Universidad de las Ciencias Informáticas

Facultad 3

TÍTULO: "SISTEMA PARA LA EVALUACIÓN TECNOLÓGICA EN EL DESPLIEGUE DE SOFTWARE DE GESTIÓN EMPRESARIAL"

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

Autores: Dayani Del Toro Lorenzo

Alejandro Rodríguez Fernández

Tutor: MsC. Carlos Abel Capeáns Hurtado

Co-tutor: Ing. Dayannis Estrada Duarte

La Habana, junio 2015

"Año 57 de la Revolución"

"En la tierra hace falta personas que trabajen más y critiquen menos, que construyan más y destruyan menos, que prometan menos y resuelvan más, que esperen recibir menos y dar más, que digan mejor ahora que mañana."

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 firmo la presente a los ____ días del mes de _____ del año _____.

Dayani Del Toro Lorenzo

_____

Firma del Autor

Alejandro Rodríguez Fernández

_____

Firma del Autor

MsC. Carlos Abel Capeáns Hurtado

_____

Firma del Tutor

AGRADECIMIENTOS

Agradezco a toda mi familia por apoyarme en todo momento en especial a mi tío y padrino Pepe por ser un segundo padre para mí.

A mi mamá por siempre estar ahí para guiarme y apoyarme en todas mis decisiones. Por no dejarme sola en los peores momentos.

A mi papá por ser mi ejemplo a seguir y ser tan buen padre para mí y estar ahí presente.

A mis hermanos por brindarme amor, cariño y apoyarme siempre y a mi sobrino hermoso por ser mi inspiración.

A todos mis vecinos por ser parte mi familia y ayudarme en todo en especial a Cusa y familia.

A todos los compañeros y amigos que hice en esta universidad y que durante estos 5 años me han apoyado en todos los momentos difíciles.

A mi tutor Carlos por todo el tiempo dedicado y por apoyarnos en cada momento.

A mi novio, amigo y compañero de tesis por todos mis logros, por quererme tanto y aguantar todas mis malcriadeces.

A mi abuela Cuca y mi tía Lita que aunque no están entre nosotros fueron mi apoyo.

Dayani Del Toro Lorenzo

Agradezco a mi familia por brindarme su ayuda incondicional, ayudándome a sobrepasar incluso los momentos más difíciles.

A mi mamá que quiero tanto y se preocupa siempre por mí.

A mi papá que es ejemplo para mí por su sacrificio y por siempre orientarme cuando lo necesitaba.

A mi hermana que siempre me anima y apoya incondicionalmente, por brindarme siempre su amor y cariño.

A todos mis amistades del IPVCE, en especial a El negro, a Toledo, a David, al Toto y a Chiquito por brindarme su amistad y su apoyo.

A mis amistades de la UCI del grupo 3501 en especial a mi tocayo Alejandro por ayudarme cuando lo necesitaba.

A mi tutor Carlos por todo el tiempo dedicado y por apoyarnos en cada momento.

A mi novia y compañera de tesis que me ha ayudado tanto en todos estos años, por aconsejarme cuando lo necesitaba y por quererme incondicionalmente.

Alejandro Rodríguez Fernández

DEDICATORIA

A mis padres por todo el esfuerzo realizado y a toda mi familia por el apoyo.

A todas las personas que me han ayudado a alcanzar este logro.

Dayani Del Toro Lorenzo

A toda mi familia en especial a mis padres por haberme apoyado durante todos estos años y a mi hermana por servirme de inspiración.

Alejandro Rodríguez Fernández

RESUMEN

Actualmente muchos proyectos de despliegue de Software de Gestión Empresarial fracasan teniendo como principal causa la incorrecta evaluación tecnológica durante la implantación del software. El método para la evaluación tecnológica de proyectos de despliegue de Sistemas de Gestión Empresarial propuesto en (Capeáns 2014) tiene como finalidad realizar la evaluación de la compatibilidad del software con la Infraestructura Tecnológica Informática de la organización, con el objetivo de recolectar información sobre los componentes tecnológicos que posee esta y la posibilidad de hacer uso de los mismos en el despliegue. Sin embargo la aplicación del método se realiza de forma manual utilizando algoritmos matemáticos para el cálculo de distancia, proximidad relativa y el trabajo con matrices trayendo consigo que el tiempo necesario para su ejecución aumente. Es por ello que el presente trabajo de diploma persigue como objetivo general desarrollar un sistema informático que facilite la aplicación del método para la evaluación tecnológica de proyectos de despliegue de Software de Gestión Empresarial contribuyendo a la disminución del tiempo de desarrollo del mismo.

Para la construcción del sistema se utilizó la metodología de desarrollo para la actividad productiva de la Universidad de las Ciencias Informáticas, como marco de trabajo Symfony2, como sistema gestor de base de datos PostgreSQL 9.2 y entorno integrado de desarrollo NetBeans 7.4. El diseño y la implementación fueron evaluados cuantitativamente a través de las métricas de diseño y las pruebas de caja negra. Se realizó la validación de la investigación, confirmando la obtención de un sistema que permitió disminuir el tiempo de desarrollo del método.

Palabras claves: Evaluación de la compatibilidad tecnológica, despliegue de Software de Gestión Empresarial, Infraestructura Tecnológica Informática.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El constante desarrollo de las tecnologías y la necesidad de las empresas de ser cada vez más competitivas han incrementado en las últimas décadas la incorporación de sistemas informáticos para la gestión de información, denominados Software de Gestión Empresarial (SGE). Los SGE permiten llevar el control sobre una empresa, independientemente del sector en el que esta se encuentre, y de todas las partes que componen la misma, con el fin de tener un conocimiento automatizado que ayude en la toma de decisiones y en el manejo diario de todos los aspectos de la empresa. Sin embargo, teniendo factores como el compromiso de la organización, el cambio de la forma de trabajar de las personas e incompatibilidad del sistema con la infraestructura tecnológica informática (ITI del inglés *Infrastructure Information Technology*) ocurre que en muchas ocasiones fracasa el despliegue de dichos software.

Numerosos autores plantean que la incompatibilidad tecnológica es uno de los principales factores que pueden ocasionar el fracaso del despliegue. Según (Díaz *et al.* 2005) en la mayoría de las organizaciones, la implantación del SGE requiere reemplazar u optimizar la infraestructura existente. Esta actividad puede incrementar el riesgo del proyecto, debido a que este recibe una importante inyección de capital adicional, se requiere habilidades de especialización y, en algunos casos, la posibilidad de parar el negocio temporalmente para su implantación.

El sector empresarial en Cuba no cuenta, en su mayoría, con una ITI acorde a los requerimientos demandados por los actuales SGE. En aras de no prescindir de los beneficios que aporta el uso de la tecnología en este sector es preciso realizar un análisis objetivo, previo a la puesta en marcha de los sistemas informáticos que soportan los procesos productivos del país, que permita tener una visión clara de los recursos disponibles y adoptar las medidas necesarias para mejorar su uso.

El impacto tecnológico que puede ser provocado por la implantación de un SGE, debido a su complejidad, unido a los problemas económicos enfrentados por el sector empresarial cubano induce a que en muchas ocasiones no pueda ser gestionada la ITI idónea para el despliegue. Se hace necesario en todo momento garantizar la utilización óptima de los recursos disponibles. Teniendo en cuenta los elementos anteriores, es esencial analizar la compatibilidad del software de gestión empresarial con la ITI, que se basa en la realización de una evaluación de la tecnología existente en la organización. Este análisis tiene como objetivo recolectar información sobre los componentes

tecnológicos que posee la organización y la posibilidad de hacer uso de los mismos en el despliegue del software de gestión propuesto y de ser necesario, los requisitos tecnológicos que deben ser adquiridos para la puesta en marcha del software en cuestión.

En (Capeáns 2014) se propone un método para la evaluación tecnológica de proyectos de despliegue de Software de Gestión Empresarial que contribuye a mejorar el tiempo de dichos proyectos a partir del análisis de la compatibilidad tecnológica de un determinado software con la ITI de la organización donde este será desplegado. Sin embargo la aplicación dicho método se puede ver afectada, debido a la complejidad de los diferentes algoritmos matemáticos que utiliza, lo cual afecta el tiempo de ejecución del mismo.

En el análisis de los resultados de la aplicación del método en (Capeáns 2014), la usabilidad es el indicador más bajo de los evaluados. Esto se debe a que en el desarrollo del método una de las actividades que se realiza es la identificación de los criterios de evaluación, donde en caso de que un proyecto tenga asociados muchos indicadores, los cálculos se pueden tornar aún más complejos y se puede atrasar de forma significativa la evaluación realizada, destinándose a estos cálculos el tiempo que podría ser empleado en mejorar la estimación del peso de cada criterio o en la obtención de los expertos que son factores que influyen directamente en la efectividad del método.

