Universidad de las Ciencias Informáticas

FACULTAD 3

Título: **"Desarrollo de la solución para la obtención de gráficos del Sistema de Planificación de Actividades SIPAC"**

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

Autor: Antonio Díaz Cedeño

Tutor(es): Ing. Dionny Cardoso Carmona

Co-tutor: Ing. Maria Teresa Rosales González

Ing. Yordi Chaveco Bustamante

La Habana, junio de 2013

El más terrible de los sentimientos es el sentimiento de tener la esperanza perdida.

Federico García Lorca.

DECLARACIÓN DE AUTORÍA

Yo Antonio Díaz Cedeño declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Antonio Díaz Cedeño

_____

Firma del Autor

Ing. Dionny Cardoso Carmona

_____

Firma del Tutor

Ing. Maria Teresa Rosales González

_____

Firma del co-Tutor

Ing. Yordi Chaveco Bustamante

_____

Firma del co-Tutor

DATOS DE CONTACTO

Tutores:

Tutor: Ing. Dionny Cardoso Carmona
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Categoría docente: Instructor
Años de experiencia en el tema: 5
Años de graduado: 5
Correo Electrónico: dionny@uci.cu

Co-tutor: Ing. María Teresa Rosales González
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Categoría docente: ----
Años de experiencia en el tema: 1
Años de graduado: 1
Correo Electrónico: mtrosales@uci.cu

Co-tutor: Ing. Yordi Chaveco Bustamante
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Categoría docente: ----
Años de experiencia en el tema: 1
Años de graduado: 1
Correo Electrónico: ybustamante@uci.cu

AGRADECIMIENTOS

Antes de todo quiero agradecer a mis padres por todo su amor, dedicación y confianza depositada en mi durante estos largos años, sin ellos no hubiese sido posible realizar este sueño.

A mis hermanos Danielito y Luisito por siempre estar presentes en las buenas y las malas.

A mi amigo Disnel por todo su apoyo durante la carrera y comportarse siempre como un hermano.

A mis tutores Teresa y Dionny por toda la ayuda brindada, y ser amigos sobre todas las cosas.

A mis compañeros del aula, por todas las horas compartidas de estudio, tristezas y felicidad.

A la gente del proyecto por darme la oportunidad de compartir con ustedes y ayudarme siempre que lo necesité.

A todos los que de una forma u otra estuvieron presentes en este largo camino y forman parte de mi crecimiento como persona y profesional.

A Dios por verme dado la oportunidad de estar aquí hoy, de agradecerles a todos ustedes.

DEDICATORIA

A mis padres, Nancy y Daniel por servirme de ejemplos de humildad y sencillez,
por enseñarme a nunca perder la esperanza cuando todo parezca perdido,
por todo su apoyo, afecto y amor desinteresado en cada etapa de la vida,
por su preocupación y confianza depositada en mí en cada minuto de estos 5 años,
por servirme de guías y ejemplos de persistencia, en lo que se quiere lograr,
por enseñarme a comprender que no existe nada imposible, solo es proponérselo.

A mi hermano Danielito por todos sus consejos, preocupación, y dedicación demostrada.

A mi hermano Luisito por su cariño, sencillez y humildad en su forma de ser.

RESUMEN

Con el objetivo de informatizar el proceso de planificación estratégica y operativa adoptados por las entidades cubanas, se desarrolla el Sistema de planificación de actividades SIPAC, el cual permite realizar la planificación de actividades. Independientemente de las funcionalidades que brinda SIPAC, en la actualidad no es posible agrupar ni graficar los elementos de la planificación en indicadores claves. Para solucionar este problema fue realizado un estudio de algunos sistemas informáticos que proporcionan generación de gráficos, así como de las diferentes bibliotecas para graficar que utilizan los mismos, posibilitando adquirir experiencia acerca de la obtención de gráficos y las tecnologías utilizadas para este propósito. Lo anteriormente expuesto evidenció la necesidad de desarrollar una solución para la obtención de gráficos en el sistema de planificación de actividades SIPAC. Con el desarrollo de la solución para la obtención de gráficos se garantiza la representación de la información sobre la composición y el estado en que se encuentran los elementos de la planificación, el porciento de cumplimiento de los elementos asociados en un mismo entorno y el total de involucrados por cada uno de ellos, posibilitando analizar y estudiar de forma dinámica la información relacionada con los indicadores claves.

Palabras claves:

Indicadores de la planificación, gráfico, modelo de planificación estratégica u operativa, sistema de planificación de actividades.

Índice

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

A partir de la implementación paulatina de los lineamientos aprobados en el VI Congreso del Partido Comunista de Cuba (PCC), se ha evidenciado en el país el surgimiento de un conjunto de transformaciones económicas, organizacionales y sociales, propiciando la concepción de nuevos mecanismos que permiten el desarrollo de la totalidad de los procesos en las instituciones cubanas. Derivado de estas nuevas concepciones, se hace necesario crear un mecanismo que permita realizar una planificación que se adapte a las necesidades cubanas. Que se rija por un conjunto de indicadores que posibiliten la interacción con los elementos de la planificación, denominada "Planificación estratégica y operativa de actividades".

La planificación estratégica y operativa constituye un nuevo modelo de planificación que tiene como principal función, planificar y organizar los procesos de: establecimiento, aprobación y puntualización de los planes, sus sistemas de control y factores que influyen en dichos planes. El mismo se va a ocupar de establecer resultados finales hacia los cuales se dirigen las actividades organizacionales e individuales, en aras de cumplir los objetivos económicos y sociales que demanda el desarrollo integral de la sociedad cubana. La puesta en funcionamiento en las entidades de este nuevo modelo de planificación, posibilita observar organizadamente la totalidad de las actividades a cumplir por parte de los involucrados. Permitiendo una planificación eficaz y una petición más acertada de los recursos; lo que posibilita que se defina una prioridad en los objetivos trazados por la entidad (Artola, 2013).

Las actividades de planificación por objetivos no han quedado exentas del proceso de informatización en el que se encuentra enfrascado el país. Por lo cual la dirección del estado cubano en conjunto con la Universidad de las Ciencias Informáticas (UCI), decidieron desarrollar un producto informático que cumpla con estos fines. Esta tarea fue llevada a cabo por un grupo de especialistas, profesores y estudiantes del Centro de Gestión de la Informatización de Entidades (CEIGE) perteneciente a la Facultad 3 de esa universidad, arrojando como principal resultado la primera versión del Sistema de Planificación de Actividades (SIPAC).

Dicho sistema 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, diseñada sobre las bases de la compartimentación de la información. Independientemente de las funcionalidades con las que cuenta el sistema actualmente no es posible:

Representar gráficamente la información en el sistema agrupada por indicadores claves como: cantidad de involucrados, el porciento de cumplimiento, la composición y el estado de cumplimiento de los elementos de la planificación.

Efecto:

Se dificulta el seguimiento de las tareas principales de la entidad como parte de procesos de ejecución y control de la planeación estratégica u operativa, de manera que los usuarios del sistema encargados del análisis del cumplimiento de los objetivos deben realizar búsquedas que pueden resultar engorrosas para determinar el desglose de los elementos de la planificación, así como el cumplimiento de los mismos. Por otra parte el cúmulo de información que deben consultar puede comprometer la interpretación de la información.

Teniendo en cuenta lo antes mencionado, se define como **problema a resolver:** En el Sistema para la Planificación de Actividades (SIPAC) no se realiza la representación gráfica de los indicadores clave de la planificación, lo que dificulta los procesos de Ejecución y Control de la Planeación Estratégica u Operativa.

Con vista a darle solución al problema planteado se definió como **objeto de estudio:** la generación de gráficos en sistemas informáticos. Enmarcándose en el **campo de acción:** la generación de gráficos en sistemas de gestión empresarial.

Idea a defender: Con el desarrollo de la solución para la obtención de gráficos 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, mediante la representación gráfica de los indicadores clave de la planificación.

Para darle solución al problema antes descrito se define como **objetivo general** de la investigación: Desarrollar la solución para la obtención de gráficos en el Sistema de Planificación de Actividades (SIPAC), de manera que facilite la realización de los procesos de Ejecución y Control de la Planeación Estratégica y Operativa; quedando desglosado este en los siguientes **objetivos específicos:**

1. Realizar la elaboración del marco teórico para fundamentar la investigación.
2. Realizar el análisis y diseño para obtener y describir la solución a implementar teniendo en cuenta las necesidades del cliente.
3. Implementar la solución que responda a las necesidades del cliente.
4. Validar la solución mediante la utilización de las técnicas de Caja blanca y Caja negra.

Para darle cumplimiento a estos objetivos específicos, se definen las siguientes **tareas de la investigación:**

1. Investigación y análisis de la generación de gráficos en sistemas informáticos y bibliotecas.
2. Estudio de Arquitectura Base definida en el proyecto.
3. Estudio del Modelo de Desarrollo definido por el centro CEIGE.
4. Elaboración del Modelo Conceptual.
5. Identificación de los requisitos de software.
6. Especificación de requisitos de software
7. Validación de los requisitos de software identificados.
8. Elaboración de los Diseños de Casos de Prueba.
9. Actualización del Modelo de datos.
10. Definición de los prototipos de Interfaz de Usuario.
11. Elaboración de los diagramas de clases del diseño.
12. Elaboración de los diagramas de interacción de acuerdo a los requisitos definidos.
13. Validación del Diseño a través de métricas.
14. Actualización del Modelo de componentes del sistema.
15. Elaboración del modelo de despliegue.
16. Implementación de las funcionalidades del sistema.
17. Validación de la solución mediante Casos de prueba.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

1.1 Introducción

En el presente capítulo se describen las principales características de algunas soluciones, tanto nacionales como internacionales, que implementan alguna variante de representación de indicadores mediante el uso de gráficos. Así mismo se detallan un conjunto de elementos que caracterizarán y orientarán el desarrollo de la solución, teniendo como base el Modelo de desarrollo de CEIGE. Además se ofrece una visión de las tecnologías y las herramientas que se utilizarán durante el desarrollo de la investigación.

1.2 Estado del arte de sistemas que generan gráficos

Como elemento importante de esta investigación, se realizará un estudio y caracterización de algunos sistemas informáticos tanto de nivel nacional como internacional que proporcionan la generación de gráficos para el análisis de información.

1.2.1 Selenne ERP

Constituye un sistema de gestión empresarial desarrollado por la empresa *SynerPlus.* Tiene como principal objetivo optimizar la gestión empresarial a través de sus soluciones. Selenne ERP brinda la posibilidad de obtener gráficos en formato de barras, pastel, líneas y pirámide a través de su solución *BPM*[1]*V3* mediante la utilización de Cuadros de Mando, tecnología que forma parte de las soluciones que implementan *Business Intelligence*[2]. Lo que ofrece la posibilidad de realizar un análisis de los aspectos claves del negocio, así como generar de forma dinámica la información; permitiendo extraer de forma inmediata todos los datos. Esto constituye una nueva forma de interactuar con la información, debido a que su análisis se realiza dinámicamente y los gráficos se pueden realizar en 2D y 3D. Selenne ERP esta desarrollado con tecnología .NET y trabaja sobre una base de datos SQLServer, con la implementación de un *Datawarehouse*[3]. (SynerPlus, 2012)

Observación: Selenne ERP presenta como principal inconveniente que su licencia es propietaria y las herramientas en las cual está implementado también lo son, por lo que se dificulta su adaptación,

[1] *Business Process Management*

[2] Conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos existentes en una organización o empresa, estrategias a seguir, así también como de sus resultados (Universidad Experimental Simón Rodriguez, 2010).

[3] Proceso mediante el cual una organización o empresa almacena todos aquellos datos e información necesarios para el propio desempeño de la misma (definiciónabc, 2013).

personalización e integración con SIPAC. Otra desventaja importante de este sistema es que realiza la generación de gráficos mediante la utilización de una plataforma externa al sistema de gestión (el *Datawarehouse*).

1.2.2 Open ERP

Sistema de gestión empresarial que contiene una serie de módulos básicos, que posibilitan la observación de datos mediante gráficos dinámicos entre los que se destacan:

- Estadísticas.
- Gestión de proyectos.
- Planificación de Proyectos.

Se caracteriza por ser software libre y brindar la facilidad de trabajar remotamente mediante una interfaz web o aplicación de escritorio multiplataforma (Windows, Linux y Mac). Está implementado en tecnología Python/XML con base de datos PostgreSQL y utiliza PyChart como biblioteca para la generación de gráficos en formato de barra, pastel y líneas. (OpenERP, 2009)

Observación: A pesar de ser software libre, la adaptación, personalización e integración de los módulos que incluyen la generación de gráficos de OpenERP con SIPAC resultaría muy engorrosa, debido a que el mismo está desarrollado bajo el lenguaje de desarrollo Python/XML lo que resulta incompatible con los lenguajes usados en el desarrollo de SIPAC, dificultando específicamente la integración de la biblioteca PyChart utilizada por el mismo en la generación de los diferentes gráficos.

1.2.3 Isis ERP Manager

Sistema desarrollado por la empresa *Quality Soft Argentina S.A* que genera mediante diferentes combinaciones y filtros disponibles la estadística de unos 250 reportes.

Esta información es generada precisamente en el módulo de Estadísticas, que incluye una opción gráfica, para que los totales de ventas y compras que muestra sean visualizados en formato de barras paralelas. Una ventaja adicional que brinda este software constituye la posibilidad de exportar a formato PDF, a una planilla Excel o a cualquier formato disponible en dicho reporte. Isis ERP Manager está desarrollado con tecnología de base de datos *Microsoft SQL Server* y utiliza como tecnología de reportes *Crystal Reports*. Este sistema opera únicamente sobre la familia de Sistemas Operativos Windows, específicamente en sus versiones *XP, Vista y Seven*. (Quality Soft Argentina S.A, 2011)

Observación: Isis ERP Manager presenta como principal desventaja el hecho de ser software privativo, por lo que se dificulta su adaptación, personalización e integración con SIPAC. Además de

imposibilitar la integración de *Crystal Report* como tecnología utilizada por este sistema para la generación de reportes con información gráfica, debido a su incompatibilidad con los lenguajes en que está desarrollado SIPAC. Otro inconveniente de Isis ERP Manager es que opera solamente sobre plataforma Windows, lo que atenta contra la política de soberanía tecnológica de Cuba.

1.2.4 Generador Dinámico de Reportes

Sistema desarrollado por el Centro de Tecnologías de Gestión de Datos (DATEC), perteneciente a la UCI, que se encarga de la gestión de la información de cualquier empresa o institución, facilitando la toma de decisiones mediante la generación de reportes en varios formatos con gran variedad de opciones en su diseño. Cuenta con 6 módulos: diseñador de modelos, diseñador de reportes, diseñador de consultas, visor de reportes, administrador de reportes y seguridad. Específicamente el módulo diseñador de reportes posibilita entre sus funcionalidades incluir gráficos en el reporte que se adaptan al negocio o sea construir reportes con información gráfica, los gráficos pueden ser: de barra, pastel, curva o línea. Este sistema tiene como característica específica que opera sobre el sistema operativo GNU/Linux y utiliza como servidor de base datos PostgreSQL, se encuentra desarrollado en php5 y para la generación de gráficos emplea la biblioteca GD[4]. (Centro Tecnologias Gestion Datos, 2012)

Observación: El Generador Dinámico de Reportes constituye únicamente una herramienta para la generación de reportes de apoyo a la toma de decisiones en las instituciones u organismos donde se despliegue, por lo que no se puede modificar el contenido de dichos reportes, por esta razón no se puede integrar al SIPAC, porque para dar solución al problema planteado se necesita interactuar con los gráficos generados de forma que se logre la modificación de los elementos representados.

1.2.5 GESPRO

La herramienta Gespro constituye un paquete para la gestión de proyectos, desarrollado por la UCI y permite realizar una dirección integrada de proyectos a partir de indicadores objetivos. Cuenta con un módulo que genera reportes estadísticos que permiten visualizar mediante gráficos, en forma de barras, pastel o de curva, los indicadores claves de la gestión de proyectos. Además proporciona la generación de diagramas de Gantt. Entre las diferentes tecnologías que soporta el paquete, se encuentra el sistema operativo Ubuntu Server 10.0.4, Apache 2, Máquina virtual Java 6.0 y como servidor de base datos PostgreSQL 9.01.1. (Laboratorio de Gestión de Proyectos, UCI, 2010)

[4]Las siglas GD proceden originalmente de "*gif draw*" o "*graphics draw*" y son unas bibliotecas de código abierto desarrolladas en C para la creación dinámica de imágenes en aplicaciones.

