Universidad de las Ciencias Informáticas

Facultad 6

Componente Flujo de Trabajo v2.0 para el Sistema de Planificación de Actividades SIPAC.

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

Autor: Amado Enrique Somoano Pérez

Tutores: Ing. Mairelys Fernández González
Ing. Tania Pérez Ramírez

La Habana, marzo de 2015

"Año 56 de la Revolución"

DECLARACIÓN DE AUTORÍA

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

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

_____

Amado E. Somoano Pérez

Autor

_____ _____

Mairelys Fernández González

Tutora

Tania Pérez Ramírez

Tutora

AGRADECIMIENTOS

Agradezco a mi familia por todo el amor y comprensión que me brindaron, en especial a mi mamá por ser el ejemplo que siempre guiará todos mis pasos. Jesús, mi amigo, mi amor, gracias por tu apoyo y comprensión, a mi papá, gracias por tu apoyo y amor incondicional, a mis abuelos, a mi hermana, a mi sobrinito y mi tía May-Ling por iluminarme con su existencia, gracias. ¡Los amo mucho!

A todos los amigos que hicieron inolvidable mi estancia en esta Universidad, en especial a Dalinda, Aylin y Laurita por aguantarme y quererme como soy. A mis compañeros de aula y profesores en especial profesora Aliosmi y Teresa de CEIGE han sido todos de gran ayuda.

Gracias también a mis amigos fuera de la UCI, Yaumel, Lien, que siempre me dieron fuerzas y ánimos para seguir. ¡Los quiero!

A mis tutoras Mairelys y Tania por la exigencia y el apoyo brindado, sin ustedes no hubiera podido hacer este trabajo, gracias por demostrarme que si se puede, son dos ángeles en mi vida y dos grandes amigas.

Muchas gracias por el apoyo y los consejos del tribunal, han sido muy decisivos en mi formación académica y cada señalamiento de ustedes será puesto en práctica durante toda mi vida profesional

Agradezco también a mi equipo de trabajo en la Facultad 3, Yisel, Leo, Sasha, al equipo de CSCPC, a los trabajadores de la Facultad 8 donde comencé mi formación profesional, a mi Directora Municipal en Joven Club y agradezco también a las personas que decían que no podía llegar, me hicieron aferrarme y luchar, cada día más por demostrar que como dice Silvio, (...) No hacen falta alas (...) a todos, muchas gracias.

.....Amado Enrique Somoano Pérez.....

DEDICATORIA

El presente trabajo de diploma va dedicado a mis padres y a mi mejor amigo y compañero de vida Jesús, por ser las personas que me aman por encima de su propia vida, a pesar de mis defectos. Por enseñarme que nada es imposible cuando de hacer un sueño realidad se trata. Por demostrarme lo que es el amor incondicional y por llenar mi vida de tantas alegrías y vivencias que siempre guardaré en lo más profundo de mi corazón. Los amo.

.....Amado Enrique Somoano Pérez.....

RESUMEN

El Sistema de Planificación de Actividades SIPAC constituye una herramienta para la gestión de las actividades a todos los niveles organizacionales, basado en la Instrucción no.1 del Presidente de los Consejos de Estado y de Ministros para la planificación de objetivos y actividades en los Órganos, Organismos de la Administración Central del Estado, Entidades nacionales y Administraciones Locales del Poder Popular. Cuenta con varios módulos encargados de generar las configuraciones necesarias para el seguimiento de las tareas principales de cada entidad, gestión de los posibles involucrados, nomencladores y niveles de subordinación basados en reglas de la compartimentación de la información. Independientemente de las funcionalidades que brinda el sistema, resulta engorroso ejecutar el proceso de Aprobación - Conciliación de los diferentes documentos para la planificación desde y hacia los diferentes niveles de dirección tanto en una entidad como a nivel de gobierno. El presente trabajo de diploma comprende el desarrollo del componente Flujo de trabajo v2.0 para el Sistema para la Planificación de Actividades SIPAC, en correspondencia con el proceso de Aprobación-Conciliación definido en la Instrucción No. 1. Dicho componente posibilita la especificación de estados y transiciones de los documentos de la planificación, definiéndose permisos de acceso a la información por niveles.

Palabras claves: aprobación, conciliación, estados, transiciones, documentos para la planificación, sistema.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El término planificar se atribuye al proceso de establecer objetivos con el fin de alcanzar determinados resultados y elegir un futuro curso de acciones para lograrlos. Particularmente la planificación se traza preguntas como: ¿Qué debe hacerse?, ¿Quién debe hacerlo? ¿Dónde, cuándo y cómo debe hacerse?, buscando obtener los resultados óptimos, en concordancia con el tiempo y los recursos disponibles. Sin embargo, la planificación no solo está referida al planteamiento de los objetivos y la previsión de los medios para lograrlos, sino que además comprende la constante toma de decisiones y la reducción de la incertidumbre a partir de conocimientos lo más aproximado a la realidad posible.

La planificación establece metas organizacionales, define estrategias, objetivos y políticas para lograr estas metas, desarrolla planes detallados buscando asegurar la implantación de las estrategias y así obtener el éxito esperado. Planificar no es más que diseñar un futuro deseado e identificar las formas y los recursos para lograrlo corroborando que la planificación constituye la base de una sólida gestión.

Se realizaron investigaciones sobre la planificación en diferentes entidades de Cuba, donde se detectaron problemas tales como la falta de integralidad en los planes económicos y en la mayoría de los casos no basados en objetivos concretos, además de una inadecuada vinculación entre los planes económicos con la planificación de las actividades y falta de coordinación de los planes de actividades. Estas dificultades estaban provocadas por la ausencia de un Órgano de Dirección del Gobierno para dirigir el proceso de planificación de actividades, que se encargara de asegurar el cumplimiento de los objetivos sumando a esto la ausencia de documentos rectores del gobierno que encauzara estos procesos.

Partiendo de la imperiosa necesidad de una planificación óptima y funcional surge en septiembre del 2011 la Instrucción No. 1 del Presidente de los Consejos de Estado y de Ministros; donde se describe cómo debe ser 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 como un sistema integral.

Por la insuficiencia de herramientas informáticas para la gestión de actividades a todos los niveles organizacionales y basándose en la Instrucción No. 1 surge el Sistema de Planificación de Actividades SIPAC que cuenta con varios módulos encargados de generar las configuraciones necesarias para el seguimiento de las tareas principales de cada entidad, la gestión de los posibles involucrados, nomencladores y niveles de subordinación basados en reglas de la compartimentación de la información.

Independientemente de las funcionalidades que brinda el sistema, no posibilita el establecimiento de los estados y transiciones de los documentos que regulen el Proceso Aprobación-Conciliación en cada entidad lo que dificulta el seguimiento y control de la Planeación Estratégica y Operativa tanto para la entidad como a nivel de gobierno teniendo en cuenta que no es posible:

- Definir los permisos de acceso a la información en cada nivel.
- La creación de estados que determinen el comportamiento de los documentos de manera uniforme.
- Establecer qué estado sucede a otro en el flujo de la información mediante precondiciones y/o post-condiciones que determinen la aprobación-conciliación de los documentos.

Dada la problemática expuesta se identifica el siguiente **problema a resolver:**

¿Cómo realizar la configuración del flujo de trabajo en el Sistema de Planificación de Actividades SIPAC, para una mejor ejecución del Proceso Aprobación – Conciliación de los diferentes documentos de la planificación?

El **objeto de estudio** en el cual se enmarca el problema anteriormente planteado es:

El Proceso Aprobación – Conciliación de los documentos para la planificación.

Con el propósito de darle solución al problema formulado se establece como **objetivo general:**

Desarrollar el Componente Flujo de Trabajo v2.0 para el Sistema de Planificación de Actividades SIPAC, para una mejor ejecución del proceso Aprobación – Conciliación de los diferentes documentos de la planificación.

Por consiguiente esta investigación se encuentra enmarcada en el **campo de acción:**

Proceso Aprobación – Conciliación de los documentos en el Sistema de Planificación de Actividades SIPAC.

Para dar cumplimiento al objetivo general se definieron las siguientes **tareas de la investigación**:

1. Caracterización del proceso Aprobación – Conciliación en el Sistema de Planificación de Actividades SIPAC.

2. Análisis de la Arquitectura del SIPAC para conocer las características fundamentales, marcos de trabajo, herramientas y tecnologías definidas para el desarrollo, así como la estrategia de integración entre los diferentes componentes.
3. Estudio de la Metodología de desarrollo para la UCI
4. Estudio de los patrones de Diseño.
5. Actualización del Modelo conceptual y el Glosario de términos asociados a la solución Flujo de Trabajo v1.0.
6. Actualización de los requisitos funcionales del software.
7. Actualización del Modelo de Datos del SIPAC.
8. Actualización del diseño de las clases asociado a la solución.
9. Validación del diseño aplicando las métricas adecuadas para esta etapa.
10. Implementación de los requisitos de software.
11. Validación mediante la aplicación de pruebas de caja blanca y pruebas de caja negra.

Métodos investigativos utilizados:

Métodos Empíricos:

- ✓ **Observación:** Se centraliza la información en la situación actual del Sistema de Planificación de Actividades SIPAC, y en el funcionamiento del flujo de trabajo

Métodos Teóricos:

- ✓ **Histórico-lógico:** Se realiza un estudio sobre la ejecución del proceso de Aprobación-Conciliación en la anterior versión de la herramienta SIPAC y otras herramientas. (1)
- ✓ **Modelación:** Se utiliza para realizar una abstracción y dar una explicación detallada de la estructura en términos de diseño de la solución propuesta.(1)
- ✓ **Analítico-sintético:** Se realiza una interpretación de la bibliografía recopilada con el objetivo de caracterizar todos los procesos asociados a la solución propuesta.(1)

Estructura del documento:

Capítulo 1: Fundamentación teórica.

En este capítulo se aborda el estado del arte referente a la ejecución del Proceso Aprobación-Conciliación en Cuba, basado en el estudio de sistemas informáticos empleados para la planificación. Se identifican y

caracterizan las herramientas y tecnologías utilizadas para el diseño e implementación de la solución y se describe la metodología a utilizar.

Capítulo 2: Análisis y Diseño de la solución.

En este capítulo se describen las características esenciales que debe tener la solución propuesta, dando cumplimiento a las fases de modelación y descripción de requisitos definidas en la metodología de desarrollo aplicada. Se obtiene el diseño del paquete de funcionalidades y la arquitectura que soportará el mismo.

Capítulo 3: Implementación y validación.

Este capítulo comprende todos los aspectos referentes a la implementación del Componente Flujo de Trabajo v 2.0 para SIPAC. Además se aplican una serie de pruebas a las funcionalidades implementadas con el fin de garantizar la calidad y validación de la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo se aborda el estudio realizado sobre los conceptos y procesos asociados al dominio del problema. Se estudian soluciones informáticas para la planificación buscando conocer el tratamiento que realizan al proceso aprobación-conciliación de los documentos. El autor ofrece una visión de las tecnologías y las herramientas a utilizar durante el desarrollo de la investigación y la descripción de la metodología a utilizar.

1.2 Marco Conceptual.

1.2.1 Plan.

Es un modelo matemático que se elabora antes de realizar una acción, con el propósito de dirigirla y encausarla. En este sentido un plan también es un documento que precisa los detalles necesarios para realizar una misión. (2)

1.2.2 Objetivos.

Propósito o meta que se plantea a cumplir en un tiempo definido e indicando la finalidad hacia la cual deben dirigirse los recursos y esfuerzos para darle cumplimiento a dichos fines. (2)

1.2.3 Actividades.

Conjunto de operaciones o tareas, propias de una persona o entidad, destinadas para cumplir determinados objetivos. (2)

1.2.4 Instrucción No. 1 del Presidente de los Consejos de Estado y de Ministros.

En este documento se plantean determinados designios que deben influir directamente sobre todos los procesos de planificación o herramientas informáticas que se utilicen en Cuba con esta finalidad teniendo en cuenta que una estricta adherencia a sus doctrinas arrojará como resultado una planificación óptima, funcional y eficiente.

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

Entre sus propósitos se encuentra lograr una adecuada organización y coordinación de todos los factores implicados, que intervienen en el proceso en interés del cumplimiento de los objetivos de trabajo. También pretende garantizar la eficiencia y eficacia en la comprobación de la ejecución de acciones mediante las diferentes formas de control. Propone una participación directa y activa de los jefes, pues son solo ellos quienes aprueban los documentos de sus subordinados.

Una característica fundamental de todos los procesos referentes a la planificación a raíz de este documento es la flexibilidad pues debe permitir cambiar (introducir, quitar, modificar o trasladar) objetivos de trabajo o actividades, producto de cambios en los escenarios o nuevas necesidades. En la Instrucción se hace hincapié en la importancia de llevar un registro y control sobre la marcha del cumplimiento de los planes y recalca que esta aplica en todos los niveles de dirección y estructuras subordinadas. (3)

1.3 Estudio del Estado del Arte.

El estudio del estado del arte se enfoca en obtener buenas prácticas que puedan aplicarse a la solución, basados en la ejecución del proceso de aprobación de varias aplicaciones; además se profundiza en el funcionamiento del Componente Flujo de Trabajo v 1.0 en el Sistema de Planificación de Actividades SIPAC para definir sus deficiencias.

1.3.1 P-Trab.

El P-Trab es un sistema informático cubano que informatiza el Proceso de Planificación por Actividades desde finales de la década del 90.

Observaciones: Es un sistema antiguo que no satisface el nuevo modelo cubano de planificación, pues no tienen en cuenta el manejo de planes ni de objetivos. La bibliografía no dice nada al respecto del Proceso de Aprobación-Conciliación y además en la ejecución de sus procesos no cumple con las indicaciones de la Instrucción No. 1 del Presidente de los Consejos de Estado y de Ministros.

1.3.2 Microsoft Office.

Del Paquete de Office, se utilizan muchas herramientas en las empresas cubanas para gestionar la planificación. Diversas entidades encuentran auxilio en herramientas como Outlook, Project y Excel.

Observaciones: El trabajo con el Paquete de Office no potencia los principios de independencia tecnológica de Cuba, por otra parte el proceso de Aprobación-Conciliación de los documentos se realiza a través del correo electrónico, no se encuentra definido un flujo de trabajo lógico y además es privativo.

1.3.3 Windows Workflow Foundation.

Windows Workflow Foundation es el modelo de programación, motor y herramientas para generar con rapidez las aplicaciones habilitadas por flujo de trabajo en Windows. Windows Workflow Foundation es un marco que permite a los usuarios crear flujos de trabajo de sistema o humanos en sus aplicaciones para Windows Vista, Windows XP y la familia Windows Server 2003. Se puede utilizar para resolver los escenarios simples como mostrar los UI de controles basados en datos proporcionados por el usuario o los escenarios complejos que se producen en las empresas grandes, como procesamiento del orden y control de inventario.

En este ejemplo se muestra el uso de varias características de Windows Workflow Foundation (WF) y Windows Communication Foundation (WCF) al mismo tiempo. Juntas implementan un escenario de proceso de aprobación de un documento. Una aplicación cliente puede enviar documentos para su aprobación y aprobar documentos. Existe una aplicación de administrador de aprobaciones para facilitar las comunicaciones entre los clientes y aplicar las reglas del proceso de aprobación. El proceso de aprobación es un flujo de trabajo que puede ejecutar varios tipos de aprobación. Existen actividades para obtener una aprobación única, una aprobación de quórum (un porcentaje de un conjunto de aprobadores) y un proceso de aprobación compleja que consta de una aprobación de quórum y una aprobación única en una secuencia.

Desde la perspectiva del cliente, el proceso de aprobación funciona del siguiente modo:

1. Un cliente de WCF la envía a un servicio de WCF hospedado por la aplicación de administrador de aprobaciones.
2. Se devuelve un Id. de usuario único al cliente. El cliente puede participar ahora en los procesos de aprobación.

