UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

Título: Módulo de flujos de trabajo para SIGE.

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

Autores: Lisandra Morgado Legrá

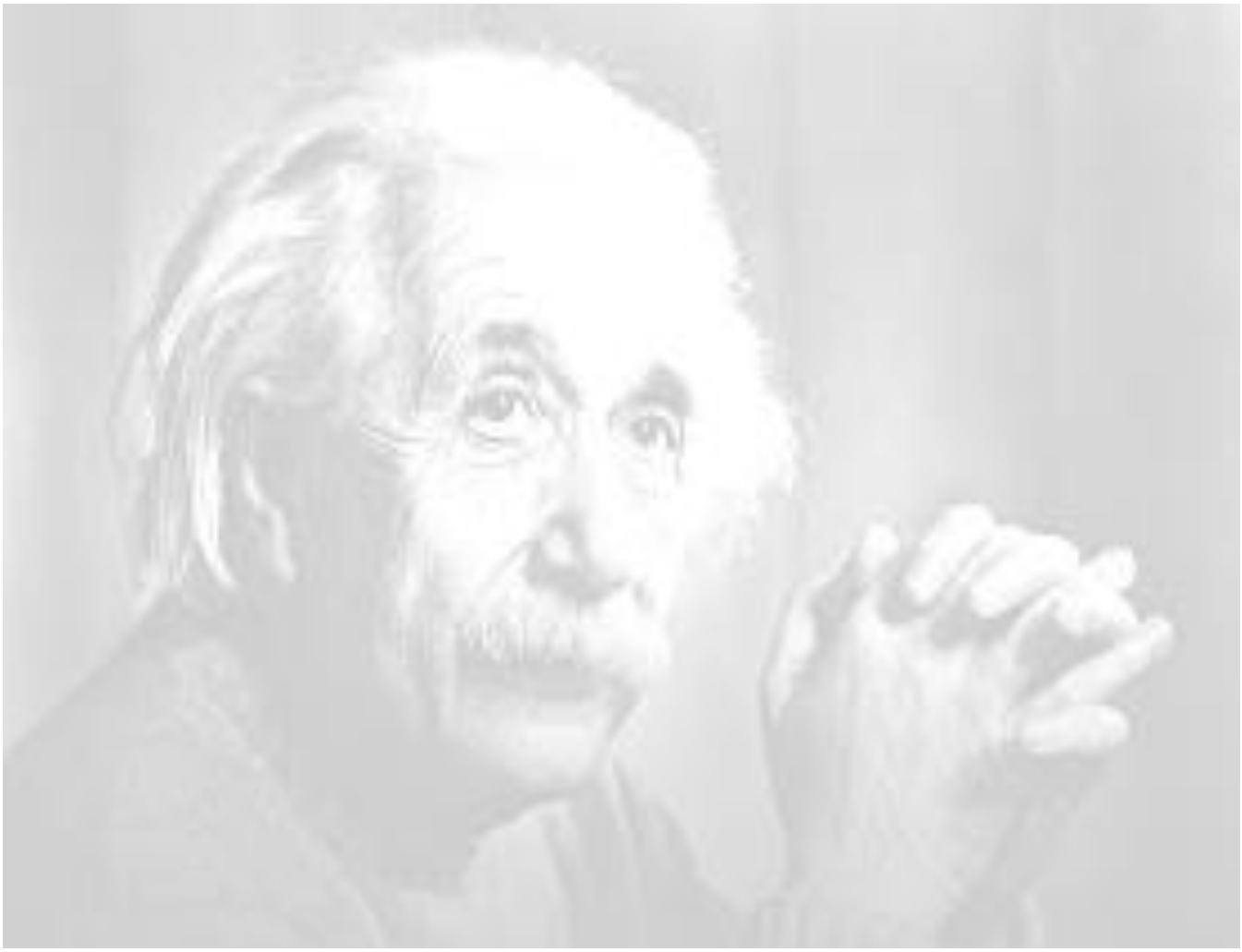
Jorge Luis Medina Pérez

Tutores: MSc. Alexis René Rodríguez León

Ing. Yunier Leyva Quintanilla

La Habana, junio de 2015

"Año 56 de la Revolución"



"Nunca consideres el estudio como un deber, sino como una oportunidad para penetrar en el maravilloso mundo del saber."

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Lisandra Morgado Legrá

_____

Firma del autor

Jorge Luis Medina Pérez

_____

Firma del autor

Yunier Leyva Quintanilla

_____

Firma del tutor

Alexis René Rodríguez León

_____

Firma del tutor

DATOS DE CONTACTO

Tutor

MsC. Alexis René Rodríguez León: Profesor graduado de Ingeniero en Ciencias Informáticas en el año 2007, en la Universidad de las Ciencias Informáticas. Se desempeña como profesor en la facultad 6 y como investigador en el Grupo de Bioinformática del Centro de Estudios de Matemática Computacional.
Contactar al correo arodriguezl@uci.cu

Tutor

Ing. Yunier Leyva Quintanilla: Profesor recién graduado de Ingeniero en Ciencias Informáticas en el año 2013, en la Universidad de las Ciencias Informáticas. Se desempeña como programador y jefe de la línea de desarrollo del proyecto SIGE perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC).
Contactar al correo yleyva@uci.cu

AGRADECIMIENTOS

Hoy se hizo realidad mi sueño después de muchos años de sacrificio he alcanzado la meta que una vez me propuse y por eso quiero agradecer a todas aquellas personas que de una forma u otra han estado apoyándome en todo momento.

A ***Dios*** *por haberme dado la vida, por guiar mis pasos y permitirme el haber llegado hasta este momento tan importante.*

A las personas más importantes en mi vida:

A mi Mamucha *por su amor, confianza, dedicación, apoyo, por ser guía ejemplar y constante, por ser una madre excepcional y lo que más amo de esta vida. Esto sueño es tuyo también.*

A mi papá *que aunque las circunstancias de la vida no nos dio la oportunidad de estar cerca igual sabes que te amo. Gracias por estar en mi vida y ser parte de ella*

A Manolo *por ser mi segundo padre, aunque sé que eso ya lo sabes. Por quererme como tu hija, no existen palabras en el mundo para expresarte cuanto te amo.*

A mis abuelos *por ser cómplices de mi niñez y estar pendientes de mí en todo momento. Por ser como unos padres para mí. Solo mi corazón sabe cuán grande es el amor que siento por ustedes. Los quiero mucho.*

A Amauri *por tu infinita paciencia, por tu tierna compañía y tu inagotable apoyo. Apenas tienes una idea de todo lo que significas para mí, por ser esa persona que ocupas un lugar en mi corazón, por apoyarme para nunca renunciar, gracias por tu amor incondicional. TE amo.*

A mis tíos Cary y Gustavo *por ser una hija para ustedes, por creer siempre en mí y aconsejarme con espíritu alentador, contribuyendo a lograr mis metas y objetivos propuestos. No hay forma humana ni palabras que me permitan expresar cuanto los quiero.*

Alina, Frank y a mi hermano*, por confiar en mí y estar siempre conmigo apoyándome en todo. Los adoro.*

A toda mi familia *y especialmente a mis tíos habaneros Elena y guille por acogerme en su casa y por permitirme conocer algunos lugares de la habana que jamás hubiera podido ir. En general los quiero y les doy las gracias porque todos han puesto su granito de arena para que mi sueño se hiciera realidad.*

A Jisel: *Dos hermanas, amigas y compañeras en los buenos y malos momentos de estos 5 años, después de muchos sacrificios han alcanzado la meta propuesta. Por haber hecho el papel de una familia verdadera en todo momento, gracias por la amistad que me has brindado desinteresadamente, por aceptarme como soy, y soportarme; porque juntas hemos hecho realidad este sueño.*

A una personita que a pesar de su carácter fuerte supo ocupar un lugarcito en mi corazón, a ella que su palabra favorita era NO cuando le pedía su cuenta, gracias a la amistad tan grande que tenemos te ganaste el nombre de mama, sé que no quieres que me vaya para Moa pero aunque la distancia nos separe nuestra amistad nunca desvanecerá esa eres tu ***Karo.***

A mis amigos*, por los buenos momentos que pasamos juntos y los recuerdos que quedaron grabados para siempre, en especial a mi goldis, Yude, mari, gleibis, fito, el flaco,Yuraysi,Bidot, Thaimi, laura, Aylin, Dudi,Viltre,Migdalia. Gracias por dejarme encontrar en ustedes una gran amistad.*

A mis tutores *Yunier y Alexis por su dedicación y apoyo incondicional, en especial a Yunier por ser un ejemplo a seguir como profesional y como persona. Gracias por tus críticas, por exigencia y optimismo, pues como dijera pepilla pónganle corazón a esto, este logro es tuyo también.*

A mi dúo de tesis Jorge, por ser el mejor compañero, gracias por tu comprensión y colaboración en la realización de este trabajo, en este día tan especial para las dos quiero decirte que aprendí a quererte y que sin ti esto no hubiera sido posible.

A mis viejos amigos, Taylor, Nane, Leudis, Yosvanis, Leudis, Blanca, Juliesqui, Yordanis, Javier en fin a todos por compartir conmigo tantos momentos lindos de la vida. Aunque algunos no estén aquí sé que nunca se han olvidado de mí.

Lisandra

AGRADECIMIENTOS

Si me preguntaran: ¿Qué fue lo más difícil de estos cinco años? Sin dudas respondería que este preciso momento, donde tengo que agradecer en tan solo unas líneas a todas aquellas personas que han estado a mi lado haciendo único cada segundo de mi vida.

Durante estos cinco años hemos formado parte de varios grupos de clases, pero ninguno como el primero, el maravilloso grupo 6106, estoy seguro que donde quiera que estemos o cuando hablemos de la UCI, los primeros recuerdos serán aquellos que dirigidos por una incomparable profesora Irina mencionaremos con orgullo. A todos los que empezaron esta larga travesía que hoy culmina y no están a nuestro lado como Jaime, Matos, Camué, Arlety, Diego, Jordan, Rolando, Alexey, Ilinay, Julio, y especialmente Randy; a todos ellos y los que hoy cumplen su sueño, GRACIAS, sin ustedes no hubiera sido lo mismo.

Quisiera agradecer especialmente a una persona que no se cansa de repetir "Jorge la tesis es de ustedes", pero con tanto amor y sacrificio ha hecho parecer que sea de él también. Cada segundo que nos dedicaste, cada documento que revisaste. No nos alcanzan las palabras para agradecerte y decir que eres un ejemplo a seguir, gracias a mi tutor Yunier.

De manera general a todos aquellos de SIGE que fueron capaces de dedicarme su tiempo cuando más lo necesitaba, a mi otro tutor Alexis René, a Glennis, Adisley.

Agradecer además, a ese fabuloso e incomparable equipo de trabajo, con quienes comparto una pasión en común, la electrónica, que sin ellos no hubiese podido ser lo que soy, han sido otra escuela para mí, no se imaginan la satisfacción que siento cuando arreglo algo y digo eso me lo enseñó el Guille o el Joe. Gracias Joe por acogerme y hacerme sentir como si fuera un hijo más, y al Guille que ha sido como el hermano que siempre quise tener. También agradecer a Darling, Adri, Lisset, Adela, Yamila y a todos aquellos que formado parte de mi formación como profesional.

Una vez escuché: "amigos para toda la vida son lo que se hacen en la universidad", quisiera poder responderle y explicarle que conocí a Yerandy, Bidot, Fuma, Erik, Miguel, Leo, Rolo, Raciel, Roberto, Karel, Nelson, Castrillon, Rubert, Brian, Carlos, Tomás, Randy por solo mencionar a algunos, a todos ellos GRACIAS.

Ha sido difícil y si alguien quedó sin mencionar, disculpen, es que son tantas personas que no sabía por dónde empezar, llegue de esta forma el AGRADECIMIENTO general a todos.

Jorge Luis

DEDICATORIA

Este gran triunfo está dedicado a mi madre, mi padre, a Manolo, mis tíos Cari y Gustavo y mis abuelos Pichila y Braulio, a quienes les debo todo lo que tengo y lo que soy en esta vida porque sin ellos nada de esto hubiera sido posible, por ser lo más grande que tengo en el mundo y haberme apoyado mucho estos 5 años que llevo distantes de toda mi familia. Les dedico este trabajo ante todo porque no me han dejado claudicar en el logro de mis sueños. Son mi ejemplo a seguir, mis guías, especialmente por su amor, dedicación, comprensión y ayudarme a lograr este sueño de poder regalarles esta satisfacción de poder verme graduada.

Lisandra

Quisiera dedicar este trabajo a las personas más importantes de mi vida:

A mis padres Cuqui y Jorge Luis que con el mayor esfuerzo del mundo me han educado de forma excepcional, gracias por todo su apoyo, amor y dedicación, no tengo palabras para describirlo. Si el título admitiera más de un nombre, de seguro, ahí estuvieran ustedes.

A mis abuelos Daniel y Aleida quienes me han consentido toda mi vida, con su amor y ternura me han transformaron en el hombre que soy hoy, los amo con todo mi corazón y no me va a alcanzar la vida de retribuirles su amor.

A mi novia Gisselle, quién ha sido el sustento para tantos momentos de tensión, quién ha soportado los cambios de ánimos estando presente cuando más lo necesitaba. Por poner mi corazón en sus manos y haberlo cuidado de forma mágica. Te amo.

Especialmente se lo dedico a mi abuela Nena, que donde quiera que esté, estoy seguro que no existe sentimiento alguno para describir lo que estaría sintiendo. Abu ya soy Ingeniero, te extrañamos.

Jorge Luis

RESUMEN

SIGE es un sistema que permite informatizar la gestión de los procesos estadísticos realizados por la ONEI. Para esto, utiliza plantillas de formularios y encuestas como soporte en la captura de información. Las plantillas de encuestas transcurren por un ciclo de vida dividido en tres etapas: Diseño, Publicación y Captación. En la etapa de Publicación, transitan por dos estados: digitación y archivo. Este proceso de transición se caracteriza por ser totalmente rígido y dependiente. Se desarrolló un módulo de flujos de trabajo para SIGE que garantiza dos procesos fundamentales: Definir y Monitorear flujo de trabajo. Este módulo ofrece la posibilidad, al personal autorizado, de definir un flujo de trabajo y llevar un control minucioso del cumplimiento y comportamiento del mismo, permitiéndole a SIGE ajustarse a diferentes escenarios de negocios logrando realizar el mecanismo de cambios de estados de las encuestas de manera flexible y configurable.

Palabras claves: Definir**,** Flujo de trabajo, Monitorear**,** SIGE.

ABSTRACT

SIGE is a software that allows computerizing the management of statistical processes performed by the ONEI. For this, it uses form and survey templates to support the capture of information. The survey templates run through a life cycle divided into three stages: Design, Publishing and Collection. They go through other two states during the Publishing stage: typing and filing. This transition process is characterized for being quite rigid and user dependent. A workflows module was developed for SIGE for ensuring two fundamental processes: to define and to monitor workflows. This module offers the possibility to the authorized personnel to define a workflow and to keep a rigorous control of its compliance and performance, allowing SIGE to adjust itself to different business scenarios and to perform the mechanism of state changes of the surveys in a flexible and configurable way.

Keywords: Define, Monitor, SIGE, Workflows.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha propiciado que la informática constituya una rama fundamental para el desarrollo de la humanidad, pues es una materia importante que interviene en muchos procesos de la sociedad actual, presente en la gestión integral de cada organización, apoyando la toma de decisiones en cada una de las infraestructuras y elevando la eficacia en los procesos para alcanzar los objetivos y metas propuestas.

Cuba se encuentra inmersa en el proceso de informatización de la sociedad con el propósito de mejorar los procesos que se realizan en diversas instituciones del país. Para ello se han incorporado aplicaciones que gestionen su información, logrando alcanzar una mayor rapidez y exactitud en los procesos de negocios.

La Universidad de las Ciencias Informáticas (UCI) es una universidad docente productiva, en la cual sus estudiantes son capaces de producir software y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación y de esta forma contribuir en el avance económico, así como alcanzar la informatización del país. La infraestructura productiva de la UCI cuenta con varios centros de desarrollo, uno de ellos es el Centro de Tecnologías de Gestión de Datos (DATEC) perteneciente a la facultad seis.

DATEC tiene la misión de crear bienes y servicios informáticos relacionados con la gestión de datos, área del conocimiento que agrupa tanto a los sistemas de información, como a los denominados sistemas de inteligencia empresarial o de negocios, cuyo propósito fundamental es apoyar el proceso de toma de decisiones. Dicho centro ofrece una gama de productos entre los que se encuentra el Sistema Integrado de Gestión Estadística (SIGE), el cual surge a partir de una solicitud realizada por la Oficina Nacional de Estadística e Información (ONEI) como aplicación encargada de informatizar la gestión de sus procesos estadísticos. Actualmente el sistema atraviesa por un proceso de evolución para ajustarse a las necesidades de otros negocios, incluyendo una serie de mejoras entre las que se encuentran: integración de mapas, almacenes de datos, minería de datos, con el objetivo de mejorar su análisis estadístico.

SIGE es un sistema integrado por módulos que colaboran entre sí para darle cumplimiento a los procesos de gestión estadística realizados por la ONEI. Para gestionar dicha información, el sistema garantiza un modelo de persistencia de datos escalable mediante la utilización de plantillas. Estas constituyen un soporte para la captura de información conformada por Encabezado, Cuerpo y Pie de Firma.

El ciclo de vida de estas plantillas, está dividido en tres etapas fundamentales: Diseño, Publicación y Captación. Tanto el Diseño como la Publicación se realizan en el Módulo Gestor de Plantillas (MGP). Primeramente, se selecciona el tipo de plantilla a diseñar (Formulario o Encuesta), en donde se define la estructura que tendrá (cantidad de páginas, validaciones, entre otras). Con la plantilla lista, finaliza la etapa de Diseño cuando el usuario aprueba su Publicación. En esta fase, la plantilla ya puede entrar en el proceso de Captación, llevado a cabo en el Módulo Entrada de Datos (MED), donde el usuario debe ingresar la información necesaria para confeccionar el modelo. Este representa una plantilla con la información asociada y se genera por el sistema una vez concluida la etapa de Captación.

Un formulario en SIGE es una forma de captación de datos estadísticos en forma tabular, compuesta básicamente por indicadores y aspectos, los cuales representan filas y columnas, respectivamente. Además, los indicadores se pueden agrupar visualmente en secciones. Estas filas pueden tener a su vez asociadas validaciones de expresiones aritméticas para realizar procesos estadísticos, donde las entradas siempre son numéricas. Un formulario, partiendo de la etapa de Publicación, atraviesa por tres estados: digitación, validación y archivo.

Por su parte, las encuestas están compuestas por preguntas y estas, a su vez, pueden agruparse visualmente en secciones. Entre los tipos de preguntas disponibles se encuentra el campo tabla, con el cual puede captarse información similar a la soportada por los formularios y además presentarse junto con otros tipos de preguntas. Las encuestas presentan un ciclo de vida guiado por las mismas etapas que los formularios. Partiendo de la etapa de Publicación, transita por dos estados: digitación y archivo.