Observación: Esta herramienta de gestión de proyectos, es un paquete que proporciona a los usuarios un control más preciso de su comportamiento en el proyecto donde se desempeña; basándose en indicadores medibles que proporcionan la estadística necesaria para realizar una valoración posterior del cumplimiento de sus tareas. Sin embargo de acuerdo a su característica de ser una herramienta para la gestión de proyectos específicamente; se dificulta el uso de la misma para la generación de gráficos, mediante los indicadores claves del modelo de planificación implantado en Cuba. Además es importante señalar que su integración con SIPAC resultaría muy complicada debido a que ambos sistemas están desarrollados bajo tecnologías diferentes.

1.3 Estado del arte de bibliotecas que generan gráficos

Como parte del desarrollo del estudio del estado del arte se tuvieron en cuenta bibliotecas que permiten la generación de gráficos. Las bibliotecas que se caracterizan seguidamente son las utilizadas por los sistemas que anteriormente se analizaron.

1.3.1 Pychart

Entre las bibliotecas que utilizan los sistemas informáticos estudiados para la generación de gráficos, se encuentra pyChart la cual fue creada con el objetivo de generar gráficos circulares, se desarrolló con el lenguaje de programación Python y trabaja sobre los sistemas operativos Windows, fue creada y es distribuida bajo licencia GNU General Public License (GPL). Actualmente genera gráficos de líneas, de barras, de gama de relleno, y de pastel y posibilita la creación de gráficos de alta calidad en formatos PDF, PNG o gráficos SVG. (Dickens, 2011)

1.3.2 Graphics Draw (GD)

La biblioteca de gráficos *Graphics Draw(GD)* inicialmente se desarrolló en el lenguaje de programación C, pero en la actualidad existen interfaces desarrolladas en otros lenguajes de programación como PHP, Perl, Python, Pascal, entre otros. Esta biblioteca permite crear imágenes a partir de líneas, arcos, texto, otras imágenes, o múltiples colores, así como la manipulación de las mismas en formato GIF, JPEG, PNG, y WBMP. A partir de PHP 4.3 se incluye una versión de la biblioteca GD y desde la versión 5.3 puede ser usada opcionalmente otra instalación GD además de la versión incluida, posibilitando obtener más características. (Boutell.Com, 2013).

1.3.3 Highcharts

Highcharts es una biblioteca escrita en JavaScript y Ajax capaz de crear gráficos estadísticos interactivos de todos tipos y estilos, como por ejemplo líneas, circulares, de barras, áreas, columnas, entre otras. Una de las ventajas principales de este script es su utilización sin necesidad de instalar

plugins externos, como Flash o Java, al navegador para que pueda visualizar las gráficas. Además es compatible con cualquier navegador web que tenga habilitado el uso de JavaScript. Para su funcionamiento solo necesita dos archivos: la biblioteca JavaScript *highcharts.js* y la conocida biblioteca *jQuery* o *Mootools,* lo que permite que se mantenga una constante actualización con los valores del servidor. Asimismo con el módulo de exportación habilitado, los usuarios pueden exportar el gráfico a PNG, JPG, PDF o SVG en el tecleo de un botón, o imprimir el gráfico directamente desde la página web. Highcharts cuenta con una ayuda o API libre y que puede ser descargada publicada en internet. Posee varios tipos de licencia entre las que se destaca la licencia libre. (Kuan, 2012)

1.3.4 Valoración de las blibliotecas estudiadas.

Con el estudio de las bibliotecas Pychart, Graphics Draw (GD) y Highcharts, se llegó a la conclusión de que no se puede utilizar Pychart por ser incompatible con las tecnologías de desarrollo de SIPAC, debido a que está desarrollada en el lenguaje de programación Python. Sin embargo la biblioteca Highcharts desarrollada en JavaScript, se puede integrar al SIPAC y permite obtener los tipos de gráficos que se necesitan para dar solución a los objetivos propuestos, también posibilitando añadir, eliminar y modificar las series, función necesaria para poder adaptar a la solución propuesta los gráficos generados por esta biblioteca. Su implementación es sencilla, dependiendo solamente de la integración de dos archivos JavaScript al SIPAC, dicho elemento marca una diferencia con la biblioteca GD, que aunque presenta una interfaz desarrollada en PHP, para su uso se requiere la instalación del paquete php5-gd que incluye funcionalidades para la gestión de imágenes, no necesarias en la solución para la obtención de gráficos. Asimismo se considera como principal inconveniente que la construcción del gráfico se realiza en el servidor de aplicaciones por estar implementada en lenguaje PHP, a diferencia de la biblioteca Highcharts, que permite la conformación del gráfico en el cliente, no requiriendo hacer peticiones al servidor de aplicaciones para la construcción del mismo. Tomando en cuenta lo anteriormente propuesto y la experiencia existente en la UCI relacionada al trabajo con la biblioteca Highcharts, así como el volumen de documentación disponible para su aprendizaje, que incluye una API o ayuda de desarrollo, Se decide utilizar la misma para para obtener los resultados deseados.

1.4 Valoración del estado del arte

Luego de haber realizado el estudio y valoración de diferentes sistemas que generan gráficos estadísticos, se evidencia la inexistencia de un sistema con las características adecuadas al modelo de planificación cubano. Por lo que se hace necesario el desarrollo de un producto netamente adaptado a las necesidades existentes en la planificación de actividades, utilizando tecnologías y herramientas

libres que permitan obtener un software que muestre la información de forma dinámica mediante el uso de gráficos estadísticos. Sin embargo el estudio de estos sistemas propició obtener experiencia en el funcionamiento de los mismos, específicamente en las tecnologías usadas por estos para graficar los diferentes indicadores. Evidenciándose el uso de plataformas externas y bibliotecas, como tecnologías para graficar los diferentes negocios, que sirve de ayuda en la implementación y desarrollo de la solución del cual es objeto el presente trabajo de diploma, el cual se basará en el uso de la biblioteca Highcharts para obtener los resultados deseados.

1.5 Lenguajes de modelado y programación

Los lenguajes de modelado y desarrollo que se emplean para la obtención de la solución que se propone, han sido establecidos en el modelo de desarrollo de CEIGE, en consecuencia con las tecnologías, herramientas y marco de trabajo definido para la confección de sus productos.

1.5.1 Lenguaje de modelado

Se denomina lenguaje de modelado de objetos al conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software (Sistemas Computin, 2010). Para la modelación de la solución que se propone se emplearán como lenguaje de modelado los siguientes:

UML: El Lenguaje de Modelado Unificado (*Unified Modeling Language*) como notación orientada a objetos cuenta con una notación estándar y semánticas esenciales para el modelado de sistemas, se empleará con el fin de especificar y documentar un sistema de software, de un modo estándar incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema. UML implementa un lenguaje de modelado común para todos los desarrollos por lo que se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. (López, Patricia, 2012)

1.5.2 Lenguaje del lado del cliente

HTML: Es el lenguaje que se utilizará para definir páginas clientes de la aplicación. Se compone por un conjunto de etiquetas utilizadas para definir y ubicar los distintos elementos que componen una página web. Como un lenguaje de marcación de elementos para la creación de documentos hipertexto, HTML puede describir hasta un cierto punto la apariencia de un documento. Puede incluir uno o varios scripts, como por ejemplo JavaScript o PHP, los cuales pueden afectar el comportamiento del HTML. Su principal desventaja es que todos los navegadores no interpretan el código HTML de la misma manera. (DesarrolloWeb, 2010)

JavaScript: Se recurrirá al lenguaje JavaScript para acceder a los objetos de la capa de presentación de la solución que se propone con la presente investigación. Este lenguaje es basado en objetos y principalmente se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Entre las acciones típicas que se pueden realizar en JavaScript existen dos vertientes. Por un lado los efectos especiales sobre páginas web para crear elementos dinámicos. Por el otro, permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se puede crear páginas interactivas. Sin embargo en caso de ser un script sobrecargado de datos, tomará mucho más tiempo visualizarse en el navegador.

Hojas de Estilo en Cascada (CSS): Las hojas de estilo en cascada serán utilizadas para representar todo lo referente a los estilos (dígase tamaños, colores, iconos, imágenes, tipografías, espacios y bordes). Constituyen el estándar para la inserción de estilos a documentos estructurados, como por ejemplo, páginas HTML o XML. El objetivo de la definición de este estándar del W3C[5] es permitir la separación entre las normas de presentación y el propio contenido a mostrar. (Iván Pérez Nieto, 2011)

XML (*Extensive Markup Language*): XML como un metalenguaje de definición de documentos estructurados mediante marcas o etiquetas se usará en la creación de las reglas básicas que permiten el intercambio de información estructurada entre aplicaciones; se emplea también para tareas de validación y configuraciones en el sistema. Se trata de un estándar del W3C que posibilita compartir la información de una manera segura, fiable y fácil. Además permite compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Es también un lenguaje del lado del servidor. (W3C, 2010)

1.5.3 Lenguaje del lado del servidor

PHP v5.2: PHP es el lenguaje que se empleará para programar del lado del servidor. Es un lenguaje de programación interpretado, completamente orientado al desarrollo de aplicaciones web dinámicas. Es gratuito, fácil de usar y aprender, portable, de código abierto y multiplataforma. Presenta interfaces para una gran cantidad de sistemas de base de datos, así como bibliotecas incorporadas para muchas tareas web. Sin embargo este lenguaje debido a su flexibilidad presenta como principal inconveniente, que mal utilizado, puede convertir al sitio en un punto de fácil acceso por piratas informáticos. Por otra parte con respecto a otros lenguajes de programación como Java o C++ es mucho menos robusto y la programación orientada a objetos es aún muy deficiente para aplicaciones grandes. (Answers Corporation, 2012)

[5] World Wide Web Consortium

1.5.4 Frameworks

Un marco de trabajo, en el desarrollo de sistemas computarizados, es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Muchos los definen como un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. (spimeWiki, 2013)

SAUXE: Sauxe es el marco de trabajo que se empleará para el desarrollo de la solución que se propone. El cual fue desarrollado en la UCI como fruto del paradigma de independencia tecnológica por el que aboga el país. Este marco de trabajo, fusionado bajo tecnología totalmente libre (entre ellas PHP, PostgreSQL, Apache) posee el desarrollo de tecnologías propias basadas en otros marcos de trabajo como ZendFramework para el manejo de la lógica de negocio, Doctrine para el acceso a datos y ExtJS para la capa de presentación. Cuenta con una arquitectura en capas que a su vez presenta en su capa superior un MVC[6]. Contiene un conjunto de componentes reutilizables que proveen la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (Baryolo, y otros)

ExtJS 2.2: ExtJS es el marco de trabajo que se empleará para el desarrollo de la capa de presentación. Está basado completamente en la programación orientada a objeto. Cada objeto contiene: propiedades, métodos y eventos. Basa toda su funcionalidad en JavaScript a través de bibliotecas. Así, en tiempo de ejecución, carga y crea todos los objetos HTML a través del uso intenso de DOM[7]. Los datos son obtenidos con AJAX a través de XML. Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos. Además permite que exista un balance entre el Cliente – Servidor, posibilitando que la carga de procesamiento se distribuye, permitiendo que el servidor al tener menor carga, pueda manejar más clientes al mismo tiempo. (Cutter', y otros, 2008)

Zend Framework: Se utilizará el marco de trabajo Zend Framework para el manejo de la lógica de negocio. Este marco de trabajo de código abierto brinda facilidades de uso y funcionalidades. Está diseñado para la versión 5 de PHP y posee buenas capacidades de ampliación. Proporciona un

[6] Modelo Vista Controlador.

[7] *Document Object Model* o Modelo de Objetos del Documento.

sistema de caché de forma que se puedan almacenar diferentes datos, así como los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador. Consta de mecanismos de filtrado y validación de entradas de datos. Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX y provee capacidades de búsqueda sobre documentos y contenidos. (Zend Technologies, 2012)

Doctrine: Para la capa de acceso a datos se empleará Doctrine. Este es un sistema ORM (en inglés *Object Relational Mapper*) para PHP 5.2 o superior que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL, manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Además, exporta una base de datos existente a sus clases correspondientes y convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (Vesterinen, 2012)

1.6 Tecnologías y herramientas de desarrollo

- **AJAX**

Ajax por sus siglas en inglés *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML) es la técnica de desarrollo web que se usará para poder hacer consultas asíncronas al servidor sin necesidad de recargar la página. Esta surge de la combinación de tres tecnologías ya existentes: HTML (o XHTML) y Hojas de Estilo en Cascada (CSS) para presentar la información*, Document Object Model* (DOM) y JavaScript, para interactuar dinámicamente con los datos, además de XML y XSLT, para intercambiar y manipular datos de manera desincronizada con un servidor web. (Eguiluz, 2013)

- **Herramienta CASE: Visual Paradigm**

Se empleará Visual Paradigm 5.0 como herramienta CASE. Utiliza UML 8.0 como lenguaje de modelado, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite dibujar diferentes tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Ayudando a construir aplicaciones de calidad más rápido, mejor y con un costo más bajo. (Visual Paradigm, 2012)

- **Sistema de Control de Versiones: Subversion**

La versión 1.6.14 de Subversion (SVN) es la herramienta de entorno colaborativo que se utilizará para el control de versiones. Este se encuentra preparado para funcionar en red y se distribuye bajo licencia libre. Mantiene versiones no sólo de archivos, sino también de directorios y versiones de los metadatos asociados a esos directorios. Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre. Ofrece mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. (Apache , 2012)

- **Entorno Integrado de Desarrollo (IDE): NetBeans**

La presente solución se desarrollará sobre el IDE de programación multiplataforma NetBeans 7.1. El cual tiene soporte para la versión 5 de PHP, ExtJS, el diseño de Hojas de Estilo (CSS) y HTML. Se integra con varias herramientas como el Subversion y servidores web, presenta una gran estabilidad. Es un producto de código abierto, con todos los beneficios disponibles de forma gratuita. Hace uso de *plugins*[8] para ampliar sus funcionalidades, lo que le da una gran facilidad de uso.

- **Servidor Web: Apache**

Se utilizará como servidor web Apache 2.0 pues es una tecnología gratuita, de código abierto, compatible con muchos Sistemas Operativos. Tiene todo el soporte que se necesita para obtener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Cuenta con una excelente configuración de la creación y gestión de registros de las actividades. Apache permite la creación de ficheros de registro a disposición del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. (Apache Software Foundation, 2012)

- **Sistema Gestor de Bases de Datos: PostgreSQL**

Como Sistema Gestor de Base de Datos (SGBD) se empleará la versión 9.1 de PostgreSQL. Este es un sistema de gestión de bases de datos relacional orientada a objetos. Es una herramienta de código abierto, de bajo coste y multiplataforma. Se destaca en ejecutar consultas complejas, subconsultas y uniones de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad. Soporta transacciones, claves ajenas con comprobaciones de integridad referencial y almacenamiento de objetos de gran tamaño. Cuenta con varias herramientas gráficas de diseño y administración de bases de datos como el pgAdmin. (PostgreSQL, 2013)

[8] Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

- **Navegador web: Mozilla Firefox**

Se utilizará como navegador web Mozilla Firefox 20.0.1. Este navegador multiplataforma es libre y es compatible con los estándares web (señálese HTML, XML, CSS y JavaScript) que se emplearán para el desarrollo de la solución que se propone. Incluye entre sus funcionalidades un mecanismo para añadir funcionalidades mediante extensiones. Precisamente Firebug 1.8.3 es el complemento que se integrará a Mozilla Firefox para ayudar a desarrollar, evaluar y depurar la aplicación, controlando el CSS y HTML en tiempo real, midiendo el tiempo de carga para optimizar la página o corrigiendo los posibles inconvenientes con JavaScript.

1.7 Modelo de desarrollo

El desarrollo de la propuesta de solución de la cual es objeto este trabajo de diploma está guiado por el modelo de desarrollo de CEIGE, el cual agrupa un conjunto de buenas prácticas y principios del desarrollo de software a nivel mundial y está orientado fundamentalmente para las particularidades de Cedrux. Entre las principales características de este se encuentran:

- **Centrado en la arquitectura:** La arquitectura determina la línea base del desarrollo del software y agrupa los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Además interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.
- **Orientado a componentes:** Las iteraciones son orientadas por el nivel de representación arquitectónica de los componentes, los cuales constituyen abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, y son la unidad de medición y ordenamiento de las iteraciones.
- **Iterativo e incremental:** El equipo de arquitectura, los clientes y la alta gerencia, planifican y coordinan las iteraciones, estas constituyen el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo la evolución incremental del producto.
- **Ágil y adaptable al cambio:** El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados. (Hernández, 2006)

1.7.1 Modelo de ciclo de vida de los proyectos del CEIGE

El modelo de ciclo de vida que se presenta en la Figura 1 define las fases por las que transitarán los proyectos de desarrollo de software del CEIGE.