3. Una vez admitido, un cliente puede enviar un documento para su aprobación mediante un proceso de aprobación única.
4. Al hacer clic en un botón de la interfaz del cliente, se inicia una instancia de flujo de trabajo en un host de servicio de flujo de trabajo del cliente.
5. El flujo de trabajo envía una solicitud de aprobación a la aplicación de administrador de aprobaciones.
6. El administrador del flujo de trabajo inicia un flujo de trabajo por su parte para representar un proceso de aprobación.
7. Una vez ejecutado el flujo de trabajo de aprobación del administrador, los resultados se devuelven al cliente.
8. El cliente muestra los resultados.
9. Un cliente puede recibir una solicitud de aprobación y responder a la solicitud en cualquier momento.
10. Un servicio de WCF hospedado en el cliente puede recibir una solicitud de aprobación procedente de la aplicación de administrador de aprobaciones.
11. La información del documento se presenta en el cliente para revisión. El usuario puede aprobar o rechazar el documento.

Desde el punto de vista de la aplicación de administrador de aprobaciones, el proceso de aprobación funciona del siguiente modo:

1. Un servicio de WCF en el administrador de aprobaciones recibe una solicitud para formar parte del sistema del proceso de aprobación.
2. Se genera un identificador único para el cliente. La información sobre el usuario se almacena en una base de datos.
3. Se recibe una solicitud de aprobación. El administrador de aprobaciones ejecuta un proceso de aprobación.
4. El administrador de aprobaciones recibe una solicitud de aprobación, lo que inicia un nuevo flujo de trabajo.
5. Las actividades Send y Receive con correlación se utilizan para enviar la solicitud de aprobación al cliente para su revisión y recibir la respuesta. El resultado del flujo de trabajo del proceso de aprobación se envía al cliente.

Observaciones: Las prácticas implementadas en este flujo de trabajo son excelentes y se aplican mucho a las doctrinas de Cuba. En el caso del proceso de aprobación en Cuba solo se realiza una aprobación única y en una sola dirección; teniendo en cuenta las doctrinas de la indicación No 1 del Presidente de los Consejos de Estados y de Ministros, por lo demás el proceso de Windows Workflow Foundation puede ser fundamental en la solución que se desea implementar.

1.3.4 Componente Flujo de Trabajo en SIPAC v1.0.

El Sistema de Planificación de Actividades (SIPAC) forma parte del paquete de soluciones integrales de gestión CEDRUX para las entidades presupuestadas y empresariales, el cual está basado en los principios de independencia tecnológica y con funcionalidades generales de los procesos y las particularidades de la economía cubana. El sistema pone al servicio de su entidad las potencialidades de la tecnología informática y provee facilidades para la integración de las diferentes áreas productivas y departamentos administrativos.

El Módulo Flujo de Trabajo en SIPAC establece cómo, quienes y en qué momento la información de la planificación puede ser accedida por los usuarios. En su primera versión, el sistema, no posibilita el establecimiento organizado de los estados y transiciones de los documentos que regulen el Proceso Aprobación-Conciliación en cada entidad lo que dificulta el seguimiento y control de la Planeación Estratégica y Operativa tanto para la entidad como a nivel de gobierno.

No existe un control de acceso a la información en cada nivel, las tareas y documentos pueden ser vistos y manipulados por cualquier usuario sin las debidas restricciones por jerarquía, permitiendo así que se duplique y/o pierda la información; al mismo tiempo consiente que sin importar el nivel del usuario pueda crear estados que determinan el comportamiento de los documentos de forma arbitraria por lo que la información puede pasar por cuantos estados estime el usuario, creando ambigüedad y haciendo muy engorroso el proceso.

Debido a este segundo problema como parte de una cadena se desarrolla otro inconveniente, pues el mismo conlleva a que la información no recorra un camino lógico de estados, no queda claro al usuario final qué estado sucede al otro lo cual debe hacerse mediante precondiciones y/o post-condiciones que determinen la Aprobación-Conciliación de los documentos, siendo esta otra dificultad que presenta el sistema y haciendo más difícil el trabajo en el Componente. (4)

Los inconvenientes antes planteados hacen que la primera versión del Componente Flujo de Trabajo del Sistema de Planificación SIPAC no cumpla con las indicaciones para una Planeación Estratégica y Operativa que propone la Instrucción No. 1 del Presidente de los Consejos de Estados y de Ministros.

1.4 Valoración del Estado del Arte.

Tras analizar varias aplicaciones y el desempeño del proceso de Aprobación-Conciliación en ellas, se arriba a la conclusión de que el proceso no se realiza de un modo que aplique a las indicaciones del país en ninguna de las soluciones informáticas estudiadas. La versión 1.0 del Módulo Flujo de Trabajo de SIPAC, tampoco cumple con las premisas, el proceso presenta ambigüedades y su ejecución es engorrosa para el usuario. Las prácticas que implementa el Windows Workflow Foundation aplican en un alto grado a lo que podría ser una propuesta de solución.

Estos elementos evidencian la imperiosa necesidad de elaborar una solución que responda acertadamente a la ejecución actual del proceso de Aprobación-Conciliación de los documentos de la planificación en las entidades cubanas.

Una segunda versión del Módulo Flujo de Trabajo de SIPAC donde se mejoren estas contrariedades es la solución ideal y además no se pone en contradicción con el modelo de planificación cubano.

1.5 Lenguajes, tecnologías y herramientas propuestos para el desarrollo de la solución.

En todo desarrollo de software, uno de los factores de mayor importancia son los lenguajes de modelado y de desarrollo que se utilizan para dar solución a un problema determinado. Dichos lenguajes permiten especificar la estructura o comportamiento que tendrá el sistema, además de propiciar una guía para la construcción del mismo. Los lenguajes, tecnologías y herramientas que a continuación se presentan han sido seleccionados y aprobados por el centro CEIGE para el desarrollo de la solución.

1.5.1 Lenguajes de modelado.

Los lenguajes de modelados son aquellos que a través de símbolos estandarizados permiten diseñar organizadamente el desarrollo de un software. Como lo planteara José Enrique González Cornejo, Gerente General de DocIRS en su ensayo *¿Qué es UML? (*Enero 2008): “El lenguaje de modelado es la

notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño". El lenguaje de modelado que se propone para la solución es el siguiente(5):

UML v2.1

El Lenguaje de Modelado Unificado o UML (Unified Modeling Language) fue concebido para modelar los elementos de un sistema de software de una manera estandarizada que incluya conceptos del proceso de negocio y funciones del sistema. Este modelado será de fácil comprensión para cualquier desarrollador con conocimientos sobre UML, y podrá ser utilizado en cualquier tipo de desarrollo.(6)

BPMN

BPMN (Business Process Modeling Notation) es empleado en el desarrollo de la solución que se propone para presentar gráficamente las diferentes etapas del proceso de Aprobación-Conciliación de los documentos de la planificación. Este estándar de modelado de procesos de negocio ha sido diseñado específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre los diferentes procesos participantes.

1.5.2 Lenguajes de Desarrollo del lado del cliente.

HTML

El Lenguaje de Marcas de Hipertexto o HTML (Hyper Text Markup Language) es un lenguaje sencillo escrito mediante etiquetas (Tags) que permiten definir y ubicar los distintos elementos que componen una página web. Se integra con lenguajes como JavaScript y PHP y no todos los navegadores interpretan este lenguaje de la misma forma, lo cual puede ser visto como una de sus desventajas.(7)

CSS

Las Hojas de Estilo en Cascada o CSS (Cascading Style Sheets) constituye un estándar que permite establecer el estilo (dígase tamaños, iconos, imágenes, tipografías, colores, espacios y bordes) de documentos estructurados, dígase la capa de presentación de una página web. Este lenguaje permite

definir el aspecto visual del documento, mientras que separa la parte semántica (HTML) de la presentacional (style sheets). (8)

JavaScript

JavaScript es un lenguaje que no requiere compilación ya que es interpretado por todos los navegadores y es utilizado para la creación de páginas web dinámicas. Dicho lenguaje permite crear diferentes efectos e interactuar con los usuarios. Además interactúa perfectamente con códigos basados en HTML, lo cual constituye una de sus principales ventajas.(9)

1.5.3 Lenguajes de Desarrollo del lado del servidor.

PHP v5.4

PHP 5.4 es el lenguaje que se empleará para programar del lado del servidor. Es interpretado y completamente orientado al desarrollo de aplicaciones web dinámicas. Es gratuito, fácil de usar y aprender, portable, de código abierto y multiplataforma. Presenta interfaces para una gran cantidad de sistemas de base de datos diferentes, así como bibliotecas incorporadas para muchas tareas web habituales. Sin embargo, este lenguaje debido a su flexibilidad, puede convertir al sitio en un punto de fácil acceso a piratas informáticos.(10)

1.5.4 Tecnologías y herramientas de desarrollo.

Marcos de Trabajo

Un marco de trabajo o framework es una estructura de soportes de programas, librerías y lenguajes de scripting. Es considerado una arquitectura de software que modela las relaciones generales de los componentes del proyecto que lo implementa; provee una estructura y manera de trabajo la cual utilizan las aplicaciones del proyecto. La finalidad de los frameworks es facilitar el desarrollo de software, permitiéndoles a diseñadores y programadores concentrarse en los requerimientos del proyecto, reduciendo los posibles problemas con las tecnologías utilizadas, así como facilitando ciertas funcionalidades básicas y comunes.(11)

Sauxe v1.0

Es un marco de trabajo, fusionado bajo tecnologías totalmente libres (entre ellas PHP, Postgresql, Apache) que posee el desarrollo de tecnologías propias basadas en otros marcos de trabajo como ZendFramework para el manejo de la lógica de negocio, Doctrine para el acceso a datos y ExtJS para la capa de presentación. Cuenta con una arquitectura en capas que a su vez presenta en su capa superior un MVC (patrón arquitectónico Modelo Vista Controlador). Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.

ZendFramework v1.5

ZendFramework2 o ZF2 es un marco de trabajo para el manejo de la lógica de negocio. Es una implementación que uso código 100% orientado a objetos. Es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP5. Brinda facilidades de uso y poderosas funcionalidades, posee buenas capacidades de ampliación y proporciona un sistema de caché de forma que se puedan almacenar diferentes datos, así como los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador. Consta de mecanismos de filtrado y validación de entradas de datos. Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX y provee capacidades de búsqueda sobre documentos y contenidos. (12)

Doctrine v1.0

Doctrine es empleado para la capa de acceso a datos. Es un sistema ORM (en inglés Object Relational Mapper) para PHP 5.2 o superior que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL, manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Además, exporta una base de datos existente a sus clases correspondientes y convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.(13)

Ext v2.2

ExtJs es el marco de trabajo empleado para el desarrollo de la capa de presentación. Está basado completamente en la programación orientada a objeto. Cada objeto contiene lo típico: propiedades,

métodos y eventos. Basa toda su funcionalidad en JavaScript a través de librerías. Así, en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso de Document Object Model (DOM). Los datos son obtenidos con AJAX. Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos. Además permite que exista un balance entre el Cliente – Servidor, posibilitando que la carga de procesamiento se distribuye, permitiendo que el servidor al tener menor carga, pueda manejar más clientes al mismo tiempo.(14)

1.5.5 Técnicas 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 usa para poder hacer consultas asíncronas al servidor sin necesidad de recargar la página. Esta surge de la combinación de tres tecnologías ya existentes: HTML (o XHTML y Hojas de Estilo en Cascada (CSS) para presentar la información, DOM y JavaScript, para interactuar dinámicamente con los datos, además de XML y XSLT, para intercambiar y manipular datos de manera de sincronizada con un servidor web.(15)

CASE: Visual Paradigm v8.0

Se emplea Visual Paradigm for UML v8.0 como herramienta CASE. Utiliza UML v2.1 como lenguaje de modelado, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Ayudando a construir aplicaciones de calidad más rápido, mejor y a bajo costo.(16)

Sistema de control de versiones: Subversión

Subversión (SVN) 1.6 es la herramienta de entorno colaborativo que se utiliza para el control de versiones. Se encuentra preparado para funcionar en red y se distribuye bajo licencia libre. Mantiene versiones no sólo de archivos, sino también de directorios y versiones de los metadatos asociados a esos directorios. Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las

operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre. Brinda facilidades de soporte tanto de ficheros de texto como binarios. (17)

Entorno integrado de desarrollo (IDE): NetBeans

La presente solución se desarrolla sobre el IDE de programación multiplataforma NetBeans 7.1 El cual tiene soporte para la versión 5.4 de PHP, JavaScript, el diseño de Hojas de Estilo (CSS) y HTML. Se integra con varias herramientas como el Subversión y servidores web. Es un producto de código abierto, con todos los beneficios del programa disponible en forma gratuita. Hace uso de plugins para ampliar sus funcionalidades, lo que le da una gran facilidad de uso.(18)

Servidor web: Apache

Se utiliza 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 tener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Tiene una alta configuración en la creación y gestión de registros de actividad. Apache permite la creación de ficheros de registro a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. (19)

Sistema gestor de bases de datos: PostgreSQL

Como Sistema Gestor de Base de Datos (SGBD) se emplea la versión 9.1 de PostgreSQL. Constituye un sistema de gestión de bases de datos relacional. Es una herramienta de código abierto, de bajo coste y multiplataforma. Se destaca en ejecutar consultas complejas, subconsultas y uniones de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Soporta transacciones, claves ajenas con comprobaciones de integridad referencial y almacenamiento de objetos de gran tamaño. Cuenta con varias herramientas gráficas de diseño y administración de bases de datos como el pgAdmin. (20)

1.6 Metodología de Desarrollo para la Actividad Productiva de la UCI

Como metodología para la realización de la solución se utiliza la Metodología de Desarrollo para la Actividad Productiva de la UCI. Esta propuesta responde a una variación que se le realiza al Proceso Unificado Ágil (AUP) por sus siglas en inglés.

1.6.1 AUP

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- ✓ Desarrollo Dirigido por Pruebas (test driven development - TDD en inglés)
- ✓ Modelado ágil
- ✓ Gestión de Cambios ágil
- ✓ Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva. (21)

1.6.1.1 Fases de AUP

1. **Inicio:** El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. **Elaboración:** El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. **Construcción:** Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. **Transición:** El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

1.6.1.2 Disciplinas AUP

AUP define 7 disciplinas (4 ingenieriles y 3 de gestión de proyectos), las disciplinas son:

1. **Modelo.** El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se abordan en el proyecto, y determinar una solución viable para resolver el problema de dominio. Agrupa los flujos de trabajos de Modelado de negocio, Requisitos y Análisis y Diseño.
2. **Implementación.** El objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.
3. **Prueba.** El objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos.
4. **Despliegue.** El objetivo de esta disciplina es la prestación y ejecución del sistema y que el mismo este a disposición de los usuarios finales.
5. **Gestión de configuración.** El objetivo de esta disciplina es la gestión de acceso a herramientas de su proyecto. Esto incluye no sólo el seguimiento de las versiones con el tiempo, sino también el control y gestión del cambio para ellos.
6. **Gestión de proyectos.** El objetivo de esta disciplina es dirigir las actividades que se lleva a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc.), coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que es entregado a tiempo y dentro del presupuesto.
7. **Entorno.** El objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario.

1.6.1.3 Variación de AUP para la UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI[1]-DEV v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Se redujo a 1 la cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11. (21)

1.7 Conclusiones del capítulo

Teniendo en cuenta el estudio del arte realizado sobre la ejecución del Proceso de Aprobación-Conciliación, se establece la siguiente conclusión:

- Los documentos para la planificación deben pasar por una serie de estados previamente definidos antes de su aprobación y solo ser vistos y manipulados por el personal autorizado para esas actividades. Este es un proceso complicado y delicado que requiere de buenas prácticas al momento de su desarrollo y en la herramienta SIPAC 1.0 no se permite su correcto tratamiento.

- El estudio de lenguajes y tecnologías de desarrollo aportó un mayor conocimiento que facilitará la posterior utilización de los mismos. Toda la selección se soporta en la anterior versión del sistema con la finalidad de obtener una buena integración de esta nueva versión. Finalmente la compresión de la metodología a utilizar hará mucho más viable la realización de la solución

[1] Capability Maturity Model Integration.