Teniendo en cuenta lo expuesto anteriormente se plantea como **problema a resolver:** La aplicación del método para la evaluación tecnológica de proyectos de despliegue de Software de Gestión Empresarial se torna complicada debido a sus cálculos matemáticos lo cual afecta el tiempo de desarrollo del método.

Como **objeto de estudio** se define los procesos de despliegue de Software de Gestión Empresarial, centrando el **campo de acción** en los sistemas de evaluación de la compatibilidad tecnológica en el despliegue de Software de Gestión Empresarial.

Se traza como **objetivo general:** Desarrollar un sistema informático que facilite la aplicación del método para la evaluación tecnológica de proyectos de despliegue de Software de Gestión Empresarial contribuyendo a la disminución del tiempo de desarrollo del método.

Para alcanzar el objetivo general se definen los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación a partir de la revisión bibliográfica de los principales conceptos asociados a la evaluación tecnológica en el despliegue de Software de Gestión Empresarial y del estudio de las tecnologías necesarias para el desarrollo de un software que informatice dicho proceso.
- Realizar el análisis y diseño del sistema para la evaluación tecnológica en los despliegues de Software de Gestión Empresarial.
- Implementar el sistema para la evaluación tecnológica en los despliegues de Software de Gestión Empresarial.
- Validar la propuesta de solución mediante pruebas de Caja negra y el objetivo de la investigación a través de un pre experimento.

Según el problema planteado se propone como **hipótesis:** Si se desarrolla un sistema informático que facilite la aplicación del método para la evaluación tecnológica de proyectos de despliegue de Software de Gestión Empresarial se contribuirá a disminuir el tiempo de desarrollo del método.

Para el **diseño de la investigación** se utiliza el pre experimento pretest - postest de un solo grupo, el cual consiste en observar o medir un determinado grupo para luego introducir un factor que producirá algún cambio en el mismo y por último, volver a observar o medir el grupo.

A raíz de la presente investigación se espera como **posible resultado:** Sistema informático para la evaluación tecnológica en los despliegues de Software de Gestión Empresarial.

Para dar solución a los objetivos específicos se definen las siguientes **tareas de investigación:**

1. Búsqueda de bibliografía asociada al objeto de estudio.
2. Selección de la bibliografía relevante.
3. Análisis de la bibliografía seleccionada.
4. Definición de las herramientas y tecnologías a utilizar para el desarrollo de la propuesta de solución.

5. Definición del modelo de desarrollo de software a utilizar.
6. Construcción de los artefactos asociados al modelo de desarrollo de software seleccionado.
7. Implementación de las funcionalidades definidas.
8. Validación de las funcionalidades del software implementadas mediante pruebas de caja negra.
9. Validación del objetivo de la investigación.

Para el desarrollo de la investigación fueron utilizados varios **métodos científicos**, tanto teóricos como empíricos los cuales facilitaron el progreso y correcto avance de la misma.

Métodos teóricos

Analítico Sintético: Se utiliza en el análisis de la documentación relacionada con el proceso de despliegue de software de gestión empresarial, extrayendo los elementos más importantes relacionados con el campo de acción.

Hipotético Deductivo: Este método es utilizado en el planteamiento de la hipótesis sobre la cual se desarrolla la investigación.

Modelación: Se utiliza para realizar modelos y diagramas con el objetivo de lograr una mejor visibilidad y entendimiento de cómo se debe desarrollar el sistema.

Métodos empíricos

Entrevista: Se utiliza en el intercambio con el cliente[1] para adquirir información sobre el negocio y los requisitos que se deben cumplir en el desarrollo del sistema.

[1] Teniendo en cuenta que el método a implementar no está siendo utilizado, funciona como cliente para el desarrollo de la solución el MsC Carlos Abel Capeáns Hurtado. Sin embargo los clientes potenciales son los proyectos de despliegue de SGE de la Universidad de las Ciencias Informáticas.

La **estructura del trabajo** consta de introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. Para un mejor entendimiento de este documento, a continuación se describe brevemente cada uno de los capítulos en que se encuentra estructurado.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA: En este capítulo se recogen los principales términos y conceptos relacionados con el tema de investigación. Se lleva a cabo un estudio para el entendimiento del método para la evaluación tecnológica de proyectos de despliegue de software de gestión empresarial. Además se justifica el empleo de cada una de las herramientas, metodologías, lenguajes de desarrollo, marcos de trabajo y tecnologías para el desarrollo de la solución propuesta.

CAPÍTULO II: ANÁLISIS Y DISEÑO: Este capítulo tendrá como núcleo fundamental el análisis y diseño del sistema a implementar. Se realiza el modelado y descripción del negocio y se presenta el modelo conceptual asociado al dominio de la investigación. De igual manera se detallan las técnicas de captura de requisitos empleadas, se especifican detalladamente los requisitos funcionales y no funcionales y se especifican las técnicas de validación utilizadas. También se presentan los artefactos generados según la metodología escogida y el diseño de la solución propuesta.

CAPÍTULO III: IMPLEMENTACIÓN Y VALIDACIÓN: En este último capítulo se realiza la descripción de la implementación, mostrando la estructura del sistema, los diagramas obtenidos en esta disciplina y los estándares de codificación usados. Posteriormente se realiza la validación del diseño mediante métricas y la validación del sistema obtenido mediante las pruebas de software especificadas. Por último se valida que la propuesta de solución cumpla con el objetivo planteado.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se hace referencia a los principales términos y conceptos relacionados con el tema de investigación. Se caracteriza el método para la evaluación tecnológica en el despliegue de SGE propuesto para el desarrollo del sistema, mostrando sus principales procedimientos y ecuaciones. Muestra además un análisis del entorno de desarrollo donde se trabajó con el objetivo de fundamentar la selección de la metodología, herramientas de desarrollo, lenguajes de programación y patrones de software que fueron utilizados para el análisis, diseño e implementación de la solución propuesta.

1.2 Marco conceptual

Para un mejor entendimiento y comprensión de la investigación, a continuación se definen una serie de conceptos relacionados con la temática abordada a partir de una revisión bibliográfica realizada.

Software de Gestión Empresarial

Un Software de Gestión Empresarial (SGE) es un paquete de software que permite a las empresas evaluar, implementar, automatizar, integrar y gestionar de forma eficiente las diferentes operaciones que se presentan en estas. Se caracteriza por su modularidad, integración de la información, universalidad, estandarización e interfaces con otras aplicaciones, son sistemas abiertos y en la mayoría de los casos multiplataforma (Finazzi 2013).

(Hardcastle 2008) Plantea que los mismos pueden ser utilizados para apoyar la planificación, control, coordinación, toma de decisiones y las actividades operacionales de la organización.

Para la investigación se define SGE como un sistema de información integral que soporta los procesos de una organización. Este permite llevar el control sobre una empresa con el fin de tener un conocimiento automatizado que ayude en la toma de decisiones y facilite la gestión de todos los recursos, a través de la integración de la información de los distintos departamentos y áreas funcionales.

Despliegue de software

El despliegue de software es el conjunto de actividades que hacen que un sistema de software esté disponible para su uso, desde la liberación del software al final del ciclo de desarrollo hasta la instalación y activación del mismo (Dearle 2007).

En (Figueredo 2009) se define despliegue como el proceso mediante el cual se instalan, configuran y cargan los datos necesarios de un software con el objetivo de perfeccionar el flujo de información de una entidad, mejorando la eficiencia de los procesos que esta realiza. En el despliegue también se realiza una transformación organizacional en función de las características de las soluciones que se adquieran. Esta incluye el entrenamiento de los usuarios y la construcción de archivos de datos necesarios para utilizar el producto instalado.

A partir del estudio de algunas de las metodologías de despliegue utilizadas a nivel nacional e internacional se identifican las principales actividades asociadas al despliegue de software (Desoft 2007; Isasi-Genix y Gómez-Acosta 2012; Megal 2004; Rivera 2011; Tomé 2009):

- Planificación: Conjunto de actividades que permiten el diagnóstico de las entidades, determinación de recursos humanos, materiales necesarios y preparación de la formación.
- Formación del personal: Implementar el plan de capacitación de las aplicaciones que conforman la solución informática.
- Puesta en marcha: Entrega del sistema en su totalidad ajustado a las condiciones del cliente y la formación de los usuarios finales en los procesos de negocio y el uso del sistema.
- Evaluación del proyecto: Obtener información sobre el desempeño del proceso y lograr su mejora.