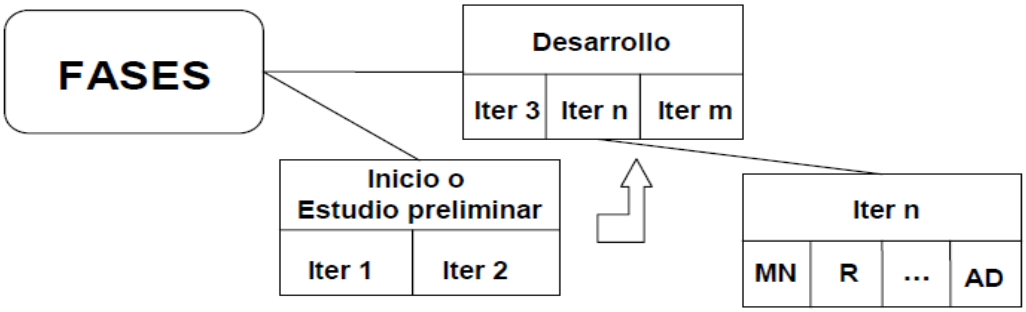

Figura 1 Fases del ciclo de vida de proyectos del CEIGE

1.7.2 Descripción de las fases del ciclo de vida de los proyectos

- **Inicio o Estudio preliminar**: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto. Tiene como principales objetivos asegurar la factibilidad del proyecto y establecer un plan para la ejecución del proyecto. Sus hitos son el Plan de desarrollo de software y el acta de inicio del proyecto firmada.
- **Desarrollo:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. El objetivo de esta fase es obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales. Como hito de desarrollo se tiene el producto liberado por una entidad certificadora de calidad.

En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación.

Ciclo de vida de proyectos de CEIGE

El ciclo de vida de los proyectos del CEIGE (Figura 2) tiene en cuenta las actividades de cada una de las fases y áreas de procesos que plantea el nivel dos de CMMI establecido en la UCI. Esta abarca el total de acciones que se realizan en las distintas líneas de desarrollo para la elaboración del servicio o

producto final, sin embargo, se debe adaptar a las características particulares del proyecto que puede que no ejecute determinada disciplina, así como la elaboración de determinados artefactos del total aquí definido.

Figura 2 Ciclo de vida de proyectos del CEIGE.

- **Modelado del negocio:** Es la fase destinada a comprender los procesos de negocio de la organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito.
- **Requisitos:** El esfuerzo principal en la fase de Requisitos es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de artefactos que describen todas las interacciones que tendrán los usuarios con el software y que responden a los requisitos funcionales del sistema. Se especifican los requisitos funcionales y no funcionales.
- **Análisis y diseño:** Durante esta fase es modelado el sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima fase. Los artefactos generados en esta etapa son más formales y específicos de una implementación. En caso de llevarse a cabo la reutilización de componentes software ya desarrollados, durante esta fase se ajusta el modelado existente a los requisitos actuales.

- **Implementación:** A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. Al reutilizar componentes software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes.
- **Pruebas internas:** Durante esta fase se desarrollan las pruebas del grupo de calidad del centro verificando el resultado de la implementación. Permite identificar posibles errores en la documentación y el software, es decir requisitos que el producto debería cumplir y que aún no los cumple.
- **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.

1.8 Patrones de diseño empleados en la solución propuesta

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. (Puebla, y otros, 2009)

Los patrones de diseño empleados se definieron durante la elaboración del diseño de las clases y las relaciones entre ellas.

Patrones GRASP

Estos patrones representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. Es el acrónimo para *General Responsibility Assignment Software Patterns* (Patrones Generales de Software para Asignar Responsabilidades) (García, 2012). Entre los principales patrones *GRASP* a utilizar se encuentran:

Experto: Este patrón se encarga de asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad; si esta se asigna en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, proporcionando la oportunidad de reutilizar los componentes en futuras aplicaciones. (Astudillo, 2013)

Creador: Este patrón se encarga de orientar quien debe ser el responsable de crear una nueva instancia de alguna clase. Además guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.

Tiene como propósito fundamental, encontrar un creador que se debe conectar con el objeto producido en cualquier evento, al escogerlo como creador, se da soporte al bajo acoplamiento.

Se recomienda asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A o
- B utiliza especialmente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.

Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A. (Astudillo, 2013)

Controlador: Es un objeto de interfaz no destinado al usuario que se encarga de manejar los eventos en el componente, separando así la lógica de negocios de la capa de presentación. De esta manera el controlador delega a las clases modelo las actividades a realizar manteniendo así una alta cohesión entre ellas.

Patrones *GoF*

Es el acrónimo de *Gang of Four* (Banda de Cuatro), llamado así por los cuatro autores del libro Patrones de Diseño, el cual describe alrededor de 23 patrones de los más utilizados. Estos se clasifican según tres propósitos en:

Creacional: Cómo se puede crear un objeto, habitualmente esto incluye aislar los detalles de la creación del objeto, de forma que su código no dependa de los tipos de objeto que hay y por lo tanto, no tenga que cambiarlo cuando añada un nuevo tipo de objeto.

Estructural: Esto afecta a la manera en que los objetos se conectan con otros objetos para asegurar que los cambios del sistema no requieren cambiar esas conexiones. Los patrones estructurales suelen imponerlos las restricciones del proyecto.

Comportamiento: Objetos que manejan tipos particulares de acciones dentro de un programa, estos encapsulan procesos que quiere que se ejecuten, como interpretar un lenguaje, completar una

petición, moverse a través de una secuencia (como en un iterador) o implementar un algoritmo. (Villa, 2013)

Entre los principales patrones *GoF* utilizados en la solución se encuentran:

Cadena de Responsabilidades: Este patrón de diseño busca un bajo grado de acoplamiento entre los objetos que hacen una solicitud y los objetos que la atienden. Cuando se tiene más de un objeto que puede atender una solicitud se sugiere que se pueda atender secuencialmente con estos objetos formando una cadena. Cada objeto tiene una referencia al objeto siguiente, el primer objeto decide a que objeto pasa y así sucesivamente hasta llegar al final (Soto, 2009).

1.9 Conclusiones del capítulo

Luego de realizar el estudio del estado del arte para los sistemas y bibliotecas que permiten la generación de gráficos, se evidenció la necesidad de desarrollar una solución que se adapte a las particularidades y tecnologías de SIPAC, escogiendo para este propósito la biblioteca Highcharts como tecnología a integrar al SIPAC para la generación de gráficos. Se tomará como guía el modelo de desarrollo de CEIGE, debido a que este abarca los principales elementos que dirigirán el proceso de desarrollo de la solución y las herramientas que define cuentan con las características necesarias para este fin.

CAPÍTULO 2: ANÁLISIS Y DISEÑO.

2.1 Introducción

En el presente capítulo se definirán las principales características que tendrá la solución que se propone desarrollar. Se definen y especifican los requisitos funcionales del sistema que garantizan los servicios que debe proporcionar el mismo, así como los no funcionales a utilizar para el desarrollo de la solución, los cuales especifican las restricciones de los servicios o funciones ofrecidos por el sistema. Además se especifican los patrones y estilos arquitectónicos a utilizar para la solución propuesta, permitiendo fundamentar el uso de los patrones empleados durante el modelado de los diagramas que describen la relación entre las clases existentes del diseño, posibilitando su validación mediante la aplicación de métricas al mismo, así como la presentación del modelo de datos de la solución.

2.2 Propuesta de solución

Actualmente en SIPAC se hace necesaria la implementación de una solución que permita dar seguimiento al comportamiento de los indicadores por el cual basa su funcionamiento la planificación de actividades. Asimismo se requiere que esta solución sea atractiva y dinámica para que mejore el análisis de la información y que de esa forma permita al usuario interactuar gráficamente con la misma. Incluyendo además la generación de salidas para el sistema teniendo en cuenta lo obtenido previamente en los estos gráficos. Teniendo en cuenta todos estos elementos se propone la solución de la cual es objeto el presente trabajo de diploma y se representa seguidamente en el Modelo conceptual.

2.2.1 Modelo Conceptual

Un modelo conceptual define los principales conceptos de una solución, posibilitando de esta forma una mayor comprensión del negocio abordado. En el caso de la solución para la obtención de gráficos, el modelo conceptual definido (ver Figura 3) se centra en los elementos de la planificación, los cuales pueden ser Objetivos, Actividades, Factores que influyen en el plan (FIP) o Planes, a los cuales se asocian las Áreas de resultados claves (ARC). Estos elementos son caracterizados por los Indicadores claves, en función de los cuales se generan los gráficos.

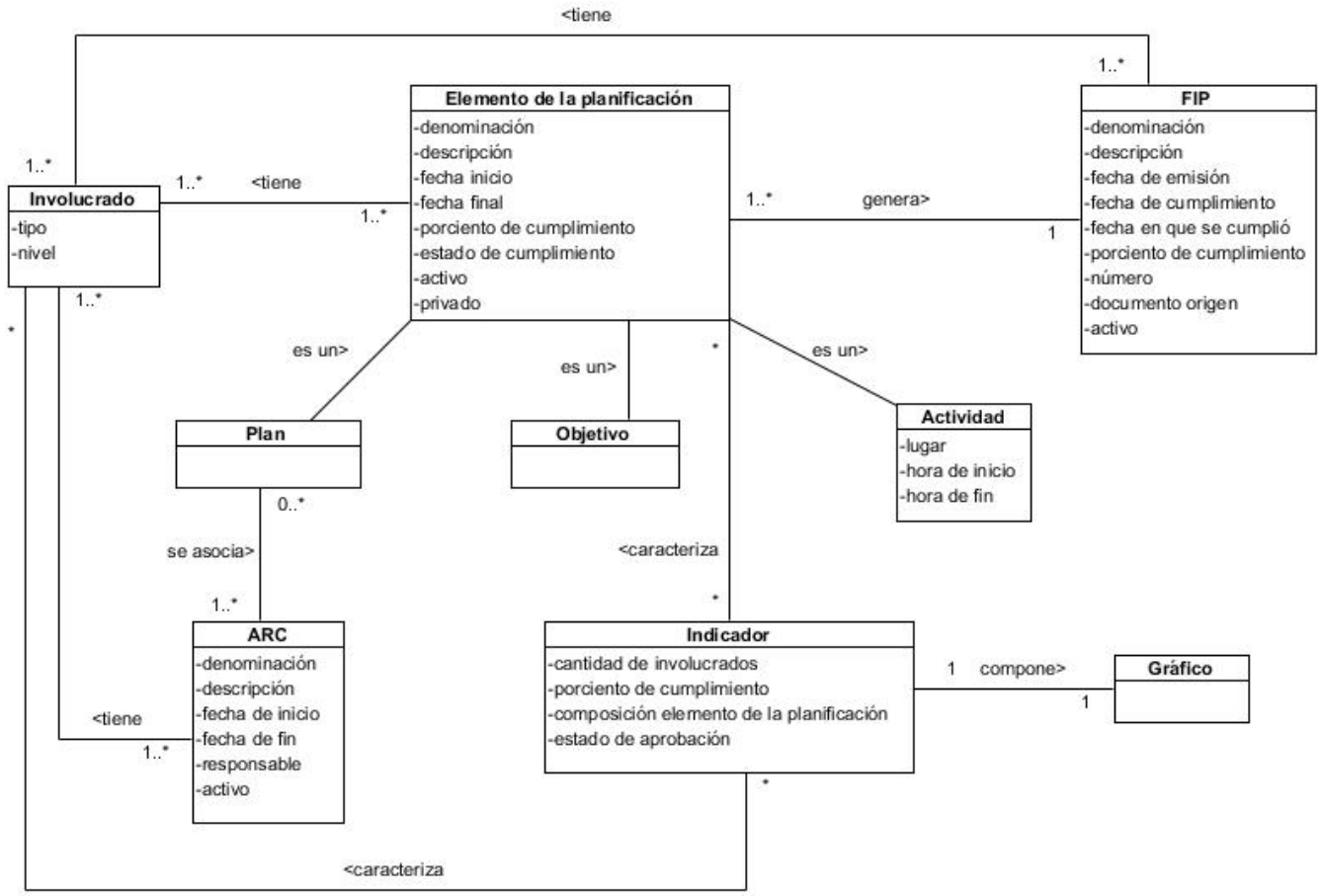

Figura 3 Modelo Conceptual de la Solución para la Obtención de Gráficos

2.3 Requisitos

Un requisito es una condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta (Ibañez, 2010). Estos deben ser bien documentados para su posterior cumplimiento.

Técnica de captura de requisitos.

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema (Escalona, et al., 2002). Para extraer los requisitos del sistema se utilizaron las siguientes técnicas:

Entrevistas: Se utilizó en los diferentes encuentros con el cliente, dígase funcionales o especialistas de Planificación del Consejo de Estado y los OACE y se realizaron preguntas con el objetivo de obtener toda la información posible sobre la visión que el entrevistado tiene de los requisitos y de esta forma comprender los propósitos de la solución buscada.

Tormenta de ideas: Esta técnica se implementó mediante la realización de reuniones y encuentros con todos los involucrados en el desarrollo del sistema donde cada cual expresa sus ideas y criterios. Su objetivo fundamental es dar una visión general de las necesidades del sistema.

Talleres: Se realizaron reuniones con los involucrados en el desarrollo del sistema, estos encuentros son más organizados donde se realiza una preparación previa y es dirigida por un experto (en este caso el funcional o jefe del proyecto SIPAC). Su objetivo fundamental es detallar cada requisito y sirve como base para la posterior especificación de los mismos.

Requisitos Funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en determinadas situaciones. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (Gabriel, 2010)

De esta forma se muestran los requisitos funcionales identificados para el desarrollo de la solución que se propone, los cuales quedan agrupados como Gestionar gráficos.

RF 1. Obtener Gráfico: El sistema debe permitir visualizar el gráfico definido por el usuario en sesión, en formato de barra o pastel mediante la captura de información suministrada desde la base de datos del sistema SIPAC, permitiendo observar estadísticamente el comportamiento de los indicadores de la planeación.

RF 2. Exportar Gráfico: El sistema debe posibilitar exportar el gráfico obtenido, en diferentes formatos establecidos como son JPEG, PNG, SVG y PDF.

RF 3. Imprimir Gráfico: El sistema debe permitir imprimir la imagen del gráfico obtenido.

RF 4. Abrir Especificación: El sistema debe permitir abrir la especificación con los principales atributos de un elemento seleccionado por el usuario en sesión.

RF 5. Modificar Especificación: El sistema debe posibilitar que el usuario en sesión realice modificaciones a los atributos del elemento especificado y luego muestre el gráfico nuevamente con estos datos actualizados.

Especificación de Requisitos Funcionales

Mediante la especificación de los requisitos se detalla de forma sencilla el flujo de eventos a realizar por la funcionalidad en el sistema.

De esta manera a continuación se muestra la especificación del requisito funcional Obtener Gráfico, la especificación de los restantes requisitos funcionales se encuentran descritos en el expediente de proyecto SIPAC 2.0 en el artefacto CIG-SPA-N-i2603.

TABLA 1 Especificación del RF Obtener Gráfico

Precondiciones	Debe existir el elemento de la planificación.	
Flujo de eventos		
Flujo básico <<Obtener gráfico>>		
1	El usuario selecciona el elemento del cual quiere obtener el gráfico.	
2	El usuario selecciona la opción Obtener gráfico.	
3	El sistema muestra la interfaz principal de la solución.	
4	El usuario selecciona el tipo de gráfico que desea obtener.	
5	El sistema muestra el gráfico que seleccionó el usuario.	
6	Se concluye el requisito.	
Pos-condiciones		
1	El gráfico se obtiene.	
Flujos alternativos		
Flujo alternativo N/A		
Pos-condiciones		
1	N/A	
Validaciones		
1	N/A	
Conceptos	N/A	N/A
Requisitos especiales	RNF1-RNF10	
Asuntos pendientes	N/A	

Requisitos no funcionales

Los requisitos no funcionales definen propiedades y restricciones del sistema, estos pueden ser por ejemplo confiabilidad, tiempo de respuesta y requisitos de almacenamientos (Sommerville, 2004). Los requisitos no funcionales de la solución se rigen por los definidos en la arquitectura del sistema que se encuentran en el artefacto CIG-SPA-N-i3514-RNF (Artola, 2013) perteneciente al expediente del proyecto SIPAC 2.0.