CAPÍTULO 2: PROPUESTA DE LA SOLUCION.

2.1 Introducción.

En el presente capítulo describen las características de la solución propuesta. De esta forma se detalla una propuesta de solución para la evaluación del cumplimiento de los objetivos, se definen los principales conceptos asociados al negocio, se describen y especifican los requisitos funcionales, así como los no funcionales. Se realiza una descripción de la solución en términos de componentes y se definen los patrones de diseño utilizados.

2.2 Propuesta de solución.

Partiendo del análisis realizado en el capítulo 1 sobre las deficiencias que presenta el Componente Flujo de Trabajo en la primera versión de la aplicación se propone modificar este módulo para posibilitar el seguimiento y control de la Planeación Estratégica y Operativa tanto para la entidad como a nivel de gobierno de la siguiente forma:

La herramienta deber permitir gestionar los estados de una manera uniforme, por consiguiente debe admitir la gestión de las transiciones de la información en las cuales se empleen como estados iniciales o finales los ya previamente definidos. Se asociarán a los estados denominados, usuarios y tipos de documentos de la planificación (planes, objetivos, actividades). Esta nueva versión permitirá definir los usuarios que pueden ver la información en cada momento, además desde el componente se van a poder asociar cuando un estado genera una puntualización o cuando genera una notificación.

2.3 Modelación del negocio.

Los artefactos correspondientes a esta disciplina se encuentran en el expediente del proyecto SIPAC en la dirección SIPAC 2.0\Expediente 3.4\Ingeniería\Modelo de Negocio.

2.3.1 Diagrama de procesos de negocio.

En el presente acápite se modela el proceso de negocio Aprobar-Conciliar documentos de la planificación. Para la representación del mismo se hizo uso de la notación *Business Process Modeling Notation* o BPMN (en español, Notación para el Modelado de Procesos de Negocio), la cual permitió representar las etapas

del proceso de la solución propuesta. Se relacionan así roles, eventos y artefactos que intervienen en su desarrollo.

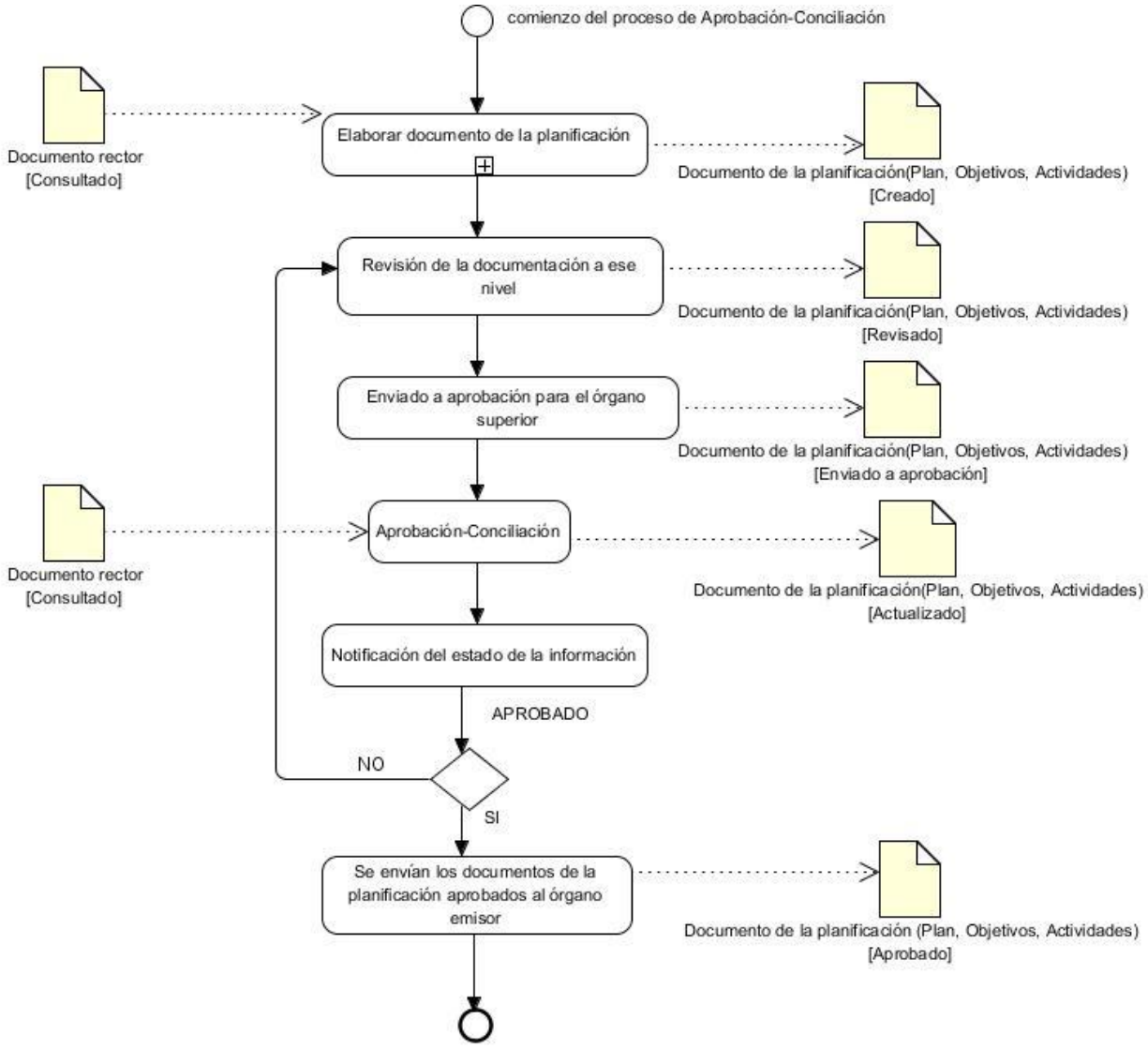


Figura 1. Proceso de negocio: Aprobar-Conciliar documentos de la planificación.

2.3.2 Especificación del proceso de negocio: Aprobar-conciliar los documentos de la planificación.

Tabla 1. Sistema de aprobación-conciliación de los documentos de la planificación.

Objetivo	Aprobar un documento

Evento(s) que lo genera(n)	Se emite un documento a aprobar (Objetivos, actividades, plan) y el documento rector
Pre condiciones	N/A
Marco Legal	N/A
Clientes internos	Puntualizar plan. Establecer sistema de control.
Clientes externos	N/A
Entradas	N/A

Flujo de eventos

Flujo Básico

1. Se elabora sobre la base del documento rector el documento de la planificación que se desea aprobar puede ser Plan, Objetivos o Actividades
2. Se revisa el documento a ese nivel.
3. Se envía el documento a aprobación al nivel superior.
4. . Los funcionarios nivel superior leen y corrigen de ser necesario el documento recibido. Se estudian además el documento rector porque sobre la base de este es que se han formulado los documentos que se desean aprobar.
5. Se notifica al emisor de la información el estado de los documentos aprobado o rechazado. Si se rechazan ver el *flujo alternativo 5.a.*
6. Se envían los documentos aprobados al organismo que lo originó. El órgano de planificación envía los documentos los órganos de dirección que los generaron.
7. Concluye el proceso

Pos-condiciones

1. Se ha aprobado el documento

Salidas

1. Documento rector. (Documento Word)
2. Objetivos. (Documento Word)
3. Actividades. (Documento Word)
4. Plan. (Documento Word)

Flujos Paralelos

N/A

Pos-condiciones

N/A

Salidas

N/A

Flujos alternativos

Flujo alternativo 5.a Se rechaza el documento de la planificación a aprobar

1. Se envía el documento al órgano de dirección que lo originó. El documento rechazado se envía al órgano de dirección que lo originó con las recomendaciones contenidas en el dictamen, para que pueda ser arreglado y ser sometido nuevamente al proceso de aprobación.
2. Concluye el proceso.

Pos-condiciones

1. Se ha rechazado el documento y devuelto al órgano de dirección que lo originó.

Salidas
1. Documento rector. (Documento Word)
2. Objetivos. (Documento Word)
3. Actividades. (Documento Word)
4. Plan. (Documento Word)
Asuntos pendientes
N/A

2.3.3 Validación del proceso de negocio.

El proceso de negocio Aprobar-Conciliar documentos de la planificación se encuentra validado a través de los criterios definidos en el documento CIG-SPA-N-Validación de procesos de negocio-SPA del expediente de proyecto de SIPAC 2.0 del centro CEIGE. Estos criterios son:

- El proceso debe tener el proveedor incluido en el listado de proveedores válidos.
- El proceso debe tener un identificador único.
- El proceso está completo si, y solo si, se cumplen los siguientes elementos: si se han descrito todos los flujos básicos, alternativos y paralelos del proceso; si se han completado todas las secciones de la planificación; si todas las figuras, tablas y diagramas están etiquetados; si todos los acrónimos, abreviaturas, siglas, términos y unidades de medidas se han definido.
- El proceso debe ser consistente (seguro) si: se ha identificado su proceso padre, si procede; si se han identificado los subprocesos, si procede; si se han representado todas su relaciones con otros procesos.
- El resultado del proceso debe ser evaluado de positivo.

2.3.4 Modelo conceptual.

El modelo conceptual es una representación global de los principales elementos lógicos de la realidad que se analiza. Comprende y describe las clases más importantes dentro del contexto del sistema.

En el capítulo anterior se encuentran explicados los conceptos de **plan**, **objetivos** y **actividad** por lo que no se repiten a continuación dando paso a otros conceptos que forman parte del Modelo Conceptual que se presenta.

Elemento de la planificación: Determina cuáles serán los componentes que se deberán tomar en consideración dentro de la Planificación estratégica y operativa. Se refiere a elementos que son considerados claves en la planificación dígase planes, objetivos y actividades.

Estados: En SIPAC se llamará estado a cada fase o etapa que deba recorrer cualquier documento de la planificación. Los mismos pueden adoptar la denominación que el usuario desee y deben estar asociados a planes, actividades u objetivos en dependencia de las necesidades del usuario o del negocio. Cada documento de la planificación debe tener un estado inicial y un estado final.

Transiciones: Consiste en establecer qué estado sucede a otro en el flujo de aprobación de los documentos y qué precondición y/o post-condición se deben cumplir en cada caso; dicho de otro modo, las transiciones de estado especifican la consecución de los estados por los que pueden pasar los documentos, o lo que es lo mismo el orden que tendrán estos estados.

La siguiente **figura** representa el modelo conceptual asociado al proceso de aprobación-conciliación de los documentos de la planificación:

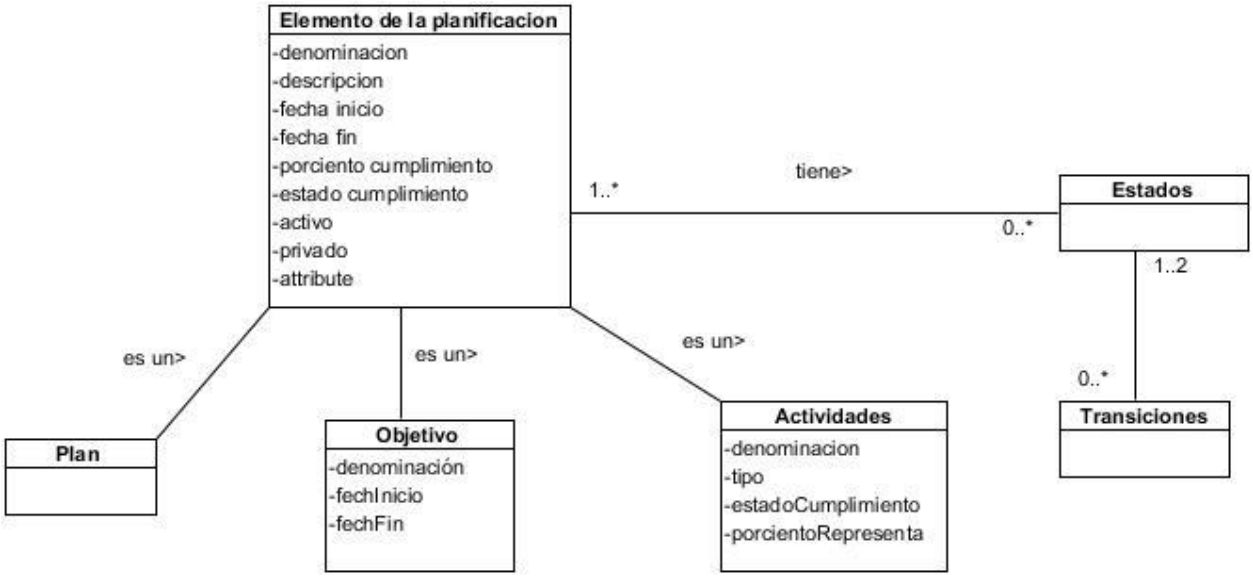


Figura 2. Modelo conceptual del Proceso Aprobación-Conciliación.

2.4 Requisitos.

Un requisito de software es, según la IEEE una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Se puede ver también como una condición o capacidad que

tiene que ser alcanzada o poseída por un sistema para satisfacer formalmente un contrato definido entre las partes interesadas. (22)

En este acápite se abordarán los elementos referentes a la descripción de los requerimientos que debe cumplir la solución implementada para lograr la calidad de esta y la satisfacción del cliente.

2.4.1 Técnicas de captura de requisitos.

Siguiendo el procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del centro CEIGE y teniendo en cuenta las características del cliente, se decidió utilizar como técnicas que permitirán a los analistas la recopilación y obtención de la información necesaria para la licitación de requisitos la entrevista, la observación, la tormenta de ideas y los talleres.

Se combinaron las siguientes técnicas: entrevista, observación y tormenta de ideas. Se entrevistaron a varios especialistas y funcionarios de los diferentes ministerios donde ya se había implementado SIPAC, en visitas realizadas a las instalaciones, ya que las entrevistas permiten un intercambio más abierto con los especialistas, mediante preguntas que esclarecen con precisión el funcionamiento de todo el trabajo. La observación se llevó a cabo para percibir cómo se desarrolla el proceso de Aprobación-Conciliación de los documentos de la panificación por los diferentes niveles, pudiendo captar directamente las particularidades del mismo en las visitas realizadas; y la tormenta de ideas donde el equipo de desarrollo acumuló ideas para tener una perspectiva general de las necesidades del sistema.

2.4.2 Listado de requisitos funcionales.

Los 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 situaciones particulares.(23)

De esta forma quedan plasmados a continuación los requisitos funcionales identificados para el desarrollo del Componente Flujo de Trabajo v 2.0 para el Sistema para la Planificación de actividades SIPAC:

RF1 Gestionar estados.

1.1 Adicionar estados.

1.2 Modificar estados.

RF3 Cambiar el estado de aprobación de los documentos.

1.3 Eliminar estados.
1.4 Ver detalles.

RF2 Gestionar transiciones.
2.1 Adicionar transición.
2.2 Modificar transición.
2.3 Eliminar transición.
2.4 Listar transiciones.
2.5 Buscar transición.

RF4 Establecer estado inicial.
4.1 Establecer estado inicial para planes.
4.2 Establecer estado inicial para objetivos.
4.3 Establecer estado inicial para actividades.

RF5 Establecer estado generador de puntualizaciones.

RF6 Establecer estado generador de notificaciones.

2.4.3 Especificación de requisitos funcionales.

Mediante la especificación de requisitos se obtiene una descripción completa del comportamiento de las funcionalidades que se van a desarrollar. Para su redacción se utiliza un lenguaje sencillo, de forma que sea fácilmente comprensible para todas las partes involucradas en el desarrollo.

A continuación se muestra la especificación del requisito *Adicionar transición*, la especificación de los restantes requisitos funcionales se encuentran descritos en el expediente de proyecto SIPAC 2.0 en el artefacto CIG-SPA-N-i2605.

Tabla 2. Especificación de requisito: Adicionar transición.