Los términos despliegue e implantación son utilizados indiferentemente durante el transcurso de la investigación para referirse a proceso de distribución del software a los usuarios finales, incluyendo todas las actividades orientadas a poner en producción el software y a capacitar a los usuarios para el uso del mismo.

Infraestructura tecnológica informática

Se denomina Infraestructura Tecnológica Informática (ITI del inglés *Infrastructure Information Technology*) a los recursos informáticos de una organización que constituyen una base para las aplicaciones empresariales actuales y futuras. Esta comprende activos físicos, activos intelectuales y procedimientos relacionados (Dai *et al.* 2005). Los activos físicos son los componentes técnicos compartidos a través de las unidades de la empresa. Estos incluyen plataformas, arquitecturas comunes, redes y bases de datos. Los activos intelectuales se refieren a la experiencia y gestión de conocimientos tecnológicos que existen en la empresa. Los procedimientos son reglas que especifican cómo se evalúan, adquieren, implementan y utilizan otros activos de la ITI(Dai *et al.* 2005).

Se identificaron tres enfoques teóricos de ITI en la literatura. El enfoque tecnológico se basa en una arquitectura de componentes técnicos, compartida en toda la organización. Los investigadores que solo han utilizado el dominio técnico para definir ITI utilizan consistentemente cuatro categorías: plataformas, redes y telecomunicaciones, datos y aplicaciones básicas. El enfoque orientado a componentes adopta una perspectiva más amplia de ITI, vista como que tiene dos elementos distintos, técnicos y humano. Los componentes técnicos no son diferentes a las del enfoque anterior. Los componentes humanos se refieren a los conocimientos y habilidades que posee el personal de la organización. Estos dos enfoques teóricos se centraron en la estructura de ITI, o sea sus componentes. El enfoque orientado a proceso toma un punto de vista más amplio, incorporando los procesos y actividades que utilizan estos componentes(Fink y Neumann 2009). Para el desarrollo de la investigación se decide utilizar el enfoque orientado a componentes, debido a que este tiene en cuenta el factor tecnológico y el factor humano como parte de la ITI en el despliegue de software.

Teniendo en cuenta lo mencionado anteriormente se puede definir la ITI como parte del núcleo fundamental de la implantación de los SGE, formando parte de ella todos los elementos tecnológicos que son utilizados en la entidad para dar soporte a las aplicaciones informáticas, así como las habilidades y conocimientos técnicos que posee el personal relacionado al proceso. Entre ellos se encuentran los servidores, redes, sistemas operativos, bases de datos, lenguajes de programación, herramientas de soporte y administración, además de todos los conocimientos informáticos y tecnológicos que ostentan las personas que laboran en esta área.

Evaluación de la compatibilidad tecnológica

La evaluación tecnológica, es un conjunto de principios, métodos y técnicas para la efectiva evaluación del valor potencial de una tecnología y su contribución a la competitividad y la rentabilidad de la empresa. Esta debe proporcionar información para ayudar a las personas involucradas en el desarrollo de estrategias que permitan examinar nuevas ideas, identificar y analizar las causas o cambio potencial, desarrollar y planificar las posibles soluciones y, finalmente, seleccionar e implementar la tecnología propuesta (Bakouros 2000).

La compatibilidad designa a la condición que posibilita que un programa y un sistema consigan un cierto entendimiento entre sí (Hohenegger *et al.* 2007). Se puede definir compatibilidad en el contexto tecnológico como la capacidad que tiene cierto sistema para funcionar simultáneamente con otros sistemas, permitiendo o mejorando el funcionamiento del conjunto. Se entiende por sistema tanto software como hardware, de modo que se puede analizar la compatibilidad entre dos software, entre un software con un hardware o entre dos hardware (Capeáns 2014).

Por tanto podemos definir evaluación de la compatibilidad tecnológica como un proceso destinado a evaluar la capacidad que tiene el software de gestión empresarial para funcionar conjuntamente con la infraestructura informática tecnológica existente en una organización. Para el desarrollo y despliegue de este software se debe analizar que este sea compatible con el máximo número de sistemas posibles y su impacto teniendo en cuenta sus necesidades básicas, en cuanto a herramientas y tecnologías necesarias, para distintos escenarios, dígase pequeñas, medianas o grandes empresas.

1.3 Método para le evaluación tecnológica de proyectos de despliegue de Software de Gestión Empresarial

Luego de realizar un estudio enmarcado en el campo de acción definido, no fueron encontrados sistemas informáticos destinados a realizar la evaluación de la compatibilidad tecnológica en el despliegue de SGE. Además se realizó un estudio de las principales metodologías y métodos de despliegue utilizadas actualmente a nivel nacional e internacional, los cuales no profundizan en la evaluación tecnológica, encontrando como única guía para desarrollar la propuesta de solución el método para la evaluación tecnológica de proyectos de despliegues de SGE. A continuación se realiza una pequeña explicación del método anteriormente mencionado.

El método propuesto en (Capeáns 2014) está compuesto por tres fases (Planificación, Desarrollo y Evaluación). Cada una de las fases está formada por actividades, y cada actividad describe el contenido haciendo referencia a sus principales acciones, artefactos y roles participantes. En el mismo se utilizan los métodos multicriterios AHP (del inglés *Analytic Hierarchy Process*) y TOPSIS (del inglés *Technique for Order Preference by Similarity to Ideal Solution*), los cuales se basan en la consulta a un grupo o grupos de personas que tienen grandes conocimientos sobre un tema determinado (Capeáns 2014). A continuación en la Figura 1 se presenta la estructura de dicho método.

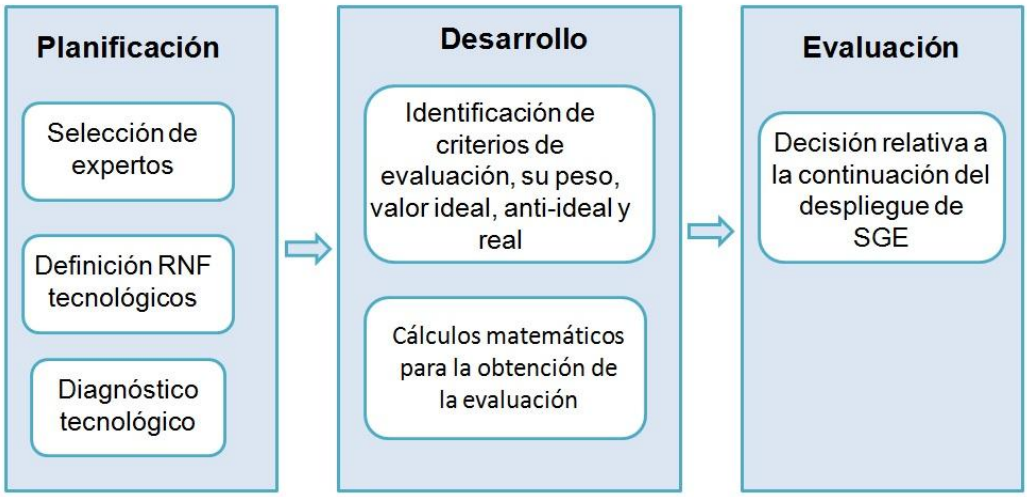

Figura 1: Método para la evaluación tecnológica de despliegues de SGE

En la fase Planificación se realizan las actividades selección de expertos, definición de los Requisitos No Funcionales (RNF) tecnológicos y el diagnóstico tecnológico. La selección de los expertos tiene como objetivo seleccionar un grupo de especialistas con conocimientos sobre el despliegue de software. Cada experto podrá aportar a la discusión general la idea que tiene sobre el criterio a evaluar desde sus conocimientos. La definición de los requisitos no funcionales tecnológicos relacionados al software en despliegue es realizada para obtener el entorno donde el mismo debe funcionar. Sin embargo el diagnóstico tecnológico es realizado a la organización donde será implantado el software con el fin de conocer las características tecnológicas que esta presenta.

En la fase Desarrollo se efectúa la identificación de los criterios de evaluación y el peso asociado a cada uno de estos a través de AHP, determinación de la ITI ideal y anti-ideal, determinación de la ITI real, cálculo de la distancia, cálculo de proximidad relativa y salida de valores lingüísticos. A partir de

la selección de los criterios de evaluación son identificados los específicos para el software a desplegar y cada experto realiza un análisis de la importancia de cada uno de ellos con respecto a los demás para identificar el peso asociado a ellos. El peso de cada criterio se obtiene a partir del promedio del peso obtenido en la evaluación de cada experto. La determinación de la ITI ideal, anti-ideal y real es la evaluación de los criterios definidos a partir de las etiquetas lingüísticas que serán modeladas mediante números triangulares difusos. El cálculo de distancia se realiza entre la ITI real de la organización con respecto a la solución ideal positiva (ITI ideal) y a la solución ideal negativa (ITI anti-ideal) respectivamente. Se utiliza el método del vértice con el objetivo de calcular la distancia entre conjuntos difusos.