Este proceso de transición puede definirse como rígido y dependiente. Rígido porque el sistema, en su diseño, tiene preconcebido los diferentes estados por los que atraviesa una encuesta, siendo una configuración inalterable que limita el alcance del mismo, pues las plantillas pueden tener un ciclo de vida enriquecido y adaptable al negocio de gestión de información de los clientes del sistema. Se considera dependiente porque cada transición entre estados se ejecuta únicamente como consecuencia de la acción de los usuarios del sistema, restringiendo cualquier posibilidad de modificarla para que se ejecute automáticamente ante condiciones particulares como pueden ser: valores de preguntas, fechas y eventos definidos por el sistema. Estas características limitan a entidades como la ONEI, principal cliente de SIGE, la posibilidad de ajustarse a diferentes escenarios de un negocio.

A partir del análisis realizado anteriormente, se identifica el siguiente **problema de investigación**: ¿Cómo personalizar el proceso de transición de estados de las encuestas en SIGE? En correspondencia con el problema planteado, el **objeto de estudio** de la presente investigación está centrado en: Los sistemas de gestión de información estadística, enmarcado en el **campo de acción**: Flujos de trabajo para sistemas de gestión de información estadística.

Con el propósito de solucionar el problema planteado se define como **objetivo general** de la investigación: Desarrollar un módulo para la definición y monitoreo de flujos de trabajo para SIGE.

A partir del objetivo general se desglosan los siguientes **objetivos específicos**:

1. Caracterizar los diferentes sistemas que utilizan flujos de trabajo para la construcción del marco teórico.
2. Definir los conceptos, metodología, tecnologías y herramientas necesarias para el desarrollo del Módulo de flujos de trabajo para SIGE.
3. Definir el análisis y diseño del Módulo de flujos de trabajo para SIGE.
4. Implementar las funcionalidades del Módulo de flujos de trabajo para SIGE.
5. Validar el correcto funcionamiento del módulo a partir de la realización de pruebas.

Para desarrollar un plan de trabajo acorde a las necesidades reales del problema a resolver se plantean las siguientes **preguntas de investigación**:

1. ¿Cuáles son las bases teóricas que fundamentan los flujos de trabajo para SIGE para la construcción del marco teórico?
2. ¿Cuáles son las características y capacidades que el Módulo de flujos de trabajo para SIGE debe tener para definir qué hacer y bajo qué circunstancias hacerlo?
3. ¿Cómo obtener una guía legible y comprensible para agregar posibles mejoras al Módulo de flujos de trabajo para SIGE?
4. ¿Cómo traducir el diseño obtenido para el Módulo de flujos de trabajo para SIGE en un producto con capacidad operacional?
5. ¿Cómo valorar el correcto funcionamiento del Módulo de flujos de trabajo para SIGE?

Se definen como **tareas de la investigación:**

1. Búsqueda, revisión y procesamiento de referentes bibliográficos de los sistemas de gestión que utilizan flujo de trabajo existentes en el mundo para la construcción del marco teórico.
2. Caracterización de los sistemas de gestión de información que utilizan flujo de trabajo para entender su funcionamiento, sintaxis y aplicación práctica.
3. Caracterización de la metodología, las herramientas y las tecnologías a utilizar durante el desarrollo del módulo para confeccionar el diseño teórico de la investigación.
4. Identificación de las necesidades funcionales y tecnológicas para definir lo que el Módulo de flujos de trabajo para SIGE debe hacer y bajo qué circunstancias debe hacerlo.
5. Elaboración de los diagramas correspondientes al diseño para construir la línea base de la arquitectura del Módulo de flujos de trabajo para SIGE.
6. Confección del modelo de diseño del Módulo de flujos de trabajo para SIGE a partir de las funcionalidades identificadas para guiar el proceso de implementación.
7. Definición de los patrones de arquitectura y de diseño para determinar la estructura del Módulo de flujos de trabajo para SIGE.
8. Implementación del Módulo de flujos de trabajo para SIGE para dar cumplimiento a los requisitos definidos para el mismo.
9. Diseño de los casos de pruebas definidos a partir de los casos de uso para determinar el comportamiento del Módulo de flujos de trabajo para SIGE bajo distintos escenarios.
10. Realización de pruebas de software para comprobar el correcto funcionamiento del módulo.

Para el desarrollo de la investigación se utilizaron diferentes métodos de investigación agrupados en teóricos y empíricos. De los métodos teóricos se utiliza el analítico-sintético y la modelación. Como método empírico se realizó el análisis de documentos.

El método **analítico-sintético** permitió el análisis de la bibliografía utilizada, además de garantizar la fiabilidad de la información empleada. Permitió obtener, describir y resumir elementos importantes que intervienen en el proceso de flujo de trabajo para el desarrollo del módulo.

Se evidencia el uso del método de **modelación** a partir de la creación de diagramas que permitieron explicar las relaciones y propiedades fundamentales de la aplicación.

El **análisis de documentos** permitió definir los diferentes requerimientos del sistema y recopilar información referente a las características y funcionamiento de los sistemas que utilizan flujos de trabajo.

Para una mejor comprensión, el presente documento posee una estructura que se divide en 4 capítulos:

Capítulo 1: Fundamentación teórica: En el capítulo quedan sustentadas las bases teóricas de la investigación. Se abordan los conceptos fundamentales relacionados con el objeto de estudio del trabajo, se realiza un análisis de las herramientas, las tecnologías y la metodología a utilizar para dar solución al problema planteado.

Capítulo 2: Características del sistema: En este capítulo se profundiza sobre las funcionalidades que debe tener el sistema. Se presentan los requisitos funcionales y no funcionales, además se definen los casos de uso del sistema y la descripción de cada uno de ellos, así como los actores que interactúan con el mismo.

Capítulo 3: Diseño de la aplicación: En este capítulo se describe el diseño de la solución propuesta. Se representa a través del modelo de dominio los conceptos significativos del negocio, también se modela el diagrama de clases del diseño y el diagrama de despliegue. Se define la arquitectura y los patrones de diseño que se utilizan a la hora de diseñar el módulo.

Capítulo 4: Implementación y prueba del Módulo de flujos de trabajo para SIGE: En este capítulo se confecciona el modelo de implementación para la construcción del módulo propuesto. Para evaluar el correcto funcionamiento del módulo se describe el método de prueba y se elaboran los casos de pruebas.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se exponen los conceptos fundamentales asociados al objeto de estudio del trabajo, que justifican el correcto entendimiento del problema que se trata y de la solución que se propone. Se realiza el estudio del estado del arte, que recoge los diferentes sistemas de gestión que utilizan flujo de trabajo y se especifica la metodología, las herramientas y las tecnologías seleccionadas para dar solución al problema en cuestión.

1.2 Conceptos asociados al dominio del problema

Sistemas de gestión

Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización, ayuda a lograr los objetivos de la organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado (Cintra, 2013).

Herramienta de gestión sistemática y transparente que permite dirigir, controlar y evaluar el desempeño institucional en términos de calidad y satisfacción social en la prestación de los servicios (Rivera, 2012).

De acuerdo con los conceptos abordados anteriormente los sistemas de gestión permiten obtener, controlar y evaluar la información de una organización en aras de garantizar que la misma realice todas las tareas necesarias para alcanzar los objetivos propuestos. Lo que conlleva al apoyo a la toma de decisiones.

Sistemas de Información Estadístico (SIE)

El SIE permite producir y difundir información estadística de una forma distribuida, homogénea, intuitiva y asistida, sin necesidad de conocer herramientas específicas, lo que redunda por un lado en un incremento de la cantidad y calidad de la información con independencia del Departamento u Organismo que ejecute la operación, y por otro, en agilizar la planificación de los Departamentos u Organismos Gestores (Castaño, 2000).

Permiten, a través de la Web, realizar análisis espacial y temporal de la Información estadística de manera inmediata y comparada con acceso a datos oficiales de las principales fuentes (Cortés, 2010).

De acuerdo con los conceptos expuestos se asume que un SIE permite obtener, procesar, almacenar y distribuir información contribuyendo al proceso de toma de decisiones y control en una organización. En este tipo de sistema debe estar presente la estrecha relación entre desarrolladores y clientes con el objetivo de mantener la concordancia entre las necesidades a satisfacer y su satisfacción real. En la actualidad, estos tipos de sistemas son muy utilizados por empresas que manejan datos estadísticos, por ejemplo la Oficina Nacional de Estadísticas e Información (ONEI) es una institución gubernamental adscrita al Ministerio de Economía y Planificación cuyo objetivo es organizar, dirigir, controlar y regular la actividad de la estadística en el ámbito nacional. Para la gestión de la información estadística se apoya en el Sistema Integrado de Gestión Estadística.

Sistema Integrado de Gestión Estadística (SIGE)

SIGE es un sistema distribuido con arquitectura modular, desarrollado con tecnologías libres, con el cual se garantiza la automatización de los procesos estadísticos, así como su concepción genérica, de adaptarse a diferentes negocios que cumplan con los estándares establecidos por la ONEI. El sistema responde a los procesos fundamentales identificados en el marco de gestión de la información estadística. Tales procesos comprenden la creación de formularios y encuestas (MGP[1]), la captura de la información a partir del diseño de los mismos (MED[2]), la restricción de acceso a las funciones del sistema (MSEG[3]), la caracterización de unidades de observación (MGC[4]). Ver Figura 1: Arquitectura del Sistema Integrado de Gestión Estadística.

SIGE permite el diseño de formularios y encuestas para la captura de información estadística asociada a las empresas, contribuyendo de esta manera al proceso de toma de decisiones, logrando establecer una valoración del comportamiento del progreso de una empresa con respecto a las metas establecidas.

Las funcionalidades de SIGE están agrupadas en módulos, los cuales colaboran entre sí para darle cumplimiento a los procesos de gestión estadística mencionados anteriormente. Actualmente SIGE está

[1] Módulo Generador de Plantillas

[2] Módulo de Entrada de Datos

[3] Módulo de Seguridad

[4] Módulo Gestor de la Configuración

compuesto por módulos, de los cuales en el desarrollo de la presente investigación se hará énfasis en 3 de ellos: Módulo Gestor de Plantillas, Módulo Entrada de Datos y Módulo de Seguridad. A continuación se detallan las principales características de cada uno de ellos.

- ✓ **Módulo Gestor de Plantillas (MGP):** Permite el diseño de plantillas como soportes para la captura de la información estadística en las unidades de observación. Estas plantillas se presentan de dos maneras: formularios (solo pueden captar datos numéricos) y encuestas (formada por secciones y diferentes tipos de preguntas) logrando así una gestión de información estadístico-descriptiva a todos los niveles de las unidades de observación.
- ✓ **Módulo Entrada de Datos (MED):** Permite la captura de la información estadística teniendo como entrada los productos finales del Módulo Gestor de Plantillas, es decir, plantillas de formularios o encuestas en las que se definan un subconjunto de datos a captar. Este módulo es el responsable de validar la información capturada a partir de reglas definidas durante el proceso de diseño de plantillas de formularios y/o encuestas, lo cual constituye un mecanismo de confiabilidad para la toma de decisiones luego de procesar la información (Zaldivar, 2013)
- ✓ **Módulo de Seguridad (MSEG):** Es el módulo que se encarga de la seguridad de la aplicación permitiendo restringir el acceso a la ejecución de las acciones del sistema. El área de trabajo del módulo está estructurada por las opciones que brinda el propio módulo, las cuales son: Gestión de Usuarios (Usuarios), Gestión de Roles y Perfiles (Roles y Perfiles), Autenticación LDAP[5] y Trazas (Columbie, 2013).

SIGE es un sistema que garantiza la seguridad, el diseño de plantillas de formularios y de encuestas, y estos a su vez posibilitan la captura de información, entre otras funcionalidades. Actualmente el proceso de transición de estados por lo que atraviesan las encuestas durante su ciclo de vida en el sistema, no garantiza que el usuario pueda generar un conjunto de estados acorde a las necesidades del negocio, y cada acción a ejecutar depende totalmente de un usuario. Para darle solución a este problema se hace necesario la utilización de una máquina de estados.

[5] Lightweight Directory Access Protocol (en español Protocolo Ligero/Simplificado de Acceso a Directorios)

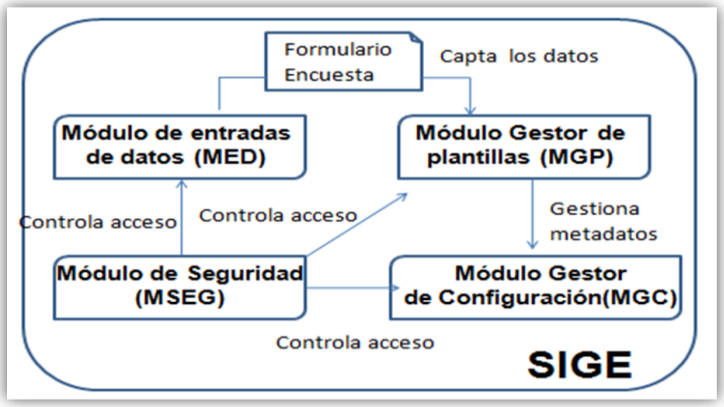


Fig. 1 Vista global a nivel de módulos del Sistema Integrado de Gestión Estadística (SIGE)

Máquinas de estados

Una máquina de estados es un comportamiento que especifica las secuencias de estados por las que pasa un objeto a lo largo de su vida en respuesta a eventos (Booch, 2006).

Es un gráfico de estados y transiciones que muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro. Es decir: es un modelo de todas las posibles historias de vida de un objeto de una clase (Contreras, 2011).

A partir del concepto abordado anteriormente se concluye que una máquina de estados permite determinar el comportamiento de un objeto en dependencia del estado en que se encuentre, así como describir la secuencia a seguir. La misma está estructurada por estados y transiciones, eventos que disparan una transición y actividades que son la respuesta a una transición.

Una clasificación de las máquinas de estados es:

- ✓ **Máquinas de estados finitos:** sirven para realizar procesos bien definidos en un tiempo discreto. Reciben una entrada, hacen un proceso (pueden cambiar a otro estado o permanecer en el mismo) y nos entregan una salida (Orozco, 2008).

Para el desarrollo de la solución se utilizará una máquina de estados finitos, que se caracterizan por tener números finitos de estados y transiciones, lográndose realizar un proceso bien definido en un tiempo moderado partiendo desde un estado inicial hacia un estado final de aceptación.

Las máquinas de estados son usadas generalmente con el objetivo de definir la secuencia de un flujo de trabajo.

Flujo de trabajo

Se define como flujo de trabajo (Workflow en inglés) a un sistema informático que organiza y controla tareas, recursos y reglas necesarias para completar el proceso de negocio. El uso de flujo de trabajo permite monitorear el estado actual de los datos, así como observar el cumplimiento del mismo. La implantación de un sistema de flujo de trabajo aporta numerosos beneficios como:

- ✓ Ahorro de tiempo, mejora de la productividad y eficiencia debido a la automatización de los procesos de negocios.
- ✓ Mejora en los procesos, mayor flexibilidad de acuerdo con las necesidades existentes.
- ✓ Mejor atención y servicio al cliente, lográndose una mayor coherencia de los procesos de negocio, lo que da lugar a una mayor previsibilidad en los niveles de respuestas a los clientes.
- ✓ Optimización de la circulación de información interna con clientes y proveedores.
- ✓ Correcta integración con sistemas de información actuales tales como bases de datos (Pixelwarex, 2012).

De acuerdo a lo planteado anteriormente se puede concluir que: flujo de trabajo es la automatización del flujo de un proceso de negocio. Desencadena de manera secuencial y ordenada un conjunto de decisiones, actividades y tareas para conseguir un resultado, de manera que este satisfaga plenamente los requerimientos del cliente al que va dirigido y permita monitorear el cumplimiento de los procesos que desarrolla una determinada entidad en menos espacio de tiempo y sin la dependencia total de un determinado usuario.

En un flujo de trabajo, las actividades pueden ser ejecutadas por personas o por funciones del sistema, el mismo permite describir el orden de ejecución y la dependencia de las relaciones así como su duración. Con el fin de implementar un sistema de flujo de trabajo es necesario encontrar un medio adecuado para el diseño y modelado, por ejemplo las Redes de Petri.

Redes de Petri

Se le llama Redes de Petri a un lenguaje formal y gráfico, adecuado para modelar sistemas con concurrencia e intercambio de recursos. Es una generalización de la teoría de autómatas, de tal manera

que el concepto de eventos que ocurren, puede expresarse al mismo tiempo. Este lenguaje de modelado contiene los siguientes objetos básicos:

Lugares: son inactivos y análogos a las bandejas de entrada en un sistema basado en la oficina. Se muestran como círculos en un diagrama de redes de Petri. Cada red de Petri tiene un lugar de inicio y un lugar final, pero cuenta con cualquier número de lugares intermedios.

Transiciones: son activos y representan tareas a realizar. Se muestran como rectángulos en un diagrama de redes de Petri.

Arcos: se unen a un solo lugar para una sola transición. Se muestran como líneas de conexión. Un arco hacia adentro va de un lugar a una transición y un arco hacia el exterior va de una transición a un lugar.

Fichas: representan el estado actual de un proceso de flujo de trabajo. Se muestran como puntos negros dentro de lugares en un diagrama. Un lugar puede contener cero o más fichas en cualquier momento en el tiempo.

En una red de Petri los triggers[6] pueden ser ejecutados de 4 maneras:

- ✓ **Sistema:** La tarea es disparada de forma automática.
- ✓ **Manual:** La tarea es disparada por la acción de un usuario.
- ✓ **Temporal:** La tarea se dispara en un tiempo programado.
- ✓ **Evento:** La tarea es disparada mediante un evento externo como un mensaje.

En el estudio realizado sobre dicho lenguaje se hizo énfasis en aquellos aspectos que permitieron realizar el desarrollo de la solución propuesta exitosamente, logrando cumplir los objetivos planteados. Uno de los aspectos tratados fue la forma de ejecución de los triggers (transiciones), y la manera de monitorearlo. Las redes de Petri representan una teoría ampliamente utilizada para implementar herramientas de modelado que permitan automatizar y simplificar procesos.

1.3 Soluciones existentes

Existen herramientas que permiten modelar flujos de trabajo a partir de los modelos de negocio diseñados. Las herramientas BonitaSoft, OpenERP y PIPE fueron las candidatas, pues se caracterizan por tener licencia libre, es decir, no requieren coste alguno para ser usadas, cumpliendo de esta manera con la política del centro de migrar a software libre. A continuación se realizará un estudio detallado

[6] Disparadores

enfatizando en la manera en que cada una de estas soluciones define y maneja el concepto de flujo de trabajo, tomando como base aquellos aspectos que permitirán sustentar los conocimientos para el desarrollo del Módulo de flujos de trabajo para SIGE.