Precondiciones	El usuario ha registrado estados de la información en el sistema.
Flujo de eventos	
Flujo básico	
1	Se seleccionan los datos de la transición: Transición válida para Estado inicial Estado final Usuarios asociados a la transición
2	El sistema valida (ver Validaciones) los datos seleccionados.
3	Si los datos son correctos el sistema los registra.
4	El sistema confirma el registro de la transición.
5	Concluye el requisito.
Pos-condiciones	
1	Se registró en el sistema una nueva transición de la información.

Flujos alternativos		
Flujo alternativo 3.a Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos de la transición.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CIG-ERP-N-DPO-i2201.	
Relaciones	**Requisitos Incluidos**	N/A.
	Extensiones	N/A.
Conceptos	**Transición de la información**	Visibles en la interfaz: Transición válida para Estado inicial Estado final Usuarios asociados Utilizados internamente: N/A.
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

2.4.4 Prototipo de Interfaz de Usuario para el RF Adicionar transición.

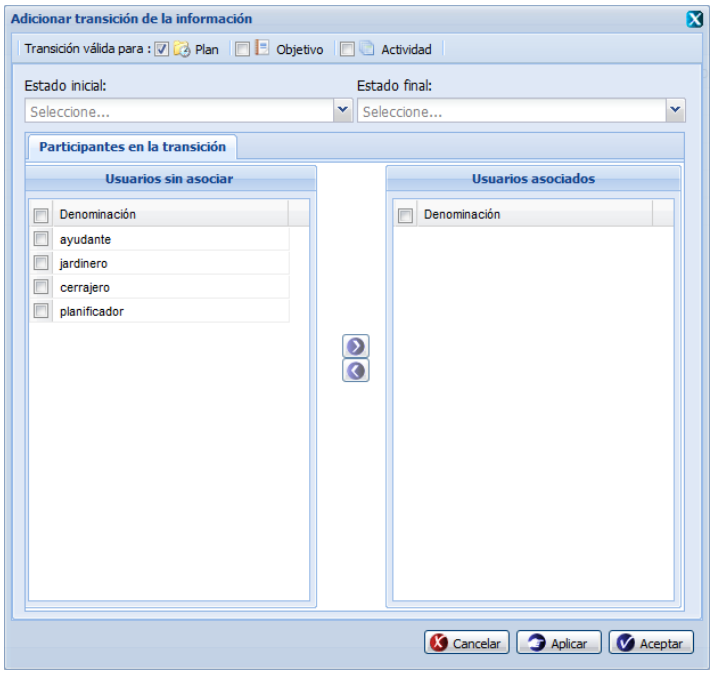


Figura 3. Prototipo elemental de interfaz gráfica de usuario: Adicionar transición.

2.4.5 Validación de los requisitos funcionales.

La validación permite ratificar los requisitos, es decir, verificar que todos los requisitos que aparecen en el documento especificado son los que realmente satisface la petición del cliente. De esta forma puede asegurarse que los requisitos validados representan una descripción, por lo menos, aceptable, del sistema que se debe implementar. Esto implica verificar que los requisitos sean consistentes y que estén completos**.** (24)

Para la validación de los requisitos funcionales identificados se utilizó la técnica: **Construcción de prototipos**. Se aplicaron además: Criterios para validar los requisitos del cliente y se realizó el Acta de aceptación de los requisitos.

2.4.6 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. 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 perteneciente al expediente del proyecto SIPAC 2.0.

Los requisitos no funcionales que impactan en la solución: Componente flujo de trabajo v2.0 para el sistema de Planificación de Actividades SIPAC se encuentran descritos en el expediente de proyecto SIPAC 2.0 en la siguiente dirección: 1. Ingeniería\ 1.1 requisitos\Descripciones de requisitos\04-Flujo de Trabajo.

2.5 Modelado de la solución.

2.5.1 Diagrama de clases del diseño web.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. En la figura que se muestra a continuación se presenta el diagrama de clases del diseño para Gestionar transiciones, una de las funcionalidades de la solución propuesta el cual está basado en estereotipos web y en el patrón arquitectónico MVC. Para obtener una imagen ampliada de las diferentes partes del patrón MVC que se utiliza en el diagrama de clases del diseño web, ver anexos 4, 5 y 6.

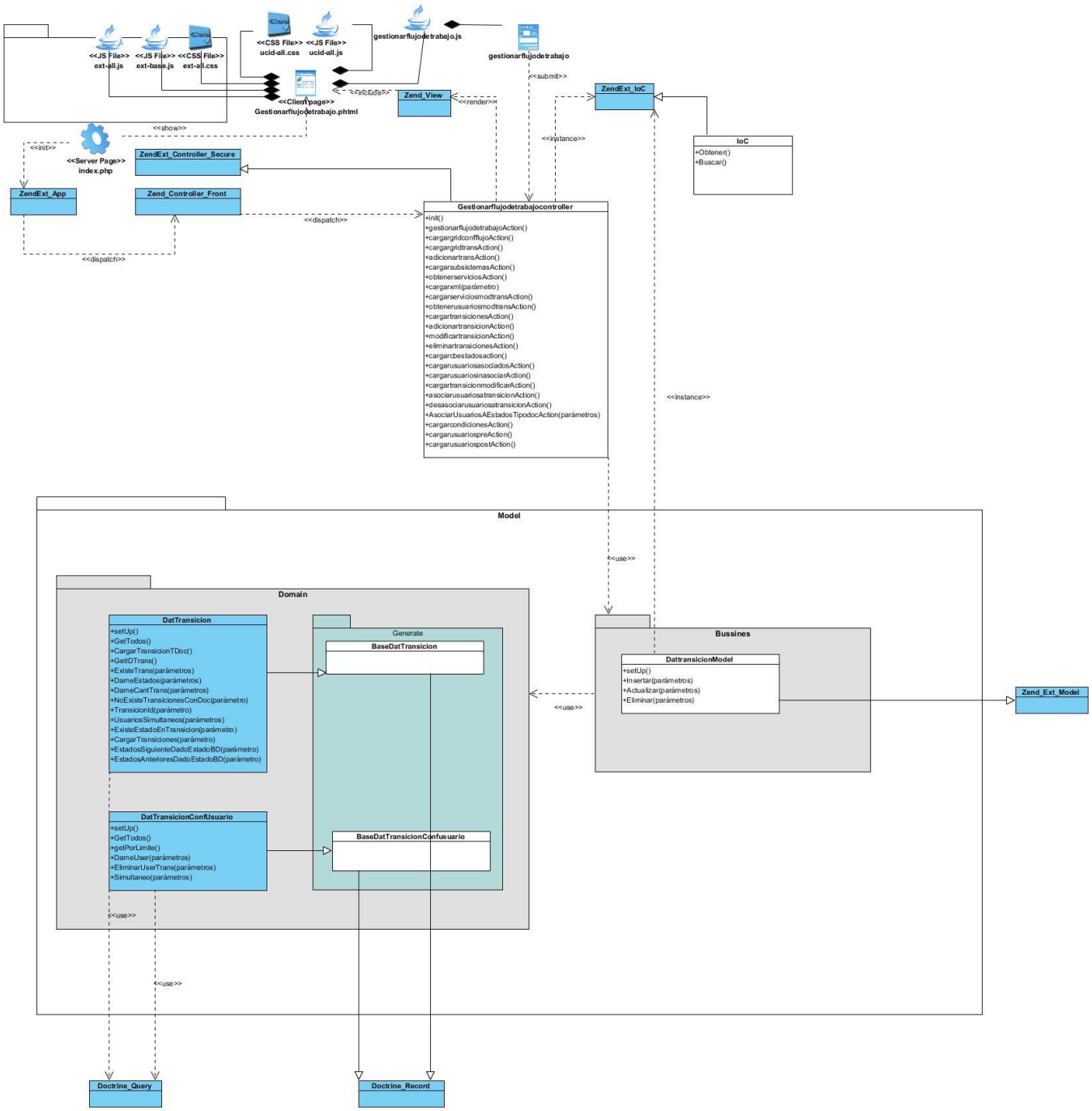


Figura 4. Modelo de clases del diseño de la funcionalidad Gestionar transiciones en el flujo de trabajo.

Tabla 3. Descripción de las principales clases del diagrama de clases del diseño.

Clase	Descripción
Gestionarflujodetrabajo.phtml	Página encargada de visualizar, a través de los js que debe incluir, la información necesaria para generar la transición de estados.
Gestionarflujodetrabajo.js	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
Gestionarflujodetrabajocontroller.php	Clase encargada de controlar la comunicación entre la Vista y el Modelo. Es la responsable de realizar las diferentes funcionalidades sobre los objetivos, según las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
ZendExt_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos.
Extjs	Contiene los componentes generados a través de la librería JavaScript Extjs.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.

2.5.2 Diagrama de secuencia.

Los diagramas de secuencia muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo. Se modelan para cada requisito funcional y contienen detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el mismo, además de los

mensajes intercambiados entre los objetos. A continuación se presenta el diagrama de secuencia correspondiente al requisito funcional Adicionar Transición, como parámetro de evaluación de un objetivo:

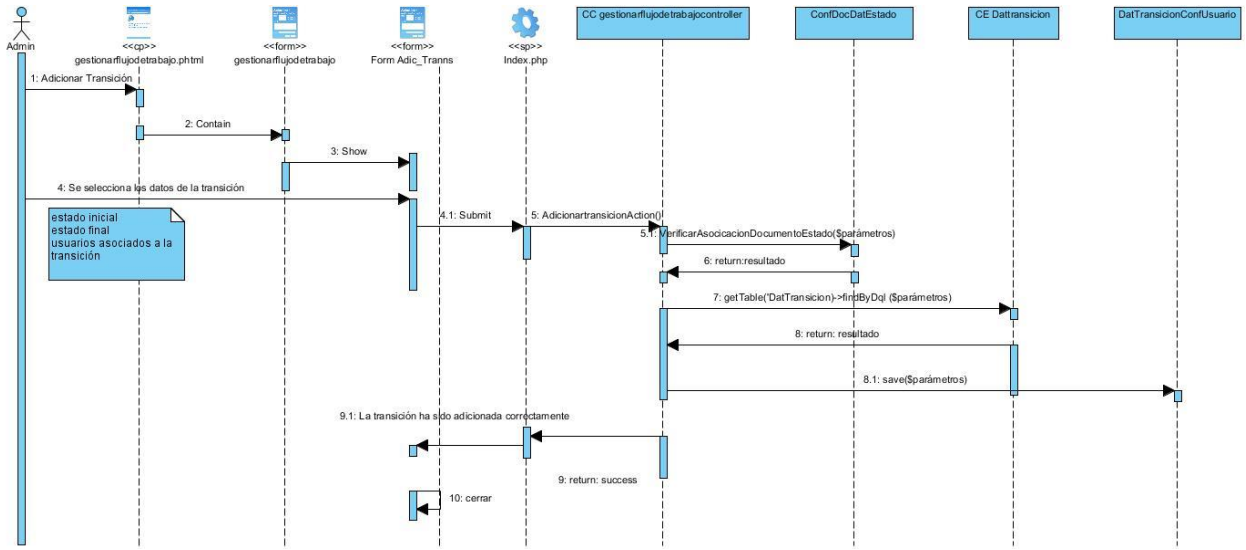


Figura 5. Diagrama de secuencia Adicionar transición.

Inicialmente el usuario accede a la interfaz principal Transiciones de la información donde se muestran listadas las diferentes transiciones y las opciones Adicionar, Modificar, Eliminar. A continuación selecciona la opción Adicionar. Paso seguido se muestra un formulario donde el usuario debe seleccionar los datos de la transición (tipo de documento, estado inicial, estado final, usuarios asociados) y presiona el botón Aceptar para enviar los datos de su selección. La clase controladora *GestionarFlujoTrabajoController* envía una petición a la clase *ConfDocDatEStado* para verificar que los estados estén asociados a los tipos de documentos, esta clase le devuelve un resultado positivo a la controladora que mediante el método getTable y findByDql de Doctrine pregunta a la clase *DatTransicion* si la transición que se desea hacer ya está hecha previamente. *DatTransicion* envía un resultado negativo a la clase controladora la cual manda a actualizar la tabla de transiciones con los datos de la transición adicionada en la clase *DatTransicionConfUsuario*. Finalmente el sistema muestra un mensaje donde ratifica que la transición fue adicionada correctamente.

2.5.3 Modelo de datos.

Un modelo de datos es una colección de conceptos que se emplean para describir la estructura de una base de datos. Está compuesto por entidades, atributos y relaciones. La mayoría de los modelos de datos poseen un conjunto de operaciones básicas para especificar consultas y actualizaciones de la base de datos.(25)

Inicialmente la persistencia de datos en el sistema asociada a la aprobación de los documentos estaba representado por la columna *estadoaprobacion* en la tabla *dat_elementos* en el esquema *mod_objetivos.* A continuación se presenta la actualización del modelo de datos que incluye la definición de un nuevo esquema, en este caso: *mod_flujotrabajo*, el cual resume los conceptos y define las relaciones, ajustándose a las necesidades de almacenamiento de datos del sistema para gestionar los estados y transiciones de los elementos de la planificación almacenados. Posee tercera forma normal y cuenta con 9 tablas que se muestran en la figura.

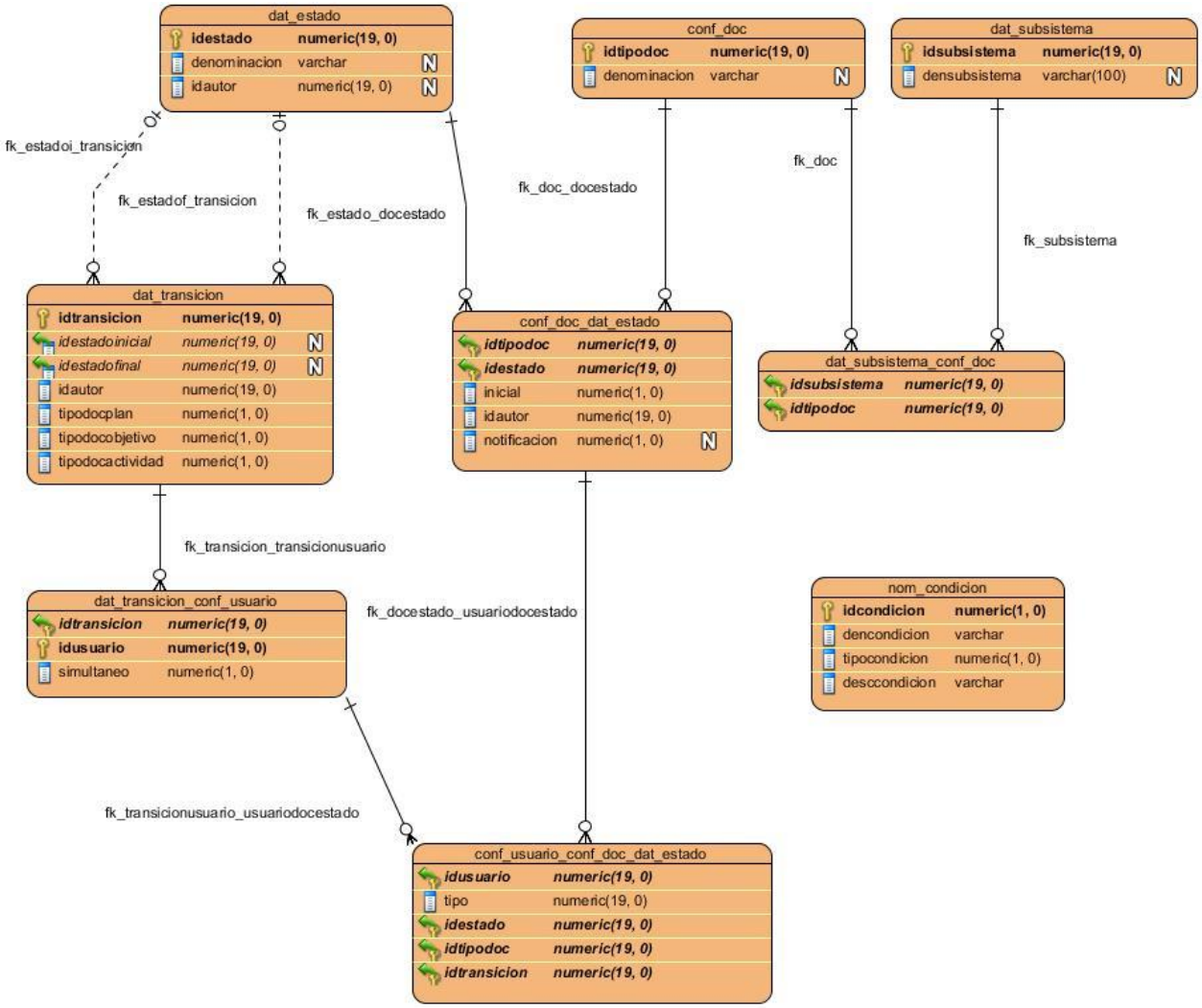


Figura 6. Modelo de datos actualizado para el Componente Flujo de Trabajo en SIPAC.