Sean A = $(a_1,\ a_2,\ a_3)$ y B = $(b_1,\ b_2,\ b_3)$ dos números triangulares difusos, se plantea la ecuación del método del vértice para calcular la distancia entre ellos:

$$\boldsymbol{d}_{(A,B)} = \sqrt{\frac{1}{3}\left[\left(a_1 - b_1\right)^2 + \left(a_2 - b_2\right)^2 + \left(a_3 - b_3\right)^2\right]} \qquad (1)$$

Sean $\overline{\boldsymbol{v}_J^+}$ y $\overline{\boldsymbol{v}_J^-}$ números triangulares difusos que constituyen los valores ideal y anti-ideal de cada criterio y $\overline{\boldsymbol{v}_{iJ}}$ el valor real de cada criterio para cada alternativa. Para el cálculo de la distancia entre la evaluación de cada criterio de la ITI real y las alternativas ficticias de la ITI ideal se utiliza la Ecuación 2, donde $\overline{\boldsymbol{D}_J^+}$ constituye el valor de la distancia ideal de cada alternativa:

$$\overline{\boldsymbol{D}_J^+} = \sum_i^n \boldsymbol{d}_{(\overline{\boldsymbol{v}_{iJ}},\overline{\boldsymbol{v}_J^+})} \qquad (2)$$

Teniendo en cuenta que en muchas ocasiones ocurre que la evaluación real de un determinado criterio es mejor que la ideal propuesta, el cálculo de distancia del valor real de cada criterio a su ideal se realiza como se muestra a continuación:

$$\boldsymbol{d}_{(\overline{\boldsymbol{v}_{iJ}},\overline{\boldsymbol{v}_J^+})} = \begin{cases} 0,\ \overline{\boldsymbol{v}_{iJ}} > \overline{\boldsymbol{v}_J^+} \\ Ecuación\ 1,\ \overline{\boldsymbol{v}_{iJ}} \leq \overline{\boldsymbol{v}_J^+} \end{cases}$$

Para el caso de la ITI anti-ideal se utiliza la Ecuación 3, donde $\overline{\boldsymbol{D}_J^-}$ constituye el valor de la distancia ideal de cada alternativa:

$$\overline{D_J^-} = \sum_i^n d_{(\overline{v_{ij}}, \overline{v_J^-})} \qquad (3)$$

$$d_{(\overline{v_{ij}}, \overline{v_J^-})} = \begin{cases} 0, & \overline{v_{ij}} < \overline{v_J^-} \\ Ecuación\ 1, & \overline{v_{ij}} \geq \overline{v_J^-} \end{cases}$$

Después de realizar el cálculo de distancias de la ITI real con respecto a la solución ideal positiva y negativa se realiza el cálculo de la proximidad relativa de la ITI real a la solución ideal positiva y negativa mediante el índice de proximidad, así como de las alternativas ficticias correspondientes a las etiquetas lingüísticas asociadas a la investigación.

Siendo $\overline{R_J}$ la proximidad relativa de cada alternativa su valor se calcula mediante la Ecuación 4.

$$\overline{R_J} = \frac{\overline{D_J^-}}{\overline{D_J^+} + \overline{D_J^-}} \qquad (4)$$

Luego de obtener el coeficiente de proximidad, los valores obtenidos son reales y por lo tanto tenemos un orden definido entre ellos, así como la posibilidad de encontrar la mínima distancia de la ITI real a las alternativas ficticias.

En la fase Evaluación se toma la decisión relativa a la continuación del despliegue del software teniendo en cuenta el resultado de la salida de los valores lingüísticos, lo que conlleva a la realización o no del despliegue.

El método comprende cuatro roles principales de los cuales el líder del proyecto, el analista y los expertos forman parte del proyecto desarrollador del software, mientras que por el lado del cliente se encuentra el informático de la organización.

1.4 Entorno de desarrollo

La correcta selección de las herramientas, lenguajes y tecnologías que se utilizan en el desarrollo de un software se traduce en ahorro de tiempo y trabajo dentro de cualquier proyecto, además influye en la calidad del producto. A continuación se expone una breve descripción de cada uno de los elementos que serán utilizados.

1.4.1 Metodología de desarrollo

El uso de una metodología es esencial para guiar el desarrollo de un sistema informático. En dependencia de las características de un determinado proyecto, la correcta selección de la metodología puede representar el éxito o el fracaso de dicho proyecto.

Se propone para el ciclo de vida del software la metodología de desarrollo para la Actividad productiva de la Universidad de las Ciencias Informáticas (UCI), la cual se fundamenta en una variación de la metodología Proceso Unificado Ágil (AUP del inglés *Agile Unified Process*) en unión con el modelo CMMI (del inglés Capability Maturity Model Integration)-DEV v 1.3. AUP es una versión simplificada del Proceso Unificado Racional (RUP del inglés *Rational Unified Process*) el cual describe de una forma fácil de entender cómo desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

La metodología seleccionada establece las diferentes fases por las que se debe transitar durante el desarrollo de un producto de software y los distintos artefactos que se generan en cada una de ellas. Las fases definidas son: Inicio, Ejecución, y Cierre. La solución que se desea implementar se enmarca en la fase de Ejecución, que tiene como objetivo obtener un sistema que satisfaga las necesidades de los usuarios finales. Esta fase cuenta con 8 disciplinas definidas de las cuales se utilizaran 6 de ellas, las mismas son: modelado del negocio, requisitos, análisis y diseño, implementación, pruebas internas y pruebas de aceptación (Rodríguez 2014).

Se decide utilizar esta metodología, debido a que la misma es establecida por la universidad para el desarrollo de software con el objetivo de erradicar la variedad de estas en los proyectos y además la misma se ajusta a las necesidades de desarrollo de la solución.

1.4.2 Lenguaje de modelado

Realizar el modelado de un sistema es fundamental para lograr una correcta estructura y organización del mismo. El lenguaje unificado de modelado (UML del inglés *Unified Modeling Language)* es un lenguaje de modelado visual que permite la modelación, visualización, especificación, construcción y documentación de los artefactos de un software, creando una idea inicial de cómo quedará estructurado el proyecto. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para

entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas (Rumbaugh *et al.* 2000).

Como lenguaje de modelado se decide utilizar UML en su versión 2.0 porque este permite modelar sistemas haciendo uso de técnicas orientadas a objetos. Además especifica todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.

1.4.3 Notación de modelado

La Notación para el Modelado de Procesos de Negocio (BPMN del inglés *Business Process Modeling Notation*) es una notación gráfica que describe la lógica de los pasos de un proceso de Negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades. BPMN permitirá modelar los procesos de forma unificada y estandarizada, creando un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos (Bizagi 2014). Además nos brinda una representación de la secuencia de actividades y los artefactos que se generan en el modelado de negocio.

1.4.4 Lenguaje de programación

PHP Hypertext Preprocessor (PHP) es un lenguaje de código abierto del lado del servidor que está especialmente pensado para el desarrollo web y que puede ser incrustado en páginas HTML. Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales (Php 2015).

Se escoge como lenguaje de programación PHP en su versión 5.4, por presentar entre sus principales características (Php 2015):

- Puede emplearse en todos los sistemas operativos principales, incluyendo Linux, muchas variantes de Unix, Microsoft Windows, Mac OS X, RISC OS y probablemente otros más.
- Admite la mayoría de servidores web de hoy en día, incluyendo Apache y otros.
- Soporte para un amplio abanico de bases de datos.

- Tiene útiles características de procesamiento de texto, las cuales incluyen las expresiones regulares y muchas extensiones.

1.4.5 Herramienta de modelado

Visual Paradigm for UML es una herramienta de ingeniería de software asistida por computación (CASE del inglés *Computer Aided Software Engineering*) que proporciona asistencia a los analistas, ingenieros de software y desarrolladores. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Además permite generar código desde diagramas, generar documentación, realizar ingeniería tanto directa como inversa, entre otras funciones. Puede integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad (Paradigm 2015). Esta herramienta en su versión 8.0 fue la seleccionada para la creación de modelos y diagramas necesarios durante todo el ciclo de vida de la solución y entre sus principales características se pueden mencionar (Paradigm 2015):

- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.