BonitaSoft:

BonitaSoft es un sistema que permite definir los flujos de procesos orientados al usuario, siendo el motor de Bonita el que mantiene la lógica del flujo de trabajo de manera independiente al modelo de negocio de la organización, por lo que el usuario simplemente se encarga de definir el flujo de proceso. Existen versiones de Bonita tanto para el sistema operativo Windows como para Linux. Utilizar BonitaSoft trae consigo los siguientes beneficios:

- ✓ Incentivar la eficiencia de equipos de trabajo fomentando la colaboración. Un equipo puede visualizar las tareas concurrentes y cada individuo en tiempo real puede conocer el estado de un proceso permitiendo tener estadísticas a nivel de proceso e instancia de un proceso, tiempos de atención de cada tarea y otras métricas en el mismo ámbito.
- ✓ Reducir los costos y riesgos de automatización de procesos persona- persona y sistema- sistema. Los procesos pueden correr en organizaciones que funcionen distantes geográficamente, permitiéndole enlazarlos y aprovecharlos de manera eficiente.
- ✓ Manejar eficientemente situaciones inesperadas, pues permite redefinir de manera dinámica y segura un proceso de suerte en el que se pueden incluir eventos que no fueron previamente identificados.
- ✓ Toma beneficios de algunas características provistas por el servidor de aplicaciones JEE (Java Enterprise Edition) como pueden ser el uso de transacciones, autenticación basada en roles y ciclo de vida de aplicaciones, así como la conexión con sistemas externos.

Este software está orientado a la creación y modificación de los modelos de procesos de negocios sobre la red. Exporta las definiciones de los flujos de trabajo en extensión (*.XPDL) para su posterior modelación en la web y ofrece una paleta para la construcción de flujos de trabajo mediante la cual es posible garantizar la mayor cantidad de patrones de diseño de flujos de trabajo para la integración y orquestación de procesos de negocio de manera eficiente y fácil. Posee además un motor de flujo de trabajo BPM[7], que es intuitivo y fácil de configurar y usar (Herrera & Kris, 2008).

[7] Gestión de Procesos de Negocio (en inglés: *Business Process Management*)

OpenERP[8]

OpenERP (conocido anteriormente como TinyERP, y ahora conocido como Odoo) es un Sistema de Planificación de Recursos Empresariales integrado, de código abierto, gratuito y multiplataforma, actualmente producido por la empresa belga Odoo S.A. OpenERP define un flujo de trabajo como un artefacto técnico para gestionar un conjunto de "cosas que hacer", además se utiliza para estructurar, y gestionar los ciclos de vida de objetos, y documentos. El flujo de trabajo es una forma de nivel superior para organizar, automatizar y simplificar los procesos en cada una de las áreas inherentes a cualquier modelo de negocio. Más específicamente, un flujo de trabajo es un grafo dirigido donde los nodos se llaman "actividades" y los arcos se llaman "transiciones", elementos fundamentales que componen el mismo, los cuales sirvieron de ejemplo para el diseño a realizar por el Módulo de flujos de trabajo para SIGE.

- ✓ Las actividades definen el trabajo que se debe hacer dentro del servidor OpenERP, como cambiar el estado de algunos registros, o el envío de correos electrónicos.
- ✓ Las transiciones controlan cómo el flujo de trabajo progresa de una actividad a otra (OpenERP, 2012).

En un sistema OpenERP se le puede asociar flujos de trabajo a cualquier objeto y son enteramente personalizables, es decir permiten de esta manera definirlo. Por ejemplo, se pueden poner condiciones y señales que desencadenan a las transiciones, por lo que el comportamiento del mismo depende de las acciones del usuario (un ejemplo se evidencia al hacer clic en un botón), además se producen cambios en los registros.

PIPE

PIPE (Platform-Independent Petri Net Editor) es una herramienta de código abierto, multiplataforma, desarrollada en Java para asegurar la independencia con la plataforma, proporcionando una interfaz de usuario fácil de usar que permite crear, guardar y cargar redes de Petri incluyendo las estocásticas generalizadas. Esta herramienta se basa en el modelado de redes de Petri, como lenguaje formal para la definición de flujos de trabajo, posibilita exportar los diagramas en formato PNG. La misma garantiza un componente gráfico intuitivo para la simulación y modelado del comportamiento y rendimiento de los sistemas (Charalambous, 2010).

[8] Sistema de Planificación de Recursos Empresariales (en inglés: *Enterprise Resource Planning*)

Después de realizar un estudio previo referente a los sistemas antes mencionados, se tomó de cada uno, aquellos aspectos que facilitarán el desarrollo de la solución propuesta. Del sistema BonitaSoft se analizó cómo el mismo mantiene la separación de la lógica del flujo de trabajo y la del modelo de negocio, por ejemplo: actualmente SIGE incluye un proceso de transición de estados a la hora de diseñar una plantilla de encuesta (Edición, Publicada y Captada) y otro cuando va a ser captado un modelo de encuesta (Digitación y Archivo), este proceso se mantendrá inalterable, por otro lado el módulo permitirá definir el proceso de transición de estados de las plantillas de encuestas sin que se vea afectado el modelo de negocio anteriormente descrito.

En el caso de OpenERP el estudio se enfocó en la simplicidad conque el mismo implementa el flujo de trabajo debido a la estructura de los componentes que modela (Nodos y Transiciones), por ejemplo los **nodos** (estados o actividades) pueden ser de tipo inicial, intermedio o final; las **transiciones** incluyen su tipología (automática o manual) y las precondiciones a ejecutar, no siendo el caso de PIPE el cual necesita de un componente **arista** (arco) donde se incluye la tipología. PIPE es una plataforma que se basa en el modelado de Redes de Petri de la cual se enfatizó en los tipos de transiciones automáticas, manuales, temporales o eventos, además de la estructura de base de datos que propone y cómo realizar el proceso de monitoreo.

1.4 Metodologías, tecnologías y herramientas empleadas en la solución

En el proceso de desarrollo del software es fundamental la correcta elección de las herramientas, tecnologías y metodología que mejor se adapten y mayores facilidades brinden para darle solución al problema planteado. A continuación se define qué metodología de desarrollo de software, tecnologías y herramientas se utilizarán durante el proceso de desarrollo de la solución propuesta, logrando definir las actividades a realizar.

1.4.1 Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a producir software (Villegas, 2010). En los últimos tiempos la industria del software ha evolucionado de tal manera, que ha sido necesario desarrollar metodologías, para sostener la demanda de producción de sistemas cada vez mayores en complejidad y tamaño, logrando su construcción de forma óptima y eficiente. Para el desarrollo de este trabajo se decidió optar por la metodología del Proceso Unificado Abierto (OpenUp).

OpenUp

La metodología de desarrollo de software OpenUp generalmente es apropiada para proyectos pequeños de corto plazo, al igual que RUP mantiene las principales características de ser orientada por casos de uso. Es una metodología ágil que genera poca documentación. Es un proceso mínimo que incluye solo el contenido fundamental completo porque se manifiesta como proceso entero para construir un sistema, extensible porque puede utilizarse como base para agregar o adaptar más procesos (Balduino, 2011).

Se decidió utilizar OpenUP como metodología para el desarrollo del módulo propuesto, pues el tiempo de desarrollo estimado puede adaptarse a 8 meses y el equipo de trabajo es pequeño, lo que permite disminuir las probabilidades de fracaso e incrementar las probabilidades de éxito. Se detectan errores en fechas tempranas del desarrollo a través de un ciclo iterativo, evitando la elaboración de documentos, diagramas e iteraciones innecesarias. Por política del centro se define esta metodología pues el módulo a desarrollar forma parte del proyecto SIGE perteneciente al Centro DATEC, lo cual se define en el documento de arquitectura del proyecto.

1.4.2 Lenguaje de modelado

UML

Open UP utiliza el Lenguaje Unificado de Modelado (UML[9] por sus siglas en inglés) el cual es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos obtenidos durante el ciclo de vida completo del proceso de desarrollo de software. Permite modelar procesos de negocio y funciones del sistema. UML 2.1 es un lenguaje de modelado, no es una guía para realizar el análisis y diseño orientado a objetos, por lo que no se utiliza para determinar qué se debe hacer en primer lugar o cómo diseñar el sistema sino que ayuda a visualizar el diseño y hacerlo más comprensible (Larman,1999), por lo que se considera un lenguaje visual fácil de entender por cualquier persona del equipo de desarrollo y permite toda la documentación del ciclo de vida completo del sistema, facilitando el desarrollo o mejora de futuras versiones.

[9] *UML por sus siglas en inglés (Unified Modeling Language)*

En el desarrollo del Módulo de flujos de trabajo para SIGE se hace uso de este lenguaje para construir muchos de los artefactos que necesita OpenUp como metodología seleccionada, entre ellos se encuentran los diagramas: clases del diseño, Casos de Uso del Sistema, de componente y despliegue. Dentro de las herramientas CASE[10] que utilizan UML como lenguaje de modelado para generar artefactos de un sistema de software, se encuentra Visual Paradigm.

1.4.3 Herramienta CASE

Visual Paradigm

Visual Paradigm para UML 8.0, es una herramienta para el modelado con soporte multiplataforma, proporciona excelentes facilidades de interoperabilidad con otras aplicaciones, permitiendo crear los diagramas correspondientes durante todo el ciclo de vida de desarrollo de software. Esta herramienta brinda un entorno de desarrollo amigable, pues permite organizar el trabajo según la comodidad de cada usuario. Admite la integración con diferentes entornos de desarrollo, además permite que múltiples usuarios trabajen a la vez sobre el mismo proyecto. Proporciona soportes a varios lenguajes en generación de código e ingeniería inversa a través de plataformas Java, C++, entre otras. Tiene apoyo adicional en cuanto a generación automática de artefactos. Es un poderoso generador de documentación y reporte UML, PDF, HTML y otros (Pressman, 2002).

A partir de las características abordadas anteriormente resulta ventajosa la utilización de esta herramienta en el desarrollo del módulo, pues es gratuita, fácil de instalar, utilizar y actualizar. Se encuentra disponible para distribuciones del sistema Operativo Linux, lo cual sigue la política de migración a software libre trazada por el departamento. Se selecciona esta herramienta para realizar el modelado de todos los diagramas necesarios para el desarrollo de las funcionalidades propuestas en el módulo, pues la misma forma parte de las tecnologías y herramientas propuestas por el departamento de DATEC en el desarrollo del sistema SIGE.

1.4.4 Sistema gestor de Base de Datos

Un Sistema Gestor de Bases de Datos (SGBD), es una interfaz para que el usuario se pueda comunicar dinámicamente con la base de datos. Estos sistemas manejan de forma clara la información de modo que su uso es bastante transparente, disminuyendo el tiempo de desarrollo y aumentando la calidad del

[10] CASE, por sus siglas en inglés (Computer Aided Software Engineering)

sistema. Entre los principales SGBD que existen en la actualidad a nivel mundial se encuentra el privativo Oracle y los libre MySQL y PostgreSQL.

PostgreSQL 9.1

Es un SGBD distribuido bajo la licencia de Berkeley Software Distribution (BSD) el cual dispone libremente su código fuente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos, por lo que un fallo en uno de los procesos no afectará al resto y el sistema continuará funcionando, garantizándose la estabilidad del mismo (Martínez, 2010). La conectividad, velocidad y seguridad además de su rica interacción con el lenguaje a utilizar para el desarrollo del módulo propuesto, hace que sea compatible y de mayor beneficio su uso.

La utilización de este sistema resulta ventajosa para el desarrollo del Módulo de flujos de trabajo para SIGE, pues el mismo se caracteriza por ser multiplataforma, es decir disponible en la mayoría de los sistemas operativos como Linux, UNIX, Windows. Es altamente escalable debido a la gran cantidad de datos que puede manejar de manera clara, sencilla y ordenada. Se escoge este sistema gestor de base de datos porque el módulo a desarrollar forma parte del proyecto SIGE perteneciente al Centro DATEC, el cual se define en el documento de arquitectura del proyecto.

PgAdmin III como interfaz de administración para PostgreSQL

Es una aplicación gráfica para gestionar el SGBD PostgreSQL, con licencia Open Source (Código Abierto). Está diseñada para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas de una manera más fácil, debido a que se le ha incorporado una nueva interfaz gráfica, la cual resulta de gran ayuda a la hora de gestionar las bases de datos. Esta interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración.

Es una herramienta de código abierto y multiplataforma respaldada por una amplia comunidad de desarrolladores que encaminan sus esfuerzos al perfeccionamiento de la misma. Establece su conexión con el servidor a través del protocolo TCP/IP (pgAdmin, 2011).

1.4.5 Lenguaje de programación

Un lenguaje de programación puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos respectivamente permitiendo a un programador especificar

de manera precisa sobre qué datos una computadora debe operar, si estos serán almacenados o transmitidos y qué acciones debe realizar bajo una determinada circunstancia (Muller, 2007).

PHP 5.3

PHP es un lenguaje de programación del lado del servidor. El mismo consiste en un lenguaje interpretado de alto nivel, embebido en páginas HTML y ejecutado generalmente tomando como entrada el código PHP y creando páginas web como salida. Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos (Asenjo, 2012).

PHP es un lenguaje multiplataforma. Permite la conexión a la mayoría de motores de bases de datos que se utilizan actualmente. Es libre, permite aplicar técnicas de Programación Orientada a Objetos (POO) y posee una amplia documentación en internet, brindando una gran variedad de ejemplos y ayudas. Debido a que SIGE está desarrollado bajo este lenguaje es conveniente implementar el módulo utilizando este lenguaje para ganar en flexibilidad e integridad entre las aplicaciones.

JavaScript

Es un lenguaje de programación del lado del cliente, pues es el navegador el que soporta la carga de procesamiento, permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Es interpretado, no requiere compilación y es soportado por la mayoría de los navegadores como Internet Explorer, Opera y Mozilla Firefox (Pérez, 2009).

En otras palabras, los programas escritos con JavaScript pueden probarse directamente en cualquier navegador sin necesidad de procesos intermedios. El mismo maneja objetos dentro de una página web y sobre ese objeto se definen diferentes eventos, por lo que se considera un lenguaje basado en objetos y guiado por eventos. Se selecciona este lenguaje de programación del lado del cliente para el desarrollo del Módulo de flujos de trabajo para SIGE, pues es multiplataforma, por lo que puede ser desplegado en casi todos los sistemas operativos, principalmente en Linux, pues cumple con la política del centro de migración a software libre. El mismo está basado en objetos, permitiendo emplear herencia, aspecto típico en la Programación Orientada a Objetos (POO).

1.4.6 Marco de trabajo

En el desarrollo del software, un marco de trabajo o framework en su traducción al inglés, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de

software concretos, en la cual otro proyecto de software puede ser fácilmente organizado y desarrollado. Típicamente, incluye soporte de programas, bibliotecas y un lenguaje interpretado que ayuda a desarrollar y unir los diferentes componentes de un proyecto (Riehle, 2000). Para el desarrollo de la propuesta de solución se utilizará Symfony y Ext.JS.

Symfony 1.1.7

Symfony es un marco de trabajo que permite optimizar el desarrollo de aplicaciones web y está desarrollado completamente con PHP5, siendo sencillo de usar pero lo suficientemente flexible como para adaptarse a casos más complejos. Es una herramienta fácil de instalar y configurar en la mayoría de los sistemas Windows y Unix estándares. Es compatible con la mayoría de gestores de bases de datos, además separa la lógica del negocio, la lógica de la aplicación y la presentación de la aplicación web (incorpora el patrón Modelo Vista Controlador MVC), lo que mantiene el código organizado y permite a la aplicación evolucionar fácilmente en el tiempo (Potencier, 2010).

Se utiliza Symfony como marco de trabajo pues el mismo trabaja con el patrón de arquitectura Modelo Vista Controlador y el ORM[11] Propel para el mapeo de objetos a bases de datos. Es un framework sencillo para cualquier programador independientemente que sea principiante o un experto en PHP. Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5. Además se selecciona este framework debido a que cumple con las políticas definidas para el proyecto en el documento de arquitectura, haciendo uso de las ventajas que ofrecen las tecnologías libres.

ExtJS 3.4

Constituye un framework de presentación JavaScript del lado del cliente para el desarrollo de aplicaciones web con interfaces muy similares a la de una aplicación de escritorio. Ofrece múltiples opciones para el trabajo con validaciones y manejo de errores en el cliente. Es una herramienta multiplataforma (Cryself Villa, 2008). Usar ExtJS permite tener además los siguientes beneficios:

- ✓ Existe un balance entre Cliente – Servidor: la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.

[11] Mapeo Objeto-Relacional (por su nombre en inglés: *Object-Relational Mapping)*

- ✓ Comunicación asíncrona: en este tipo de aplicación el motor renderizado puede comunicarse con el servidor sin necesidad de estar sujeto a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente lo note.
- ✓ Eficiencia de la red: el tráfico de red puede disminuir al permitir que la aplicación elija qué información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

La utilización de este framework resulta ventajosa para el desarrollo del Módulo de flujos de trabajo para SIGE, pues posibilita distribuir correctamente la carga de procesamiento permitiendo que el servidor pueda atender más clientes al mismo tiempo, además de mantener aplicaciones muy grandes debido a que tiene su código bien organizado en el marco de trabajo para el desarrollo del sistema del lado del cliente utilizado en la implementación de SIGE.

1.4.7 Entorno Integrado de Desarrollo

Un Entorno Integrado de Desarrollo (en inglés Integrated Development Environment o IDE) proporciona herramientas que dan soporte al proceso de desarrollo del software. Puede estar pensado para su utilización con un único lenguaje de programación o bien puede dar cavidad a varios de estos. Para el desarrollo de la solución se seleccionó como IDE Netbeans.

Netbeans 7.1

La plataforma NetBeans es un Entorno Integrado de Desarrollo (IDE) libre y gratuito sin restricciones de uso, cuya función es proporcionar un marco de trabajo fácil y amigable. Diseñado principalmente para el lenguaje de programación Java y para lenguajes dinámicos como PHP, JavaScript. Permite editar programas en java, compilarlos, ejecutarlos, depurarlos y construir rápidamente la interfaz gráfica de una aplicación. Se utiliza para crear aplicaciones de escritorio, empresariales y web (Netbeans, 2013).

Para la implementación del módulo se seleccionó como IDE NetBeans en su versión 7.1. El mismo es libre y gratuito sin restricciones de uso, proporciona un marco de trabajo fácil y amigable. Ofrece una amplia documentación y recursos de capacitación. Se utiliza para crear aplicaciones de escritorio, empresariales y web. NetBeans es fácil de instalar y se puede ejecutar tanto en Windows, Linux, Mac. También se tuvo en cuenta que es una de las herramientas usadas por el centro DATEC.

1.4.8 Bibliotecas

Las bibliotecas son un conjunto de subprogramas utilizados para desarrollar software. Las mismas contienen código y datos que proporcionan servicios a programas independientes pasando a formar parte de estos.

Dracula: Biblioteca Gráfica 0.0.3

Dracula es un conjunto de herramientas para construir gráficos interactivos de diseño, se basa en JavaScript y SVG[12] permitiendo ser menos pesado en el servidor y lograr una buena compatibilidad en los navegadores. El código está liberado bajo la licencia MIT[13] por lo que puede utilizarse y extenderse libremente. Permite definir la estructura que tendrá el gráfico, es decir construir los nodos y bordes, proporcionar diferentes tamaños y formas, colores, posición del texto y los algoritmos de posicionamiento según desee el usuario para adaptarlo a las necesidades de lo que se esté desarrollando (Silva, 2011). Se selecciona la misma para definir las estructuras que tendrán los grafos en cuanto a sus componentes, o sea se especifican cada uno de los nodos (estados) y entre cuáles estados va a existir una arista (transición). Dándole paso a la biblioteca Raphaël la cual se encarga del renderizado del grafo.