El modelo de datos de la solución va estar caracterizado principalmente por las tablas: *DatEstado*, la cual se encarga de almacenar los estados por los que va a transitar la información relacionada con los elementos primarios de la planificación, representados por las entidades *DatPlanPdo*, *DatObjetivos*, *DatFip*, *DatActividades* del esquema mod_*objetivos*, también se caracteriza por almacenar en la tabla *dat_transición* la información relacionada con las transiciones de la información, dicha tabla tiene una relación de (m...m) con la tabla *conf_usuario_conf_doc_dat_estado*, con el objetivo de identificar el usuario que crea la transición y al tipo de elemento de la planificación con el que está asociado.

2.5.4 Diagrama de componentes.

El sistema SIPAC se encarga de interrelacionar objetivos de trabajo y actividades en tiempo real de tal forma que se garantice el seguimiento del desarrollo y control del cumplimiento de los objetivos y tareas principales en las entidades. La arquitectura base de SIPAC presenta una estructura que responde a diferentes niveles de empaquetamiento: subsistema, componente, funcionalidad y requisito.(26)

De esta forma el subsistema Configuración está constituido inicialmente por el siguiente componente:

Flujo de trabajo: El Componente Flujo de Trabajo o workflow como lo indica su significado en inglés permite gestionar los estados que tomarán los documentos de la planificación así como las transiciones por las que deberá pasar la información.

En la siguiente figura se muestran las relaciones que se establecen entre los subsistemas que integran SIPAC y el Componente Flujo de Trabajo.

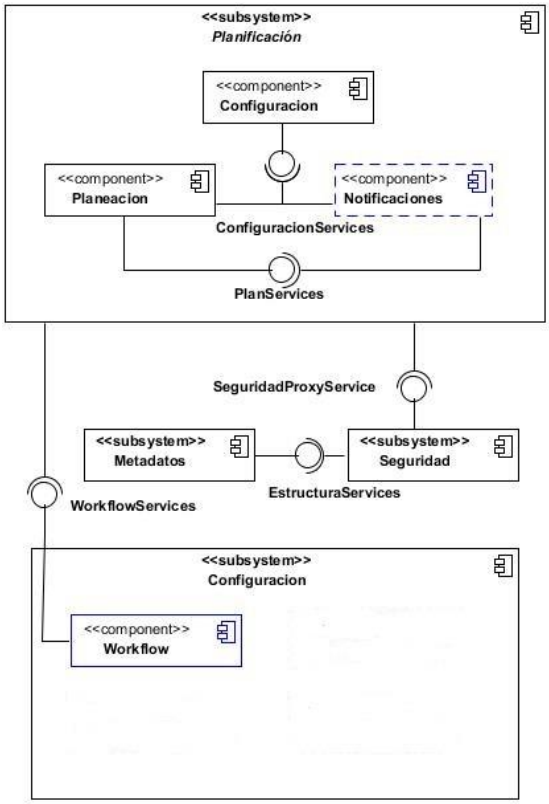


Figura 7. Representación de SIPAC y sus componentes.

El subsistema Planificación consume los servicios brindados por el subsistema Configuración donde se encuentra el Componente Flujo de Trabajo. Desde Planificación se realiza la consulta a estados y transiciones para la aprobación de los documentos.

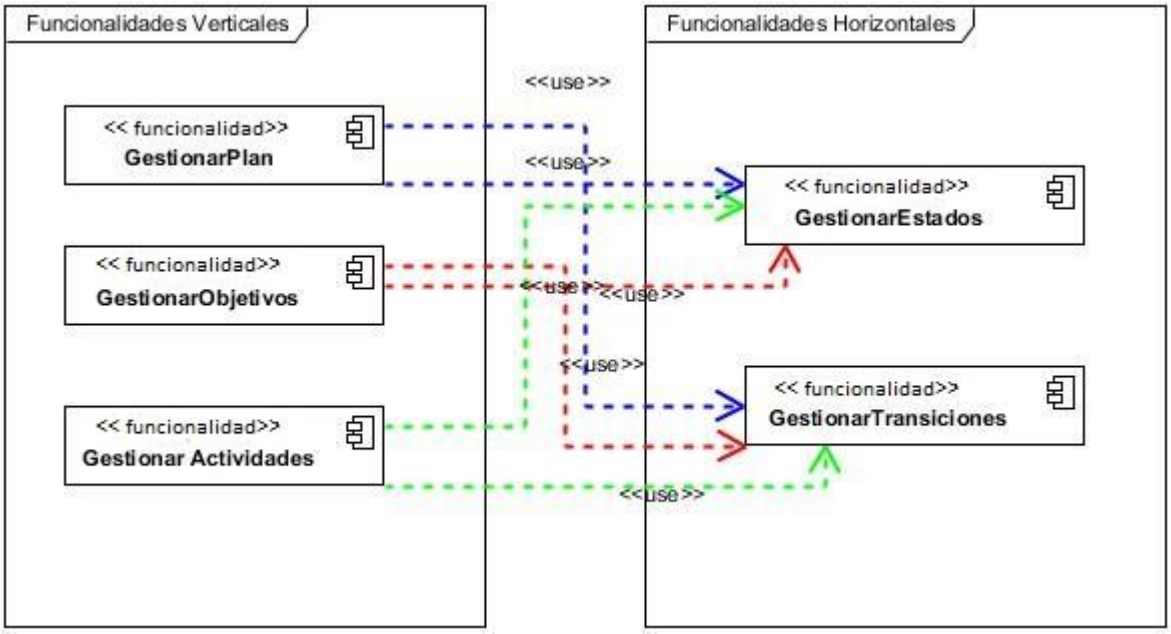


Figura 8. Diagrama de funcionalidades del componente Flujo de trabajo en SIPAC.

2.6 Patrones del diseño.

Los patrones de diseño son soluciones bien pensadas a problemas conocidos de programación lo cuales proporcionan catálogos de elementos reusables en el diseño de sistemas de software, evitando así la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.

2.6.1 Patrones GRASP.

Los patrones GRASP (General Responsibility Assignment Software Patterns) son patrones generales de software para asignación de responsabilidades.(27)

Con el objetivo de obtener un sistema flexible y reusable, se pusieron en práctica los siguientes patrones GRASP en el diseño de la solución:

- **Experto:** Determina la clase que posee mayor jerarquía para asignarle una responsabilidad según la información que posee, garantizando el encapsulamiento de la información y facilitando el bajo acoplamiento en las aplicaciones. El uso de este patrón se evidencia en las clases *Dattransicion, DattransicionConfUsuario* las cuales contienen la información necesaria para manejar todos los datos referentes a las transiciones y a la asociación de las mismas con los usuarios respectivamente
- **Creador:** Se utiliza cuando se quiere a partir de una clase con alta jerarquía obtener clases descendientes o instancias a partir de las clases obtenidas. Se evidencia su uso en la clase *GestionarFlujoTrabajoController* la cual se encarga de crear las instancias de la clase *DatTransicionModel*, para así usar las funcionalidades de esta.
- **Bajo Acoplamiento:** Se encarga de mantener las clases lo menos ligadas entre sí posible. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- **Alta Cohesión:** Se basa en que la información que almacena una clase sea coherente y esté en la medida de lo posible, relacionada con la clase. Con el uso de este patrón se logra un incremento de la claridad y comprensión, lo que simplifica el mantenimiento. Este patrón fue utilizado en el diseño del paquete de funcionalidad de manera general.
- **Controlador:** Define quién debe responder a determinados eventos. Una clase controladora debe ser la encargada de manejar los eventos dentro de la funcionalidad, así la aplicación del patrón conlleva a separar la lógica de negocios de la capa de presentación, al aplicar estos principios, el controlador no realiza las actividades mencionadas sino que las delega en otras clases. Se evidencia su uso en la clase *GestionarFlujodeTrabajoController.*

2.6.2 Patrones GoF.

Los patrones GoF (Gang Of Four) describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos y otros ámbitos referentes al diseño de interacción o interfaces.(27) Estos se clasifican según su propósito en 3 grupos: Creacional, Estructural, Comportamiento**.**

En la solución propuesta fueron utilizados los siguientes patrones GoF:

- **Fachada (Estructural):** Este patrón proporciona una interfaz unificada de alto nivel para un subsistema, que oculta las interfaces de bajo nivel de las clases que lo implementan simplificando así la interacción con el subsistema. Se utiliza para proporcionar un fácil acceso a subsistemas complejos. En el diseño de la solución que se propone, la clase que se utiliza como fachada es *ZendEXt_IoC*, para acceder a los servicios de otros componentes.
- **Cadena de Responsabilidad (Comportamiento):** Este patrón permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada. Se evidencia al distribuir las responsabilidades, ya que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo es manejado por el Modelo, creando una nueva excepción de tipo *ZendExt_Exception*. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error.

2.7 Conclusiones del capítulo.

En el presente capítulo se obtuvo los principales artefactos para la implementación mediante el análisis y diseño de la solución. Se logró un mayor entendimiento de negocio mediante la definición de los principales conceptos y procesos asociados al proceso de aprobación – conciliación de los documentos de la planificación. Se establecieron tanto los requisitos funcionales como no funcionales que debe cumplir la solución propuesta, mediante la captura, descripción y validación de los mismos. Quedaron establecidas las bases para el desarrollo de la solución con el diseño en términos componentes y la aplicación de patrones de diseño.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN

3.1 Introducción.

En el presente capítulo se detalla la arquitectura de la solución, los estándares de codificación por los que se regirá el código fuente y demás elementos referentes a la implementación de la solución: Flujo de trabajo v2.0 para el Sistema de Planificación de Actividades SIPAC. Además se valida la solución a través de un conjunto de pruebas para garantizar que cumpla con las métricas de calidad establecidas y corregir posibles errores existentes.

3.2 Implementación de la solución propuesta.

En esta sección se detalla la estructura física de la solución implementada en correspondencia con lo definido por el marco de trabajo Sauxe y el diagrama de despliegue correspondiente a la misma.

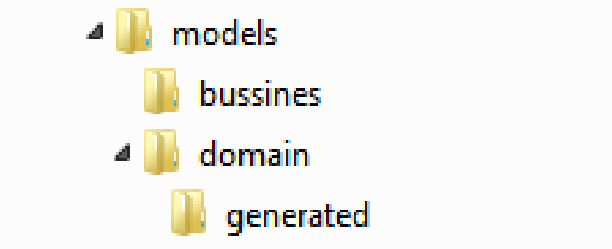
3.2.1 Estructura física de la solución.

La estructura física que presenta la solución responde a la arquitectura contenedora definida por el Departamento de Tecnologías de CEIGE. Dicha estructura permite una mejor organización de los subsistemas, componentes y clases que conforman el sistema SIPAC. A continuación se detalla la misma enfocada a la solución propuesta.

Dentro de la carpeta raíz de SIPAC se encuentra la carpeta ***apps*** la cual se encarga de manejar la lógica del negocio. Esta carpeta contiene las clases modelos y controladoras de todos los componentes de cada uno de los subsistemas, en este caso del Componente Workflow dentro del subsistema de Configuración.

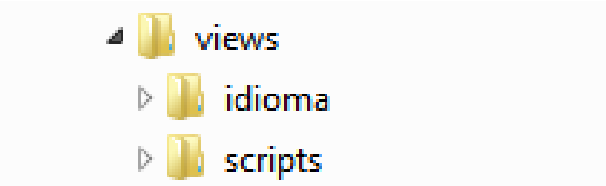
La carpeta ***workflow*** contiene a:

- ***controllers***: carpeta encargada de contener las clases controladoras para gestionar las funcionalidades del sistema.
- ***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**.



La ***bussines*** debe contener las clases necesarias para acceder a los datos que persisten en la base de datos. Por otro lado ***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. Cada una de estas clases heredará de clases bases que contienen los atributos y dependencia entre las tablas. Estas últimas son igualmente generada por el Doctrine las cuales se ubican dentro del paquete llamado ***generated***.

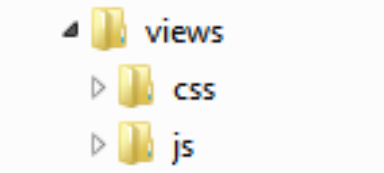
- ***service:*** el paquete incluye todas las clases y funcionalidades de los servicios que va a ofrecer el paquete de funcionalidades.
- ***views***: contiene los paquetes idioma y scripts, que se encargan de contener el idioma en que se va a mostrar la aplicación y las páginas clientes respectivamente.



Otra de las carpetas contenidas en la raíz de SIPAC es la carpeta ***web***, la cual 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.

La carpeta ***workflow*** contenida en ***web*** incluye el paquete de funcionalidades denominado Gestionar flujo de trabajo, que igualmente incorpora el fichero index.php.

También contiene el paquete ***views*** específico para la solución el cual contiene los ***css*** y ***js*** necesarios para visualizar todo el contenido de la presentación de la solución para ejecutar el proceso de Aprobación-Conciliación de los elementos de la planificación en el SIPAC



- ***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.

3.2.2 Estándares de codificación.

Los estándares de codificación permiten establecer reglas sobre la escritura del código fuente, permitiendo seguir un estilo de programación homogéneo de forma tal que todos los participantes de un proyecto puedan entender el mismo y el código, en consecuencia, sea mantenible.(28) A continuación se definen las tres partes fundamentales dentro de un estándar de programación:

- ✓ Convención de nomenclatura: Define cómo nombrar variables, funciones y clases.
- ✓ Convenciones de legibilidad de código: Es la forma de organizar el código y lograr que independientemente de quien desarrolle se entienda como un todo.
- ✓ Convenciones de documentación: Define cómo establecer comentarios, archivos de ayuda, entre otros.

Nomenclatura de las clases

Los nombres de las clases deben comenzar con mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing, la cual define que los identificadores y nombres de variables, métodos y clases que están compuestos por múltiples palabras juntas, inicia cada palabra con letra mayúscula y sin usar ningún artículo posibilitando que con solo leer el nombre de la clase ya se reconozca la función de la misma.

Nomenclatura según el tipo de clases

Clases controladoras: Las clases controladoras después del nombre llevan la palabra: “Controller”. Ejemplo: GestionarFlujoTrabajoController.

Clases de los modelos:

- Business (Negocio): Las clases que se encuentran dentro de Business después del nombre llevan la palabra: “Model”. Ejemplo: GestionarFlujoTrabajoModel.
- 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: DatTransicion.
- 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: BaseDatTransicion.

Nomenclatura de las funcionalidades

El nombre a emplear para las funciones 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 PascalCasing con la excepción de la primera letra.

Nomenclatura según la clase donde se encuentren las funciones:

- En la clase controladora: Las principales funcionalidades de las clases controladoras se les pone el nombre y seguida la palabra: “Action”. Ejemplo: adicionarTransAction().
- En las clases de los modelos: Las funcionalidades se nombran de manera que al leerlo se identifique su propósito. Ejemplo de función: dameCantTrans ().