1.4.6 Entorno de Desarrollo Integrado

NetBeans es un entorno de desarrollo integrado (IDE del inglés *Integrated Development Environment*) hecho principalmente para Java pero puede servir para cualquier otro lenguaje de programación. Es una herramienta pensada para escribir, compilar, depurar y ejecutar programas, donde existe además, un número importante de módulos para extenderla. Este IDE es libre y gratuito, sin restricciones de uso. Presenta una biblioteca extensa, así como varias características de utilidad que facilitan el trabajo

con el mismo, como son: auto completamiento de código, visor de clases, métodos y componentes y ayuda incluida (Netbeans 2015). Se elige como IDE NetBeans en su versión 7.4 debido a las características mencionadas anteriormente.

1.4.7 Marco de trabajo

Symfony2 es un marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web que separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Symfony2 está desarrollado completamente con PHP 5 y es compatible con la mayoría de gestores de Bases de Datos, como MySQL, PostgreSQL, Oracle y SQL. Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows. Este marco de trabajo cuenta con las siguientes características (Eguiluz 2013):

- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Independiente del sistema gestor de bases de datos.
- Fácil de extender, lo que permite su integración con bibliotecas desarrolladas por terceros.

Las características antes mencionadas se ajustan perfectamente a las necesidades de la solución propuesta. Symfony2 es un marco de trabajo sencillo que permite que el equipo de trabajo se familiarice fácilmente con él. Este admite el uso de diferentes patrones, entre ellos el patrón arquitectónico Modelo Vista Controlador que es seleccionado para el desarrollo de la propuesta de solución. A partir de lo anteriormente expuesto se decide utilizar Symfony2 como marco de trabajo.

1.4.8 Sistema Gestor de Base de Datos

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[2] y con su fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. Este sistema gestor utiliza un modelo cliente/servidor y usa multiprocesos, en vez de multihilos, para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Se selecciona PostgreSQL en su versión 9.2 por presentar las siguientes características (Postgresql 2009):

- Soporta distintos tipos de datos.
- Incluye herencia entre tablas.
- Disponible para Linux y UNIX en todas sus variantes y Windows.

1.4.9 Servidor web

Apache es un servidor de páginas web que permite acceder a páginas alojadas en un ordenador. Es un software de código abierto multiplataforma(Apache 2012). Se decide utilizar este servidor en su versión 2.4 por ser una herramienta modular, que permite incorporarle nuevas funcionalidades. Además de contar con abundante documentación y ser capaz de soportar varios lenguajes de programación, dentro de los que se encuentran los seleccionados para la solución propuesta.

1.5 Patrones para el desarrollo de software

Según (Larman 1999) un patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos. Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software sino intentan codificar el conocimiento, las expresiones y los principios ya existentes (Larman 1999). Para un mejor diseño de la solución se emplearan un grupo de patrones de diseños así como un patrón arquitectónico.

[2] Del inglés *Berkeley Software Distribution*

1.5.1 Patrones de Diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos fundamentales de una estructura común de diseño que la hacen útil para la creación de un diseño orientado a objetos reutilizable. Los patrones de diseño están orientados a describir el cómo se debe construir el software de manera tal que se utilicen correctamente las clases y los objetos que lo componen (Gamma *et al.* 2003). Para la implementación del sistema se emplearon varios patrones de diseño tanto GRASP (del inglés *General Responsibility Asignment Software Patterns*) como GOF (del inglés *Gang of Four*). La utilización de estos patrones refleja un diseño sencillo, pensado para la creación de un sistema robusto y fácil de entender.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman 1999). A continuación se mencionan los patrones que fueron utilizados en el diseño del software para la asignación de responsabilidades:

- Experto: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- Creador: Asignarle a la clase B la responsabilidad de crear una instancia de clase A.
- Alta Cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta.
- Bajo Acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento.
- Controlador: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase (Larman 1999).

Existen 23 patrones GOF que se clasifican según su propósito en Creacionales, Estructurales y de Comportamiento y según su ámbito en Objeto y de Clase.

- Creacionales: abstrae el proceso de instanciación de objetos. Su misión es permitir construir sistemas independientes de la forma de creación, composición o representación de objetos.
- Estructurales: controla como se combinan las clases u objetos en la construcción de estructuras mayores. Un patrón estructural de clases utiliza la herencia para componer interfaces o

implementaciones. Un patrón estructural de objetos describe la forma en que se componen objetos para obtener nueva funcionalidad.

- De Comportamiento: se relaciona con algoritmos, la forma en la que interactúan las clases u objetos y la asignación de responsabilidades entre ellos. Los patrones de comportamiento de clases utilizan la herencia para distribuir el comportamiento entre las clases. Por su parte los patrones de comportamiento de objetos cooperan como un grupo de objetos interconectados para realizar una tarea que un solo objeto no puede realizar por sí solo (Gamma *et al.* 2003).

Dentro de este grupo de patrones la solución propuesta hará uso de los patrones Método de Fábrica y Decorador.

El uso de estos patrones garantizó asignar a cada clase la responsabilidad que le corresponde, obtener el menor número de relaciones y dependencias entre clases, aumentando las posibilidades de reutilización de las mismas. Todos estos elementos ayudaron a la confección de un diseño de la solución con claridad y rendimiento.

1.5.2 Patrones de Arquitectura

Los patrones arquitectónicos representan el nivel más alto dentro del sistema de patrones y expresan el esquema de la estructura fundamental de la organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. La selección de un patrón arquitectónico debe ser conducida por las propiedades generales de la aplicación a desarrollar (Almeira y Pérez 2007).

El patrón arquitectónico utilizado para la implementación del sistema es el Modelo Vista Controlador (MVC del inglés *Model View Controller*). Este patrón divide una aplicación interactiva en tres componentes. El modelo contiene la funcionalidad central de la aplicación y los datos, las Vistas: despliegan la información al usuario y los Controladores: manejan la entrada del usuario (Almeira y Pérez 2007).

1.6 Pruebas de software

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Una vez generado el código

fuente, el software debe ser probado para descubrir (y corregir) el máximo de errores posibles antes de su entrega al cliente. Las pruebas requieren que se descarten ideas preconcebidas sobre la corrección del software que se acaba de desarrollar y se supere cualquier conflicto de intereses que aparezca cuando se descubran errores (Pressman 2006).

Las pruebas de caja negra también denominadas, pruebas de comportamiento, se concentran en los requisitos funcionales del software. Permiten al ingeniero del software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa (Pressman 2006). Según (Pressman 2006) para realizar este tipo de pruebas existen varias técnicas entre ellas que se encuentra:

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

Para validar la implementación de la solución propuesta se aplicarán las pruebas de caja negra, que permitirán probar el comportamiento del software a partir de un conjunto de datos de entrada, todo esto mediante de la técnica partición equivalente. Estas pruebas permitirán ejercitar completamente todos los requisitos funcionales implementados, centrándose en las funcionalidades que realiza la solución sin tener en cuenta su estructura interna.

1.7 Conclusiones del capítulo

Con el desarrollo del presente capítulo se pudo arribar a las siguientes conclusiones:

- ✓ Se enunciaron los principales términos asociados al dominio de la investigación, lo cual permitió adquirir un mayor conocimiento sobre el objeto de estudio de la investigación.
- ✓ El estudio del método para la evaluación tecnológica de despliegues de SGE permitió un mejor entendimiento para la posterior implementación de la solución.
- ✓ La identificación de los lenguajes, tecnologías y herramientas, así como la selección de la metodología de desarrollo a utilizar en la investigación, aportó un mayor conocimiento sobre los mismos, lo que facilitará su utilización.

CAPÍTULO II: ANÁLISIS Y DISEÑO

2.1 Introducción

En el presente capítulo se puntualizan los elementos relacionados con el análisis y diseño del sistema a desarrollar, mostrando las características que presenta el negocio mediante el modelado y descripción de los procesos de negocio. Se muestran los requisitos funcionales y no funcionales que fueron definidos, así como las técnicas de captura y validación de requisitos utilizadas. Se describe el diseño del sistema, presentando los artefactos generados por la metodología seleccionada y los patrones de diseño que se evidencian en la implementación.

2.2 Modelado del negocio

Un proceso de negocio es una colección de actividades que tomando una o varias clases de entradas crean una salida que tiene valor para un cliente. Modelar el proceso de negocio es una parte esencial de cualquier proceso de desarrollo de software, pues permite al analista capturar el esquema general y los procedimientos que gobiernan el negocio. El modelado provee una descripción de dónde se va a ajustar el sistema de software (Sparks 2000). Esta actividad permite asegurar que los clientes, usuarios finales, desarrolladores y otros involucrados tengan una visión común de la organización.