Técnicas para la Validación de Requisitos

Para validar que los requisitos anteriormente identificados y descritos cumplan con las expectativas del cliente y el equipo de desarrollo, se emplea la Técnica de validación de requisitos: construcción de prototipos por cada requisito funcional. Además en el expediente de proyecto se incluye los artefactos Criterios para validar Requisitos del Cliente y Criterios para validar Requisitos del Producto con este mismo fin.

También es importante señalar que se realizaron diversas revisiones, por el administrador de calidad y la analista principal de la línea de Planificación, a los artefactos relacionados con la disciplina de Levantamiento de requisitos para lograr una correcta interpretación de la información transmitida, los señalamientos planteados fueron recogidos y aplicados posteriormente.

2.4 Diseño de la solución en términos de componentes

SIPAC es el encargado de interrelacionar objetivos de trabajo, actividades y recursos en tiempo real; garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades. El Modelo de Componentes de dicho sistema está integrado por el componente Configuración que es el responsable de la gestión de los grupos de usuarios del sistema, de los permisos respectivos y de la gestión de los nomencladores. Además lo integra el componente Planeación que abarca la relación entre todos los elementos de la planificación. La solución para la obtención de gráficos de los indicadores de la planificación está estrechamente relacionada con los elementos de la planificación del sistema SIPAC, permitiendo la visualización mediante gráficos de la información relacionada con los mismos; los cuales a través de sus respectivas interfaces de comunicación establecen un estrecho vínculo.

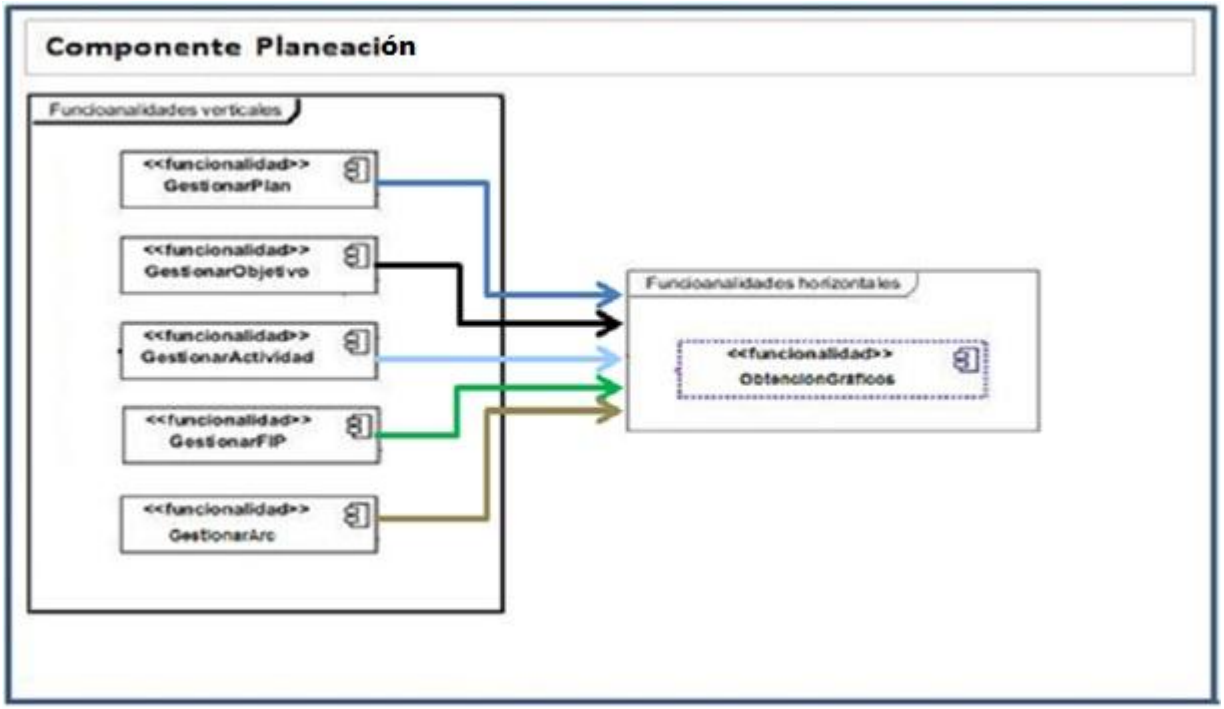

Figura 4 Diagrama de Componente de la Solución para la Obtención de Gráficos

El modelo de componentes de la solución para la obtención de gráficos va estar caracterizado por el componente planeación, el cual contiene un conjunto de funcionalidades verticales que describen la interrelación existente entre los elementos de la planificación y la funcionalidad horizontal para la obtención de gráficos, esta a su vez se va a obtener como resultado de la implementación y desarrollo de los requisitos funcionales descritos anteriormente.

2.5 Diseño de las clases

Para la transición a la fase de implementación es de mucha importancia tener definidas las clases que intervendrán y las relaciones existentes entre ellas. Es por esta razón que a continuación se presenta el diagrama de clases donde se representan las clases de la solución y sus interrelaciones.

Diagramas de clase del diseño

El diseño de la solución para la obtención de gráficos basa su definición en el patrón arquitectónico MVC. Contando en la capa de la Vista con las clases cmpgestionargraficos.phtml y cmpgestionargraficos.js, que se compone del formulario cmpgestionargraficos. Estas clases se relacionan con la clase Controladora cmpgestionargraficosController.php la cual es la encargada de controlar la comunicación entre la Vista y el Modelo. Este último se representa en el diagrama

mediante las clases cmpgestionargraficosModel.php, en la cual se implementará la lógica de negocio para cumplir con los requisitos definidos anteriormente. En esta capa se incluyen también las clases de acceso a datos desde las cuales se realizará la captura de la información necesaria para gestionar los gráficos. El diagrama de clases del diseño de la solución para la obtención de gráficos de los indicadores de la planificación se muestra en la siguiente figura.

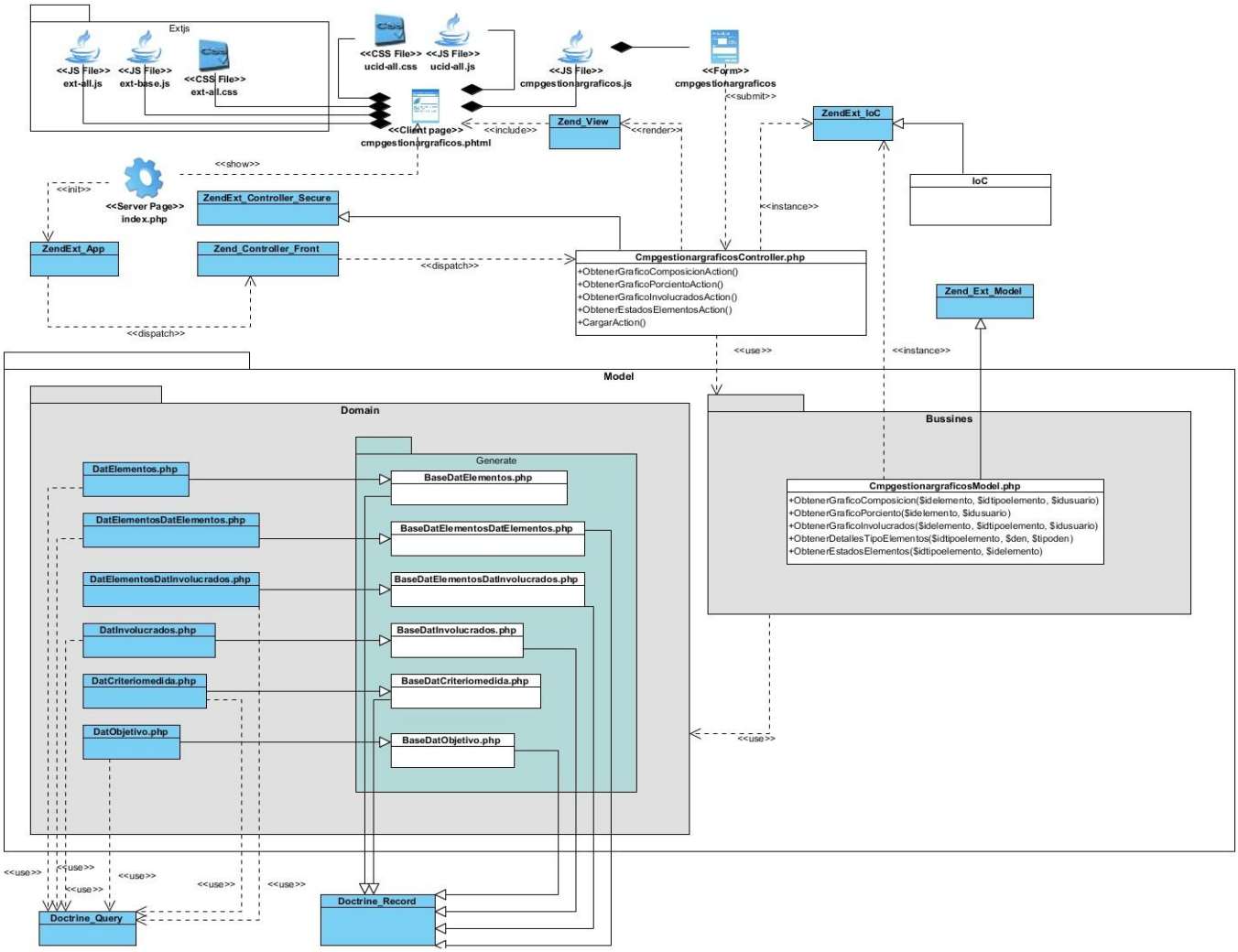

Figura 5 Diagrama de Clases del Diseño Solución para la Obtención de Gráficos

Patrones empleados en el diseño

En el diseño de la solución se evidencia el uso del patrón experto, específicamente en las clases *DatElementos* y *DatElementosDatElementos* las cuales contienen la responsabilidad de manejar toda la información relacionada con los elementos de la planificación, permitiendo la reutilización de la misma por varios componentes.

El uso del patrón creador se ve evidenciado durante el diseño de la solución en la creación de objetos de la clase *DatElementos* para hacer instancias de funcionalidades específicas en la clase *CmpGraficadorModel.*

El patrón controlador es evidenciado su uso, en las funcionalidades implementadas en la clase *CmpgraficadorController* las cuales sirven de punto intermedio que permite hacer una separación de la lógica de negocio de la capa de presentación, cumpliendo el objetivo específico de dicho patrón.

El patrón GoF cadena de responsabilidades es principalmente usado en el tratamiento de Excepciones, si se produce un error en una consulta a la base de datos esta es manejada por el Modelo quien crea una excepción de tipo *ZendExt_Exception* que debe ser enviada al Controlador, quien la envía a la Vista ya traducida y esta muestra en un lenguaje entendible al usuario la notificación del error.

Diagramas de Secuencia

El diagrama de secuencias, que se define en UML, es uno de los más utilizados para identificar el comportamiento de un sistema, por representar los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por una transacción del sistema (Generación del diagrama de secuencias de UML 2.1.1 desde esquemas preconceptuales., 2008).

La figura que se muestra seguidamente constituye un ejemplo de diagrama de este tipo, pertenece al RF Obtener Gráfico, específicamente del escenario Composición. El flujo representado en este diagrama comienza cuando el Usuario da clic en la opción Obtener gráfico en cualquier clase cliente de los elementos de la planificación dígase planes, objetivos y actividades. Luego esa petición va a la clase controladora cmpgestionargraficosController.php envía la petición a la clase modelo cmpgestionargraficosModel.php la cual realiza las llamadas a las clases expertas para obtener la información con la que se va a construir el gráfico. Esta información se envía a la clase Vista cmpgestionargraficos.js que es la encargada finalmente de mostrar el gráfico en su formulario.

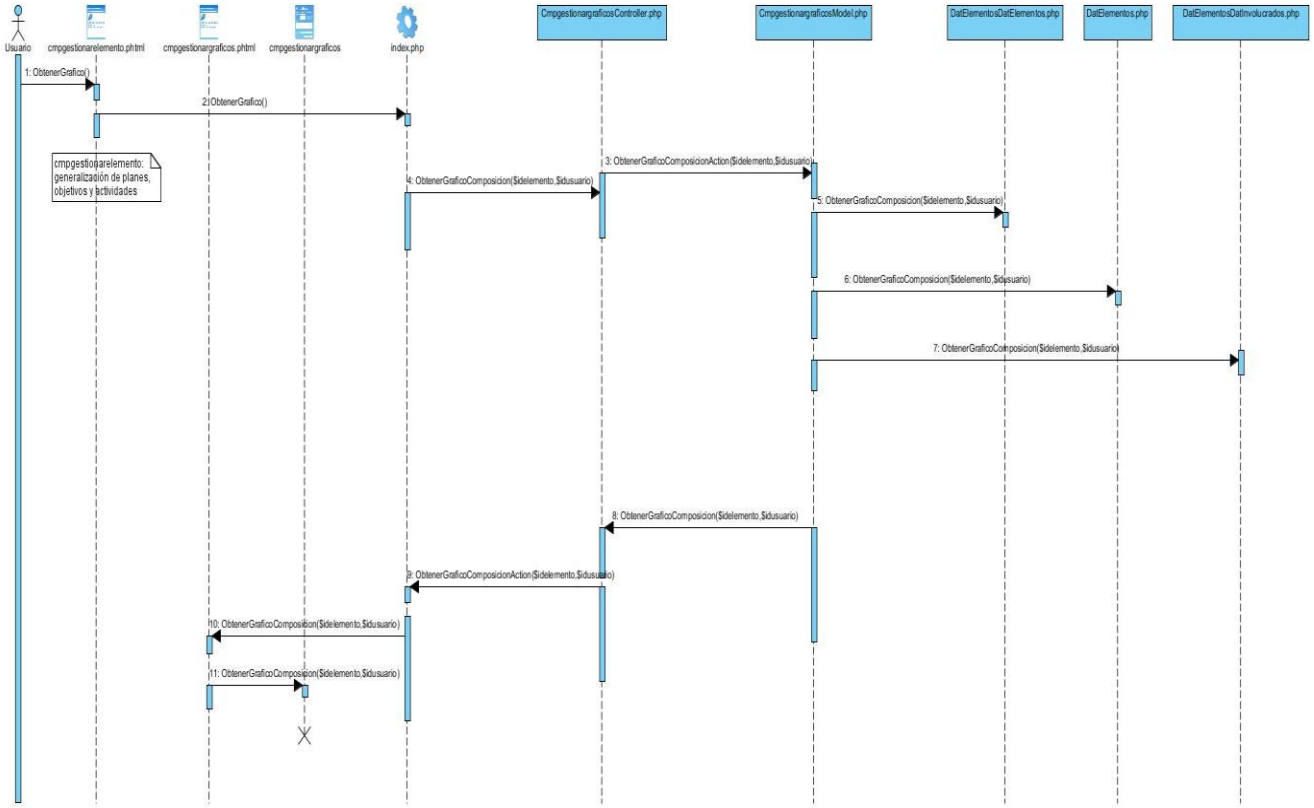

Figura 6 Diagrama de Secuencia del RF Obtener Gráfico (Composición)

Los restantes diagramas de secuencias descritos para la solución de obtención de gráficos de SIPAC se encuentran especificados en el expediente de proyecto.

2.6 Modelo de Datos

Un modelo de datos es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de reglas de integridad, las cuales definen un conjunto de estados consistentes. (Instituto Tecnologico de Colima., 2013)

El modelo de datos de SIPAC resume los conceptos y define las relaciones, ajustándose a las necesidades de almacenamiento de datos del sistema. Posee tercera forma normal y cuenta con 52 tablas, el mismo se encuentra en el expediente de proyecto de SIPAC 2.0.