Raphaël: Biblioteca JavaScript 1.3.1

Raphaël es una pequeña biblioteca JavaScript que simplifica el trabajo con gráficos vectoriales en la web. Permite crear gráficos e imágenes, rotar y ampliar vistas, entre otras cosas. Raphaël utiliza la recomendación del W3C[14] para SVG y VML[15] como base para la creación de gráficos. Esto significa que cada objeto gráfico se crea también como un DOM[16], por lo que puede asociar controladores de eventos de JavaScript o modificar más adelante. El objetivo de Raphaël es proporcionar un adaptador para realizar el dibujo de gráficos vectoriales compatibles con múltiples navegadores de manera fácil. Actualmente

[12] Gráfico Vectorial Escalable (Scalable Vector Graphics)

[13] Instituto Tecnológico de Massachusetts (Massachusetts Institute of Technology)

[14] World Wide Web Consortium, abreviado W3C

[15] Lenguaje de Marcado de Vectores (Vector Markup Language)

[16] Modelo de Objetos del Documento (Document Object Model)

soporta: Firefox 3.0 + o superior, Safari 3.0 + o superior, Chrome 5.0 + o superior, Opera 9.5 + o superior, Internet Explorer 6.0 o superior (Muller, 2012).

Existen otras herramientas para el trabajo con gráficos en JavaScript tales como Higcharts y Charts JS las cuales incluyen animaciones y gráficos interactivos, pero su alcance se ve limitado al trabajo con vectores (creación de grafos dirigidos) más bien su uso se desarrolla sobre gráficos de barra, poligonales, de pastel, entre otros, siendo de mayor utilidad las ventajas que brinda Raphaël por su capacidad en el procesamiento y visualización de gráficos para el desarrollo del módulo, permitiendo visualizar en forma de grafo el comportamiento de un determinado flujo de trabajo. Otra de las ventajas proporcionadas por la unión de estas dos bibliotecas es la forma en que se renderiza el grafo, facilitándole a los usuarios del sistema posicionar cada elemento donde desee dentro del panel en que se encuentra.

1.5 Conclusiones parciales

A partir de la investigación reflejada en este capítulo se concluye que los sistemas estudiados no establecen una solución factible para aplicarlos a SIGE, pues no se ajustan a las políticas que establece el centro DATEC. Por lo que se hace necesario desarrollar una solución basada en los principales aspectos de las soluciones estudiadas. Teniendo en cuenta lo expuesto durante este capítulo, se decide desarrollar el Módulo de flujos de trabajo para SIGE, para ello se determinó utilizar la metodología, tecnologías y herramientas definidas por la dirección del proyecto para lograr la estandarización de todos sus módulos teniendo en cuenta los principios de soberanía tecnológica que defiende el país. La inclusión de la metodología OpenUP permitió asegurar una guía para regir todo el proceso de desarrollo de la solución propuesta, así como proponer las actividades a realizar para generar un conjunto de artefactos necesarios, que en combinación con las tecnologías y herramientas seleccionadas permitirán lograr los objetivos propuestos, partiendo de los conocimientos adquiridos sobre los diferentes conceptos abordados.

Capítulo 2: Características del sistema

2.1 Introducción

Para darle continuidad al proceso de desarrollo del módulo teniendo en cuenta la metodología seleccionada es necesario definir sus características partiendo del propósito por el cual es desarrollado y las necesidades que debe satisfacer. Una vez comprendidas las características y la lógica de funcionamiento de la solución propuesta, se definen los casos de uso del sistema, los cuales son representados mediante el diagrama de casos de uso así como la descripción de los mismos. A continuación en este capítulo se enuncian elementos importantes referentes a los aspectos planteados para lograr un mejor entendimiento de cómo tiene que ser y qué debe hacer el Módulo de flujos de trabajo para SIGE.

2.2 Solución propuesta

Como solución a la problemática planteada, se propone desarrollar un módulo que garantice dos procesos fundamentales: Definir y Monitorear flujo de trabajo. La integración de este módulo al sistema no sustituye el funcionamiento del proceso de transición de estados por los cuales atraviesa una encuesta actualmente, sino que permitirá definir un flujo que se ajuste a las necesidades de diferentes procesos de negocios. Ambos procesos se ejecutarán paralelamente y de forma independiente. Por lo que el sistema no va a depender del módulo para realizar la gestión de sus procesos estadísticos, sino que lo incorpora como una mejora que le permita ajustarse a diferentes escenarios de negocios. La solución que se propone no será aplicada en los formularios, dado que el proceso de transición de estados que presentan los mismos no sufrirá cambios, pues su uso está enfocado a la gestión de datos estadísticos, constituyendo la forma oficial, definida y aceptada para obtener datos en la ONEI.

A continuación se describen una serie de pasos lógicos a tener en cuenta para la definición de un flujo de trabajo:

1. Se selecciona una plantilla de encuesta publicada para definir el flujo de trabajo que se asociará a la misma.
2. Se adicionan los diferentes estados por los cuales transitará en dependencia de la información que se maneje en el negocio. Estos estados se clasifican en 3 tipos (inicial, intermedio y final), y en

dependencia del estado en que se encuentre, permitirá crear permisos para restringir la visibilidad que tendrá un rol a una página específica de la plantilla. Para que un flujo de trabajo tenga sentido al menos deben existir dos estados, haciendo obligatorio dicha restricción.

3. Se adicionan las transiciones entre los diferentes estados creados, ellas pueden ser: de tipo **usuario** que se van a ejecutar bajo la acción de un usuario definido en la misma y las de tipo **terminar** o **automática** donde se define una condición sobre una pregunta numérica la cual se ejecutará en el sistema de forma automática. Para que un flujo de trabajo tenga una secuencia entre sus estados se hace necesario crear al menos n – 1 transiciones, donde n es la cantidad de estados creados, además de tener en cuenta que todos los estados deben formar parte de al menos una transición.

Una vez completado satisfactoriamente el proceso anterior, el flujo estará listo para ser almacenado en el sistema. Este flujo de trabajo puede monitorearse, aunque para realizar dicho proceso, la plantilla asociada al mismo debe captarse previamente (modelo de encuesta). Después de captada se puede visualizar una amplia información del estado actual en que se encuentra dicho modelo, así como cambiar este estado si se tiene los privilegios definidos en las transiciones de tipo usuario. Para representar visualmente el flujo de trabajo asociado al modelo se utiliza un visor gráfico proporcionando una visión clara del mismo.

2.3 Modelo de dominio

El modelo de dominio es una representación gráfica de los conceptos significativos que están asociados al proceso que se desea informatizar (Pressman, 2002). Este modelo se crea para documentar el vocabulario del sistema, que ayuda a comprender los conceptos que utilizan los usuarios, con los que trabajan y con los que deberá trabajar la aplicación. En la figura 2 se representa el modelo de dominio realizado.

2.3.1 Definición de clases del modelo del dominio

SIGE: Sistema utilizado por la ONEI para gestionar la información estadística.

MGP: Módulo encargado de gestionar el diseño de plantillas de formularios y encuestas.

MED: Módulo encargado de gestionar la digitación de formularios y encuestas.

Plantillas Publicadas: Plantillas que una vez terminado su diseño completamente, están actas para soportar el ingreso de datos.

Formularios: Son plantillas que forman una estructura en forma de tabla, están compuestas por secciones, indicadores y aspectos.

Encuesta: Son plantillas que contienen preguntas, las cuales se agrupan por secciones para facilitar la captación de información.

Modelo: Es la plantilla que se genera en el proceso de captación de datos donde el usuario ingresa la información asociada a dicha plantilla.

Digitación: Es donde se ingresan los datos necesarios asociados a la plantilla.

Archivo: El modelo o encuesta está lista para ser guardada en el sistema, es decir para archivarse.

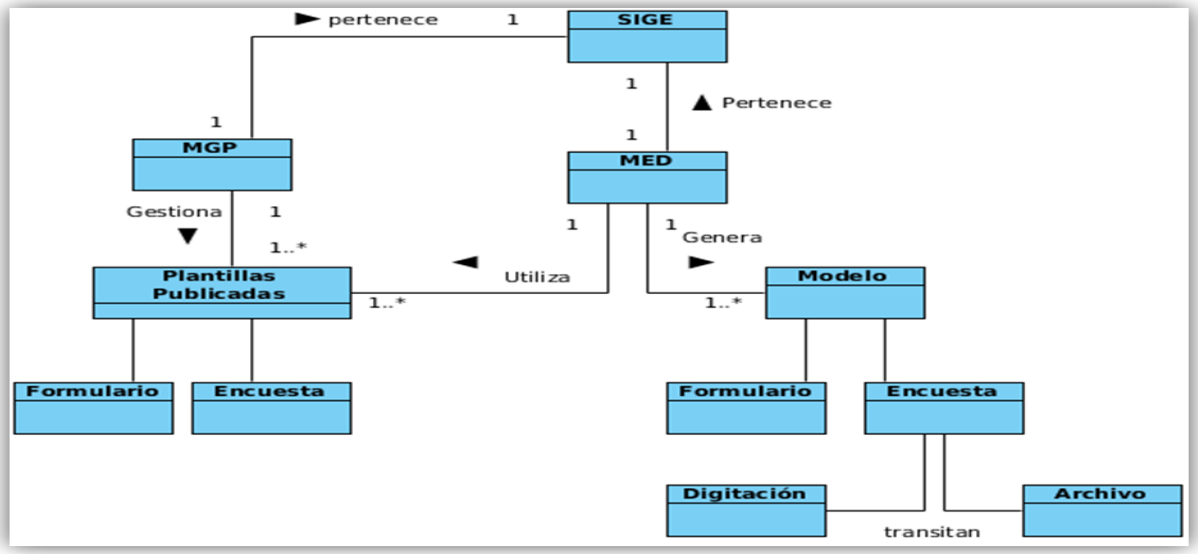


Fig. 2 Modelo de dominio del Módulo de flujos de trabajo para SIGE.

2.3.2 Descripción del flujo del diagrama de modelo de dominio

SIGE es un sistema modular que facilita la gestión de los procesos estadísticos realizados por la ONEI. Dentro de los módulos que conforman el sistema se encuentran el MGP, MED y MSEG. El MGP permite gestionar el diseño de las plantillas, de tipo formulario o encuesta. Una vez definida la estructura de la plantilla, esta pasa a la etapa de Publicada y quedará lista para ser captada. Durante la etapa de Publicación, las plantillas serán usadas en el proceso de captación, llevado a cabo por el Módulo Entrada de datos (MED), donde el usuario ingresa los datos necesarios para confeccionar el modelo (Digitación).

Luego de finalizar el modelo estará listo para ser almacenado en el sistema (Archivo). Este proceso es realizado únicamente como consecuencia de la acción de un usuario.

2.4 Requisitos del sistema

Los requisitos de un sistema representan los servicios que ha de ofrecer, así como las funcionalidades asociadas a su funcionamiento (Sommerville, 2005). Los requisitos se pueden clasificar en dos grupos: requisitos funcionales y no funcionales, es decir lo que el software debe hacer y bajo qué circunstancias hacerlo. A continuación se muestran los requisitos funcionales y los no funcionales de la solución propuesta.

2.4.1 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe ser capaz de realizar (Diaz, 2005). Estos muestran las funcionalidades que deben satisfacerse para garantizar el cumplimiento de los objetivos planteados. A continuación se describen los requisitos funcionales definidos para el desarrollo del módulo a implementar:

RF1: Adicionar nuevo flujo de trabajo.

Descripción: El módulo debe ser capaz de permitir al usuario adicionar un nuevo flujo de trabajo.

Entradas: Se introducen los datos referentes a un flujo de trabajo: nombre, descripción, selección de plantilla.

Salidas: El sistema registra el nuevo flujo de trabajo y actualiza la lista de flujos de trabajo existentes.

RF2: Listar flujo de trabajo.

Descripción: El módulo debe ser capaz de mostrar el listado de flujos de trabajo existentes.

Entradas: No se ingresan datos.

Salidas: El sistema muestra la lista de flujos de trabajo que están registrados en el sistema.

RF3: Eliminar flujo de trabajo.

Descripción: El módulo debe ser capaz de eliminar un flujo de trabajo del listado.

Entradas: Se selecciona el flujo de trabajo que se desea eliminar.

Salidas: Se elimina el flujo de trabajo seleccionado del sistema.

RF4: Modificar flujo de trabajo.

Descripción: El módulo debe permitirle al usuario modificar un flujo previamente creado.

Entradas: Se selecciona el flujo de trabajo que se desee modificar.

Salidas: Se actualizan los cambios y se muestran en el listado de flujos existentes.

RF5: Buscar flujo de trabajo.

Descripción: El módulo debe permitirle al usuario realizar la búsqueda de flujos de trabajo previamente creados ingresando un nombre.

Entradas: Nombre del flujo de trabajo.

Salidas: Se selecciona el flujo de trabajo que cumple con el parámetro de entrada y se visualizan los datos.

RF6: Adicionar nuevo Estado.

Descripción: El módulo debe ser capaz de permitir al usuario crear los estados necesarios para establecer el flujo de trabajo.

Entradas: Se ingresan los datos referentes al estado: nombre, tipo, adicionar permisos.

Salidas: El sistema registra el nuevo estado y actualiza la lista existente.

RF7: Listar Estado.

Descripción: El módulo debe ser capaz de mostrar el listado de los estados existentes.

Entradas: No se ingresan datos.

Salidas: El sistema muestra la lista de los estados que están registrados en el sistema.

RF8: Eliminar Estado.

Descripción: El módulo debe ser capaz de eliminar un estado previamente registrado en el sistema.

Entradas: Seleccionar el estado que se desea eliminar de la lista.

Salidas: Se elimina el estado seleccionado.

RF9: Modificar Estado.

Descripción: El módulo debe permitirle al usuario modificar un estado previamente creado.

Entradas: Seleccionar el estado que se desea modificar.

Salidas: Se actualizan los cambios y se muestran en el listado de estados existentes.

RF10: Adicionar nueva Transición.

Descripción: El módulo debe ser capaz de permitir al usuario crear una nueva transición.

Entradas: Se introducen los datos referentes a la transición: Selección de un estado origen, selección de un estado destino, selección del tipo de transición (Usuario ó terminar).

Salidas: El sistema registra la nueva transición y actualiza la lista.

RF11: Listar Transición.

Descripción: El módulo debe ser capaz de listar todas las transiciones creadas.

Entradas: No se ingresan datos.

Salidas: Se muestran todas las transiciones creadas.

RF12: Eliminar Transición.

Descripción: El módulo debe ser capaz de eliminar una transición previamente registrada.

Entradas: Seleccionar la transición que se desea eliminar.

Salidas: Se elimina la transición seleccionada de la lista de transiciones.

RF13: Modificar Transición.

Descripción: El módulo debe permitirle al usuario modificar una transición previamente creada.

Entradas: Transición que se desea modificar.

Salidas: Se actualizan los cambios y se muestran en el listado de transiciones existentes.

RF14: Adicionar nuevo Permiso.

Descripción: El módulo debe ser capaz de permitir al usuario adicionar una restricción de visibilidad a un rol para una página específica de la plantilla de encuesta publicada.

Entradas: Se selecciona el rol, el tipo de visibilidad y la página, para poder asignar un permiso.

Salidas: El sistema registra el nuevo permiso y actualiza el listado de permisos.

RF15: Listar Permiso.

Descripción: El módulo debe ser capaz de listar todos los permisos para restringir la visibilidad que tendrá el rol a una página específica de la plantilla de encuesta publicada.

Entradas: No se ingresan datos.

Salidas: Se muestran todos los permisos creados.

RF16: Eliminar Permiso.

Descripción: El módulo debe ser capaz de eliminar un permiso previamente registrado.

Entradas: Seleccionar el permiso que se desea eliminar.

Salidas: Se elimina el permiso seleccionado de la lista.

RF17: Modificar Permiso.

Descripción: El módulo debe ser capaz de hacer modificaciones a un permiso existente.

Entradas: Permiso que se desea modificar.

Salidas: Se actualizan los cambios y se muestran en el listado de permisos existentes.

RF18: Listar encuestas que tienen asociado un flujo.

Descripción: El módulo debe ser capaz de listar todas las encuestas que tienen asociado un flujo.

Entradas: No se ingresan datos.

Salidas: Se muestran todos los encuestas existentes que tengan asociado un flujo.

RF19: Mostrar estado actual de una encuesta.

Descripción: Luego de haberse seleccionado una encuesta el módulo debe ser capaz de mostrar el estado actual en que se encuentra.

Entradas: Encuesta seleccionada.

Salidas: Se muestra el estado actual en que se encuentra la encuesta.

RF20: Cambiar estado.

Descripción: Se abre una ventana donde te permite escoger hacia cual estado destino deseas ir si tiene los privilegios necesarios.

Entradas: Estado actual de la encuesta que se seleccionó.

Salidas: Se realiza el cambio hacia el otro estado en dependencia del rol que desempeñe el usuario que esté autenticado en el sistema el usuario.

RF21: Listar permisos del estado actual.

Descripción: El módulo debe ser capaz de listar todos los permisos asociados a ese estado actual.

Entradas: Flujo seleccionado.

Salidas: Se listan todos los permisos del estado actual en que se encuentra el flujo.

RF22: Renderizar grafo de la encuesta seleccionada.

Descripción: El módulo debe ser capaz de graficar los estados y transiciones después de haber seleccionado un flujo de trabajo.

Entradas: Flujo de trabajo seleccionado.

Salidas: Se dibujan todos los componentes del flujo de forma tal que queda representado los estados por los que pasa y transiciones a ejecutar.

2.4.2 Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el software debe de tener. Estas propiedades permitirán garantizar en el software usabilidad, confiabilidad y rapidez (Koch, 2002). Si el mismo cumple con todas las funcionalidades requeridas, entonces las propiedades no funcionales, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Los requerimientos no funcionales que debe cumplir el sistema para garantizar su correcto funcionamiento son los siguientes:

RNF1: Hardware

Cliente

- ✓ Procesador, con un mínimo de 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 256MB.

Servidor

- ✓ Procesador, con un mínimo 2.66 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 1GB.
- ✓ Disco Duro con al menos 10Gb de capacidad para instalar el sistema.

RNF2: Software.

Un servidor en el que se instale SIGE con los siguientes elementos:

- ✓ Sistema operativo GNU/Linux Ubuntu 14.04.
- ✓ Lenguaje de programación PHP y bibliotecas versión 5.
- ✓ Sistemas Gestor de Bases de Datos PostgreSQL, versión 9.1.
- ✓ En las estaciones clientes Mozilla Firefox 6.0 hasta la 37.

Restricciones del Diseño e Implementación:

Para la implementación del sistema se requiere del uso de las siguientes herramientas:

RFN3: Lenguaje y marco de trabajo (framework) para el desarrollo del módulo del lado del servidor. El módulo deberá ser implementado en el lenguaje de programación PHP en su versión 5.3. Como framework de desarrollo se utilizará Symfony 1.1.7 el cual propone una arquitectura en tres capas: modelo, vista y controlador.

RFN4: Lenguaje y marco de trabajo (framework) para el desarrollo del módulo del lado del cliente.

Para el desarrollo del módulo se utilizará el lenguaje JavaScript y como framework ExtJS pues es una biblioteca de JavaScript que permite el diseño de interfaces visuales.