2.2.1 Descripción del proceso de negocio

En el presente acápite se modela el proceso de negocio para la evaluación tecnológica en el despliegue de SGE. Para modelar el mismo se hizo uso de la notación BPMN, la cual permitió representar las etapas del proceso. Se relacionan así roles, eventos y artefactos que intervienen en su desarrollo. En la Figura 2 se muestra el modelo de procesos de negocio para el proceso de evaluación tecnológica en el despliegue de SGE, el cual se explica posteriormente. Para obtener mayor información sobre la descripción de este proceso de negocio consultar el documento entregable Descripción_de procesos_de_negocio_Evaluación_tecnlógica_en_el _despliegue_de_SGE.odt.

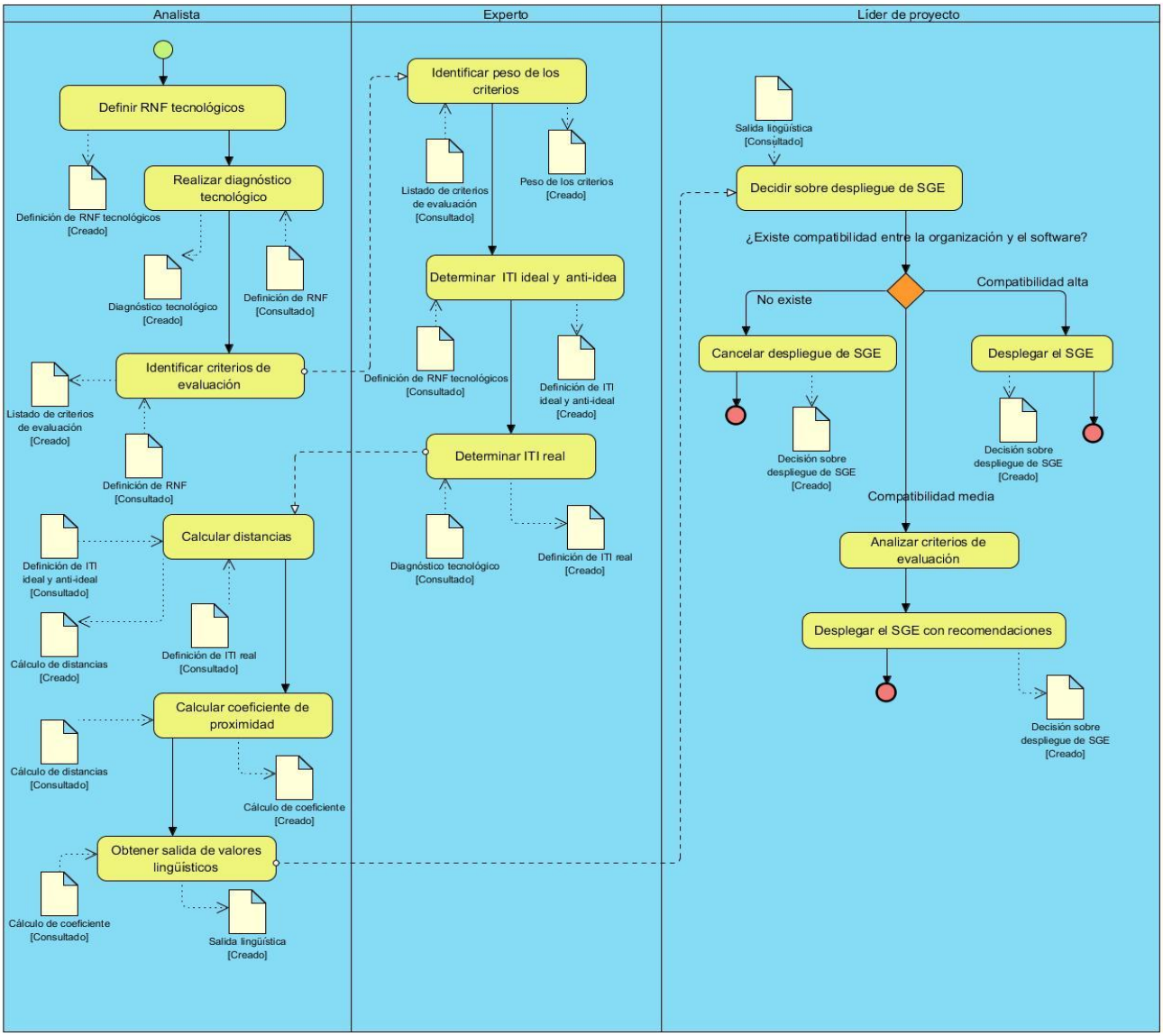

Figura 2: Proceso de negocio: Evaluación tecnológica en el despliegue de SGE

El proceso inicia con la definición de los requisitos no funcionales (RNF) tecnológicos del software. Esta actividad crea el artefacto Definición de RNF tecnológicos. Luego el analista realiza el diagnóstico tecnológico en la entidad consultando el documento creado anteriormente y teniendo como salida el Diagnóstico tecnológico. Después se identifican los criterios de evaluación consultando el artefacto Definición de RNF y creando como salida el Listado de criterios de evaluación. Posteriormente el experto consulta el listado creado anteriormente para pasar a identificar el peso de cada criterio. Esta actividad crea un artefacto llamado Peso de los criterios. De seguido el experto decide determinar la ITI ideal y la ITI anti-ideal de la empresa apoyándose en el documento Definición de RNF y creando

como salida la Definición de ITI ideal y anti-ideal. Seguidamente se pasa a determinar la ITI real de la organización, para realizar esta actividad se consulta el Diagnóstico tecnológico y crea el documento Definición de ITI real. Después el analista calcula las distancias, donde se consultan los documentos Definición de ITI ideal y anti-ideal y Definición de ITI real. Además el Cálculo de distancias pasa a ser creado. Consecutivamente se consulta el artefacto creado anteriormente para calcular el coeficiente de proximidad. En esta actividad se deja creado el Cálculo de coeficiente, el cual es consultado para obtener la salida de valores lingüísticos en la próxima actividad. La actual actividad crea el documento Salida lingüística que es necesario consultarlo para decidir sobre el despliegue. Si al realizar la decisión del despliegue no existe compatibilidad entre los valores de salida y los existentes en la organización, no se procede al despliegue del software y se finaliza el proceso. Si la compatibilidad entre los valores es alta, entonces se procede a la implantación del software y el proceso finaliza. En último caso si la compatibilidad es media se procede a realizar un análisis de los criterios de evaluación para determinar en qué casos no existe la suficiente compatibilidad entre el valor deseado y el existente en la entidad. En dependencia de la criticidad de estos se decide si se puede realizar el despliegue o no. En caso de que los criterios no sean críticos es decir que el SGE puede funcionar a pesar de que en su evaluación se procede al despliegue del software y se realizan recomendaciones en función de estos. Seguidamente finaliza el proceso.

2.3 Modelo conceptual

Un modelo conceptual explica los conceptos significativos en un dominio del problema; es el artefacto más importante a crear durante el análisis orientado a objetos (Larman 1999). Es considerado un diccionario visual de las abstracciones relevantes, vocabulario e informaciones del dominio. Constituye una primera aproximación al diseño definitivo. La Figura 3 refleja los principales conceptos del dominio, atributos y las relaciones entre ellos. Además en el documento entregable puede ser consultada la explicación de este artefacto.