Nomenclatura de los comentarios:

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

3.2.3 Interfaces de usuario de la solución versión 2.0 del Componente Flujo de Trabajo para el sistema de planificación SIPAC.

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

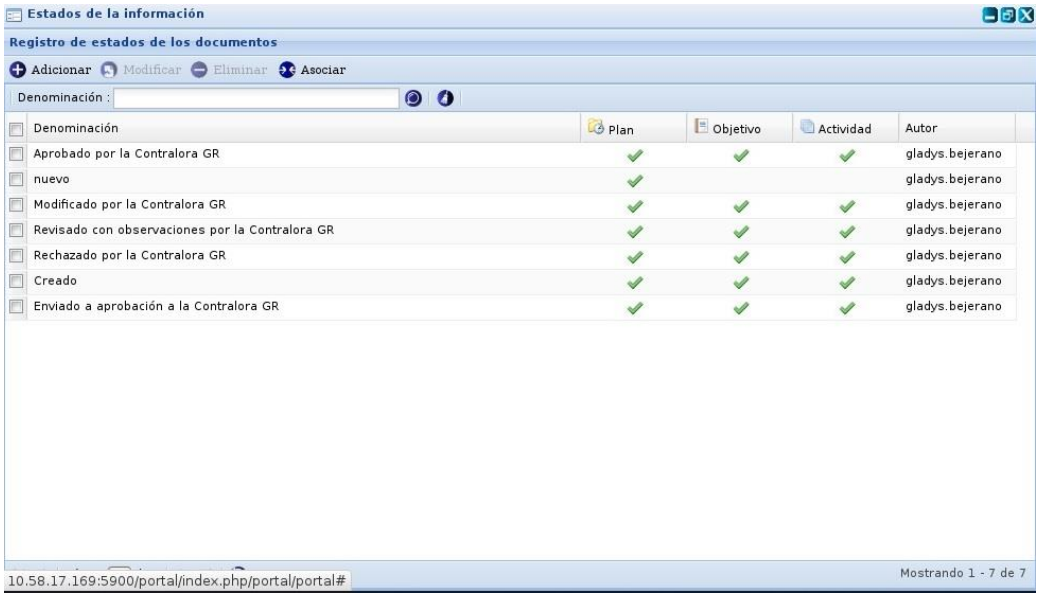


Figura 9. Estados de la información.

Figura 10. Adicionar estado.

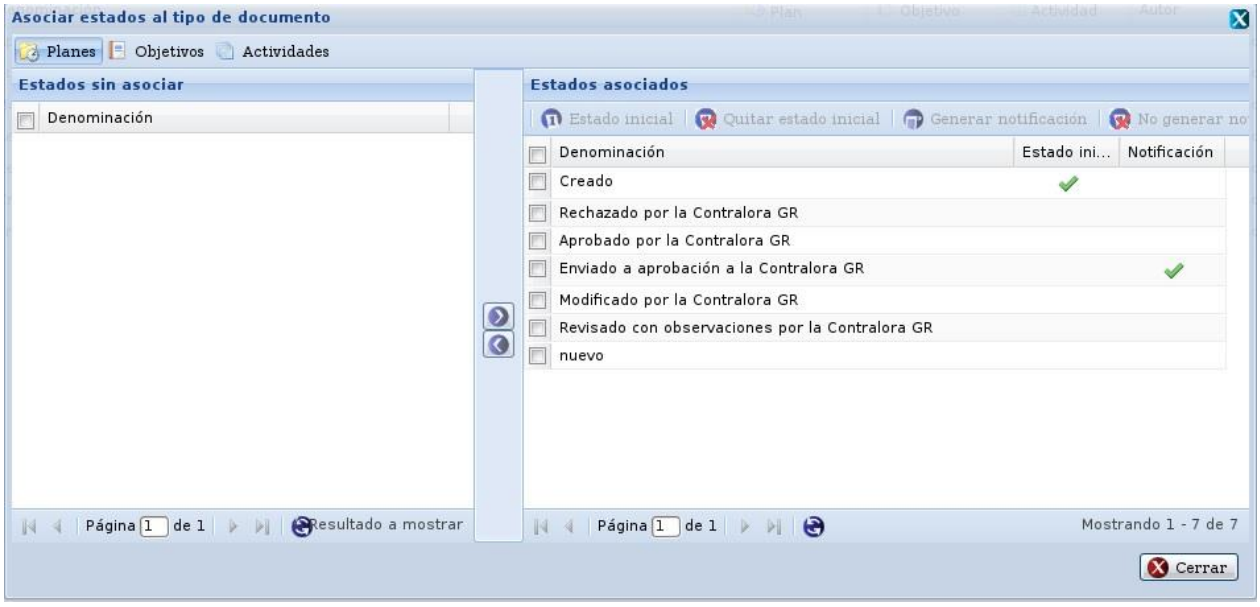


Figura 11. Asociar estados al tipo de documento.

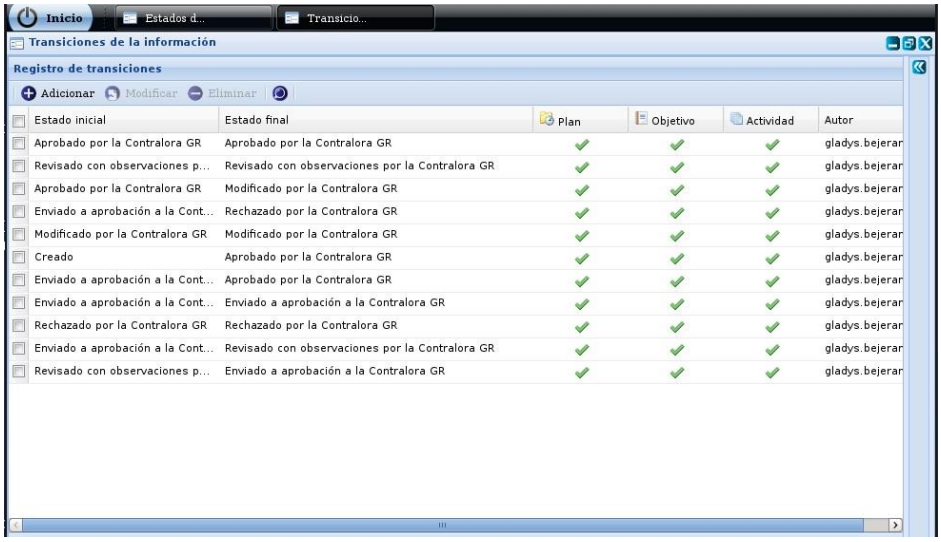


Figura 12. Transiciones de la información.

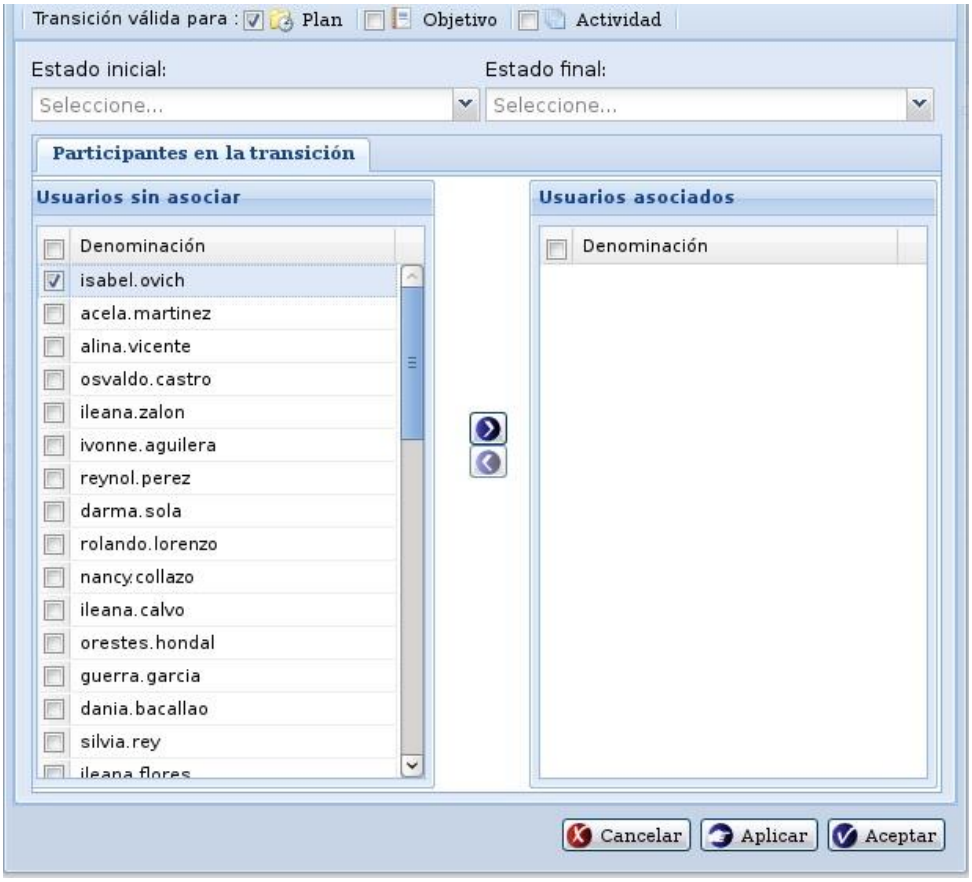


Figura 13. Adicionar transición.

3.3 Diagrama de despliegue.

A continuación se describe como queda distribuido físicamente el sistema entre los diferentes nodos de cómputo. Existen dos posibles escenarios para el despliegue de la solución: El escenario para PC cliente con disco y el escenario para PC cliente sin disco, también conocido como cliente ligero.

Figura 14. Diagrama de despliegue de escenario para PC cliente con disco.

Figura 15. Diagrama de despliegue de escenario para PC cliente sin disco.

3.4 Validación de la solución propuesta.

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es medible y varía de un sistema a otro, y se ha convertido en un elemento de gran importancia en la actualidad ya que influye directamente en la competitividad de las empresas.(29) Para detectar y corregir los posibles errores existentes en la solución implementada se detalla a continuación la validación de la solución propuesta así como las métricas para la validación del diseño propuesto y se describen las pruebas de aplicación realizadas.

3.4.1 Validación del diseño propuesto.

Para validar las clases del diseño se utilizaron las métricas: Tamaño operacional de clases (TOC) y Relación entre clases (RC). Estas métricas fueron diseñadas para validar los siguientes atributos de calidad:

- Responsabilidad: consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.
- Complejidad de implementación: grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización: grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento: grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.
- Complejidad del mantenimiento: grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.

- Cantidad de pruebas: numero o grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, modulo, clase, conjunto de clases, etc.) diseñado.

Métrica TOC: Tamaño operacional de clase.

Está dado por el número de métodos asignados a una clase. En la aplicación de esta métrica se tuvieron en cuenta las clases del Modelo y del Domain, así como la clase Controladora de acuerdo al diseño de la solución desarrollada. Se tuvo en cuenta además, la cantidad de procedimientos que contienen cada una de las clases, obteniendo los siguientes resultados:

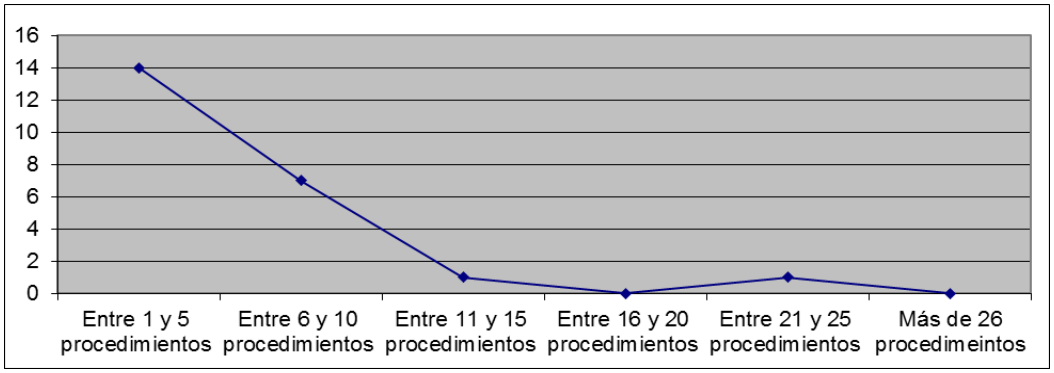


Figura 16. Representación de la cantidad de clases y el número de procedimientos que contienen.

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

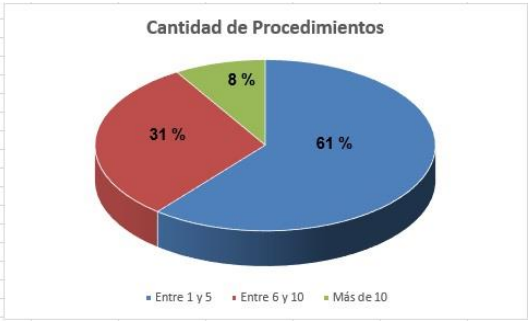


Figura 17. Representación de la cantidad de clases y el número.

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

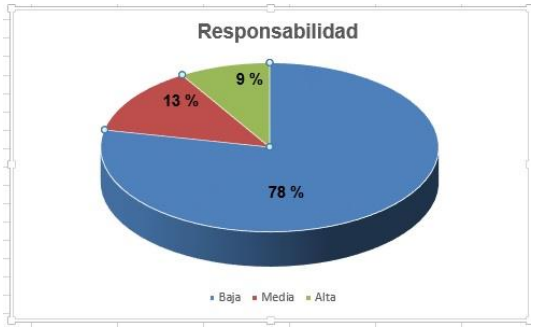


Figura 18. Representación del valor en % del atributo Responsabilidad.

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

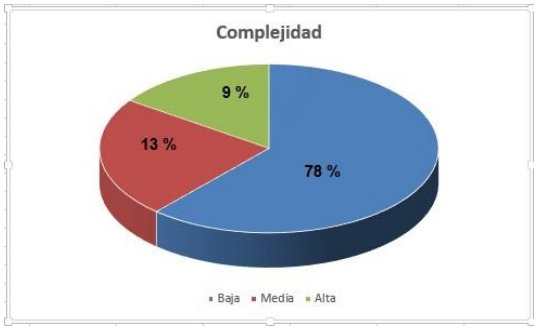


Figura 19. Representación del valor en % del atributo Complejidad de implementación.

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

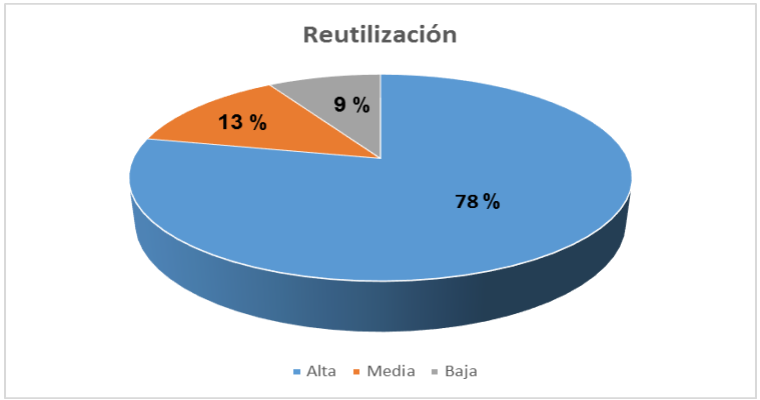


Figura 20. Representación del valor en % del atributo Reutilización.

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

Teniendo en cuenta que el 78% de las clases contienen un número de funcionalidades inferior a la media de procedimientos por clases se puede afirmar que el diseño de clases elaborado se encuentra en un nivel satisfactorio dentro de los límites de calidad. Por consiguiente, el 78% de las clases que intervienen en el diseño tienen un nivel alto de Reutilización.

Métrica RC: Relación entre clases.