En la figura que se muestra a continuación se presenta el modelo de datos de la solución para la obtención de gráficos:

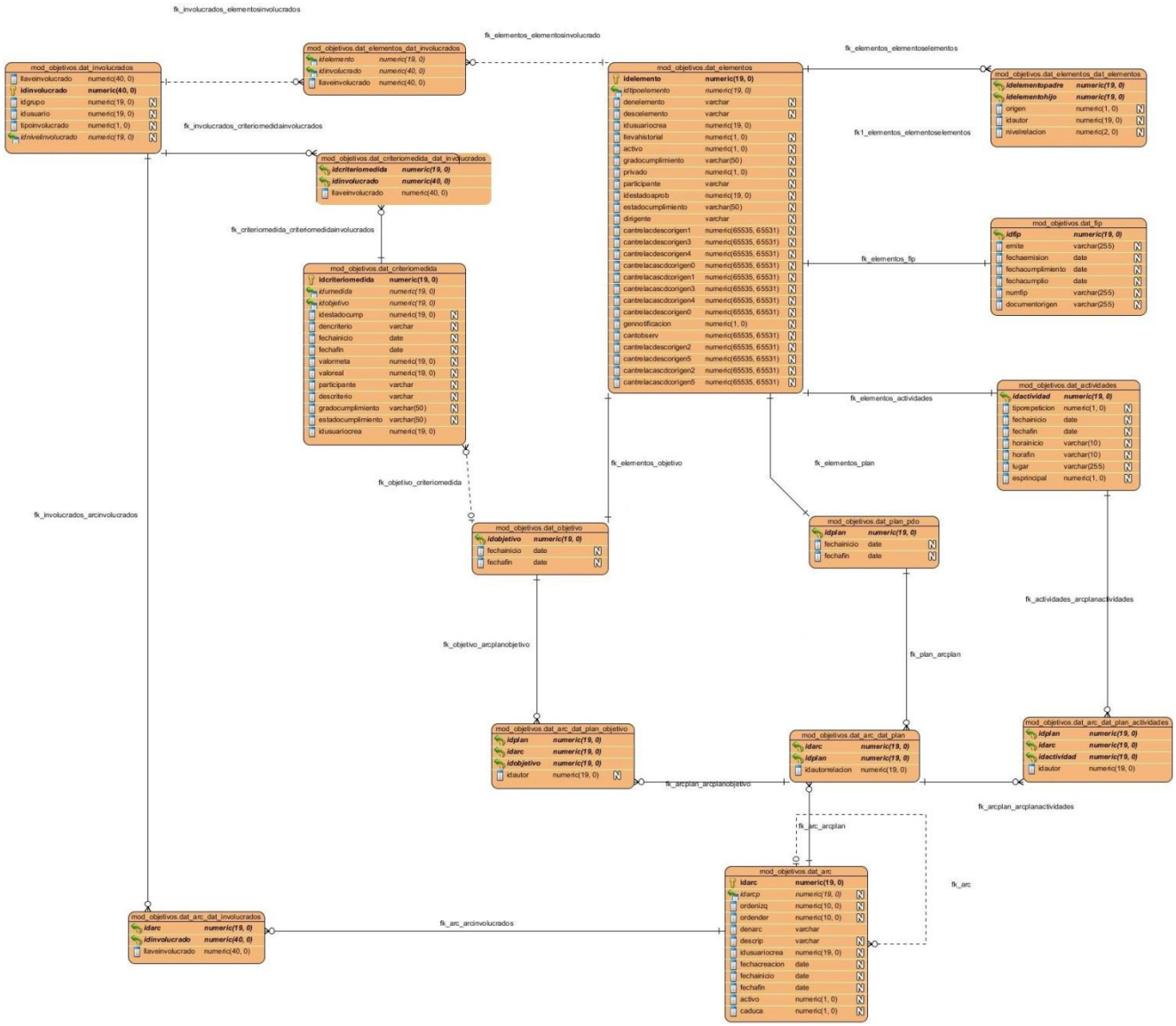

Figura 7 Modelo de Datos Solución para la Obtención de Gráficos

El modelo de datos de la solución va estar caracterizado por la entidad DatElementos, la cual se encarga de almacenar toda la información relacionada con los elementos primarios de la planificación, representados por las entidades DatPlanPdo, DatObjetivos, DatFip, DatActividades; estos estarán relacionados directamente a la entidad DatElementos a través de una relación de 1 a muchos (*). Las Áreas de Resultados Claves (ARC) estarán representadas por la entidad DatArc y esta a su vez tendrá una relación con los planes, objetivos y actividades; pudiendo cada uno de estos elementos estar relacionados a varias ARC. Los elementos de la planificación pueden tener relacionados varios involucrados asociados a la vez, estos son representados por la entidad DatInvolucrados. Los Criterios de Medidas son representados por la entidad DatCriteriomedida y estarán asociados directamente a

los objetivos mediante una relación de 1 a 1, siendo medido de esta forma cada objetivo por su respectivo criterio de medida; por otra parte estos estarán asociados a los involucrados mediante una relación de mucho a mucho, dando lugar a una nueva entidad denominada DatCriteriomedidaDatinvolucrados**.**

2.7 Validación del modelo de diseño propuesto

La aplicación de métricas al diseño e implementación de un producto de software constituyen un elemento fundamental a la hora de evaluar la calidad del mismo.

"Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad del módulo, medidas que pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes." (Linares Domínguez, y otros, 2009)

De esta forma se procede a realizar una evaluación del diseño propuesto para el desarrollo de la solución para la obtención de gráficos del sistema de planificación de actividades SIPAC.

Atributos de calidad que se abarcan durante el diseño:

Responsabilidad: Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.

Complejidad del mantenimiento: Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto.

Complejidad de implementación: Grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: Significa cuán reutilizable es una clase o estructura de clase dentro de un diseño de software.

Acoplamiento: Dependencia o interconexión de una clase o estructura de clase respecto a otras.

Cantidad de pruebas: Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto diseñado.

2.7.1 Métricas de diseño aplicadas para evaluar la calidad del diseño propuesto

Dada la necesidad de conocer la calidad del diseño realizado y teniendo en cuenta que en la solución no existe herencia, por lo que no se pueden aplicar las métricas que evalúan estos aspectos, se decidió aplicar las siguientes métricas:

Tamaño operacional de clase (TOC): Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

Relaciones entre clases (RC): Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

- **Resultados de la métrica Tamaño operacional de clase (TOC).**

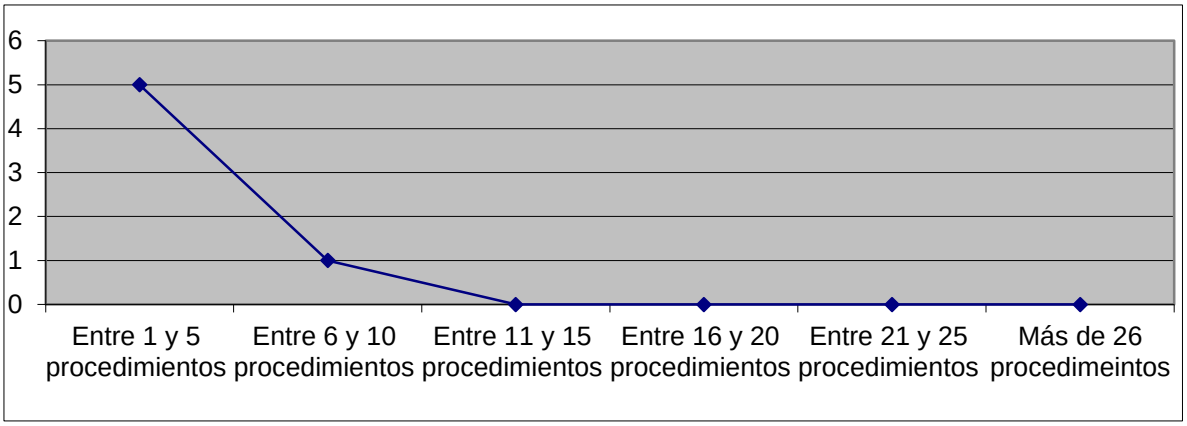

Figura 8 Cantidad de procedimientos obtenidos por intervalos

Cantidad de procedimientos presentes por intervalos, en una determinada cantidad de clases, luego de la aplicación de la métrica Tamaño Operacional de Clases. Obteniéndose como resultado entre 1 y 5 procedimientos en la mayoría de las clases utilizadas en la solución.

Resultado en % obtenido con el uso del instrumento que muestra la cantidad de procedimientos según los intervalos definidos.

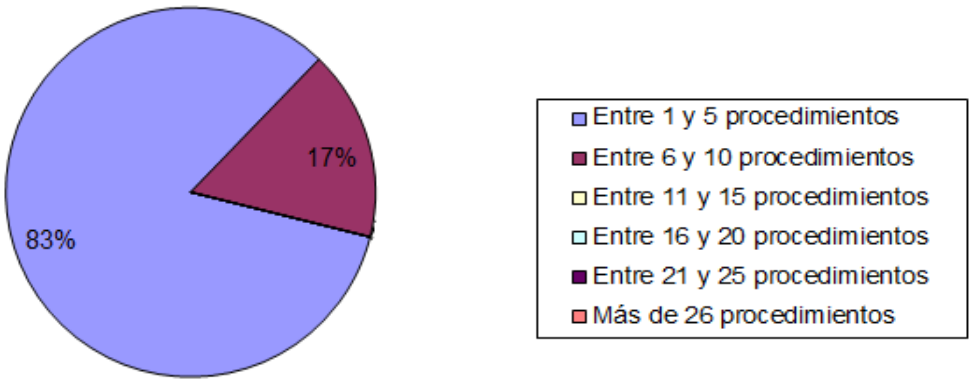

Figura 9 Porciento de la cantidad de procedimientos obtenidos por intervalos

Porciento de procedimientos presentes en una determinada cantidad de clases, luego de la aplicación de la métrica Tamaño Operacional de Clases. Obteniéndose como resultado entre 1 y 5 procedimientos en el 83% de las clases utilizadas en la solución.

Representación de los resultados obtenidos mediante la aplicación de la métrica TOC, específicamente del atributo Responsabilidad.

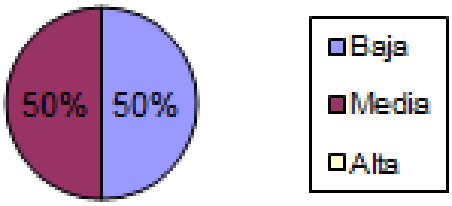

Figura 10 Representación de la incidencia del atributo Responsabilidad

Resultado en porciento de la incidencia del atributo Responsabilidad luego de la aplicación de la métrica TOC. Evidenciándose una baja Responsabilidad en el 50% de las clases utilizadas en la solución.

Representación de los resultados obtenidos mediante la aplicación de la métrica TOC, específicamente del atributo Complejidad.

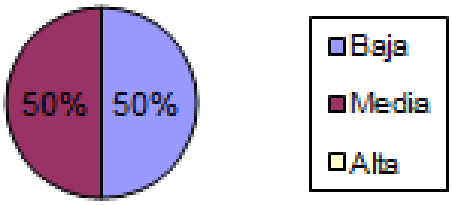

Figura 11 Representación de la incidencia del atributo Complejidad

Resultado en porciento de la incidencia del atributo Complejidad luego de la aplicación de la métrica TOC. Evidenciándose una baja Complejidad en el 50% de las clases utilizadas en la solución.

Representación de los resultados obtenidos mediante la aplicación de la métrica TOC, específicamente del atributo Reutilización.

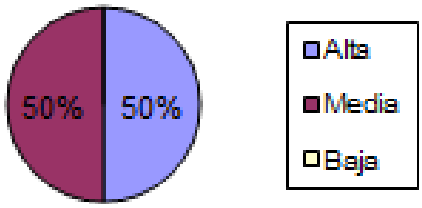

Figura 12 Representación de la incidencia del atributo Reutilización

Resultado en porciento de la incidencia del atributo Reutilización luego de la aplicación de la métrica TOC. Evidenciándose una alta Reutilización en el 50% de las clases utilizadas en la solución.

Luego de realizar un análisis de los resultados obtenidos al aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el desarrollo e implementación de la solución para la obtención de gráficos muestra resultados favorables, debido a que el 83% de las clases posee menor cantidad de operaciones. Los atributos de calidad se encuentran en un nivel satisfactorio en el 50% de las clases; de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

- **Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC).**

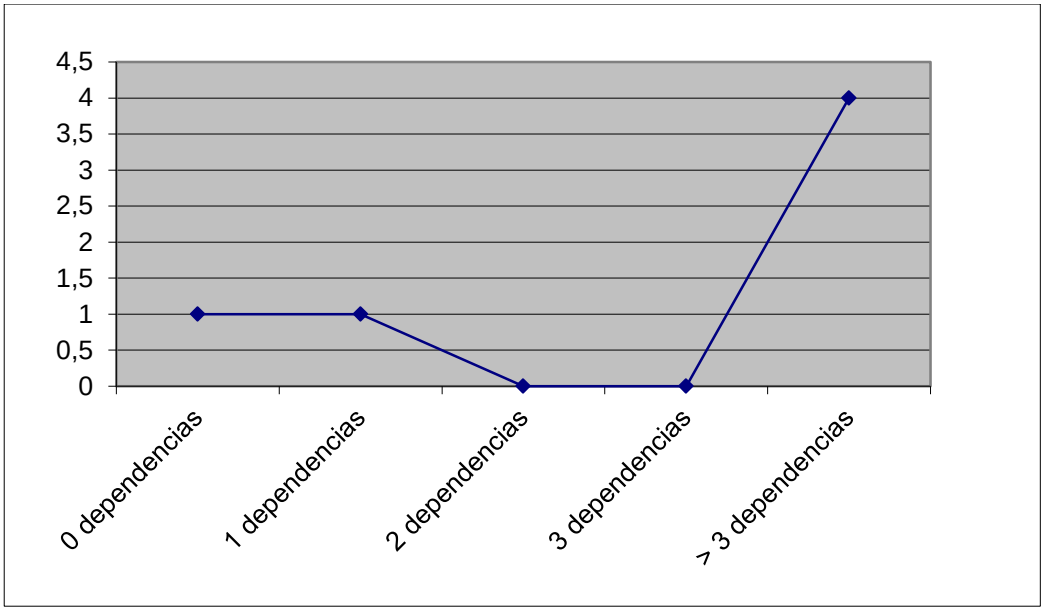

Figura 13 Cantidad de dependencias en las relaciones entre clases

Representación de la cantidad de dependencias existentes entre las clases, luego de la aplicación de la métrica Relaciones entre Clases. Obteniéndose como resultado más de 3 dependencias en la mayoría de las clases utilizadas en la solución.

Resultado obtenido en % con el uso del instrumento que muestra la dependencia entre las clases.

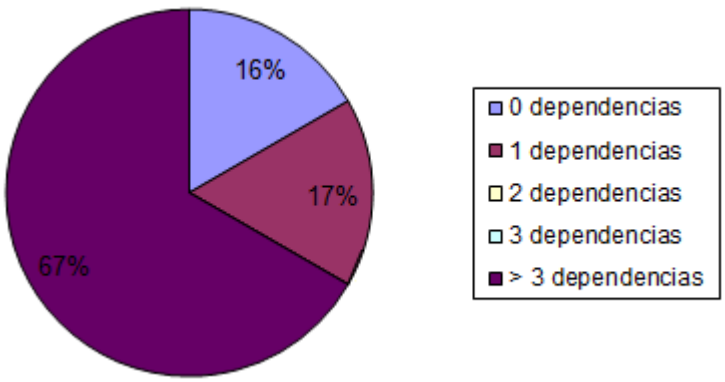

Figura 14 Porciento de las dependencias entre las clases

Representación en porciento de las dependencias existentes entre las clases, luego de la aplicación de la métrica Relaciones entre Clases. Obteniéndose como resultado una alta dependencia entre las clases utilizadas en la solución, específicamente en el 67% de ellas.

Representación de los resultados obtenidos mediante la aplicación de la métrica RC, específicamente del atributo Acoplamiento.

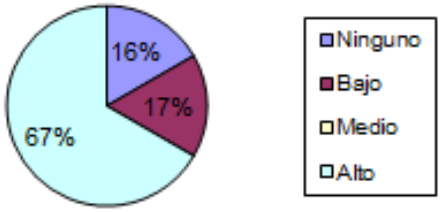

Figura 15 Representación de la incidencia del atributo Acoplamiento

Resultado en porciento de la incidencia del atributo Acoplamiento luego de la aplicación de la métrica RC. Evidenciándose un alto Acoplamiento en el 67% de las clases utilizadas en la solución, debido a la alta dependencia existente entre las mismas.