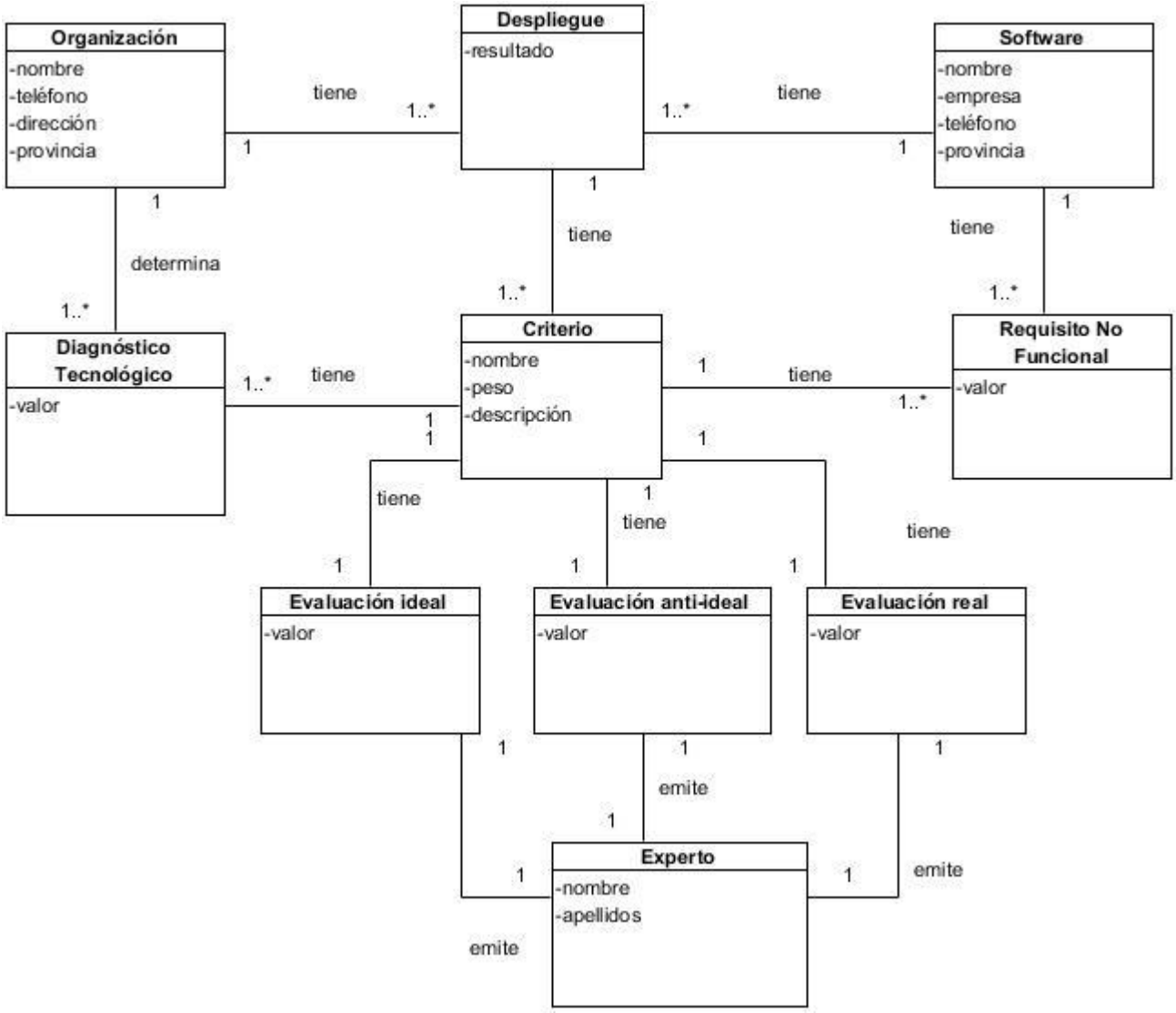

Figura 3: Modelo conceptual

2.4 Ingeniería de Requisitos

La ingeniería de requisitos es el proceso para establecer los servicios que el sistema debería proveer y las restricciones bajo las cuales debería operar y ser desarrollado. Es una actividad esencialmente de interacción con los interesados en el sistema (Sommerville 2005).

2.4.1 Obtención de requisitos

La obtención o captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. El proceso de obtención de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo, y depende mucho de las personas que participen en él (Escalona y Koch 2002). A continuación se presentan un grupo de técnicas que han sido utilizadas para esta actividad en el proceso de obtención de requisitos.

Entrevistas: esta técnica permite al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural (Escalona y Koch 2002). Las entrevistas realizadas, arrojaron información valiosa acerca de las necesidades y perspectivas de utilización del sistema, resaltando la importancia de desarrollar una aplicación sencilla, amigable y fácil de utilizar. En el Anexo 1 se puede encontrar el cuestionario utilizado para la entrevista.

Tormenta de ideas: es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la acumulación de ideas y/o información sin evaluar las mismas. Ofrece una visión general de las necesidades del sistema pero sin ofrecer detalles concretos (Escalona y Koch 2002). En varias reuniones se desarrollaron tormentas de ideas con la participación del equipo de desarrollo y el cliente. Como resultado, se logró generar una amplia gama de opiniones sobre los requisitos a satisfacer por el software.

2.4.2 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, la forma en que debe reaccionar ante ciertas entradas y cómo se debe comportar en situaciones particulares (Sommerville 2005).

Durante la disciplina de obtención de requisitos se identificaron 50 requisitos funcionales (RF) con los que debe contar la aplicación para satisfacer las necesidades requeridas por el cliente. A continuación en la Tabla 1 se muestra el listado de los mismos.

Tabla 1: Requisitos funcionales de software

Agrupación	No.	Funcionalidad	Agrupación	No.	Funcionalidad
Gestionar usuario	**RF 1**	Adicionar usuario	**Gestionar requisitos no funcionales**	**RF 25**	Adicionar requisitos no funcionales
	RF 2	Modificar usuario		**RF 26**	Modificar requisitos no funcionales
	RF 3	Eliminar usuario		**RF 27**	Eliminar requisitos no funcionales
	RF 4	Listar usuario		**RF 28**	Mostrar requisitos no funcionales
	RF 5	Buscar usuario	**Gestionar criterio**	**RF 29**	Adicionar criterio
Gestionar organización	**RF 6**	Adicionar organización		**RF 30**	Modificar criterio
				RF 31	Eliminar criterio
	RF 7	Modificar organización		**RF 32**	Listar criterio
	RF 8	Eliminar organización			
	RF 9	Listar organización		**RF 33**	Buscar criterio
	RF 10	Buscar organización	**Seleccionar criterios**	**RF 34**	Seleccionar criterios
Gestionar software	**RF 11**	Adicionar software		**RF 35**	Desmarcar criterios seleccionados.
				RF 36	Eliminar criterios seleccionados

			Identificar peso de los criterios	**RF 37**	Identificar peso de los criterios
				RF 38	Eliminar peso de los criterios
	RF 12	Modificar software	**Determinar evaluación ideal**	**RF 39**	Determinar evaluación ideal
	RF 13	Eliminar software			
	RF 14	Listar software		**RF 40**	Modificar evaluación ideal
	RF 15	Buscar software		**RF 41**	Eliminar evaluación ideal
Gestionar despliegue	**RF 16**	Adicionar despliegue	**Determinar evaluación anti-ideal**	**RF 42**	Determinar evaluación anti-ideal
	RF 17	Modificar despliegue			
	RF 18	Eliminar despliegue		**RF 43**	Modificar evaluación anti-ideal
	RF 19	Listar despliegue		**RF 44**	Eliminar evaluación anti-ideal
	RF 20	Buscar despliegue	**Determinar evaluación real**	**RF 45**	Determinar evaluación real
Gestionar diagnóstico tecnológico	**RF 21**	Adicionar diagnóstico tecnológico			
	RF 22	Modificar diagnóstico tecnológico		**RF 46**	Modificar evaluación real
	RF 23	Eliminar diagnóstico tecnológico		**RF 47**	Eliminar evaluación real

	RF 24	Mostrar diagnóstico tecnológico	**Evaluar despliegue**	**RF 48**	Evaluar despliegue
			Reportes	**RF 49**	Listar despliegues con resultados adversos
				RF 50	Listar criterios con mayor incidencias

2.4.3 Descripción de requisitos

La descripción de los requisitos es una versión completa del comportamiento de las funcionalidades que se van a desarrollar. Para su redacción se utiliza un lenguaje sencillo, de forma que sea fácilmente comprensible para todas las partes involucradas en el desarrollo. A continuación en la Tabla 2 se expone la descripción del requisito Adicionar despliegue perteneciente a la agrupación Gestionar despliegue.

Tabla 2: Descripción del requisito Adicionar despliegue

Precondiciones	Se ha registrado al menos un software en el sistema. Se ha registrado al menos una organización en el sistema.
Flujo de eventos	
Flujo básico Adicionar despliegue	
1. El usuario selecciona la opción Adicionar Despliegue.	
2. Se introducen los siguientes datos del despliegue: Software Organización	
3. Se presiona el botón Adicionar.	
4. El sistema valida los datos introducidos.	
5. Si los datos son correctos el sistema los registra.	
6. Concluye el requisito.	
Pos-condiciones	

1. Se registra un nuevo despliegue.

Flujos alternativos

Flujo alternativo 3.a Información errónea

1. El sistema muestra un mensaje al usuario señalando los datos erróneos y permite corregirlos.
2. El usuario corrige los datos.
3. Volver al paso 2 del flujo básico.

Pos-condiciones

N/A

Flujo alternativo 3.b Información incompleta.

1. El sistema señala los datos vacíos y permite corregirlos.
2. El usuario corrige los datos.
3. Volver al paso 2 del flujo básico.