Está dado por el número de relaciones de uso de una clase con otra. Dicha métrica 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. Al aplicarse la métrica Relación entre clases se obtuvo el siguiente resultado:

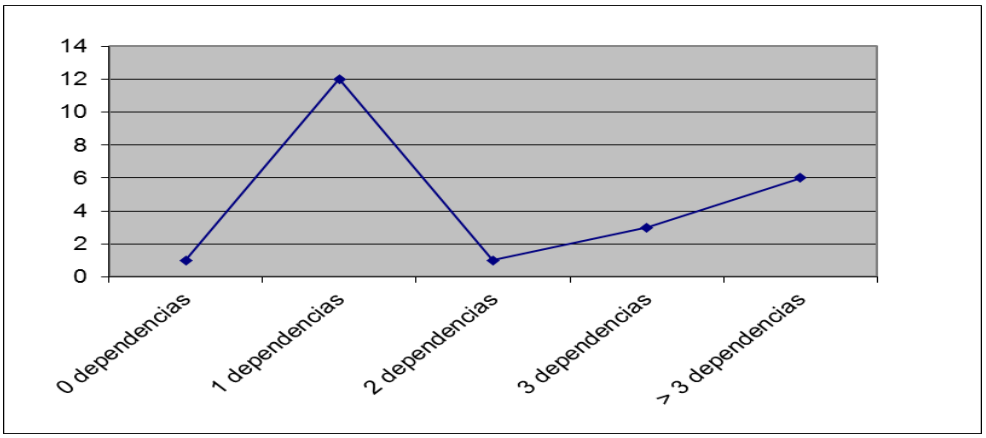


Figura 21. Representación de las asociaciones de uso por cantidad de clase.

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

Figura 22. Representación de las asociaciones de uso por cantidad de clase.

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

Figura 23. Representación del valor en % del atributo Acoplamiento.

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

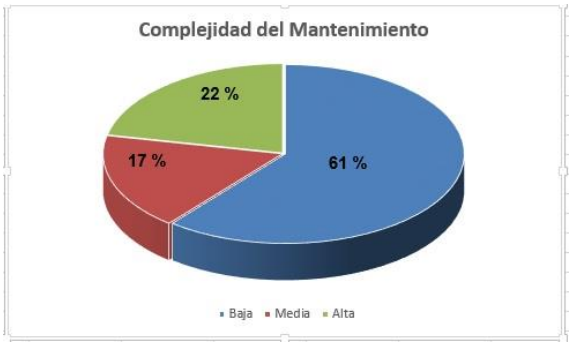


Figura 24. Representación del valor en % del atributo Complejidad de Mantenimiento.

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

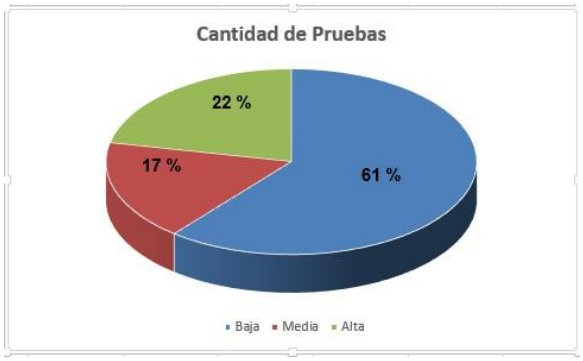


Figura 25. Representación del valor en % del atributo Cantidad de pruebas.

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

Figura 26. Representación del valor en % del atributo Reutilización.

Análisis de los resultados obtenidos en la evaluación de la métrica RC:

Después de aplicar la métrica RC se demostró que el 61% de las clases poseen menos de 3 dependencias respecto a otras. Los atributos de calidad se encuentran en un nivel satisfactorio; ya que los atributos Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 61% de las clases.

3.4.2 Pruebas de software.

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto software antes de su puesta en marcha. Básicamente, es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas bajo condiciones especificadas, los resultados son

observados o registrados, y una evaluación es realizada de un aspecto del sistema o componente.(30) Existen varios tipos de pruebas de software las cuales se describen a continuación:

- **Pruebas unitarias:** Se aplican a un componente o función del software. Su objetivo es aislar cada parte del programa y mostrar que las partes individuales son correctas. Tienen como propósito encontrar defectos en un módulo o función.
- **Pruebas de integración:** Consiste en construir el sistema a partir de los distintos componentes y probarlo con todos integrados. Su propósito es asegurar que no existan errores de interfaces y encontrar defectos en el sistema.
- **Pruebas de aceptación:** Son las únicas pruebas que son realizadas por los usuarios expertos, todas las anteriores las lleva a cabo el equipo de desarrollo. Consiste en comprobar si el producto está listo para ser implantado para el uso operativo en el entorno del usuario.
- **Pruebas funcionales:** Este tipo de prueba se realiza sobre el sistema funcionando, comprobando que cumpla con la especificación (normalmente a través de los casos de uso). Para estas pruebas, se utilizan las especificaciones de casos de prueba.

Métodos de prueba.

Los métodos de prueba del software tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo. Ayudan a definir conjuntos de casos de prueba aplicando ciertos criterios. Los métodos de prueba que se especifican a continuación permiten probar cada una de las condiciones existentes en el programa, identificar claramente las entradas, salidas y estudiar las relaciones que existen entre ellas, permitiendo así maximizar la calidad de las pruebas y de este modo obtener un sistema más estable y confiable.

- **Pruebas de caja blanca.**

También suelen ser llamadas estructurales o de cobertura lógica. Con ellas se pretende investigar sobre la estructura interna del código, exceptuando detalles referidos a datos de entrada o salida, para probar la lógica del programa desde el punto de vista algorítmico. Realizan un seguimiento del código fuente según se va ejecutando los casos de prueba que ejecutan cada posible ruta del programa o módulo, considerándose una ruta como una combinación específica de condiciones manejadas por un programa.(41) Este tipo de prueba le permite al ingeniero de software obtener casos de pruebas que:

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

Aplicación de las pruebas de caja blanca

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.

Con el objetivo de valorar la calidad con la que se llevó a cabo la implementación de la solución Componente Flujo de Trabajo v 2.0 del sistema para el Sistema para la Planificación de actividades SIPAC, fue necesario aplicar la técnica descrita anteriormente: **la prueba del camino básico**.

Para ello fue necesario seguir los siguientes pasos básicos:

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

Dando cumplimiento a los anteriores pasos básicos se enumeran cada una de las sentencias de código de uno de los procedimientos de la clase *GestionarFlujoTrabajoController*, específicamente la funcionalidad *adicionarTransicionAction.*

```
function adicionarTransicionAction() {//1
    $idusuariosession = $this->global->Perfil->idusuario;//1
    $idusuariosarelacionar = json_decode(stripslashes($this->_request->getPost('idusuariosarelacionar')));//1
    $idestadoinicial = $this->_request->getPost('idestadoinicial');//1
    $idestadofinal = $this->_request->getPost('idestadofinal');//1
    $tipodocplan = $this->_request->getPost('tipodocplan');//1
    $tipodocobjetivo = $this->_request->getPost('tipodocobjetivo');//1
    $tipodocactividad = $this->_request->getPost('tipodocactividad');//1
    $error = 0;//1
    if ($tipodocplan) {//2
        $existeasociacionestadoinicial = ConfDocDatEstado::VerificarAsociacionDocumentoEstado(0, $idestadoinicial);//3
        $existeasociacionestadofinal = ConfDocDatEstado::VerificarAsociacionDocumentoEstado(0, $idestadofinal);//3
        if (!isset($existeasociacionestadoinicial[0]) || !isset($existeasociacionestadofinal[0])) {//4
            $error = 1;//5
            $respuesta = 2;//5
        }//6
    }//7
    $existetransicion = Doctrine::getTable('DatTransicion')->findByDql("idestadoinicial = '$idestadoinicial' AND idestadofinal = '$idestadofinal'");//8
    if (isset($existetransicion[0]->idtransicion) && $existetransicion[0]->idtransicion != '') {//9
        $error = 1;//10
        $respuesta = 3;//10
    }//11
    if ($error == 0) {//12
        $objtransicion = new DatTransicion();//13
        $objtransicion->idestadoinicial = $idestadoinicial;//13
        $objtransicion->idestadofinal = $idestadofinal;//13
        $objtransicion->idautor = $idusuariosession;//13
        $objtransicion->tipodocplan = $tipodocplan;//13
        $objtransicion->tipodocobjetivo = $tipodocobjetivo;//13
        $objtransicion->tipodocactividad = $tipodocactividad;//13
        $objtransicion->save();//13
        $idtransicion = $objtransicion->idtransicion;//13
        $respuesta = 1;//13
    }//14
    echo json_encode(array('mensaje' => $respuesta));//15
}//16
```

Figura 27. Fragmento de código con el algoritmo *adicionarTransicionAction*.

Luego de realizado el procedimiento del primer paso, se hace necesario representar el grafo de flujo asociado al código presentado anteriormente. Para ello se utilizan aristas, nodos y regiones.

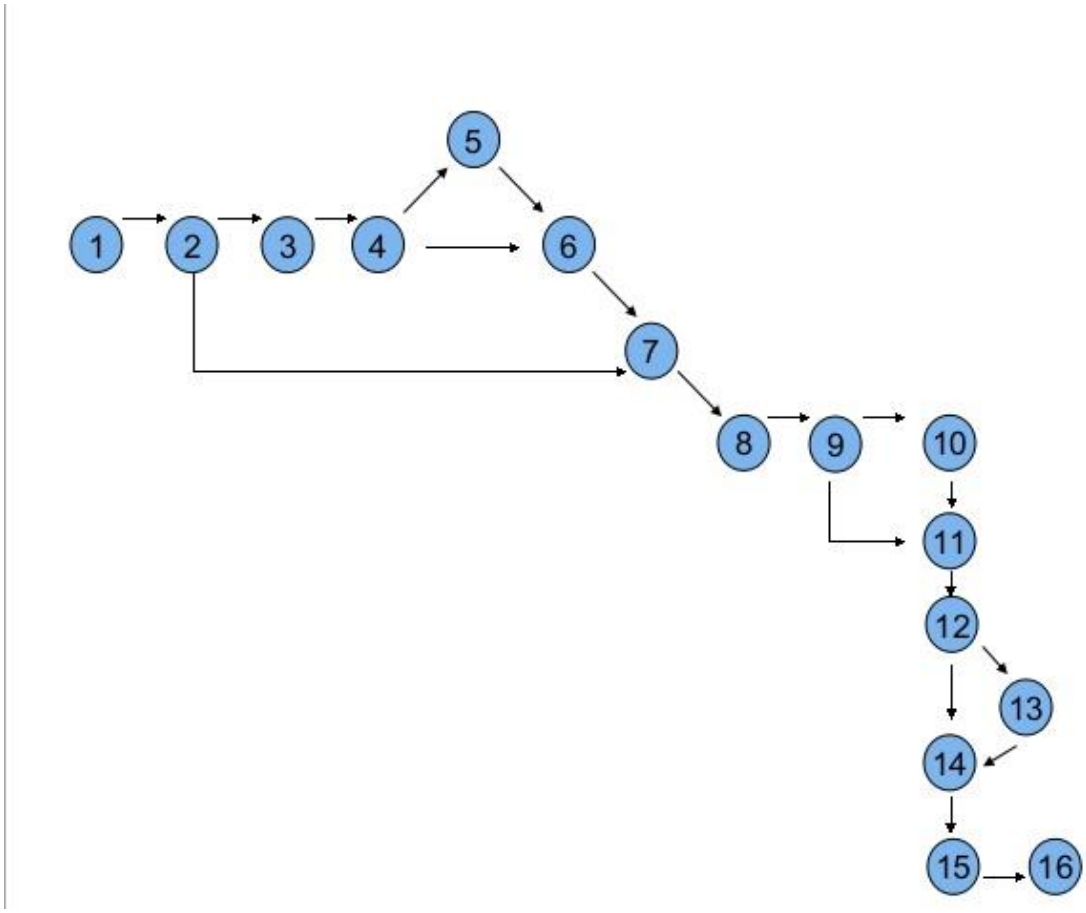


Figura 28. Grafo de flujo asociado al procedimiento *adicionarTransicionAction.*

Una vez representado el gráfico de flujo asociado al procedimiento *adicionarTransicionAction* se prosigue a calcular la complejidad ciclomática de este, la cual garantiza la menor cantidad de casos de pruebas a realizar, para que se ejecute al menos una vez cada sentencia de código. El cálculo de la complejidad ciclomática es necesario efectuarlo mediante tres vías o fórmulas, utilizando siempre el mismo grafo en cada caso

- ✓ **Fórmula 1:** V(G) = (A - N) + 2

  A: cantidad total de aristas del grafo.

  N: cantidad total de nodos del grafo.

  Resultado: V (G) = (19 - 16) + 2 = 5
- ✓ **Fórmula 2:** V(G) = P + 1

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

Resultado: V (G) =4 + 1 = 5

- ✓ **Fórmula 3:** V(G) = R

  R: cantidad total de regiones existentes en el grafo, se incluye el área exterior del grafo, como una región más.

  Resultado: V (G) = 5

Luego de calculada la complejidad ciclomática a través de la fórmulas, arrojando un resultado de 4, se concluye que existen 4 caminos básicos, lo cual representa el número mínimo de casos de prueba que se deben aplicar para valorar la calidad con que se implementó la solución propuesta. A continuación se representan los 5 caminos básicos por donde puede circular el flujo asociado al procedimiento en cuestión:

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

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

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

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

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

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo. Para definir los casos de prueba es necesario tener en cuenta:

Descripción: Se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada y así ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que serán la entrada al procedimiento.

Resultados Esperados: Se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

La prueba de caja blanca que se realizará al procedimiento *adicionartransicionAction* invocado cuando se pretende adicionar una nueva transición como parte del flujo de aprobación de los documentos:

Datos generales:

$idestadoinicial = estado inicial de la transición en este caso, “Creado”.

$idestadofinal = estado final de la transición en este caso, “Enviado”.

$tipodocplan = significa que el tipo de documento a incluir en la nueva transición es un plan. En el algoritmo se incluye en la comprobación $tipodocactividad y $tipodocobjetivo, sin embargo en las pruebas no se especifica debido a que se repite el comportamiento de la función para cada una de estas variables.

A continuación se describen los casos de prueba para cada uno de los caminos básicos determinados en el grafo de flujo:

Tabla 4. Caso de prueba para el camino básico #1.

Camino básico 1: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16	
Descripción	Se debe adicionar una nueva transición como parte del flujo de aprobación de los documentos, para ello es necesario que se comprueben que los estados iniciales y finales estén asociados a plan. Se debe comprobar además que la transición no ha sido creada previamente.
Condición de ejecución	Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: $idestadoinicial y $idestadofinal no pueden estar vacíos.
Entrada	$tipodocplan = 1 significa que el tipo de documento a incluir en la nueva transición es un plan. $existeasociacionestadoinicial = significa que el estado inicial está asociado a planes, este caso es 1. $existeasociacionestadofinal = significa que el estado inicial está asociado a

	planes, este caso es 1. $existetransicion = significa que existe la transición, en este caso: no existe. $error = significa que ha ocurrido algún error en el proceso, en este caso posee valor 0.
Resultado esperado	Teniendo en cuenta los datos de entrada se debe adicionar la transición para planes con Estado inicial: Creado y Estado final: Enviado.
Resultado	La acción fue ejecutada correctamente.

Tabla 5. Caso de prueba para el camino básico # 2.

Camino básico 2: 1-2-3-4-6-7-8-9-10-11-12-13-14-15	
Descripción	Se debe adicionar una nueva transición como parte del flujo de aprobación de los documentos, para ello es necesario que se comprueben que los estados iniciales y finales estén asociados a plan. Se debe comprobar además que la transición no ha sido creada previamente.
Condición de ejecución	Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: $idestadoinicial y $idestadofinal no pueden estar vacíos.

Entrada	$tipodocplan = 0 significa que el tipo de documento a incluir en la nueva transición es un pan. $existeasociacionestadoinicial = significa que el estado inicial está asociado a planes, este caso 1. $existeasociacionestadofinal = significa que el estado inicial está asociado a planes, este caso 1. $existetransicion = significa que existe la transición, en este caso: no existe. $error = significa que ha ocurrido algún error en el proceso, en este caso: 0.
Resultado esperado	Teniendo en cuenta los datos de entrada se debe adicionar la transición con Estado inicial: Creado y Estado final: Enviado. Se comprueba el tipo de documento al cual se le está asociando la transición.
Resultado	La acción fue ejecutada correctamente.