RFN5: Se empleará como entorno integrado de desarrollo IDE NetBeans 7.1 y el sistema gestor de base de datos PostgreSQL 9.1.

Confidencialidad

RFN6: El módulo sólo será consultado por los usuarios que tengan acceso al sistema. La información sólo será visualizada para aquellos usuarios que posean los privilegios suficientes; restringiéndose la ejecución de acciones a usuarios sin credenciales que intenten acceder a las mismas.

Interfaz

RFN7: Las interfaces de usuario serán diseñadas a modo de aplicaciones RIA (Rich Internet Application) lo que permite a los usuarios contar con aplicaciones web con una experiencia de usuario similar a la de las aplicaciones de escritorios. Para lograr este fin se usará la librería JavaScript ExtJS el cual conjuga una serie de componentes visuales que proveen funcionalidades, que ayudan al diseño de este tipo de aplicaciones WEB con apariencia de escritorio.

2.5 Diagrama de Caso de Uso del Sistema

Los diagramas de Casos de Uso del Sistema (CUS), son los encargados de recoger el comportamiento de un sistema desde el punto de vista de los usuarios. De esta manera luego de tener especificados los requisitos funcionales e identificado los actores del sistema, se puede dar paso a la realización del diagrama, el cual muestra la relación que existe entre el usuario final y las funcionalidades (Jacobson, 2000). La siguiente figura muestra el diagrama de CUS, el cual representa la relación que existe entre el actor en este caso el especialista y las diferentes funcionalidades que debe cumplir la aplicación.

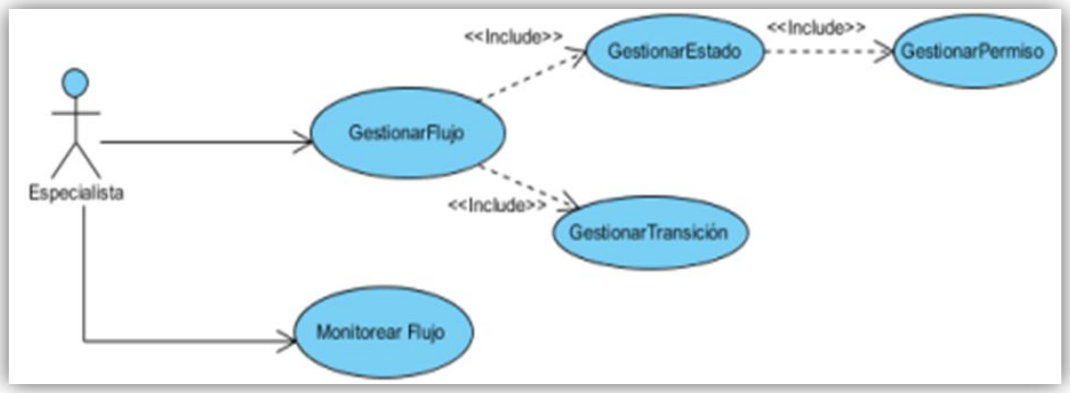


Fig. 3 Diagrama de Caso de Uso del Sistema.

2.6 Patrones de Caso de Uso

Entre los diferentes patrones de caso de uso se encuentra el CRUD Completo, que propone formar un caso de uso a partir de los requisitos funcionales relacionados con las acciones de insertar, listar o mostrar, modificar, y eliminar una determinada información. Haciendo uso de este, las funcionalidades identificadas para el desarrollo de la aplicación quedaron agrupadas en cinco casos de uso: Gestionar Flujo, Gestionar Estado, Gestionar Permisos, Gestionar Transición y Monitorear Flujo.

2.7 Descripciones textuales de los casos de uso del sistema

Una vez identificados los casos de uso del sistema, se realiza una descripción de lo que el sistema debe hacer cuando interactúa con sus actores. En ella se detalla paso a paso el flujo de eventos que ocurren en

dicha interacción y se especifican las acciones alternas de cada uno de los escenarios. A continuación se muestra la descripción textual correspondiente a los casos de uso definidos.

Tabla 1: Descripción del Caso de Uso Gestionar flujo

Objetivo	Adicionar un nuevo flujo al sistema, listar los flujos existentes en el sistema, modificar los datos de los flujos del sistema, eliminar un flujo del sistema y buscar un determinado flujo.
Actores	Especialista
Resumen	El caso de uso (CU) se inicia cuando el especialista va a realizar algunas de las siguientes operaciones: **Adicionar:** cuando el especialista desea crear un flujo, este introduce los datos necesarios para su construcción y el sistema registra los mismos, finalizando así el CU. **Buscar:** el especialista introduce el nombre por el cual desea realizar la búsqueda**,** el sistema muestra los resultados para dicha búsqueda, finalizando así el CU. **Modificar:** el especialista selecciona el flujo a modificar, el sistema muestra los datos correspondientes, el especialista los modifica, finalizando así el caso de uso. **Eliminar:** el especialista selecciona el flujo e indica eliminar, el sistema lo elimina, finalizando así el CU. **Listar:** El sistema lista los flujos según se vayan creando y los muestra en la pantalla principal en el listado de los flujos existentes con los datos**,** finalizando así el CU.
Complejidad	Alta
Prioridad	Critico
Precondiciones	Que el especialista esté autenticado en el sistema y tenga los permisos necesarios para poder realizar algunas de estas acciones.
Poscondiciones	En dependencia de la acción realizada por el especialista: Se crea un nuevo flujo de trabajo. Se modifica un flujo existente. Se elimina un flujo de trabajo. Se busca un flujo de trabajo Se lista un flujo de trabajo para su posterior uso.
Flujo de eventos	
Flujo básico < Gestionar flujo de trabajo >	

	Actor	**Sistema**

	1. Entra al sistema y selecciona la Opción definir flujo.	2. Muestra una interfaz con los aspectos necesarios para definir un flujo.
	3. Elige que opción va a realizar: a) Adicionar flujo. b) Modificar flujo. c) Eliminar flujo. *d)* Buscar un flujo. *e)* Listar flujo.	4. En dependencia de la acción solicitada por el especialista, muestra la interfaz correspondiente: a) Adicionar nuevo flujo: Ir a la sección **"Adicionar flujo".** b) Modificar flujo existente: Ir a la sección **"Modificar flujo".** c) Eliminar flujo existente: Ir a la sección **"Eliminar flujo".** d) Buscar flujo existente: Ir a la sección **"Buscar flujo".** e) Listar un flujo: Ir a la sección **"Listar flujo".**
Sección a: "Adicionar nuevo flujo"		
Flujo básico < Adicionar flujo >		
	Actor	**Sistema**
	1. Selecciona la opción nuevo flujo de trabajo.	2. Muestra una pantalla permitiendo introducir los datos necesario para crear el flujo: ✓ Nombre del flujo ✓ Descripción ✓ Selección de plantilla.
	3. Introduce los datos del flujo: ✓ Nombre del flujo ✓ Descripción ✓ Selecciona una plantilla a la cual se le realizará el flujo. Y ejecuta el botón **Siguiente.**	4. Verifica que los campos requeridos hayan sido completados.
		5. El sistema verifica que no exista un flujo con el mismo nombre.
		6. El sistema verifica que no exista una plantilla con un flujo asociado.

		7. Muestra una ventana que permite gestionar los datos de un estado. Ver **CU Gestionar estado.**
	8. Ejecuta el botón **Siguiente**	9. Muestra una ventana que permite gestionar los datos de las transiciones. Ver **CU Gestionar transición.**
	10. Si todos los datos están correctos y han sido creados los nodos y la transición, el especialista ejecuta el botón **Guardar**.	
		11. El sistema muestra un mensaje Operación exitosa y actualiza el listado de flujos existentes.
Flujos alternos		
Nº Evento <Campos obligatorios vacios>		
	Actor	**Sistema**
		4a) Si el campo nombre quedó vacío el sistema emite un mensaje: “El campo nombre es obligatorio“ y se indica en color rojo, vuele al paso 3. 4b) Sino se selecciona una plantilla el sistema emite un mensaje: “Debe seleccionar una plantilla“, vuele al paso 3.
Nº Evento <Flujo existente>		
	Actor	**Sistema**
		5a) Si el flujo ya existe en el sistema emite un mensaje “Ya existe un flujo con ese nombre”. Vuelve al paso 3.
Nº Evento <Flujo asociado a una plantilla>		
	Actor	**Sistema**
		6a) Emite un mensaje “Ya existe un flujo asociado a esa plantilla”. Vuelve al paso 3
Sección b: “Modificar flujo”		
Flujo básico < Modificar flujo >		
	Actor	**Sistema**

	1. Busca el flujo que desea modificar.	2. Muestra el flujo en caso de que exista en dependencia del nombre que lo identifique.
	3. El especialista selecciona el flujo que desee modificar del listado de flujos y presiona el botón “Modificar”.	4. Muestra una interfaz con los datos del flujo, brindando la posibilidad de modificar los mismos y permitiendo: ✓ **Guardar** donde se procederá a la modificación de los datos del flujo. ✓ **Cerrar ventana** donde se procederá a cancelar la operación.
	5. El especialista modifica los campos: ✓ Nombre del flujo ✓ Descripción ✓ Selecciona una plantilla a la cual se le realizará el flujo. y presiona el botón **“Guardar”.**	6. Valida las modificaciones realizadas.
		7. El sistema verifica que no exista un flujo con el mismo nombre.
		8. El sistema verifica que no exista una plantilla con un flujo asociado.
		9. Actualiza los datos del flujo, y muestra un mensaje **Modificación completada**, terminando de esta forma el caso de uso.
Flujos alternos		
Nº Evento <Cancelar Modificación>		
	Actor	**Sistema**
	5 a) El especialista presiona el botón **“Cancelar”.**	6a) El sistema elimina todos los datos de entrada y regresa a la pantalla anterior, terminando de esta forma el caso de uso. Vuelve al paso 3.
Nº Evento <Campos obligatorios vacíos>		
	Actor	**Sistema**
		6a) Si el campo nombre quedó vacío el sistema

		emite un mensaje: “El campo nombre es obligatorio“ y se indica en color rojo, vuele al paso 5. 6c) Sino se selecciona una plantilla el sistema emite un mensaje: “Debe seleccionar una plantilla “, vuele al paso 5.
Nº Evento <Flujo existente>		
	Actor	**Sistema**
		7a) Emite un mensaje “Ya existe un flujo con ese nombre”. Vuelve al paso 5.
Nº Evento <Flujo asociado a una plantilla>		
	Actor	**Sistema**
		8a) Emite un mensaje “Ya existe un flujo asociado a esa plantilla”. Vuelve al paso 5.
Sección c: “Eliminar flujo”.		
Flujo básico < Eliminar flujo >		
	Actor	**Sistema**
	1. Selecciona el flujo que desea eliminar del listado de flujos y presiona el botón **“Eliminar”.**	2. Muestra el mensaje de advertencia “¿Estas seguro que deseas eliminar el flujo de trabajo?“, permitiendo: ✓ **Aceptar (Si)** donde se procederá a la eliminación del flujo seleccionado. ✓ **Cancelar (No)** donde se procederá a cancelar la operación.
	3. Presiona el botón **“Aceptar”.**	4. Elimina el flujo del sistema, y actualiza el listado, terminando de esta forma el caso de uso.
Flujos alternos		
Nº Evento <Cancelar Eliminación>		
Actor		**Sistema**
3a) El especialista presiona el botón “Cancelar”.		4a) El sistema regresa a la pantalla anterior, terminando de esta forma el caso de uso.
Sección d: “Buscar flujo”		

Flujo básico < Buscar flujo >		
	Actor	**Sistema**
	1. El especialista provee los datos por los cuales puede buscar un flujo: ✓ Nombre	2. El sistema busca todos los flujos que coincidan con el criterio especificado.
		3. El sistema muestra un listado con todos los flujos resultantes de la búsqueda.
	4. El especialista selecciona el flujo del cual se desea visualizar sus datos.	5. El sistema visualiza los datos del flujo seleccionado.
Flujos alternos		
Nº Evento < No existe flujo que coincida con el criterio especificado >		
Actor		**Sistema**
		3a) El sistema muestra una lista vacía.
Sección e: "Listar flujo"		
Flujo básico < Listar flujo >		
	Actor	**Sistema**
		El sistema muestra en la pantalla principal un listado de los flujos existentes con los datos: ✓ Nombre ✓ Descripción ✓ Número ✓ Subnúmero Ofreciendo la posibilidad de encontrar el flujo que se desee permitiendo de esta forma navegar por el listado obtenido. Terminando de esta forma el caso de uso.

2.8 Conclusiones parciales

Durante el desarrollo de este capítulo se realizaron y cumplieron las actividades que propone la metodología OpenUP, definiéndose 22 requisitos funcionales para garantizar el cumplimiento de los

objetivos propuestos y las restricciones correspondientes (7) con las cuales debe cumplir el Módulo de flujos de trabajo para SIGE. Se utilizó el patrón CRUD para estructurar y organizar los casos de uso, dando paso a la confección del diagrama de casos de uso del sistema y la descripción detallada de cada uno.

Capítulo 3: Diseño de aplicación

3.1 Introducción

Una vez comprendidas las características y la lógica de funcionamiento de la solución propuesta, teniendo en cuenta la metodología seleccionada se prosigue con el Diseño, donde se representan mediante diagramas las relaciones de clases y sus responsabilidades bajo la arquitectura definida. La arquitectura define un diseño y asegura la interacción entre los elementos del software para cubrir todos los objetivos y restricciones del mismo.

3.2 Modelo de Diseño

En la elaboración y confección de un software es importante realizar previamente un correcto diseño, pues en dependencia de la eficacia con que se ha diseñado anteriormente quedarán fomentadas las bases para facilitar la implementación del mismo. Un Modelo de Diseño es básicamente un modelo físico que proporcionará la estructura y la forma que tendrá el sistema que se está construyendo.

3.2.1 Diagramas de clases del diseño

Los diagramas de clases son utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema, y además se obtiene como resultado del refinamiento del modelo conceptual (Jacobson, 2000).

El Módulo de flujos de trabajo para SIGE posee un diseño basado en el patrón MVC, el cual es implementado por el marco de trabajo Symfony1.1.7 descrito anteriormente en el Capítulo 2. Ver figura 4.

3.3 Patrones

Existe un amplio repertorio de principios generales y expresiones que guían en la construcción del software, dentro de estos se encuentran los patrones. “El patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas” (Larman, 1999). Para la realización del diseño se tendrá en cuenta un conjunto de patrones, que constituyen una guía para resolver problemas comunes en programación que generalmente están presentes en el diseño de un programa. Cada patrón explica cómo resolver un determinado problema, bajo determinadas circunstancias, facilitando la creación de sistemas fáciles de mantener,

entender, reutilizar y extender. Los siguientes patrones son los que se evidencian en el desarrollo de la solución propuesta por la presente investigación.

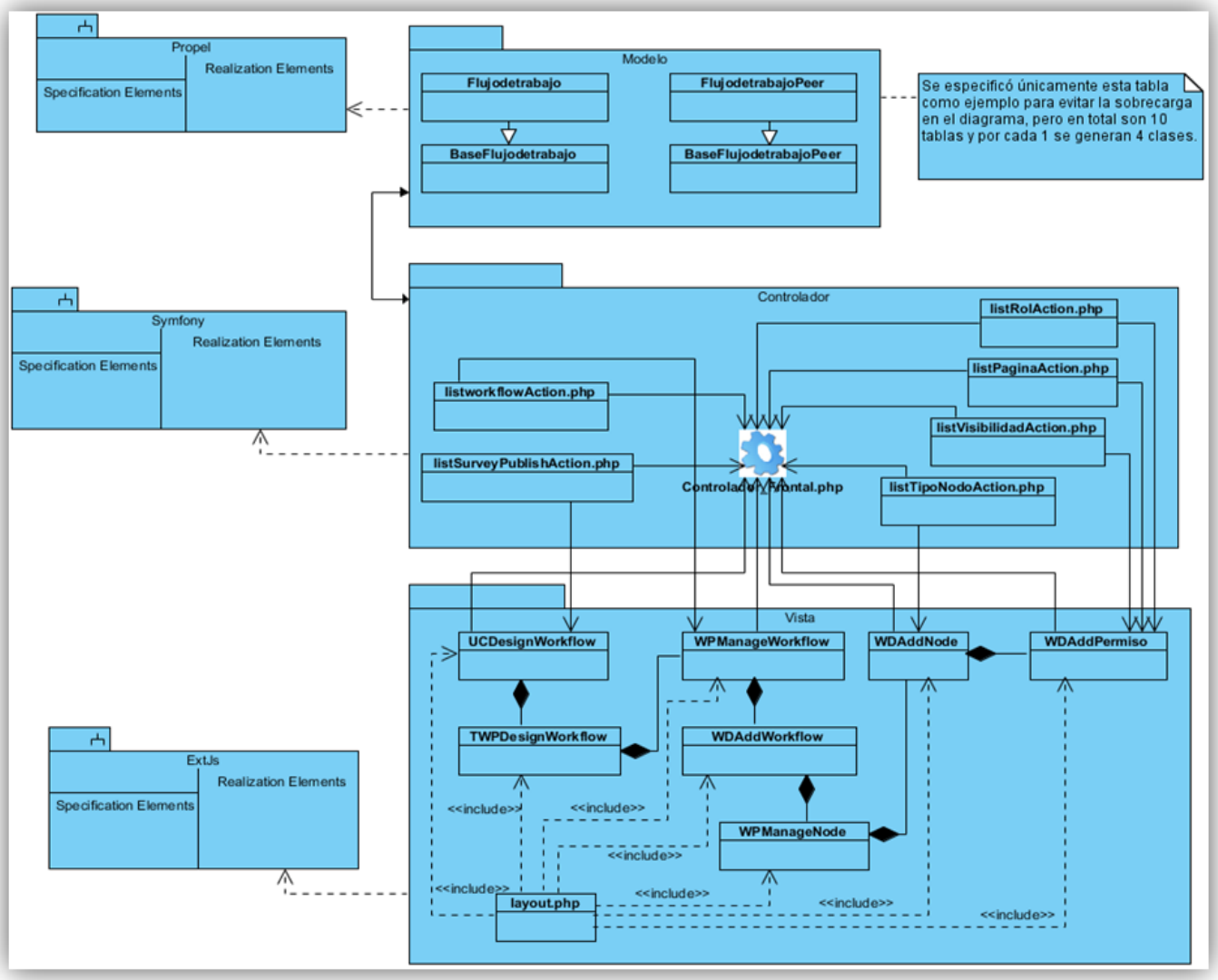


Fig. 4 Diagrama de clases del diseño del CU Gestionar Estado

3.3.1 Patrón de Arquitectura

Para el desarrollo del modelo de diseño se utilizó el patrón de arquitectura Modelo-Vista-Controlador (MVC). Pues permite el desarrollo de una arquitectura consistente, reutilizable y fácil de mantener, el mismo establece una independencia entre el modelo y la vista. Esta separación permite hacer cambios en una parte de la aplicación sin que las demás se vean afectadas y en un reducido espacio de tiempo. Además se emplea este patrón porque el marco de trabajo Symfony 1.1.7, toma lo mejor de esta arquitectura y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo (Potencier, 2010). Es un bloque de código que realiza llamadas al modelo para obtener los datos y se los envía a la vista para que los muestre al usuario. Ver figura 5. El mismo se representa a través de tres elementos fundamentales: el modelo, la vista y el controlador.