Representación de los resultados obtenidos mediante la aplicación de la métrica RC, específicamente del atributo Complejidad de Mantenimiento.

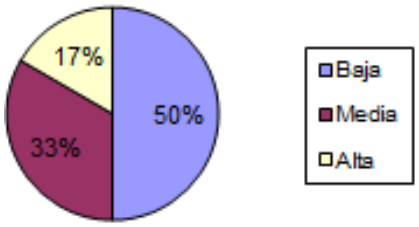

Figura 16 Representación de la incidencia del atributo Complejidad de Mantenimiento

Resultado en porciento de la incidencia del atributo complejidad de mantenimiento luego de la aplicación de la métrica RC. Evidenciándose una baja Complejidad de Mantenimiento en el 50% de las clases utilizadas en la solución.

Representación de los resultados obtenidos mediante la aplicación de la métrica RC, específicamente del atributo Cantidad de Pruebas.

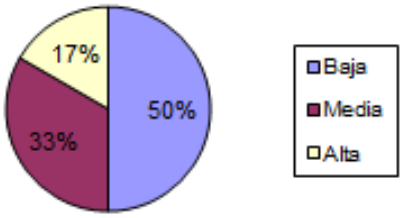

Figura 17 Representación de la incidencia del atributo Cantidad de Pruebas

Resultado en porciento de la incidencia del atributo cantidad de pruebas luego de la aplicación de la métrica RC. Evidenciándose baja la Cantidad de Pruebas en el 50% de las clases utilizadas en la solución.

Representación de los resultados obtenidos mediante la aplicación de la métrica RC, específicamente del atributo Reutilización.

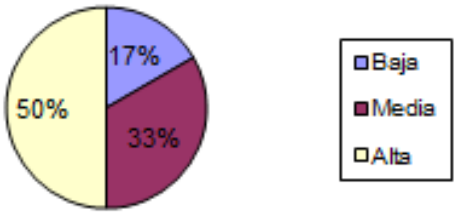

Figura 18 Representación de la incidencia del atributo Reutilización

Resultado en porciento de la incidencia del atributo Reutilización luego de la aplicación de la métrica RC. Evidenciándose una alta Reutilización en el 50% de las clases utilizadas en la solución.

Después de realizar un análisis de los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el desarrollo e implementación de la solución para la obtención de gráficos se encuentra en los límites aceptables de calidad, debido a que existe un (67%) de las clases con más de tres dependencias respecto a otras. Los atributos de calidad arrojaron como resultado que existe un alto acoplamiento entre las clases, específicamente en el 67% de ellas debido a la alta dependencia entre las mismas, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización, se comportan favorablemente para un 50 % de las clases.

2.8 Conclusiones del capítulo

El análisis y diseño realizado incluye todas especificaciones del cliente. Asimismo el diseño propuesto se comporta de forma satisfactoria teniendo en cuenta las métricas Relaciones entre Clases y Tamaño Operacional de las Clases para cada uno de los atributos evaluados.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN.

3.1 Introducción

El presente capítulo comprenderá los elementos principales de la implementación de la solución: la estructura física de la misma, la definición de los estándares de codificación, la descripción de clases y la representación de la estrategia de integración a seguir. Además se realizarán un conjunto de pruebas de caja blanca y caja negra con el objetivo de validar el cumplimiento de la idea a defender de la cual es objeto este trabajo de diploma. Asimismo estas pruebas son ejecutadas para avalar el cumplimiento de las exigencias del cliente y garantizar la calidad de la solución.

3.2 Implementación de la solución

La implementación de la solución para la obtención de gráficos de los indicadores proporcionará como resultado un producto que cumpla las exigencias y necesidades existentes en SIPAC.

3.2.1 Estructura física de la solución

La estructura física de la solución para la obtención de gráficos de los indicadores perteneciente a SIPAC se rige por la estructura del marco de trabajo definida para Cedrux.

TABLA 2 Estructura física de SIPAC

Carpeta	Descripción
apps	Almacena la lógica del negocio, es decir los controladores y el modelo de cada uno de los componentes pertenecientes a los subsistemas.
comun	Incluye la configuración concreta para el subsistema.
xml	Se encuentra dentro de la carpeta *recursos.* Contiene los ficheros: *ioc:* - Donde se publican los servicios que brindan cada uno de los componentes de SIPAC mediante un lenguaje de descripción de servicios web (WSDL).
pdo	Contiene al Subsistema de Planificación de Actividades que incluye la solución para la obtención de gráficos de los indicadores e incluye las carpetas que se describen en la

	próxima tabla.

Tabla 3 Estructura física de la solución para la obtención de gráficos

Carpeta	Descripción
comun	Contiene la configuración de la solución para la obtención de gráficos de los indicadores, incluye la carpeta *recursos* y dentro de esta una denominada *xml* que contiene los ficheros: *validation:*- Examina las precondiciones antes de ejecutar una función en el servidor dependiendo del tipo de parámetros, la acción y el usuario que la ejecute. *exception*: - Almacena las descripciones de los mensajes de las excepciones que pueden ser lanzadas por el servidor.
controllers	Contiene las clases controladoras del componente encargadas de realizar las funcionalidades.
models	Esta carpeta contiene otras dos, las cuales a su vez agrupan clases encargadas del acceso a los datos. Estas carpetas son: *bussines y domain.*
bussines	Contendrá las clases necesarias para acceder a los datos que persisten en la base de datos.
domain	Debe contener las clases generadas por el marco de trabajo Doctrine a partir de cada una de las tablas existentes en la base de datos.
generated	Contiene las clases generadas por Doctrine, son las clases bases que contienen los atributos y dependencias entre las tablas de la base de datos. De estas heredarán las

	clases que se encuentren en el paquete *domain*.
services	Este paquete contiene las clases y funcionalidades de los servicios que ofrecerá el componente.
validators	Agrupa las clases y funcionalidades correspondientes a las validaciones que realiza el servidor antes de ejecutar una determinada función.
views (propia del marco de trabajo)	Contiene los paquetes idioma y scripts, que contiene el idioma en que se va a mostrar la aplicación y las páginas clientes.
web	Contiene las vistas de todos los subsistemas y componentes de la aplicación, contiene un fichero *index.php* que almacena la dirección del archivo de configuración y a través de este inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento.
views (específica de la solución)	Esta contiene los css y js necesarios para visualizar todo el contenido de la presentación de la solución para la obtención de gráficos de los indicadores de la planificación.
css	Contiene las clases necesarias para darle la estructura gráfica a la solución, 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.

3.2.2 Estándares de código

Un estándar de código se basa en la organización y aspecto físico de un software con el objetivo de facilitar la lectura, entendimiento, mantenimiento del código, reutilización a lo largo del proceso de desarrollo y no en la lógica del programa.

Un estándar de programación no busca únicamente definir la nomenclatura de las variables, objetos, métodos y funciones, sino que además incluye reglas para el orden y legibilidad del código escrito. Teniendo como punto de partida lo anteriormente expuesto se definen 3 partes principales dentro de un estándar de programación:

Nomenclatura de las clases:

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación *PascalCasing*, esta especifica que los identificadores y nombres de variables, métodos y funciones que están compuestos por múltiples palabras juntas, deben iniciar cada palabra con letra mayúscula, lo que posibilita que con sólo leerlo se reconozca el propósito de la misma.

Ejemplo: *CmpGraficador* En este caso el nombre de clase está compuesto por 2 palabras iniciadas cada una con letra mayúscula.

Nomenclatura según el tipo de clases:

Clases controladoras: Las clases controladoras después del nombre llevan la palabra:"Controller".

Ejemplo: CmpGraficadorController

Clases de los modelos:

Business (Negocio): Las clases que se encuentran dentro de *business* después del nombre llevan la palabra: "Model". Ejemplo: CmpGraficadorModel.

Domain (Dominio): Las clases que se encuentran dentro de *domain* el nombre que reciben es el de la tabla en la base de datos. Ejemplo: DatElementos.

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

Ejemplo: BaseDatElementos.

Nomenclatura de las funcionalidades y atributos:
El nombre a emplear para las funciones y los atributos se escribe con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará notación *CamelCasing* que es similar a la antes mencionada: *PascalCasing* con la excepción de la primera letra.

Ejemplo de método: *obtenerGraficoComposicion*. El nombre de método está compuesto por 3 palabras, la primera en minúsculas y las siguientes iniciando con letra mayúscula.

Las principales funcionalidades de las clases controladoras se les asigna el nombre y seguidamente la palabra:"*Action*" Ejemplo: *obtenerGraficoComposicionAction().*

Ejemplo de atributo: *arrElementos*. El nombre del atributo está compuesto por 2 palabras, la primera en minúsculas y la segunda comenzando con letra mayúscula.

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

3.2.3 Integración de Highcharts con el Marco de Trabajo

Como se explicó anteriormente, una de las razones por las cuales se seleccionó la biblioteca Highcharts fue debido a que ya existía en CEIGE experiencia de su integración con el Marco de Trabajo Sauxe. Tomando como referencia el informe que generó esta investigación, se realizó la personalización de estas configuraciones para SIPAC. Esta personalización se encuentra en el expediente de proyecto en el artefacto CIG-SPA-N-Informe integración Highcharts.

3.2.4 Descripción de las clases y funcionalidades de la solución

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

TABLA 4 Descripción clase CmpGraficadorController

Nombre: CmpGraficadorController
Tipo de clase: Controladora

Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**:
ObtenerGraficoComposicionAction()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de obtener los elementos que forman el gráfico de composición.
ObtenerGraficoPorcientoAction()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de obtener los elementos que forman el gráfico de porciento de cumplimiento.
ObtenerGraficoInvolucradosAction()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de obtener los elementos que forman el gráfico cantidad de involucrados.
ObtenerEstadosElementosAction()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de obtener el estado de los elementos que forman el gráfico de composición.
CargarAction()	Regula el flujo de información desde la vista hasta el modelo y viceversa, con el objetivo de obtener los elementos que se desea abrir su especificación.

Clases Model:

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

TABLA 5 Descripción clase CmpGraficadorModel

Nombre: CmpGraficadorModel	
Tipo de clase: Model	
Para cada responsabilidad:	
Nombre:	**Descripción:**
ObtenerGraficoComposicion($idelemento, $idtipoelemento, $idusuario)	Funcionalidad que se encarga de gestionar la información con vista a obtener como resultado los datos necesarios para el gráfico de composición.
ObtenerGraficoPorciento($idelemento, $idusuario)	Funcionalidad que se encarga de gestionar la información, con el objetivo de obtener como resultado los datos necesarios para el gráfico de porciento de cumplimiento de los elementos de la planificación.
ObtenerGraficoInvolucrados($idelemento, $idtipoelemento, $idusuario)	Funcionalidad que se encarga de gestionar la información necesaria para obtener como resultado, los datos para el gráfico de cantidad de involucrados por tipo de elemento de la planificación.
ObtenerEstadosElementos($idtipoelemento, $idelemento)	Funcionalidad que se encarga de gestionar la información para la obtención de los estados de cumplimiento de los diferentes elementos de la planificación.
ObtenerDetallesTipoElementos($idtipoelemento, $den, $tipoden)	Funcionalidad que se encarga de gestionar la información, permitiendo listar los elementos seleccionados según el tipo de elemento de la planificación.

Domain:

TABLA 6 Descripción clase DatElementos

Nombre: DatElementos	
Tipo de clase: Domain	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
DameElementosDadoIdelementoyTipoelemento($idelemento)	Funcionalidad que posibilita obtener la cantidad de elementos por tipo elemento relacionados a un elemento padre.
DameArcElementoyTipoelemento($idelemento,$idtipoelemento, $boolean)	Funcionalidad que posibilita obtener la cantidad de áreas de resultados claves (arc) relacionados a un elemento padre.
DameObjetivosDadoIdelementoyTipoelemento($idelemento)	Funcionalidad que posibilita obtener los objetivos relacionados a un elemento padre.
DameElementosParaEstados($idelemento)	Funcionalidad que posibilita obtener el idelemento y idtipoelemento de los elementos relacionados a un elemento padre.
DameEstadoCumplimientoElementos($arrayelement)	Funcionalidad que posibilita obtener el estado de cumplimiento y la cantidad por

	estado de los elementos relacionados a un elemento padre.
DameElemento($idelemento)	Funcionalidad que permite obtener las características del elemento especificado.

3.3 Pruebas de software

Un producto tiene calidad cuando se puede garantizar que este haya pasado exitosamente por un adecuado proceso de pruebas; para el software esto se comporta de la misma manera.

De acuerdo a la IEEE[9] una prueba se define como: "Actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente". (Elizondo, 2001) Una prueba se considera exitosa si encuentra alguna deficiencia en el software. Para obtener diferentes tipos de errores en el sistema se hace necesario aplicar un amplio conjunto de pruebas. (Universidad Autónoma de Baja California, 2013)

Niveles de pruebas

Cuando se quiera evaluar dinámicamente un sistema se debe comenzar por los componentes más simples y pequeños e ir avanzando gradualmente hasta probar el software completamente. Es así que se procede a explicar los diferentes niveles de pruebas que se le pueden realizar a un software:

- **Pruebas unitarias:** la prueba unitaria o de unidad se centra en el módulo, estas constituyen comprobaciones que se le realizan a las unidades lógicas del sistema. (Universidad Autónoma de Baja California, 2013)
- **Pruebas de integración:** tienen como objetivo captar los módulos probados en las pruebas de unidad y detectar errores relacionados con la interacción entre dichos módulos. (Universidad Autónoma de Baja California, 2013)
- **Pruebas de validación:** se prueba el sistema como un todo para verificar si cumple con los RF y RNF. (Universidad Autónoma de Baja California, 2013)
- **Pruebas de aceptación:** son realizadas con los clientes y define su aceptación del sistema. (Universidad Autónoma de Baja California, 2013)

[9] Instituto de Ingenieros Eléctricos y Electrónicos.

Métodos de Prueba

Constituye un enfoque sistemático, autónomo del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba[10] para detectar diversos tipos de errores, el cual se centra en dos perspectivas diferentes (Softwarefactory, 2013):

- La lógica interna del programa utilizando técnicas de diseño de casos de prueba de "Caja blanca". (Softwarefactory, 2013)
- Los requisitos del software utilizando técnicas de diseño de casos de prueba de "Caja negra". (Softwarefactory, 2013)

Objetivo

El proceso de pruebas por el cual se regirá el aseguramiento de la calidad de la solución tiene como meta determinar cómo y en qué sentido la solución cumple con los intereses del cliente, teniendo como referencia los requisitos establecidos y las restricciones impuestas. En ese ámbito se trazan un conjunto de objetivos dentro de los que se sitúan:

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

Con la idea de diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. (Hernández, 2008)

Alcance

Las pruebas de software en el desarrollo de un producto informático abarcan un conjunto de ramas importantes que garantizan el correcto funcionamiento del mismo, desde la funcionalidad de los primeros prototipos, la estabilidad, cobertura y rendimiento de la arquitectura, hasta el producto final.

Cuando se ejecuta una prueba, se realiza una verificación de los resultados obtenidos, comprobándose si se asemejan a los resultados esperados. Si la salida no es la esperada, indica la ocurrencia de un error y en ese caso es preciso corregirlo, esto implica todo un proceso de depuración de errores a través de los casos de prueba.

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

Se debe tener en cuenta el costo de tiempo y esfuerzo que traen aparejado los errores, generalmente se presentan en situaciones complejas y en ocasiones las soluciones no las puede determinar el propio desarrollador.

3.4 Prueba de Software que será aplicada a la solución

Las pruebas de software garantizan medir el resultado del desarrollo de un producto informático, permitiendo comprobar el rendimiento y funcionamiento del mismo.

Para comprobar el correcto funcionamiento de la solución para la obtención de gráficos de los indicadores de SIPAC se decidió aplicarle pruebas para el nivel de unidad en específico, basado en los métodos de caja blanca y caja negra.

3.4.1 Descripción general de las pruebas para el nivel de Unidad

Se le denomina Prueba de Unidad al proceso de separar y probar el correcto funcionamiento de las partes individuales de un programa para asegurar que cada uno de estas se ejecuta correctamente por separado.