Tabla 6 . Caso de prueba para el camino básico # 3.

Camino básico 3: 1-2-3-4-5-6-7-8-9-11-12-13-14-15-16	
Descripción	Se debe adicionar una nueva transición como parte del flujo de aprobación de los documentos, para ello es necesario que se comprueben que los estados iniciales y finales estén asociados a plan. Se debe comprobar además que la transición no

	ha sido creada previamente.
Condición de ejecución	Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: $idestadoinicial y $idestadofinal no pueden estar vacíos.
Entrada	$tipodocplan = 1 significa que el tipo de documento a incluir en la nueva transición es un pan. $existeasociacionestadoinicial = significa que el estado inicial está asociado a planes, este caso 0. $existeasociacionestadofinal = significa que el estado inicial está asociado a planes, este caso 0. $existetransicion = significa que existe la transición, en este caso: no existe. $error = significa que ha ocurrido algún error en el proceso, en este caso: 0.
Resultado esperado	Teniendo en cuenta los datos de entrada, la variable $error toma valor 1 y se muestra un mensaje de error en el cual se indica que los estados definidos en la transición no están asociados al tipo de documento plan.
Resultado	La acción fue ejecutada correctamente.

Tabla 7. Caso de prueba para el camino básico # 4.

Camino básico 4: 1-2-3-4-6-7-8-9-11-12-14-15-16	

Descripción	Se debe adicionar una nueva transición como parte del flujo de aprobación de los documentos, para ello es necesario que se comprueben que los estados iniciales y finales estén asociados a plan. Se debe comprobar además que la transición no ha sido creada previamente.
Condición de ejecución	Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: $idestadoinicial y $idestadofinal no pueden estar vacíos.
Entrada	$tipodocplan = 1 significa que el tipo de documento a incluir en la nueva transición es un pan. $existeasociacionestadoinicial = significa que el estado inicial está asociado a planes, este caso 1. $existeasociacionestadofinal = significa que el estado inicial está asociado a planes, este caso 1. $existetransicion = significa que existe la transición, en este caso: existe. $error = significa que ha ocurrido algún error en el proceso, en este caso: 0.
Resultado esperado	Teniendo en cuenta los datos de entrada, la variable $error toma valor 1 y se muestra un mensaje de error en el cual se indica que existe una transición para el tipo de documento plan con los

	mismos estados inicial y final.
Resultado	La acción fue ejecutada correctamente.

Tabla 8. Caso de prueba para el camino básico # 5.

Camino básico 5: 1-2-3-4-5-6-7-8-9-10-11-12-14-15-16	
Descripción	Se debe adicionar una nueva transición como parte del flujo de aprobación de los documentos, para ello es necesario que se comprueben que los estados iniciales y finales estén asociados a plan. Se debe comprobar además que la transición no ha sido creada previamente.
Condición de ejecución	Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: $idestadoinicial y $idestadofinal no pueden estar vacíos.
Entrada	$tipodocplan = 1 significa que el tipo de documento a incluir en la nueva transición es un pan. $existeasociacionestadoinicial = significa que el estado inicial está asociado a planes, este caso 1. $existeasociacionestadofinal = significa que el estado inicial está asociado a planes, este caso 1. $existetransicion = significa que existe la transición, en este caso: no existe. $error = significa que ha ocurrido algún

	error en el proceso, en este caso: 0.
Resultado esperado	Teniendo en cuenta los datos de entrada, se verifica el valor de la variable $error, en este caso ha tomado valor 1, lo que indica que se ha producido un error durante la ejecución que impide que se adicione la transición.
Resultado	La acción fue ejecutada correctamente.

Luego de aplicar los distintos casos de prueba correspondientes a cada camino básico identificado en el grafo de flujo asociado, se pudo comprobar que el flujo de la función está correcto, cumple con las condiciones necesarias que se habían planteado.

- **Pruebas de caja negra.**

También suelen ser llamadas funcionales y basadas en especificaciones. Con ellas se pretende examinar el programa para verificar que cuente con las funcionalidades que debe tener y la forma de llevar a cabo las mismas, analizando siempre los resultados que devuelve y probando todas las entradas en sus valores válidos e inválidos. Las pruebas no se hacen en base al código, sino a la interfaz. (41)Este tipo de prueba intenta encontrar errores de las siguientes categorías:

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

Aplicación de las pruebas de caja negra

Existen también diferentes técnica de caja negra:

- **Partición equivalente:** Se divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Se basa en una evaluación de las clases de equivalencia para una condición de entrada.

- **Análisis de valores límites (AVL):** Permite la elección de casos de prueba que ejerciten los valores límites. Es una técnica de diseño de casos de prueba que complementa la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase, en lugar de centrarse en las condiciones de entrada, el AVL también obtiene casos de prueba para el campo de salida.

A continuación se aplica la prueba de partición de equivalencia como parte de las pruebas de caja negra realizadas sobre el requisito funcional *Adicionar Transición*.

Tabla 9. Caso de prueba de caja negra para validar el requisito funcional Adicionar transición.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar transición.	1. El sistema debe permitir que el usuario al seleccionar la opción Adicionar transición, adicione una nueva transición.	EP 1.1: Flujo Básico de eventos	1. Se seleccionan los datos de la transición: Transición válida para Estado inicial y Estado final. Se selecciona además el tipo de documento y el o los usuarios asociados a la transición. 2. Se selecciona la opción aceptar. 3. El sistema confirma el registro de la transición.
		EP 1.2: Se seleccionan datos incompletos	1. Se seleccionan los datos de la transición: Transición válida para Estado inicial y Estado final. Se dejan de seleccionar o el tipo de documento, o el usuario asociado o alguno de los estados 2. Se selecciona la opción aceptar. El sistema señala los datos vacíos y

			permite corregirlos.
		EP 1.3: Se cancela la acción.	1. Se seleccionan los datos de la transición: Transición válida para Estado inicial y Estado final. Se selecciona además el tipo de documento y el o los usuarios asociados a la transición. 2. Se selecciona la opción cancelar 3. El sistema no registra los datos de la transición y cierra la ventana de Adicionar transición.

3.4.3 Resultados de las pruebas aplicadas.

Luego de aplicados los métodos de prueba a las funcionalidades implementadas se pudo concluir que los resultados obtenidos hasta el momento han sido satisfactorios desde el punto de vista funcional. Se probaron un total de 14 funcionalidades en cada iteración. Las no conformidades detectadas fueron debidamente atendidas en aras de lograr el correcto comportamiento de la solución desarrollada ante diferentes situaciones. A continuación se muestra la cantidad de no conformidades detectadas en cada iteración de pruebas realizada:

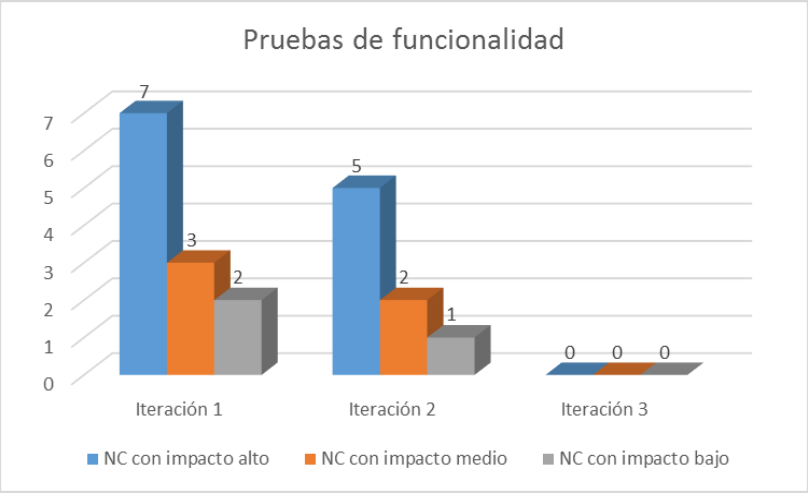


Figura 29. No conformidades detectadas durante las iteraciones de pruebas realizadas.

3.5 Conclusiones del capítulo.

El desarrollo del capítulo actual permitió arribar a las siguientes conclusiones parciales:

- La definición de los principales aspectos que influyeron en la implementación de la solución propuesta fue crucial para el desarrollo de una solución que ejecute con la calidad requerida el proceso de Aprobación-Conciliación de los documentos de la planificación
- Se definió la estructura física de la solución haciendo también referencia a los estándares de codificación utilizados, los prototipos de las interfaces con las que podrá interactuar el usuario, así como los elementos necesarios para su despliegue representados en un diagrama por cada escenario.
- Se validó además el diseño propuesto a partir de la aplicación de métricas (Tamaño operacional de clase y Relaciones entre clases) que permitieron evaluar aspectos como la complejidad de la implementación, la responsabilidad y reutilización de las clases arrojando resultados satisfactorios en cuanto al trabajo realizado
- La aplicación de pruebas de caja blanca (Técnica del camino básico) y pruebas de caja negra (Técnica partición de equivalencia) posibilitó verificar el correcto funcionamiento del Componente Flujo de Trabajo v2.0 para el Sistema de Planificación de Actividades SIPAC.

CONCLUSIONES GENERALES

La realización del presente trabajo y los resultados obtenidos con el mismo permitieron llegar a las siguientes conclusiones generales:

- El estudio de metodologías y sistemas informáticos que ejecutan Flujos de Trabajo, evidencian la necesidad de un sistema que permita ejecutar el mismo cumpliendo con las características y doctrinas del sistema de planificación cubano.
- La solución desarrollada posibilita el establecimiento de los estados y transiciones de los documentos que regulen el Proceso Aprobación-Conciliación en cada entidad.
- El sistema permite y facilita el seguimiento y control de la planeación estratégica y Operativa tanto para una entidad como a nivel de gobierno pues consiente el permiso de accesos a la información en cada nivel, la creación de estados que determinan el comportamiento de los documentos de una manera uniforme y queda claro al usuario final el camino lógico a recorrer por los documentos de la planificación.
- La obtención de una solución funcional que ejecute un flujo de trabajo correcto y bajo las indicaciones de la Instrucción No. 1 del Presidente de los Consejos de Estado y de Ministros garantiza un proceso de Planificación óptimo, muy cercano a la realidad y tangible en todos los Órganos, Organismos de la Administración Central del Estado, Entidades Nacionales y las Administraciones Locales del Poder popular.

RECOMENDACIONES

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

- Comprobar la factibilidad de la solución desarrollada tomando en cuenta diferentes entornos organizacionales.
- Agregar funcionalidades al módulo que permitan una mayor usabilidad del mismo.

BIBLIOGRAFÍA REFERENCIADA

1 MARTINTO, Msc. Pedro Carlos Pérez. *El diseño metodológico de la investigación científica. Teoría de Muestreo: población y muestra. Diseño experimental y métodos. In* : .

2 Sandy Machado Scull y Yuliet Galán Ramírez. *Glosario de términos del subsistema Planificación por Objetivos.* 1.0, Ciudad Habana, Cuba : s.n., 19 de 11 de 2009. Vol. 1.0. Documentación del Proyecto PDO. CIG-ERP-N-DPO-s1201.

3 Presidente de los Consejos de Estados y de Ministros. *Instrucción No. 1 para la planificación de los objetivos y actividades en los Órganos de la Administración Central del Estado, Entidades Nacionales y las Administraciones Locales del Poder Popular*, Ciudad Habana, Cuba: s.n., 11 de 2011

4 CEIGE. *Manual de Usuario para el Sistema de Planificación de Actividades* [online]. 2012. Available from: SIPAC 1.0/Exp 2.2/1. Ingeniería/1.3 Implementación y prueba/manuales/Manual de Usuario 1.2

5 GONZÁLEZ CORNEJO, JOSÉ ENRIQUE. *El Lenguaje de Modelado Unificado* (UML). [online]. Enero 2008. [Accessed 22 Sep 2014]. Available from: http://www.docirs.cl/uml.htm

6 *UML: Un Lenguaje Modelo*. *Cientec* [online]. [Accessed 22 Sep 2014]. Available from: http://www.cientec.com/analisis/ana-uml.html

7 *INTRODUCCION A HTML* (páginas web). [online]. [Accessed 22 Sep 2014]. Available from: http://eusalud.uninet.edu/Cursos/doc99/INTROHTML.html

8 *Manual de* CSS, *hojas de estilo.* [online]. [Accessed 22 Sep 2014]. Available from: http://www.desarrolloweb.com/manuales/manual-css-hojas-de-estilo.html

9 *Definicion de JavaScript.* [online]. [Accessed 22 Sep 2014]. Available from: http://www.pergaminovirtual.com.ar/definicion/JavaScript.html

10 *PHP: Hypertext Preprocessor.* [online]. [Accessed 22 Sep 2014]. Available from: http://www.php.net/

11 RECAMAN CHAUX, HERNANDO. *Marco de trabajo para aplicaciones web de código abierto en instituciones universitarias.* 29 February 2012.

12 ROB, ALLEN. *Introducción a Zend Framework. www.akrabat.com* [online]. 2010. [Accessed 22 May 2014]. Available from: http://alemohamad.com/tutorial-zend-framework/

13 *Doctrine 1.2 ORM Manual.* [online]. [Accessed 22 Sep 2014]. Available from: http://docs.doctrine-project.org/projects/doctrine1/en/latest/en/manual/introduction.html#about-this-version

14 *Curso extjs. Buenas Tareas* [online]. November 2011. [Accessed 22 Sep 2014]. Available from: http://www.buenastareas.com/ensayos/Curso-Extjs/3156528.html

15 *Introducción a AJAX. Libros Web* [online]. [Accessed 29 Dec 2014]. Available from: http://librosweb.es/ajax/capitulo_1.html

16 *What's new in Visual Paradigm 11.1?* [online]. [Accessed 29 Dec 2014]. Available from: http://www.visual-paradigm.com/whats-new/

17 *Apache Subversion.* [online]. [Accessed 29 Dec 2014]. Available from: http://subversion.apache.org/

18 *NetBeans IDE -The Smarter and Faster Way to Code.* [online]. [Accessed 22 May 2014]. Available from: https://netbeans.org/features/index.html

19 *Apache - Servidor HTTP Apache 2.0.* [online]. [Accessed 29 Dec 2014]. Available from: http://httpd.apache.org/docs/2.0/es/invoking.html

20 RENDÓN ARTOLA, ARIADNA. *CIG-SPA-N-i3504-DT (Vista Entorno de Desarrollo Tecnológico).*

21 UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS. *Metodología UCI*

22 IEEE: Standard Glossary of Software Engineering Terminology. [no date].

23 IAN SOMMERVILLE. *Ingeniería de Software. 7ma.* [no date].

24 ARIAS CHAVES, MICHAEL. *790-1199-1-PB (La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.).*

25 LÓPEZ JIMÉNEZ, SUSANA ALEJANDRA. *Fbd-tutorial-i2 (modelo de datos).*

26 RENDÓN ARTOLA, ARIADNA. *CIG-SPA-N-i3502-AB (Arquitectura de Software-Guía Base-SIPAC 2.0).*

27 *Patrones de Diseño.pdf.* [online]. [Accessed 20 Jan 2015]. Available from: https://eseida.wikispaces.com/file/view/Tema 6/ Patrones de Diseño.pdf

28 CENTRO DE INFORMATIZACIÓN DE ENTIDADES. *Estándares de codificación Proyectos con el marco de trabajo Sauxe del CEIGE.*

29 ACIMED - *Un enfoque actual sobre la calidad del software.* [online]. [Accessed 22 May 2014]. Available from: http://scielo.sld.cu/scielo.php?pid=S1024-94351995000300005&script=sci_arttext

30 SERGIO OCHOA D. CC51A – *Ingeniería de Software-Pruebas de Software. .*