El **Modelo** representa la información específica con la que trabaja la aplicación, es decir, su lógica de negocio. La base de datos pertenece a esta capa.

La **Vista** transforma el modelo en una interfaz visual que permite al usuario interactuar con ella.

El **Controlador** responde a eventos, usualmente a acciones del usuario, e invoca cambios en el modelo y probablemente en la vista (Potencier, 2010).

Cuando el usuario realiza una petición al sistema, internamente sucede lo siguiente: Symfony 1.1.7 ejecuta el Controlador asociado a dicha petición. Este Controlador solicita al Modelo los datos que se necesitan de la base de datos. Con los datos devueltos por el Modelo, el Controlador solicita a la Vista que cree una página mediante una plantilla y que inserte los datos del Modelo. Al finalizar este Controlador entrega al servidor la página creada por la Vista.

3.2.2 Patrones GRASP

Symfony está concebido de tal manera que obliga el uso de diferentes patrones de diseño. Los patrones de diseño empleados en la solución pertenecen al conjunto de patrones GRASP **(**Patrones Generales de Software para Asignar Responsabilidades), la utilización de estos patrones ha ayudado a refinar el diseño y a asignar las responsabilidades de las distintas clases de diseño, haciéndolas sencillas y reutilizables (Larman, 1999). Los que se evidencian en el desarrollo de la solución propuesta por la presente investigación son:

Fig. 5 Comportamiento de la arquitectura Modelo-Vista-Controlador.

Controlador: define quién deberá encargarse de atender la petición generada por el usuario (Larman, 1999). Funciona como intermediario entre la interfaz principal y el algoritmo que la implementa, de tal forma que es el que recibe los datos del usuario y los envía a las distintas clases según el método llamado. En la solución al utilizar la clase `controlador_frontal.php` para procesar los pedidos del usuario y realizar los cambios en el modelo y la vista queda evidenciado este patrón. Ver figura 6.

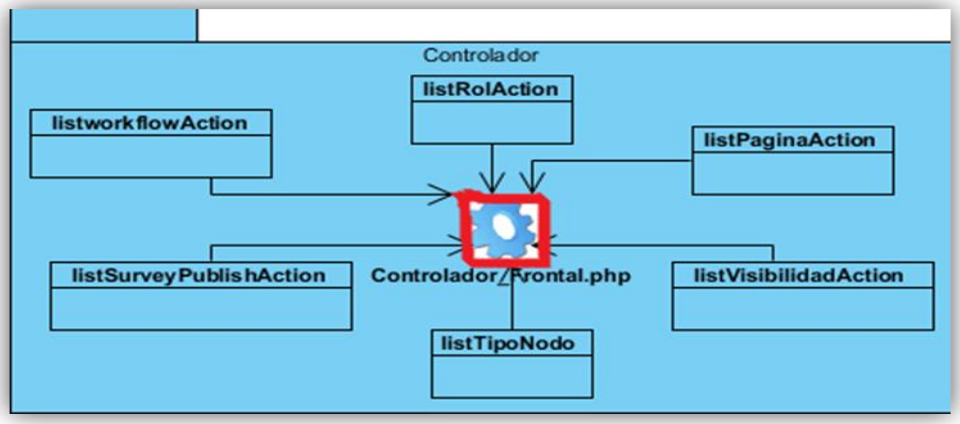


Fig. 6 Fragmento del diagrama de clases del diseño donde se evidencia la utilización del patrón GRASP: Controlador.

Experto: El patrón experto, está diseñado para asignar una responsabilidad a la clase que cuenta con la información necesaria para realizarla (Larman, 1999). Symfony genera 4 clases por cada tabla de la BD, por ejemplo para el desarrollo del módulo se tiene una tabla denominada Nodos, a partir de esta se crean las siguientes clases: Nodos, BaseNodos, NodosPeer y BaseNodosPeer.

Las clases **Base…….Peer** son las encargadas de hacer las consultas a la BD utilizando Propel, pues tienen los atributos necesarios para realizar dicha función, por tanto, deben implementar la responsabilidad de realizar las acciones directamente con la BD, evidenciándose así el uso del patrón Experto. Ver figura 7.

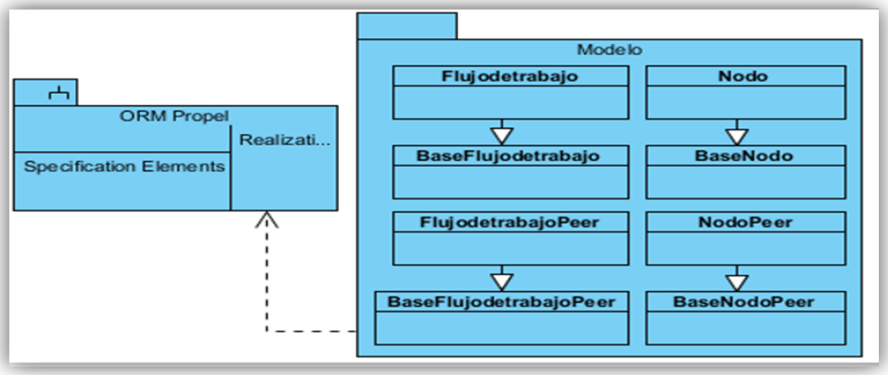


Fig. 7 Fragmento del diagrama de clases del diseño donde se evidencia la utilización del patrón GRASP: Experto.

Creador: Este patrón es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A. (Larman, 1999). Un ejemplo de esto es en las clases "action" cuando desea crear una instancia de una clase del Modelo, por ejemplo se tiene en la clase "`SaveWorkflowAction.php`" del módulo Definir Flujo de Trabajo, la misma crea una instancia de las clases `Flujodetrabajo.php`, `Nodo.php`, `Transición.php`, elementos necesarios para construir un Flujo de trabajo, evidenciándose este patrón.

Bajo Acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento (Larman, 1999).Una clase con bajo acoplamiento no depende de muchas otras. Este patrón se pone de manifiesto principalmente en el modelo de datos donde cada una de estas clases se comunican con el menor

número de clases posibles, logrando así un bajo acoplamiento entre las mismas, lo que permite una mayor reutilización. Ver figura 8.

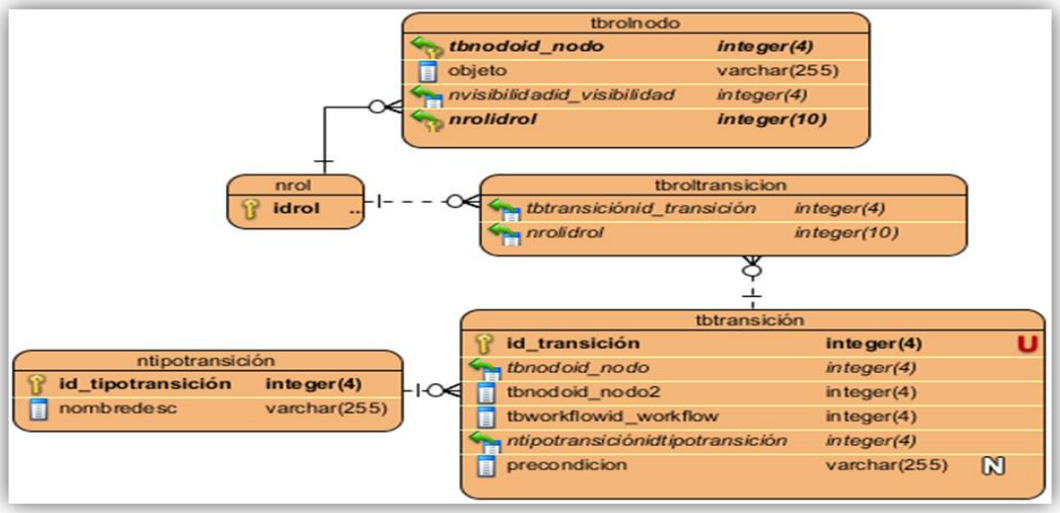


Fig. 8 Fragmento del modelo de datos donde se evidencia la utilización del patrón GRASP: Bajo Acoplamiento.

Alta Cohesión:

Asignar una responsabilidad de modo que la cohesión siga siendo alta (Larman, 1999). En la solución se encuentran varias clases actions que contienen varias funcionalidades con un propósito único, siendo estas funcionalidades las encargadas de controlar las acciones de las plantillas. Garantizando que grandes cambios conlleven a efectos mínimos y se logre una mayor capacidad de reutilización. Por ejemplo en la clase `listWorkflowActions.php` se pueden realizar grandes cambios como listar toda la información que se necesite de un flujo de trabajo conllevando a efectos mínimos, brindando una información más detallada en la clase `WPManageWorkflow.js`. Ver figura 9.

3.2.3 Patrones de Diseño GOF

Los patrones GoF son reconocidos dentro del campo del desarrollo de software en el libro Design Patterns, escrito por los que comúnmente se conocen como *Gang of Four* o "pandilla de los cuatro": Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides (Larman, 1999). Symfony implementa algunos de estos patrones:

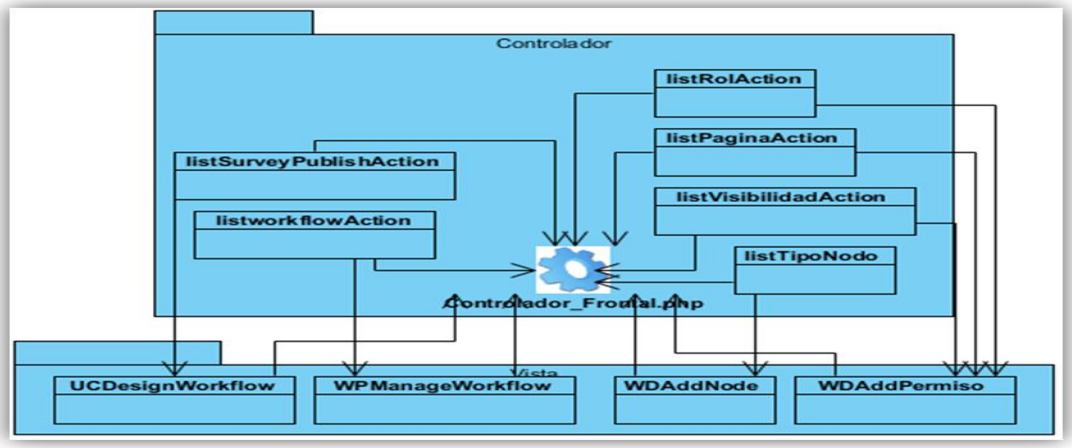


Fig. 9 Fragmento del diagrama de clases del diseño donde se evidencia la utilización del patrón GRASP: Alta Cohesión.

Patrón Decorador: El uso de este patrón se pone de manifiesto en la relación que se establece entre el layout o plantilla global y las diferentes plantillas que forman parte de la vista. El archivo llamado `layout.php` que contiene el Layout de la página, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el Layout, o si se mira desde otro punto de vista, el Layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado “Decorator”. El mismo se representa en la figura 10.

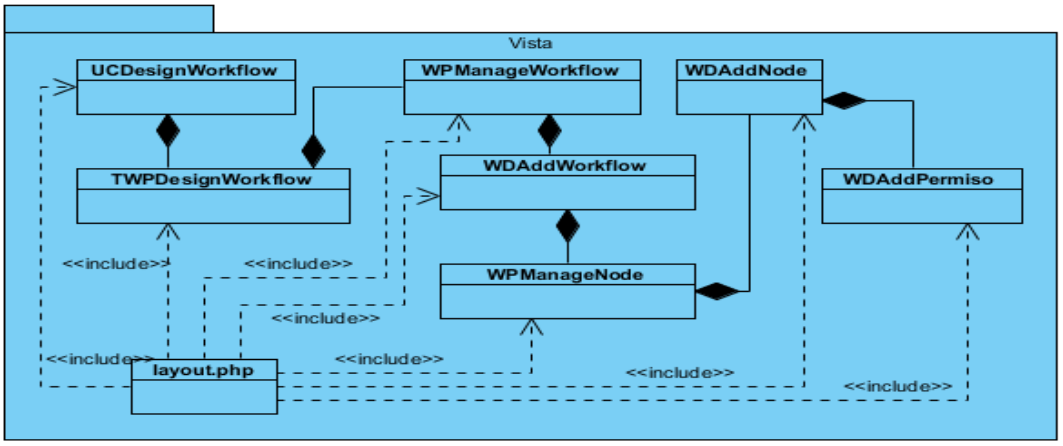


Fig. 10 Fragmento del diagrama de clases del diseño donde se evidencia la utilización del patrón GOF: Decorador.

Singleton (Instancia única): Asegurar que una clase tiene una sola instancia, proporcionando un punto de acceso global a la misma (Prieto, 2009). Este patrón se aplica en el método `createInstance ()` de la clase `sfContext.php`. La clase `sfContext.php` es una de las que utiliza el controlador frontal `(sfFrontWebController.php)` para enrutar todas las peticiones que se hagan a la aplicación. Por ejemplo al inicializarse el caso de uso Diseñar Flujo de trabajo la clase `sfContext.php` garantiza que exista una instancia única para este caso de uso. Ver figura 11.

```
 * @return sfContext                An sfContext instance
 */
static public function createInstance(sfApplicationConfiguration $configuration, $name = null, $class = __CLASS__)
{
  if (is_null($name))
  {
    $name = $configuration->getApplication();
  }

  self::$current = $name;

  self::$instances[$name] = new $class();

  if (!self::$instances[$name] instanceof sfContext)
  {
    throw new sfFactoryException(sprintf('Class "%s" is not of the type sfContext.', $class));
  }

  self::$instances[$name]->initialize($configuration);

  return self::$instances[$name];
}
```

Fig. 11 Fragmento de código donde se evidencia la utilización del patrón GOF: Singleton.

Observer: Este patrón define una dependencia de 1 a mucho de forma que cuando el objeto 1 se modifica, los n objetos también (Prieto, 2009). Ejemplo 1 flujo de trabajo tiene muchos estados o transiciones, esto se percibe cuando se añade, elimina o modifica un flujo de trabajo, repercutiendo en los n estados o transiciones que tenga el mismo.

Adapter: El uso de este patrón permite convertir la interfaz de una clase en otra interfaz esperada por los clientes (Prieto, 2009)**,** el mismo se evidencia en la clase `WPManageWorkflow.php`, cuando se ejecuta el botón add, se muestra la ventana (WDAddWorkflow) en la cual se entran los datos necesarios para adicionar un flujo de trabajo: ver figura 12.

```
var wdAddNode = new patdsi.workflow.design_workflow.WDAddNode({
    usecase: config.usecase,
    wf: this.wf,
    isNew: true
});
wdAddNode.on('addnodo', function(nodo) {
    this.addNodo(nodo);

}, this);
wdAddNode.show();
```

Fig. 12 Fragmento de código donde se evidencia la utilización del patrón GOF: Adapter.

Composite: El uso de este patrón permite componer objetos en jerarquías y permitir a los clientes tratar objetos simples y compuestos de manera uniforme (Prieto, 2009). En el siguiente ejemplo se muestra como se aplica este patrón en el desarrollo del Módulo de flujos de trabajo para SIGE: Un flujo de trabajo está formado por varias transiciones y nodos, y estos últimos a su vez contienen varios permisos. Por ejemplo: en la clase `WDAddWorkflow.js` se crea un objeto workflow (this.wf) el cual se va a ir conformando a medida que el usuario añada los nodos con sus permisos asociados y sus transiciones, este objeto `(this.wf)` se puede tratar de forma simple en cada una de las clases que se crean (`WPManageNode.js`, `WPManageTransition.js`, `WDAddPermiso.js)` pasándolo por parámetro a las mismas en el instante de su construcción. Este se evidencia en la figura 13.

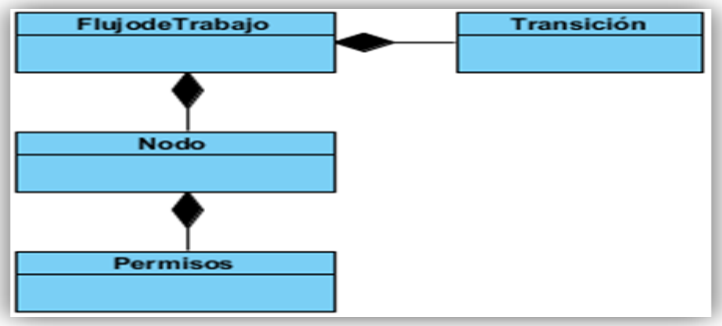


Fig. 13 Diagrama donde se evidencia la utilización del Patrón GOF: Composite.

Fachada: El uso de este patrón aporta una interfaz simple, que brinda un acceso único a las funcionalidades de varios componentes complejos, esto permite que cada componente sea más reutilizable e independiente (Prieto, 2009). Este se evidencia en la clase `WDAddWorkflow`, esta clase contiene 3 clases internas las cuales permitirán adicionar un flujo de trabajo.

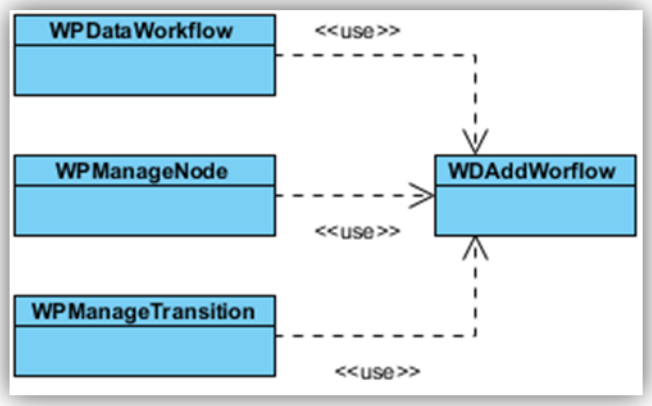


Fig. 14 Diagrama donde se evidencia la utilización del patrón GOF: Fachada

3.4 Modelo de Datos

El modelo de datos es una representación abstracta de los datos y las relaciones que existen entre ellos. Por lo general, un modelo de datos permite describir el tipo de datos que incluye la base de datos y la forma en que se relacionan, así como las restricciones de integridad y las operaciones de manipulación de

los datos (Garza, 2008). La siguiente figura (ver figura 15) muestra la estructura del modelo de datos del Módulo de flujos de trabajo para SIGE.