Ventajas de usar este tipo de prueba:

1. Los errores son más fáciles de localizar.
2. Los errores están más acotados.
3. Se fomenta el cambio.
4. Se reducen los “efectos secundarios”.
5. Se da más seguridad al programador. (Unidad de Servicio. Arquitectura e Innovación, 2007)

“Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no fluyen correctamente, todas las demás pruebas no tienen sentido”. (Universidad Autónoma de Baja California, 2013)

Durante la implementación de la solución se adquiere un conocimiento del nivel de funcionamiento del mismo, lo que posibilitará que los desarrolladores se encarguen de aplicar las pruebas para este nivel mediante los métodos antes presentados y las técnicas correspondientes que serán especificadas más adelante.

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

- **Descripción:**

La Prueba de Caja Blanca también se conoce como Prueba de Caja Transparente o de Cristal.

En ese sentido el criterio de selección de casos de prueba buscará cierta cobertura no solo para la determinación de caminos independientes, sino también de valores para las condiciones de bucles dentro y fuera de sus límites operacionales basados en el contenido de los módulos.

Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento.

Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación:

Prueba del Camino Básico: Permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos. Los casos de prueba obtenidos garantizan que durante la prueba se ejecute al menos una vez cada sentencia del programa.

Prueba de Condición: Ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.

Prueba de Flujo de Datos: Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.

Prueba de Bucles: Método de prueba que se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución todos los bucles en sus límites operacionales. (Universidad Autónoma de Baja California, 2013)

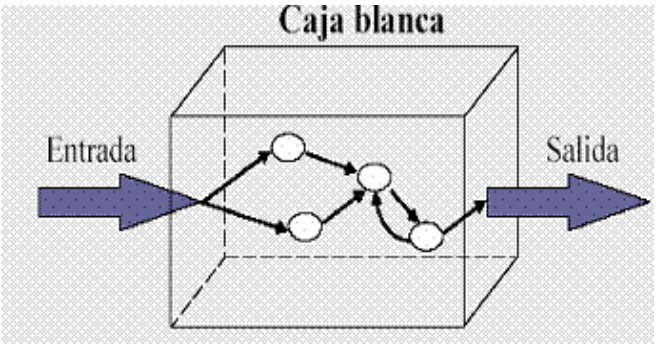

Figura 19 Prueba Caja Blanca

- **Aplicación:**

Según descripción de la prueba de caja blanca presentada con anterioridad y a partir de la necesidad de crear un producto de alta calidad que cumpla con los requisitos deseados por el cliente es preciso valorar qué tan certera ha sido la implementación de la solución para la obtención de gráficos de los indicadores de la planificación y para ello es necesario aplicar una de las técnicas que esta comprende, en este caso la 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 mediante las fórmulas pertinentes se calcula dicha complejidad:

De esta manera se analizan y enumeran las sentencias de código de uno de los procedimientos contenidos en la clase GraficadorModel, específicamente de la funcionalidad: ObtenerDetallesTiposElementos(), atendiendo a la alta importancia que tiene en el desarrollo de la solución. Esta se encarga de listar todos los elementos de la planificación de un mismo tipo seleccionado por el usuario relacionados a un elemento padre. La funcionalidad es invocada en la clase CmpgraficadorController mediante la funcionalidad Cargar.

```
//funcion para obtener los detalles de un tipo de elemento
public function ObtenerDetallesTipoElementos($idtipoelemento, $den, $tipoden) {
    $objetoDatElement= new DatElementos();          1
    $objetoDatCriterio= new DatCriteriomedida();    1

    $arrayElemento= $objetoDatElement->DameElementosParaEstados1($den); 1

  $cant=0;   1
  foreach($arrayElemento as $a){    2
  if($a['idtipoelemento']==$idtipoelemento){    3
  $arrayElementosMostrar[$cant]['idtipoelemento'] = $idtipoelemento;  3
  $arrayElementosMostrar[$cant]['idelemento'] = $a['idelemento'];     3
  $arrayElementosMostrar[$cant]['denominacion'] = $a['denelemento'];  3
  $arrayElementosMostrar[$cant]['estadocumplimiento'] = $a['estadocumplimiento']; 3
  $cant++;    3
    }   3
  }   2

  if(!isset($arrayElementosMostrar) && $idtipoelemento == 5){   4
  $arrayElemento1=$objetoDatElement->DameArcementoyTipoelemento($den, $tipoden,false);5
  $cant1=0; 5
  foreach($arrayElemento1 as $a1){  6
  $arrayElementosMostrar[$cant1]['idtipoelemento'] = $idtipoelemento; 6
  $arrayElementosMostrar[$cant1]['idelemento'] = $a1['idarc'];   6
  $arrayElementosMostrar[$cant1]['denominacion'] = $a1['denarc']; 6
  $arrayElementosMostrar[$cant1]['estadocumplimiento'] = null; 6
  $cant1++;   6
  }   6
  }   4
  if(!isset($arrayElementosMostrar) && $idtipoelemento == 4){   7
  $objetivos=$objetoDatElement->DameObjetivosDadoIdelementoyTipoelemento($den);  8
  $cant2=0;   8
  foreach($objetivos as $obj){  9
  $criterios=$objetoDatCriterio->dameCriterioMedidaNuevo($obj['idelemento']); 9
  foreach($criterios as $crit){  10
  $arrayElementosMostrar[$cant2]['idtipoelemento'] = $idtipoelemento;   10
  $arrayElementosMostrar[$cant2]['idelemento'] = $crit['idcriteriomedida'];   10
  $arrayElementosMostrar[$cant2]['denominacion'] = $crit['dencriterio'];    10
  $arrayElementosMostrar[$cant2]['estadocumplimiento'] = $crit['estadocumplimiento']; 10
  $cant2++; 10
  }     10
  }     9
  }       7
  return $arrayElementosMostrar;  11
}
```

Figura 20 Funcionalidad ObtenerDetallesTipoElementos

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

Nodo: Círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, un nodo en sí puede representar un proceso, una secuencia de procesos o una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: Saetas a través de las cuales se unen los Nodos y constituyen el flujo de control del procedimiento.

Regiones: Las regiones son las áreas delimitadas por las aristas y nodos.

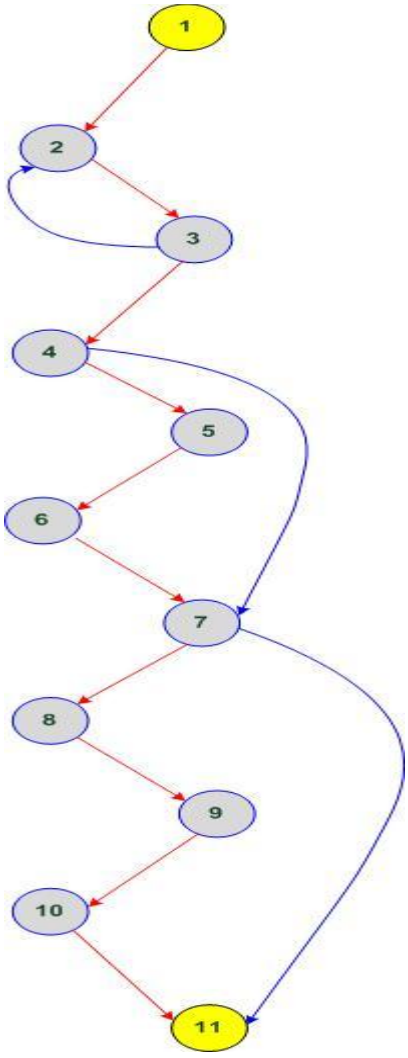

Figura 21 Grafo de Flujo asociado al algoritmo ObtenerDetallesTiposElementos

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) = (13 -11) + 2

V (G) = 4.

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) = 3 + 1

V (G) = 4.

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) = 4.

El cálculo efectuado mediante las fórmulas antes presentadas muestran una complejidad ciclomática de valor 4, de manera que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución.

Camino básico #1: 1 – 2 – 3 – 4– 7–11.

Camino básico #2: 1 – 2 – 3 – 2– 3 – 4– 7 – 11.

Camino básico #3: 1 – 2 – 3 – 4 – 5 – 6 – 7– 11.

Camino básico #4: 1 – 2 – 3 – 4 – 5 – 6 – 7– 8– 9 – 10 –11.

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

La prueba de caja blanca que se realizará a la funcionalidad ObtenerDetallesTiposElementos() invocada cuando se pretende obtener un listado de los elementos de la planificación de un mismo tipo relacionados a un elemento padre.

De forma general:
El elemento de la planificación seleccionado como padre para obtener la relación debe ser un plan (0), objetivo (1) o una actividad (3), definiéndose a su vez de esta forma el tipo de elemento del que se desee obtener la lista de elementos.

$idtipoelemento = 0, 1, 2, 3, 4,5.
$den = 9000002983. Elemento padre de la relación. (plan, objetivo, actividad).
$tipoden = 0, 1, 3 Significa que el tipo de elemento padre de la relación es un plan, objetivo o actividad respectivamente.

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

Descripción: Se debe obtener listado de elementos de un mismo tipo relacionados al elemento padre.

Condición de ejecución: Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: el $**idtipoelemento** que se desea obtener el listado, debe ser un valor numérico (0, 1, 2, 3, 4, 5), el valor correspondiente al parámetro **$den**, debe ser el id del elemento padre de la relación o sea un valor numérico. El valor correspondiente al parámetro **$tipoden** debe ser un valor numérico (0, 1, 3).

Entrada:
$idtipoelemento = 3
$den = 9000003205
$tipoden = 1

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el listado de elementos relacionados al elemento padre en este caso sean actividades:

Obteniéndose cómo resultado [3, 9000003246, Defender trabajo de diploma, Sin cumplir]

El algoritmo fue ejecutado correctamente.

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

Descripción y **Condición de ejecución** iguales que para el Caso de Prueba #1.

Entrada:

$idtipoelemento = 1
$den = 9000003205
$tipoden = 1

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el listado de elementos relacionados al elemento padre en este caso sean objetivos:

Obteniéndose cómo resultado [1, 9000003207, prueba, Sin cumplir]

El algoritmo fue ejecutado correctamente.

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

Descripción y **Condición de ejecución** iguales que para el Caso de Prueba #1.

Entrada:
$tipoelemento = 5
$den = 9000003205
$tipoden = 1

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el listado de elementos en este caso sean áreas de resultados claves.

Obteniéndose cómo resultado [5, 9000000154, prueba, null], [5, 9000000155, prueba1, null], [5, 9000000156, prueba2, null]

El algoritmo fue ejecutado correctamente.

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

Descripción y **Condición de ejecución** iguales que para el Caso de Prueba #1

Entrada:
$tipoelemento = 4
$den = 9000003205
$tipoden = 1

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el listado de elementos en este caso sean criterios de medida.

Obteniéndose cómo resultado [4, 9000000041, nuevo, Sobre cumplida]

El algoritmo fue ejecutado correctamente.

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

- **Descripción:**

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos.
Se centra en lo que se espera de un módulo, es decir, intenta encontrar casos en que el módulo no se atiene a su especificación (Maria Carmen Fernandez Panadero, 2013), esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa.

En esencia permite encontrar:

- 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.

Dentro de la prueba de caja negra se incluyen las Técnicas de Pruebas que serán descritas a continuación:

Partición de Equivalencia: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. (Universidad Autónoma de Baja California, 2013)

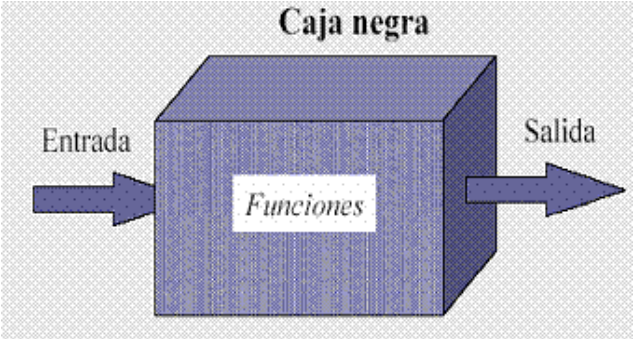

Figura 22 Prueba Caja Negra

- **Aplicación:**

Seguidamente se aplica la prueba de partición de equivalencia como parte de la realización de la prueba de caja negra sobre la interfaz que responde al requisito funcional Obtener Gráfico.

La Partición de Equivalencia divide el dominio de entrada de un programa en un número finito de variables de equivalencia. 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.

Para más detalles sobre las pruebas consultar los siguientes anexos:
Anexo 1: Caso de prueba de Caja Negra para validar el requisito funcional Obtener Gráfico.
Anexo 2. Descripción de variables para el caso de prueba.
Anexo 3. Juegos de datos a probar. (Universidad Autónoma de Baja California, 2013)

Después de haber aplicado los métodos de prueba descritos anteriormente para la funcionalidad ObtenerDetallesTiposElementos y el requisito funcional Obtener Gráfico, es necesario señalar que los resultados obtenidos son satisfactorios para el funcionamiento interno de la solución, así como para medir el comportamiento de la misma atendiendo a diferentes situaciones presentadas (entradas válidas y no válidas). Asimismo los RNF de la solución fueron validados como parte del producto SIPAC, este resultado se puede consultar en el artefacto CIG-SPA-N-i3516-CV del expediente de proyecto.

3.5 Resultados de las pruebas aplicadas

Luego de concluir la implementación de la solución, se realizaron por parte del equipo de calidad del proyecto las pruebas anteriormente descritas, las cuales arrojaron como resultado un total de 10 no

conformidades, de ellas 7 significativas y 3 no significativas, que fueron resueltas satisfactoriamente según su detección (ver figura 23). Emitiéndose de esta forma el Acta de aceptación del cliente que se encuentra en los Anexos.

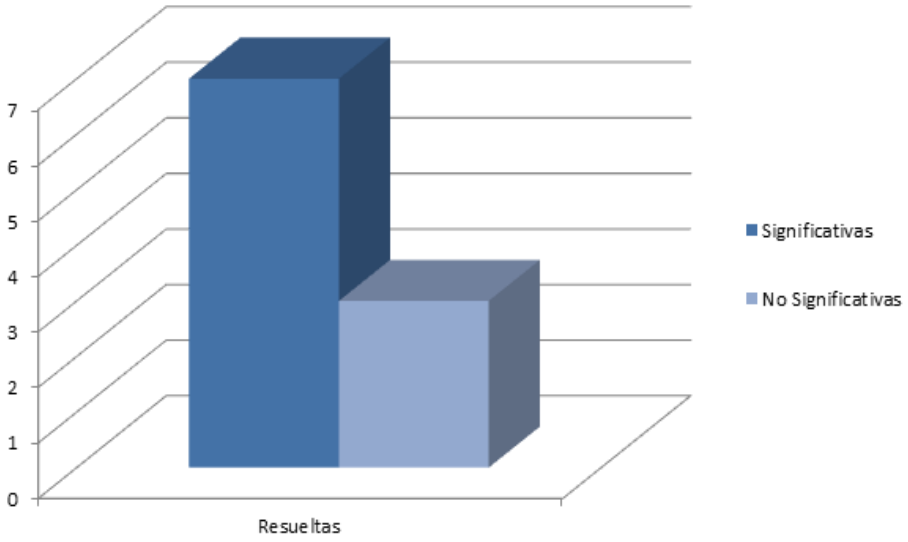

Figura 23 Cantidad de no conformidades detectadas a la solución

3.6 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 Acta de liberación en: Expedientes de proyecto\repo_doc_2.0\1. Ingeniería\1.3 implementación y prueba\ Actas de Liberación) y mediante pruebas de aceptación por parte del cliente (ver anexo: 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 la obtención de gráficos teniendo en cuenta que:

-La gestión de la planificación se realiza de una manera más interactiva hacia a todos los niveles organizacionales; los usuarios del sistema encargados del seguimiento de las tareas principales así como el análisis del cumplimiento de los objetivos de la entidad, pueden consultar el desglose de los elementos de la planificación, así como el cumplimiento de los mismos, mediante la representación gráfica de los indicadores: composición y estado de cumplimento, cantidad de involucrados, porciento de cumplimiento.

-Permite el aprovechamiento del tiempo en cuanto a la búsqueda de especificaciones de los elementos.

- Posibilita la generación de salidas del sistema en función de los indicadores claves de la planificación.

3.7 Conclusiones

Luego de realizar un conjunto de pruebas de software de caja blanca y caja negra en el nivel de unidad, mediante la utilización de casos de pruebas; para lo que se tuvo en cuenta las entradas, salidas y los resultados esperados para los mismos, se evidenció que la solución para la obtención de gráficos cuenta con un correcto funcionamiento y que satisface las necesidades del cliente.

