2009.

37. **Pérez Martinto, Pedro Carlos.***El diseño metodológico de la investigación científica. .* Universidad de las Ciencias Informáticas : s.n., 2010.

38. **A, Guido R. Rolón.** Zentyal. [En línea] 2012. [Citado el: 23 de 03 de 2013.] http://trac.zentyal.org/wiki/Documentation/Community/Spanish/HowTo/Como%20conectar%20OpenFire%20a%20un%20servidor%20Zentyal%20con%20OpenLDAP.

GLOSARIO DE TÉRMINOS

Acción: Modificación realizada sobre algún elemento de la planificación.

Actividad: conjunto de operaciones o tareas propias de una persona o entidad.

Alerta: mensaje de para comunicar la proximidad de una tarea.

ARC: Área de resultados claves, agrupación de varias especialidades o entidades con un mismo propósito que cumplir.

Cambio de estado: tipo de modificación realizada sobre algún elemento de la planificación.

Elemento de la planificación: agrupa a los elementos que se asociados a la planificación estos pueden ser: planes, objetivos y actividades.

Eliminar: tipo de modificación realizada sobre algún elemento de la planificación.

FIP: factores que Influyen en la Planificación, procesos que pueden tener lugar y que se materializan en elementos que influyen en elementos existentes de la planificación.

Involucrado: personas que participan o son responsables de los elementos de la planificación, las ARC o los FIP.

Involucrar: tipo de modificación realizada sobre algún elemento de la planificación.

Modificación: Tipo de modificación realizada sobre algún elemento de la planificación.

Objetivo: meta a alcanzar que se trace cada entidad.

Plan: modelo sistemático que se elabora antes de realizar una acción, con el propósito de dirigirla y encauzarla. En este sentido, un plan también es un escrito que precisa los detalles necesarios para realizar una misión.

Reporte: información del sistema presentada al usuario en un formato previamente definido.

Notificación: se refiere a las notificaciones generadas cuando se han realizado cambios a los elementos de la planificación.