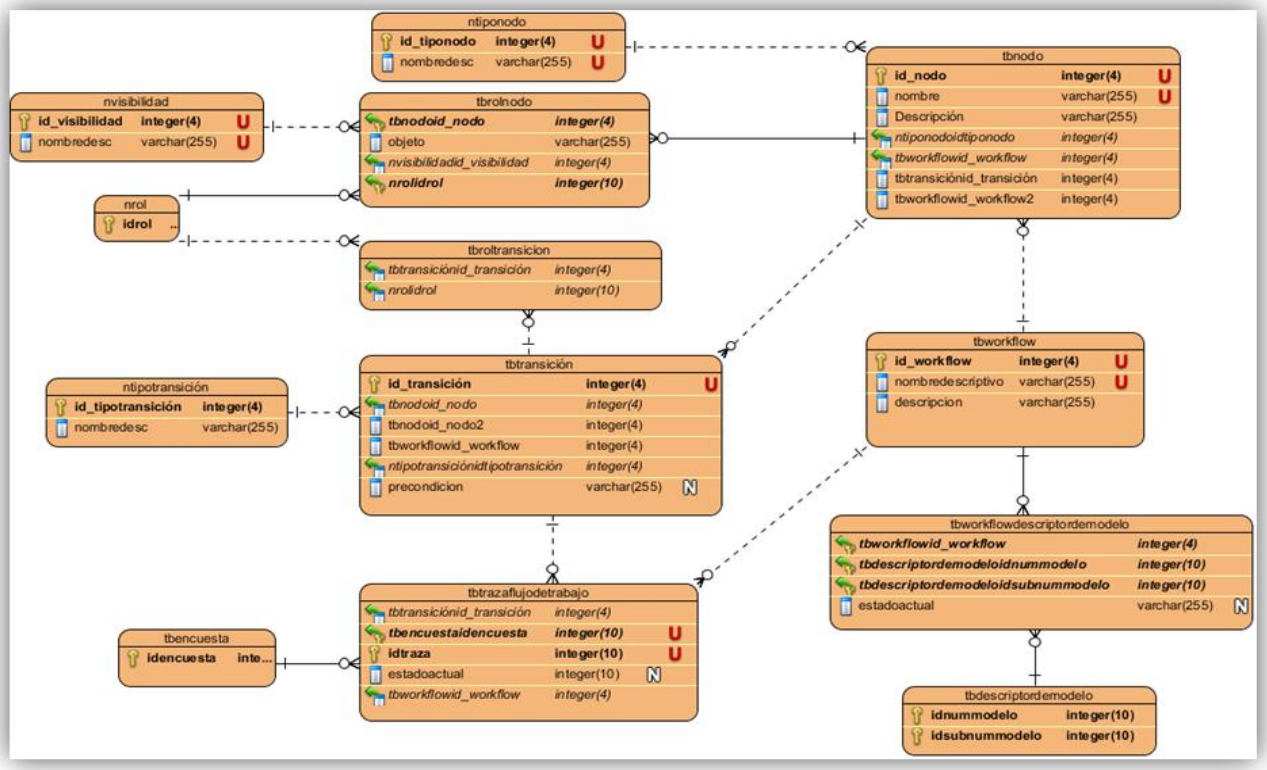


Fig. 15 Diagrama Entidad Relación.

3.5 Modelo de Despliegue

El Modelo de Despliegue describe la distribución física del sistema, muestra cómo están distribuidos los componentes de software entre los distintos nodos de cómputo. Permite comprender la correspondencia entre la arquitectura software y la arquitectura hardware (Larman, 1999). A continuación (ver figura 14) se muestra el modelo de despliegue del Módulo de flujos de trabajo para SIGE.

El nodo **PC_Cliente** que representa la computadora del usuario, se conecta mediante el Protocolo Seguro de Transferencia de Hipertexto (**HTTPS**, Hypertext Transfer Protocol Secure) al nodo **Servidor Web**. Del servidor se puede acceder al nodo **Servidor_BD** mediante el protocolo diseñado para facilitar la reutilización de código de bases de datos (**TCP/IP**).

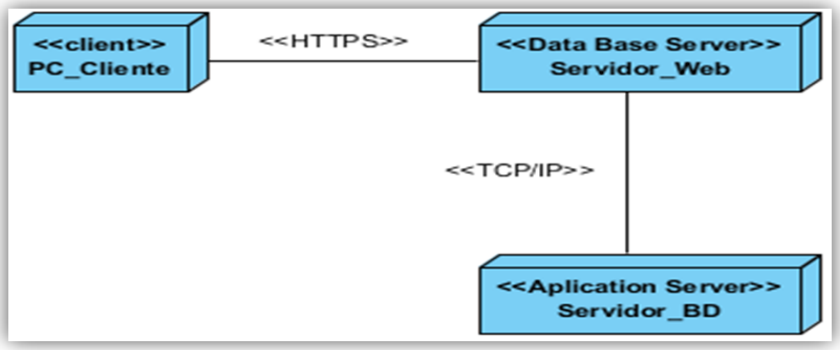


Fig. 16 Modelo de despliegue del Módulo de flujos de trabajo.

3.6 Conclusiones Parciales

En el capítulo presentado se trataron los temas más importantes referentes al diseño. Se seleccionó la arquitectura MVC que garantizó definir la estructura lógica para el Módulo de flujos de trabajo para SIGE, asegurando una mejor organización para sus componentes. Los patrones de diseño aplicados ayudaron a complementar la arquitectura del módulo, evidenciando el uso de soluciones demostradas como parte de las buenas prácticas de la programación. Además se diseñaron los artefactos que permitieron dar continuación a la implementación del mismo.

Capítulo 4: Implementación y prueba

4.1 Introducción

Una vez definido el diseño del módulo, se asegura un punto de partida para iniciar la implementación, donde se debe obtener como resultado el código fuente de la aplicación y el modelo de implementación. Luego de ser implementado se necesita elevar la calidad del mismo y validar que todas sus funcionalidades son operativas.

Para lograr dicho objetivo es necesario realizarle pruebas haciendo uso de varias técnicas de prueba. La realización de las pruebas permite detectar errores en el módulo asegurando que durante su ciclo de vida o al finalizar cada iteración de desarrollo se corrijan los problemas identificados. En el contenido del capítulo se exponen diferentes elementos asociados a la implementación y validación del Módulo de flujos de trabajo para SIGE que permiten reflejar con claridad las actividades realizadas durante esta etapa.

4.2 Modelo de Implementación.

El Modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación. Describe como los elementos de diseño se implementan en componentes. Se considera el artefacto más significativo del flujo de trabajo de Implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Este modelo está conformado por el diagrama de componentes (Jacobson, 2000)

4.2.1 Diagrama de Componentes.

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes, mostrando las dependencias que existen entre ellos. Los componentes físicos incluyen: archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables y paquetes (Jacobson, 2000). Para visualizar la estructura general del sistema se generó el diagrama de componente, el cual permitió describir el modelo de implementación y las relaciones entre los elementos de dicho modelo, el criterio utilizado se basó en la agrupación por paquetes así como la representación de dependencias entre los mismos, donde cada clase representa un componente. Ver figura 15.

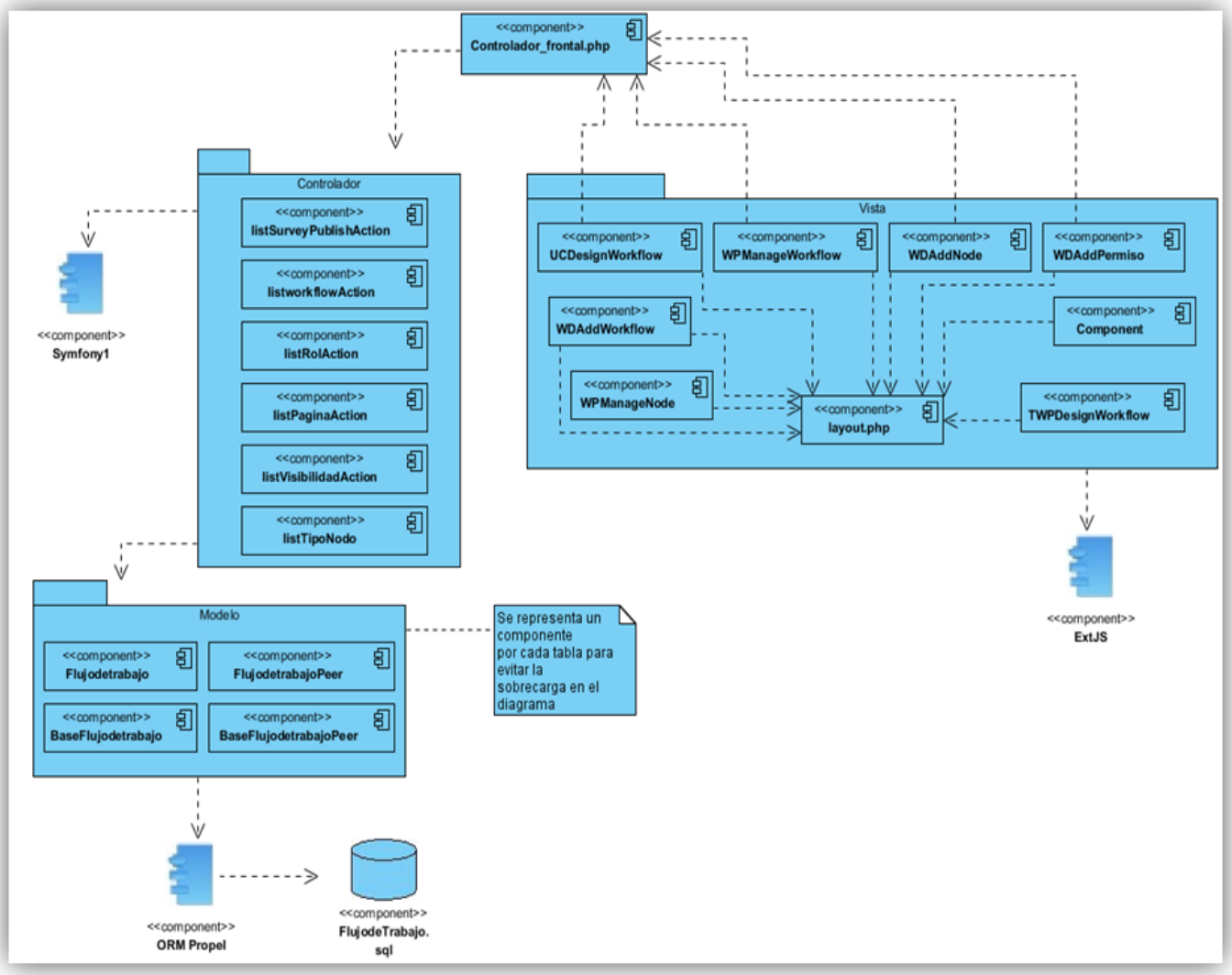


Fig. 17 Diagrama de componentes CU Gestionar Estados.

El flujo de acciones del diagrama de componentes presentado se define de la siguiente forma: los componentes identificados se agrupan en tres paquetes de implementación básicos la vista, el controlador y el modelo. En la vista se encuentran los componentes que permiten la interacción directa entre el usuario y el sistema, mostrando y recogiendo información, utilizando los componentes de ExtJS. En el controlador están los componentes que manipulan los eventos del usuario y realizan peticiones al subsistema a través del componente controlador frontal, que es la única puerta de entrada y salida a la

aplicación, en este proceso el controlador utiliza el archivo componente Symfony1.1.7. Finalmente en el modelo, se agrupan los componentes que interactúan con la base de datos y garantizan el cumplimento de las reglas del negocio, empleando Propel para realizar el mapeo de datos.

4.3 Estándares de codificación.

Los estándares de codificación son pautas que se deben seguir para permitir que el código fuente de la solución posea mayor calidad, además de posibilitar el entendimiento por parte de otros programadores con el objetivo de facilitar el futuro mantenimiento del módulo.

Para el uso de estándares de codificación se tuvo en cuenta los estilos que usa SIGE, debido a que el módulo a desarrollar será integrado a dicho sistema. A continuación se muestra los estándares utilizados:

Notaciones utilizadas:

- ✓ **PascalCasing:** En este caso los identificadores, nombres de las variables, métodos, clases y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.
- ✓ **CamelCasing:** Para definir los identificadores, nombres de las variables, métodos, clases y funciones, que están compuestas por múltiples palabras juntas, se inicia la primera palabra con letra minúscula y las demás utilizando PascalCasing.

Nomenclatura de las clases JavaScript

Los nombres deben comenzar en dependencia del componente ExtJS:

- WP: En caso de ser un panel.
- WD: En caso de ser una ventana.
- UC: Cuando se quiere definir un caso de uso.
- TWP: Para definir el panel de trabajo del caso de uso.

Posteriormente se pasa a definir el nombre que la identificará en idioma Inglés con la nomenclatura PascalCasing. Además de su extensión ".js" Ejemplo: WPManageWorkflow.js.

Nomenclatura clases php

1. **ClasesActions (acciones).**

Se utiliza la notación CamelCasing seguido de "Action.class.php" Ejemplo: listWorkflowAction.class.php.

2. **Clases Estáticas**

Se utiliza la notación PascalCasing seguido de ".class.php". Ejemplo: WorkflowService.class.php.

Nomenclatura de las funciones, métodos y variables

El nombre a emplear se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing. Ejemplo: deleteWorkflow.

Comentarios

Los comentarios de implementación son para guiar el proceso y dar un mayor entendimiento. Se utiliza: "//" y "/* */". Ejemplo: /* Esto es un comentario*/

Todas las etiquetas php deben ser completas (<?php?>)... no reducidas (<? ?>).

Los bloques de código siempre deben estar encerrados por llaves.

4.4 Pruebas de Software

Las pruebas de software forman parte de lo que podemos denominar verificación y validación del software. La verificación se refiere al hecho de comprobar si estamos desarrollando el software correctamente. La validación se encarga de comprobar si estamos construyendo el producto correcto. Son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación (Amo, 2005).

4.4.1 Niveles de pruebas

Las pruebas son aplicadas para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo, dentro de estas se distinguen diferentes niveles de prueba: Prueba a nivel de desarrollador, unidad, integración, sistema y aceptación (PRESSMAN, 2002). En el desarrollo del módulo se aplicarán solamente las pruebas siguientes para comprobar que el mismo cumple con los requisitos definidos anteriormente:

- ✓ **Pruebas de desarrollador:** es el primer nivel de pruebas, diseñadas e implementadas por el equipo de desarrollo, este tipo de prueba le permite al desarrollador realizar acciones que le posibiliten verificar que los componentes están funcionando correctamente y en caso de entradas erróneas, mostrar la capacidad de tratar errores.
- ✓ **Prueba de Integración:** en este nivel se prueba los componentes combinados para ejecutar un CUS. Se realiza la prueba para descubrir errores en las especificaciones de las interfaces de las clases con el objetivo de verificar que el sistema funcione correctamente una vez integrado. Las pruebas

funcionales de integración son similares a las pruebas de caja negra. Aquí se trata de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios prestados por otro(s) módulo(s).

- ✓ **Prueba de Aceptación:** es la prueba final antes del despliegue del módulo. Son definidas y diseñadas por el cliente con el objetivo de asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas.

4.4.2 Tipos de pruebas

Existen diferentes tipos de pruebas que se pueden aplicar para verificar que el módulo cumple con todos los requisitos identificados durante el proceso de análisis. Dentro de los tipos de pruebas se encuentran las pruebas funcionales, de usabilidad, fiabilidad, rendimiento y soportabilidad. Para el desarrollo del Módulo de flujos de trabajo para SIGE se selecciona el tipo de prueba funcional con el objetivo de probar que el módulo desarrollado cumple con las funciones específicas para las cuales fue creado.

Prueba de Funcionalidad: Se realiza este tipo de pruebas con el objetivo de validar el cumplimiento de cada funcionalidad implementada y verificar hasta qué punto se cumplieron las especificaciones iniciales del sistema apoyadas en el diseño de los casos de prueba. Además permite comprobar la integración del módulo a SIGE, para determinar si fueron cumplidos los objetivos planteados.

4.4.3 Métodos de prueba

Existen dos métodos de pruebas independientemente del nivel en que se enmarquen los tipos de pruebas: el método de caja blanca y caja negra, en el desarrollo del módulo se utilizará el método de Caja Negra, pues este tipo de prueba se centra en los requisitos funcionales del software, permitiendo probar a través de un conjunto de entradas la ejecución de los requisitos funcionales permitiendo corregir errores que puedan ser detectados. A continuación se explicará en que consiste.

Pruebas de Caja negra: Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software a probar, entendiendo por interfaz las entradas y salidas de dicho programa (EVA, 2013). Se ejecutaron todos los Casos de Pruebas basados en los Casos de Uso ingresando datos válidos y no válidos con el objetivo de verificar que cada una de las respuestas del sistema se corresponda con lo que se espera en cada una de las entradas de datos y en caso de entrada de datos no válidos, se muestren mensajes de advertencia o de error con su descripción. Dentro del Método de Caja Negra se utilizó la técnica Partición Equivalente.

La **técnica partición de equivalencia** permite examinar los valores válidos e inválidos de las entradas existentes en el software. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada (Pressman, 2002). Por lo antes expuesto se decide aplicar esta técnica al Módulo de flujos de trabajo para SIGE en aras de comprobar que el mismo cumple con los objetivos trazados.

4.4.4 Desarrollo de pruebas

Para comprobar que el Módulo de flujos de trabajo para SIGE cumple con los requisitos previamente identificados en la fase de análisis, se aplicarán las pruebas pertenecientes a los niveles de desarrollador e integración explicados en los epígrafes anteriores. Todas estas pruebas aplicadas utilizan el método de caja negra.

Diseño de caso de prueba

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Se parte de la descripción de estos últimos, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión. Para detallar el caso de uso se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y estas a su vez en escenarios, para hacer más fructífera la ejecución de las pruebas. A continuación se presentan las tablas de las secciones probadas para el caso de uso Gestionar flujo de trabajo.

Luego de realizar un caso de prueba es necesario definir una serie de variables a las cuales se le asignaran valores válidos y no válidos para comprobar que el resultado que se obtiene es el esperado, en este caso se evaluará el CU **"`Gestionar Flujo`"** y dentro de este, la sección **"`Adicionar nuevo flujo`"**, donde fueron definidas 3 variables las cuales se describen a continuación:

Tabla 2 Descripción de las variables

No.	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1.	Nombre	Texto	No	Cadena de caracteres (solo letras minúsculas y mayúsculas) que representa el nombre del flujo.
2.	Descripción	Texto	Si	Breve descripción textual asociada al flujo.
3.	Plantilla	Lista de selección	No	Campo de selección que contiene las tablas

				que puede utilizar el proceso.

A partir de la descripción de variables se realizó la matriz de datos, donde se evaluó y probó la validez de cada uno de los datos seleccionados en el sistema, utilizando un juego de datos válidos y no válidos, mediante el empleo de la técnica de partición de equivalencia.

Tabla 3 Sección Adicionar flujo

No. Esc	**Escenario**	**Variables**			**Respuesta del sistema**	**Resultado de la prueba**	**Flujo Central**
		1	**2**	**3**			
EC1.1	Adicionar nuevo flujo	V	V	V	El sistema lista el flujo que ha sido creado, lo visualiza y muestra un mensaje Operación exitosa.	Satisfactorio	1-El especialista introduce todos los datos correctamente para crear el nuevo flujo. 2-El sistema verifica que los campos requeridos hayan sido completados. 3-El sistema verifica que no exista un flujo con el mismo nombre. 4-El sistema verifica que no exista una plantilla con un flujo asociado.
EC1.2	Adicionar flujo con campos vacíos.	I	V	V	1-Si el campo nombre quedó vacío el sistema emite un mensaje: “El campo nombre es obligatorio“ y se indica en color rojo. 2-Sino se selecciona una plantilla el sistema emite un mensaje: “Debe seleccionar una plantilla“.	Satisfactorio	1-El especialista introduce los datos para crear el nuevo flujo. 2-El sistema verifica que los campos requeridos no han sido completados.
EC1.3	Flujo duplicado.	I	V	V	Emite un mensaje “Ya existe un flujo con ese nombre“.	Satisfactorio	1-El especialista introduce los datos para crear el nuevo flujo. 2-El sistema verifica que el nombre introducido ya existe.