CONCLUSIONES GENERALES

Después de terminado el presente trabajo de diploma se concluye que:

1. Mediante la elaboración del marco teórico y el estudio del estado del arte se evidenció la necesidad de desarrollar una solución para la obtención de gráficos para SIPAC que incluya a la biblioteca Highcharts.
2. Mediante la elaboración de los artefactos que define el modelo de desarrollo del CEIGE, se realizó el análisis y diseño de la solución, con el objetivo de obtener y describir la solución a implementar teniendo en cuenta las necesidades del cliente.
3. Se obtuvo la solución para la obtención de gráficos en SIPAC, que responde a las necesidades del cliente, la cual fue comprobada a través de pruebas de software efectuadas al obtenerse resultados positivos.
4. La solución para la obtención de gráficos para SIPAC permite que se realice el análisis y la interpretación de los indicadores claves de la planificación, mediante la representación visual de estos teniendo en cuenta la información almacenada en el sistema.

RECOMENDACIONES

Luego de ver concluido el presente trabajo de diploma y considerando cumplidos los objetivos trazados en el mismo, se recomienda:

- Incluir gráficos para la representación de los indicadores de los reportes que incluye 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 del Estado, Entidades Nacionales y las Administraciones Locales del Poder Popular.

- Realizar pruebas de carga y stress a la solución que permitan comprobar su eficiencia.

- Incluir una funcionalidad que permita modificar los gráficos con el objetivo de ofrecerle al usuario mejoras en la personalización de la solución.

REFERENCIAS BIBLIOGRÁFICAS

Answers Corporation. 2012. *Answers Corporation.* [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.answers.com/topic/php.

Apache . 2012. *Subversion.* [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://subversion.apache.org.

Apache Software Foundation. 2012. Versión 2.0 del Servidor HTTP Apache. [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://httpd.apache.org/docs/2.0/es.

Artola, Ariadna Rendón. 2013. *Proyecto Técnico.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2013.

—. 2013. *Requisitos no funcionales SIPAC 2.0.* Habana : s.n., 2013.

Astudillo, Marcello Visconti y Hernán. 2013. *Fundamentos de Ingeniería de Software.* [En línea] 2013. [Citado el: 15 de Marzo de 2013.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

Baryolo, Oiner Gómez, Borbón, Yoandry Morejón y Tejo, Darien Garcia. *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* Habana : s.n.

Boutell.Com. 2013. *Boutell.Com.* [En línea] 2013. [Citado el: 13 de Junio de 2013.] http://www.boutell.com/gd/.

Centro Tecnologias Gestion Datos. 2012. *Generador Dinámico de Reportes.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2012.

Cutter', Blades, Ramsay y Colin y Frederick, Shea. 2008. *Learning Ext JS.* s.l. : Packt Publishing, 2008.

definiciónabc. 2013. *definiciónabc.* [En línea] 2013. [Citado el: 8 de Junio de 2013.] http://www.definicionabc.com/tecnologia/datawarehouse.php.

DesarrolloWeb. 2010. *DesarrolloWeb.* [En línea] 8 de Noviembre de 2010. [Citado el: 27 de Noviembre de 2012.] http://www.desarrolloweb.com/manuales/21/.

Dickens, Jeff. 2011. *Softpedia.* [En línea] 23 de Diciembre de 2011. [Citado el: 13 de Junio de 2013.] http://www.softpedia.es/programa-PyChart-202891.html.

Eguiluz, Javier. 2013. *LIBROSWEB.* [En línea] 2013. [Citado el: 22 de Enero de 2013.] http://www.librosweb.es/ajax/.

Elizondo, Perla Ines Velasco. 2001. Prueba de Componentes de Software Basadas en el Modelo de JavaBeans. [En línea] Abril de 2001. [Citado el: 21 de Mayo de 2013.] http://www.cs.man.ac.uk/~velascop/publ/Tesis.pdf.

Escalona, María José y Koch, Nora. 2002. lsi.us.es. [En línea] 2002. [Citado el: 11 de 03 de 2013.] https://www.lsi.us.es/docs/informes/LSI-2002-4.pdf.

Gabriel, Olivera Sosa Angel. 2010. Reporte de Instalación de Apache. [En línea] 6 de Septiembre de 2010. [Citado el: 12 de Marzo de 2013.] http://es.scribd.com/doc/37187866/Requisitos-funcionales-y-no-funcionales.

García, Rubén. 2012. *Ingeniería del Software II.* [En línea] 11 de Septiembre de 2012. [Citado el: 15 de Marzo de 2013.] http://rhga0283.blogspot.com.

Generación del diagrama de secuencias de UML 2.1.1 desde esquemas preconceptuales. **Zapata, Carlos Mario ; Gilma Liliana Garcés. 2008.** 10, Medellín : Revista EIA, Diciembre de 2008. 1794-1237.

Hernández, Sergio Demián Gracián. 2006. [En línea] Mayo de 2006. [Citado el: 12 de Junio de 2013.] http://dgsa.uaeh.edu.mx:8080/xmlui/bitstream/handle/231104/1685/Flujo%20de%20trabajo%20del%20an%C3%A1lisis%20y%20dise%C3%B1o%20de%20Rup.pdf?sequence=1.

Hernández, Yorji Pérez. 2008. *Trabajo de Diploma para optar por el titulo de Ingeniero en Ciencias Informáticas.* Habana : s.n., 2008.

Ibañez, Joaquín. 2010. *Lider de Proyecto.com.* [En línea] 2010. [Citado el: 11 de junio de 2013.] http://www.liderdeproyecto.com/manual/los_requerimientos.html.

Instituto Tecnologico de Colima. 2013. *Instituto Tecnologico de Colima.* [En línea] 2013. [Citado el: 11 de Junio de 2013.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm.

Iván Pérez Nieto. 2011. *elcodigo.* [En línea] 2011. [Citado el: 27 de Noviembre de 2012.] http://www.elcodigo.net/tutoriales/diccionario.html#Origen.

Kuan, Joe. 2012. *Learning Highcharts.* [Libro] s.l. : Packt Publishing, 2012. 978-1-84951-908-3.

Laboratorio de Gestión de Proyectos, UCI. 2010. *Manual de usuario para GESPRO 12.05.* [En línea] 2010. [Citado el: 27 de Noviembre de 2012.] http://gespro-help.prod.uci.cu/#.

Linares Domínguez, Yasser y Aquino Leyva, Aylen. 2009. *Implementación del Modulo Contabilidad General del Sistema Integral de Gestion Cedrux.* Habana : s.n., 2009.

López, Patricia. 2012. *Ingenieria del Software I.* [Libro] s.l. : Universidad Cantabria-Facultad Ciencias, 2012.

Maria Carmen Fernandez Panadero, Julio Villena Román. 2013. uc3m. [En línea] 2013. [Citado el: 9 de 6 de 2013.] http://www.it.uc3m.es/ttrd/material/05-pruebas-de-programas.pdf.

OpenERP. 2009. *OpenERP.* [En línea] 2009. [Citado el: 27 de Noviembre de 2012.] http://www.openerpsite.com/erp-openerp-modulos.

PostgreSQL. 2013. *PostgreSQL.* [En línea] 2013. [Citado el: 22 de Enero de 2013.] http://www.postgresql.org/docs/9.2/interactive/index.html.

Puebla, Yoan Arlet Carrascoso y Gómez, Enrique Chaviano. 2009. Arquitectura orientada a servicios para el módulo de inventario del ERP cubano. [En línea] 15 de Mayo de 2009. [Citado el: 13 de Marzo de 2013.] http://www.gestiopolis.com/administracion-estrategia/erp-arquitectura-orientada-a-servicios.htm.

Quality Soft Argentina S.A. 2011. *Sistema ISIS.* [En línea] 2011. [Citado el: 27 de Noviembre de 2012.] http://www.sistemaisis.com.

Sistemas Computin. 2010. Lenguajes de modelado de objetos. [En línea] 1 de 6 de 2010. [Citado el: 8 de 6 de 2013.] http://sistemacomputin.blogspot.com/2010_06_01_archive.html.

Softwarefactory. 2013. Testing. [En línea] 2013. [Citado el: 21 de Mayo de 2013.] http://newsoftwarefactory.com/como-lo-hacemos/testing/.

Sommerville, Ian. 2004. *Requerimientos del software.* [Libro] 2004.

Soto, Oscar Armando Ortiz. 2009. *Patrones de Diseño.* [Documento] 9 de 6 de 2009. 20042020071.

spimeWiki. 2013. *spimeWiki.* [En línea] 2013. [Citado el: 22 de Enero de 2013.] http://wiki.spimebox.com/doku.php?id=framework.

SynerPlus. 2012. SynerPlus. [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.synerplus.es/ERP/Selenne-ERP/9.html.

Unidad de Servicio. Arquitectura e Innovación. 2007. *slideshare.* [En línea] Junio de 2007. [Citado el: 9 de Junio de 2013.] http://www.slideshare.net/dbaccess/pruebas-unitarias-uso-de-nunit-dentro-de-proyectos-net.

Universidad Autónoma de Baja California. 2013. *Universidad Autónoma de Baja California.* [En línea] 2013. [Citado el: 11 de Junio de 2013.] fcqi.tij.uabc.mx/usuarios/luisgmo/data/8.1%20prb-cal-mant.pdf.

Universidad Experimental Simón Rodriguez. 2010. *Universidad Experimental Simón Rodriguez.* [En línea] 27 de Mayo de 2010. [Citado el: 8 de Junio de 2013.] http://www.slideshare.net/rosmelys/trabajo-business-intelligence.

Vesterinen, Konsta. 2012. *doctrine.* [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.doctrine-project.org.

Villa, David. 2013. *Pensar en C++.* [En línea] 2013. [Citado el: 16 de Marzo de 2013.] http://arco.esi.uclm.es/~david.villa/pensar_en_C++/vol2/ch09s02.html.

Visual Paradigm. 2012. *Visual Paradigm.* [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.visual-paradigm.com.

W3C. 2010. *W3C.* [En línea] 2010. [Citado el: 27 de Noviembre de 2012.] http://www.w3.org/standards/xml/.

Zend Technologies. 2012. *Zend Technologies.* [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://framework.zend.com/manual/1.12/en/manual.html.

BIBLIOGRAFÍA

Artola, Ariadna Rendón. 2013. *Proyecto Técnico.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2013.

—. 2013. *Requisitos no funcionales SIPAC 2.0.* Habana : s.n., 2013.

Aylienn Aquino Leiva, Yasser Linares Domínguez. 2009. *Implementación del Modulo Contabilidad General del Sistema Integral de Gestion Cedrux.* Habana : s.n., 2009.

Baryolo, Oiner Gómez, Borbón, Yoandry Morejón y Tejo, Darien Garcia. *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* Habana : s.n.

Centro Tecnologias Gestion Datos. 2012. *Generador Dinámico de Reportes.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2012.

Cutter', Blades, Ramsay y Colin y Frederick, Shea. 2008. *Learning Ext JS.* s.l. : Packt Publishing, 2008.

Generación del diagrama de secuencias de UML 2.1.1 desde esquemas preconceptuales. **Zapata, Carlos Mario ; Gilma Liliana Garcés. 2008.** 10, Medellín : Revista EIA, Diciembre de 2008. 1794-1237.

Hernández, Yorji Pérez. 2008. *Trabajo de Diploma para optar por el titulo de Ingeniero en Ciencias Informáticas.* Habana : s.n., 2008.

Introducción a la Calidad de Software. **Echeverry, Ana Maria Lopez, Cabrera, Cesar y Ayala, Luz Estela Valencia. 2008.** No 39, Pereira : Scientia et Technica, 2008. ISSN 0122-1701.

Kuan, Joe. 2012. *Learning Highcharts.* [Libro] s.l. : Packt Publishing, 2012. 978-1-84951-908-3.

López, Patricia. 2012. *Ingenieria del Software I.* [Libro] s.l. : Universidad Cantabria-Facultad Ciencias, 2012.

Sommerville, Ian. 2004. *Requerimientos del software.* [Libro] 2004.

Soto, Oscar Armando Ortiz. 2009. *Patrones de Diseño.* [Documento] 9 de 6 de 2009. 20042020071.

GLOSARIO DE TÉRMINOS

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

ARC: Áreas de Resultados Claves, agrupación de varias especialidades o entidades con un mismo propósito que cumplir.

Composición del elemento de la planificación: Describe las relaciones que tienen entre ellos los elementos de la planificación.

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

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

Gráfico: Representación visual de los indicadores de la planificación, muestra el comportamiento de un elemento de la planificación dando un criterio determinado.

Indicador: Características que definen a los elementos de la planificación mediante las cuales se analizará y comparará la información.

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

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.

Porciento de cumplimiento: Describe el porciento de cumplimiento de los elementos de la planificación.

Componente: Es un fragmento de un sistema de software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas.

Métricas: medida estadística, no cuantitativa que se aplica a todos los aspectos de calidad de software, los cuales pueden ser medidos desde diferentes puntos de vista como el análisis, construcción, funcional, entre otros.

ANEXOS

Anexo 1: Caso de prueba de Caja Negra para validar el requisito funcional Obtener Gráfico.

Requisito	**Descripción general**	**Escenarios de prueba**	**Flujo del escenario**
Obtener Gráfico	El requisito funcional Obtener Gráfico consiste en seleccionar un elemento de la planificación (plan, objetivo, actividad) del cual se quiera generar los gráficos, una vez seleccionado el elemento deseado por el usuario se presiona el botón “Graficar”, mostrándose de esta forma una ventana con los tipos de gráficos a generar de acuerdo a los distintos indicadores de la planeación (Composición y Estado de los elementos de la planeación, Porciento Cumplimiento de los elementos y Cantidad Involucrados por elemento), posibilitando de esta forma escoger el gráfico deseado por el usuario.	ESC#1: Obtener Grafico.	1-Se selecciona el elemento de la planificación que se desea obtener los gráficos. 2-Se presiona el botón “Graficar” que permite mostrar la interfaz principal de la solución. 3-Se muestra la interfaz principal de la solución para la obtención de gráficos, permitiendo escoger el gráfico según el indicador analizar. 4-Se presiona sobre la imagen del gráfico que se desee generar. 5-Se muestra el gráfico generado con los datos pertinentes.

Anexo 2: Descripción de variables para el caso de prueba.

No.	Nombre del Campo	Clasificación	Puede ser Nulo	Descripción
1	Idelemento	Check Box	No	Elementos que se pueden seleccionar para graficar.

Anexo 3. Juegos de datos a probar.

Id escenario	Escenario	Idelemento	Respuesta del Sistema	Resultado
ESC 1	**Obtener Gráfico**	**V(90112)**	**Se obtienen los gráficos con los indicadores de la planificación.**	**Correcto**

Anexo 4: Acta de aceptación del cliente

CENTRO DE INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

Acta de aceptación del proyecto

UCI Universidad de las Ciencias Informáticas

En la Habana a los 25 días del mes de mayo de 2013

De una parte, la dirección del proyecto SIPAC, representado por la **Ing. Mairelys Fernández González**, quien a los fines y efectos derivados del presente documento se denominará como **PARTE CLIENTE**, y de otra Parte el equipo de desarrollo de la "Solución para la obtención de gráficos del Sistema de Planificación de Actividades SIPAC.", representado en este acto por **Antonio Díaz Cedeño**, que a los fines y efectos derivados del presente documento se denominarán como **PARTE COMERCIALIZADORA**.

Primero: Que en cumplimiento de los acuerdos, han sido efectuadas las actividades que se describen, **Las partes** DECLARAN:

CONSIDERANDO: Que se han efectuado las actividades siguientes:

1. Levantamiento de requisitos.
2. Análisis y diseño.
3. Implementación.
4. Pruebas internas y de liberación.

CONSIDERANDO: Que las actividades realizadas han sido desarrolladas con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las partes**.

CONSIDERANDO: Que las actividades que se han ejecutado cumplen con los requisitos de la **PARTE CLIENTE**.

CONSIDERANDO: que la **PARTE COMERCIALIZADORA** ha entregado la documentación que avala la ejecución de este acto a la **PARTE CLIENTE.**

POR TANTO: **Las partes** acuerdan formalizar mediante el Acta, Aceptadas las actividades que han sido ejecutadas en esta fecha.

CENTRO DE INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

Acta de aceptación del proyecto

UCI Universidad de las Ciencias Informáticas

Y para que así conste, se extiende la presente Acta en dos (2) ejemplares, rubricados por Las partes

Por la PARTE CLIENTE

Ing. Mairelys Fernández González

Por la PARTE COMERCIALIZADORA

Antonio Díaz Cedeño