EC1.4	Plantilla con flujo asociado.	V	V	I	Emite un mensaje "Ya existe un flujo asociado a esa plantilla ".	Satisfactorio	1-El especialista introduce los datos para crear el nuevo flujo. 2-El sistema verifica que la plantilla seleccionada ya tiene un flujo asociado, y emite un mensaje: "Ya existe un flujo asociado a esa plantilla".
EC1.5	Cancelar operación	NA	NA	NA	1-El sistema no realiza ningún cambio, y regresa a la interfaz anterior.		1-El especialista cancela la operación realizada. 2-El sistema elimina los datos ingresados y regresa a la interfaz anterior.

Pruebas de integración

Luego de realizar las pruebas funcionales, se integrará el módulo al proyecto SIGE para comprobar que los objetivos propuestos fueron alcanzados satisfactoriamente. A continuación se especifican los cambios realizados a la arquitectura del sistema con la integración para verificar su correcto funcionamiento. En el anexo 1 se evidencia un aval que corrobora la correcta integración del módulo al proyecto antes mencionado.

Cambios realizados en la arquitectura del sistema con la integración del Módulo de Flujos de:

- ✓ Se añadió un nuevo módulo a SIGE, generando 2 procesos dentro del paquete Workflow.
- ✓ Se añadió un nuevo schema (contiene 10 nuevas tablas) en el modelo del sistema correspondiente a la creación del flujo de trabajo.
- ✓ Se añadió un nuevo campo (estadoactual) a la tabla encuesta, el cual indica el estado actual en que se encuentra.
- ✓ Se modificó el método `getSurvey()` de la clase `Encuesta.php` y el `getTemplateSurvey ()` de la clase `Descriptordelmodelo.php` donde se le añade un campo visibilidad a cada página del modelo y plantilla para su posterior visualización.
- ✓ Se modificó la clase `saveAllRowsSurvey.php` donde se ejecuta el flujo asociado a esa plantilla o modelo en caso de tenerlo, salvando un conjunto de trazas y un estado actual para esta encuesta.
- ✓ Se modificó el método `loadSurvey ()` de la clase `WPSurveyBody.js` donde se construyen las encuestas para su posterior visualización, se desabilitó las páginas a las que no tuviera visibilidad el usuario conectado.

- ✓ Se utiliza la biblioteca `Dracula.js` para poder visualizar el flujo de trabajo en forma de grafo dirigido. Para la integración de esta biblioteca se añaden las rutas de los archivos de la biblioteca en `indexTextSuccess.php`
- ✓ Se fueron realizando pruebas funcionales de los requisitos implementados en los módulos design_workflow y engine_workflow para verificar que no afectaran el funcionamiento del sistema. Ejemplo de esto se diseñó un flujo de trabajo con el módulo design_workflow y luego se monitoreó con el engine_workflow evidenciando resultados satisfactorios y su correcta integración con el sistema.

4.5 Resultado de las pruebas

Caja Negra

Después de realizar las pruebas funcionales mediante el método de caja negra, utilizando los casos de prueba asociados a cada caso de uso, y empleando la técnica partición de equivalencia se comprobó el correcto funcionamiento del módulo. Durante las tres iteraciones realizadas, se detectaron un total de 8 no conformidades, quedando todas resueltas, como se muestra en la figura 19.

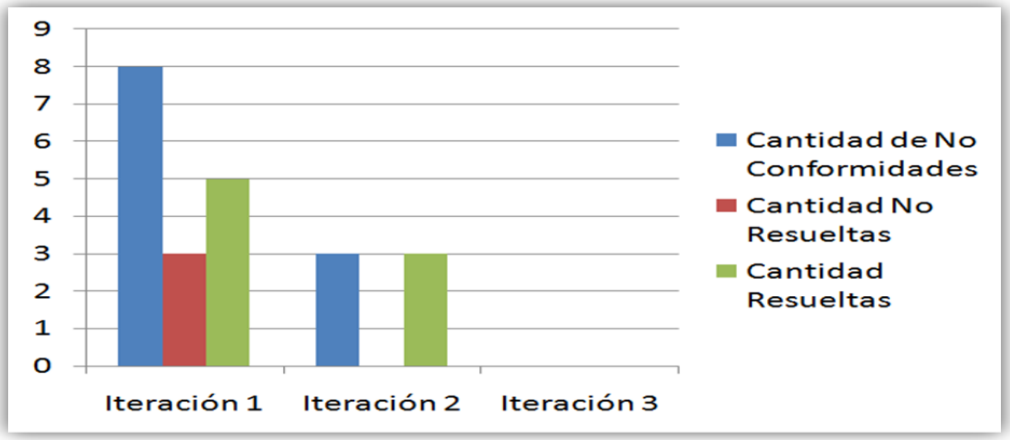


Fig. 18 Resultados de las pruebas

Prueba de Integración

Una vez integrado el módulo a SIGE, se pudo apreciar que funciona correctamente arrojando resultados satisfactorios. Cuando el usuario va a diseñar un flujo de trabajo y adiciona un nuevo flujo, el sistema permite ingresar todos los datos referentes al flujo, permitiendo la gestión de estados y transiciones asociadas al mismo. Además el usuario después de haber adicionado el flujo puede realizar todas las operaciones siguientes como listar los flujos, modificar datos asociados a ese flujo. Una vez creado el flujo

el usuario puede acceder al módulo Monitorear flujo de trabajo para chequear y verificar el cumplimiento y comportamiento del diseño. Con esta prueba se garantiza el cumplimiento del RNF Restricción de Diseño.

Prueba de Aceptación

Las pruebas de aceptación se realizan con el objetivo de validar la solución a través de un encuentro con el cliente, el cual comprueba que el sistema cumple con el funcionamiento esperado y da muestra de su conformidad en el documento oficial **Carta de Aceptación** del cliente donde quedan plasmados los resultados. Ver Anexo 1

4.6 Conclusiones Parciales

Esta etapa de desarrollo se caracteriza por resultados ya visibles para los clientes y gratificantes para los desarrolladores, pues queda implementada la aplicación con las principales funcionalidades que se definieron. Luego de ser generado el código fuente del módulo se aplicaron casos de prueba mediante la técnica de caja negra lo que facilitó determinar 5 no conformidades al culminar la primera iteración de desarrollo, 3 en la segunda iteración y en la tercera se alcanzaron resultados satisfactorios debido a que no fue identificada ninguna no conformidad, para un total de 8 NC. A partir de la realización de pruebas se validó que los requisitos funcionales definidos son totalmente operativos, evidenciados en la Carta de Aceptación del cliente.

Conclusiones

Teniendo en cuenta los objetivos trazados en la investigación, en aras de darle cumplimiento al problema propuesto, se arribó a las siguientes conclusiones:

- ✓ El estudio de los principales conceptos y los diferentes sistemas que utilizan flujos de trabajo permitió sentar las bases para el desarrollo del Módulo de flujos de trabajo para SIGE.
- ✓ A partir de la realización del análisis y diseño del Módulo de flujos de trabajo para SIGE se obtuvo como resultado los diagramas y artefactos necesarios para guiar el desarrollo del mismo.
- ✓ La implementación del Módulo de flujos de trabajo para SIGE dio cumplimiento a los 22 requisitos funcionales identificados en la fase de análisis.
- ✓ El diseño y ejecución de las pruebas a nivel de desarrollador, integración y de sistema permitió comprobar el correcto funcionamiento del Módulo de flujos de trabajo para SIGE y el cumplimiento de las funcionalidades definidas.
- ✓ Como resultado se obtuvo el Módulo de flujos de trabajo para SIGE que permite definir el conjunto de estados y transiciones por los que van a transitar las encuestas durante su ciclo de vida en el sistema, así como visualizar detalladamente el comportamiento y cumplimiento del flujo que se define.

Recomendaciones

Durante el desarrollo del trabajo han surgido ideas que podrían implementarse en versiones posteriores del módulo, de forma que se logren agregar o modificar las funcionalidades, para lo cual se recomienda:

- ✓ La incorporación de otros tipos de transiciones como las temporales y mediante eventos con el fin de enriquecer el proceso de transición de estados.
- ✓ La incorporación de otros tipos de preguntas como las de tipo lógico y de texto en el caso de las transiciones automáticas.

Referencias bibliográficas

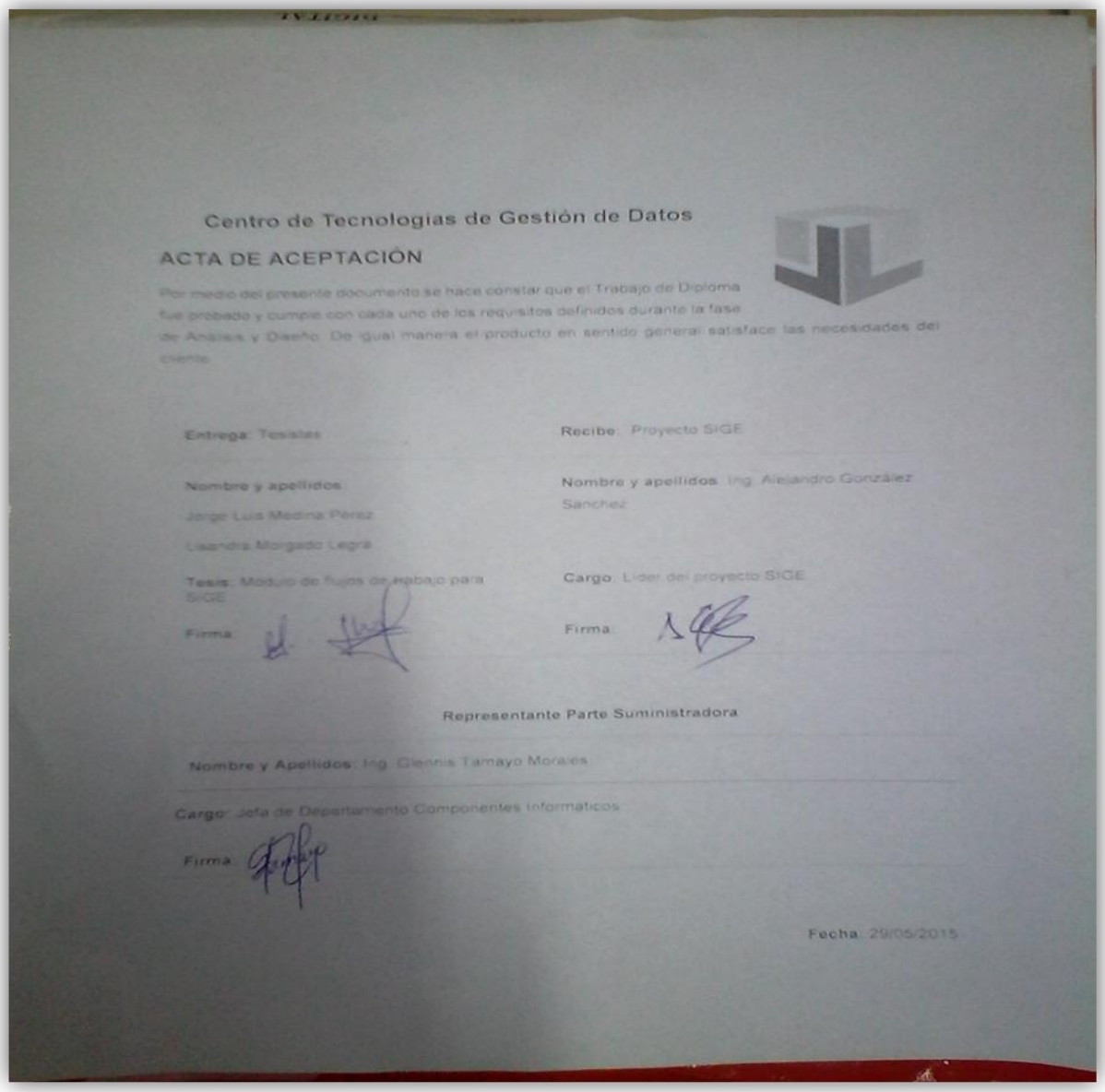
1. **Amo, Fernando Alonso. 2005.** *Introducción a la ingeniería del software.* s.l. : Delta Publicaciones, 2005. 9788496477001.
2. **Asenjo, Jorge Sánchez. 2012.** *Introducción a php.* España : s.n., 2012.
3. **Aveler Otero, Darvin. 2010.** Qué es OpenERP. [En línea] 22 de Enero de 2010. [Citado el: 7 de noviembre de 2014.] http://darv.in/2010/01/22/%C2%BFque-es-openerp/.
4. **Balduino, Ricardo. 2011.** *Introduction to OpenUP(Open Unified Process).* Sao Paulo, Brazil : s.n., 2011.
5. **Booch, Rumbaugh y Jacobson. 2006.** *El Lenguaje Unificado de Modelado.* 2006.
6. **Castaño, Miguel. 2000.** *EL sistema de información estadística en el marco del enfoque sistémico.* Madrid : Facultad de Informática Universidad Politécnica de Madrid, 2000.
7. **Charalambous, Alex. 2010.** *Extension of PIPE2 to Support Coloured Generalised Stochastic Petri Nets.* London : Department of Computing, 2010.
8. **Cintra, Yordano Yunior Osorio. 2013.** *Sistema para la gestión del banco de problemas de la UCI.* La Habana : s.n., 2013.
9. **Columbie, Claudia. 2013.** *Manual de usuario del módulo de seguridad.* La Habana : s.n., 2013.
10. **Contreras, Pamela. 2011.** *Unidad Nº8. Vista de la Máquina de Estados.* Sevilla : s.n., 2011.
11. **Cortés, Luis Toharia. 2010.** *La mejora del sistema de informacion estadistica procedente de los registros de la seguridad social .* España : s.n., 2010.
12. **Cryself Villa. 2008.** Introducción a Ext Js. . [En línea] 24 de marzo de 2008. [Citado el: 9 de noviembre de 2014.] http://crysfel.com/introduccion-a-ext-js/.
13. **Diaz, IsabeL. 2005.** *Metamorfosis: Un marco para el análisis de requisitos funcionales.* Portugal : s.n, 2005.
14. **EVA. 2013.** EVA. [En línea] 2013. [Citado el: 4 de abril de 2015.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_2/Comun/Material_de_caja_b_y_caja_n.pdf.
15. **Garza, Gómez de Silva. 2008.** *Introducción a la Computación.* s.l. : Cengage Learning Editores, 2008. 9789706867681.
16. **Herrera, Basurto y Kris, Cristhian. 2008.** [En línea] febrero de 2008. [Citado el: 10 de noviembre de 2014.] http://es.scribd.com/doc/2068969/Instalacion-de-Bonita-Workflow#outer_page_1.

17. **Jacobson, I. 2000.** *El proceso unificado de desarrollo de software.* s.l. : Pearson Educación S.A, 2000.
18. **Koch, Nora. 2002.** *Ingeniería de Requisitos en Aplicaciones para la Web-Un estudio comparativo.* España : Universidad de Sevilla, 2002.
19. **Larman, Craig. 1999.** *UML y Patrones.* México : s.n, 1999.
20. **Lobo, Armando Robert. 2008.** *Sistema Integrado para la Gestión de Estadística en Cuba.* La Habana : 6ta Jornada Científica Estudiantil de la Universidad de las Ciencias Informáticas, 2008. s.n.
21. **Martínez, Rafael. 2010.** postgresql. [En línea] 2 de diciembre de 2010. [Citado el: 4 de noviembre de 2014.] http://www.postgresql.org.es/sobre_postgresql#caracteristicas.
22. **Muller, Alex. 2012.** *ApplicationMap.Administración web y comercio electrónico en entornos de software libre .* 2012.
23. **Muller, Hans Cristian. 2007.** *Programando en Fortran.* Santa Cruz : s.n., 2007.
24. **Netbeans. 2013.** Netbeans. [En línea] 2013. [Citado el: 10 de noviembre de 2014.] http://netbeans.org/community/releases/70/index.html.
25. **OpenERP . 2012.** OpenERP. [En línea] 2012. [Citado el: 9 de noviembre de 2014.] http://openerp-server.readthedocs.org/en/latest/workflows.html.
26. **Orozco, Jorge Alejandro Gutiérrez. 2008.** *Máquinas de Estados Finitos.* s.l. : Escuela Superior de Cómputo, 2008.
27. **Pérez, Javier Eguíluz. 2009.** *introduccion a javascript.* 2009.
28. **pgAdmin. 2011.** pgAdmin PostgreSQL Tools. [En línea] 2011. [Citado el: 6 de noviembre de 2014.] http://www.pgadmin.org.
29. **Pixelwarex. 2012.** Pixelwarex. [En línea] 2012. [Citado el: 9 de noviembre de 2014.] http://www.pixelware.com/workflow-flujo-trabajo.htm.
30. **Potencier, Fabien. 2010.** *Symfony 1.4 la guia definitiva.* 2010.
31. **Pressman, Roger S. 2002.** *Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill Companies, 2002. ISBN: 8448132149.
32. **Prieto, Félix. 2009.** Patrones de diseño. [En línea] 2009. [Citado el: 23 de marzo de 2015.] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.
33. **Riehle, Dirk. 2000.** *Framework Design: A Role Modeling Approach.* s.l. : Swiss Federal Institute of Technology, 2000.

34. **Rivera, Isaura Sara. 2012.** *Componentes de los sistemas de gestion y del ciclo de vida de un sistema de información.* Barranquilla : s.n., 2012.
35. **Silva, André. 2011.** *VISUALIZING ONLINE INTERACTIONS IN MOODLE: A REPORTS MODULE.* Portugal : s.n., 2011. 4169-007.
36. **Sommerville, Ian. 2005.** *Ingeniería de Software.* Madrid : Pearson Educación, 2005.
37. **Villegas, Esmeralda. 2010.** *Investigación documental. Metodologías de desarrollo de software.* Apatzingán Michoacan : Instituto tecnológico superior de Apatzingán, 2010.
38. **Zaldivar, Hector Luis Reyes. 2013.** *Informatización de los Procesos de Gestión Estadística en Cuba.* Habana : s.n., 2013.

Anexos

Anexo 1: Carta de Aceptación



Centro de Tecnologías de Gestión de Datos

ACTA DE ACEPTACIÓN

Por medio del presente documento se hace constar que el Trabajo de Diploma fue probado y cumple con cada uno de los requisitos definidos durante la fase de Análisis y Diseño. De igual manera el producto en sentido general satisface las necesidades del cliente.

Entrega: Tesistas

Nombre y apellidos:
Jorge Luis Medina Perez
Lisandra Morgado Legra

Tesis: Modulo de flujos de trabajo para SIGE

Firma:

Recibe: Proyecto SIGE

Nombre y apellidos: Ing. Alejandro González Sanchez

Cargo: Líder del proyecto SIGE

Firma:

Representante Parte Suministradora

Nombre y Apellidos: Ing. Glennis Tamayo Morales

Cargo: Jefa de Departamento Componentes Informaticos

Firma:

Fecha: 29/05/2015