Pos-condiciones

N/A

Flujo alternativo *.a El usuario cancela la acción.

1. Concluye el requisito.

Pos-condiciones

N/A

Validaciones

1. Se validan los datos según lo establecido en el Modelo conceptual MODELO CONCEPTUAL.doc.

Conceptos	**Visibles en la interfaz:** Software Organización **Utilizados internamente:** id id_organizacion id_software resultado

Requisitos especiales	N/A
Asuntos pendientes	N/A

2.4.4 Validación de requisitos

La validación de requisitos trata de mostrar que éstos realmente definen el sistema que el cliente desea (Sommerville 2005). Para la validación de los requisitos fue aplicada la siguiente técnica:

Prototipos: Esta técnica muestra un modelo ejecutable del sistema a los usuarios finales y los clientes, para que puedan ver si dicho modelo cumple con sus necesidades reales (Sommerville 2005). Una vez identificados y descritos cada uno de los requisitos, se diseñaron prototipos de interfaces de usuario utilizando como herramienta de modelado Visual Paradigm. Los mismos fueron presentados al cliente y estos arrojaron que cumplían con sus necesidades. Se realizaron los prototipos de cada uno de los requisitos que fueron identificados anteriormente. A continuación en la Figura 4 se muestra la validación realizada mediante prototipo para el requisito especificado anteriormente.

Figura 4: Prototipo para el requisito Adicionar despliegue

2.4.5 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el software. No se refieren directamente a funciones específicas, sino a sus propiedades emergentes; la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento son algunas de ellas (Sommerville 2005). A

continuación en la Tabla 3 se enuncian los requisitos no funcionales (RNF) definidos para la propuesta de solución:

Tabla 3: Requisitos no funcionales de software

Código	Clasificación
Usabilidad	
RNF 1	La interfaz debe ser sencilla y amigable de manera que potencie la comodidad del usuario para su trabajo
Eficiencia	
RNF 2	Los tiempos de respuesta y velocidad de procesamiento de la información en el sistema serán rápidos. En las ocasiones que el sistema esté realizando mayor carga de trabajo por encontrarse procesando y almacenando datos, estos tiempos no deben exceder de 5 a 8 segundos.
Fiabilidad	
RNF 3	La aplicación deberá estar disponible siempre, asegurando el acceso en todo momento desde cualquier lugar de red a todos los usuarios que estén autorizados.
Seguridad	
RNF 4	El acceso al sistema así como la información se encontrarán protegidos contra accesos no autorizados utilizando mecanismos de autenticación propios del sistema.
RNF 5	A cada usuario se le debe otorgar y dar acceso a las funcionalidades a las cuales está autorizado a acceder.
RNF 6	El administrador del sistema como política de seguridad podrá restringir el acceso a las diferentes áreas del sistema a los usuarios o grupos de usuarios.
Interfaz de usuario	
RNF 7	Las interfaces del sistema contendrán los datos de forma estructurada, permitiendo la interpretación correcta de la información.
RNF 8	La entrada incorrecta de datos será mostrada al usuario claramente, detallando los campos donde se encuentra el error y mostrando como título el detalle del error.
Hardware	
RNF 9	Requerimientos mínimos para el servidor: • Procesador Intel Dual Core o superior. • Tarjeta de red o capacidad de conectividad.

	• 1 GB de memoria RAM o superior. • Capacidad de disco duro 120 GB
RNF 10	Requerimientos mínimos para la conexión desde una PC cliente: • Procesador Intel Pentium III o superior. • 512 MB de RAM o superior.
Software	
RNF 11	Para el funcionamiento del servidor será necesario: • Sistema operativo Linux (cualquier distribución) o Windows XP o superior.
RNF 12	Para el funcionamiento en la PC cliente será necesario: • Navegador Mozilla Firefox 32 o superior.

2.5 Diseño de la solución

2.5.1 Arquitectura del sistema

La arquitectura del sistema está basada en el patrón arquitectónico Modelo Vista Controlador el cual viene integrado a Symfony2. Este patrón garantiza la organización del código fuente de la aplicación, dividiéndola en tres componentes fundamentales: la vista, el modelo y el controlador.

La **Vista** es la capa con la que interactúan directamente los usuarios finales, dentro de ella se ubica todo lo referente a la visualización de la información en las páginas de la aplicación. En Symfony2 se evidencia esta capa a través de la biblioteca jQuery, en conjunto con los archivos CSS, JavaScript y HTML.

El **Controlador** responde a eventos que se traducen en solicitudes de servicio para el modelo o la vista, gestionando la interacción entre el usuario y los datos. Esta capa queda evidenciada a través del Controlador Frontal, mediante los archivos app.php para entornos de producción y app_dev.php para entornos de desarrollo, los cuales se conciben para la ejecución de la aplicación en el servidor de producción y para el uso de los programadores respectivamente. Los archivos de enrutamiento también forman parte de esta capa los cuales permite determinar qué controlador está asociado con la página que se solicita. También forma parte de esta capa el motor de plantillas Twig, responsable de transformar cada archivo Twig en HTML. Por último las clases Controller, que controlarán el flujo de información que se recibe y se envía hacia la vista.

El **Modelo** administra el comportamiento y los datos del dominio de aplicación. Esta capa se divide en dos subcapas, la capa de negocio y la capa de acceso a datos. La subcapa de Negocio contiene las clases gestoras encargadas del manejo de la lógica del negocio. Estas clases desacoplan los componentes del negocio de las clases controladoras. La subcapa de Acceso a datos es la encargada de establecer la conexión con la base de datos. Dentro de esta capa se encuentra ubicado el ORM Doctrine, que posibilita la separación en la aplicación del SGBD (Sistema Gestor de Base de Datos).

Se encuentra además una capa llamada **Datos** donde se encuentra el Sistema Gestor de Base de Datos PostgreSQL el cual permite que persista la información con la que trabaja la aplicación y gestionar su almacenamiento.

La estructura descrita anteriormente se muestra en la Figura 5.

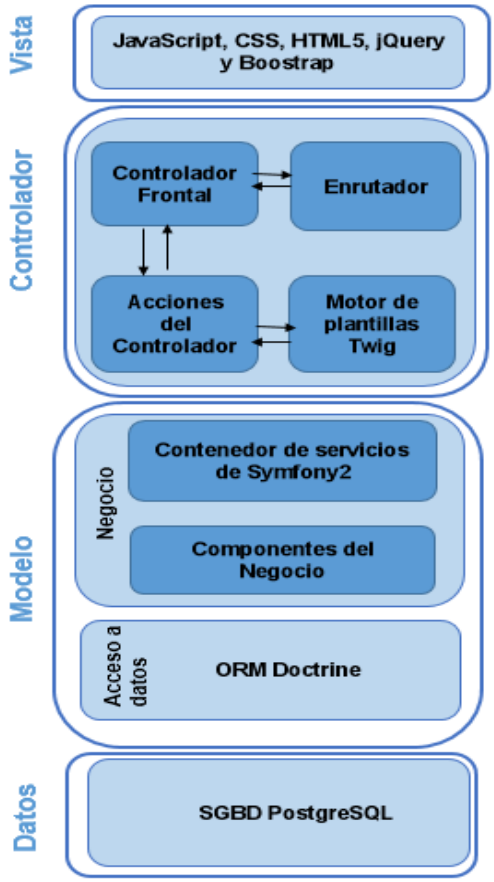

Figura 5: Arquitectura del sistema

2.5.2 Patrones de diseño

Para un mejor diseño de la solución se hace uso de patrones de diseño tanto GRASP como GOF. A continuación se describen los que fueron utilizados en el desarrollo de la propuesta de solución.

Patrones generales de software para asignación de responsabilidades (GRASP)

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

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. Se pone de manifiesto en las clases controladoras del sistema, las cuales crean los objetos de las clases que representan las entidades, evidenciando que las clases controladoras son creadoras de dichas entidades, además brinda soporte a un bajo acoplamiento y mejora la reutilización. En la Figura 6 se representa la utilización del patrón.

```
public function createAction(Request $request)     {
    $entity = new criterio();
    $form = $this->createCreateForm($entity);
    $form->handleRequest($request);

    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($entity);
        $em->flush();
```

Figura 6: Uso del patrón Creador en la clase CriterioController

Controlador: En la arquitectura definida este patrón se evidencia en las clases controladoras, responsables de implementar todas las funcionalidades pertenecientes a una interfaz determinada, además está presente en el controlador frontal de Symfony2. Todas las peticiones web son manejadas por dicho controlador frontal app.php. Una vez que este recibe la petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL introducida por el usuario. A continuación en la Figura 7 se muestra el uso de este patrón.

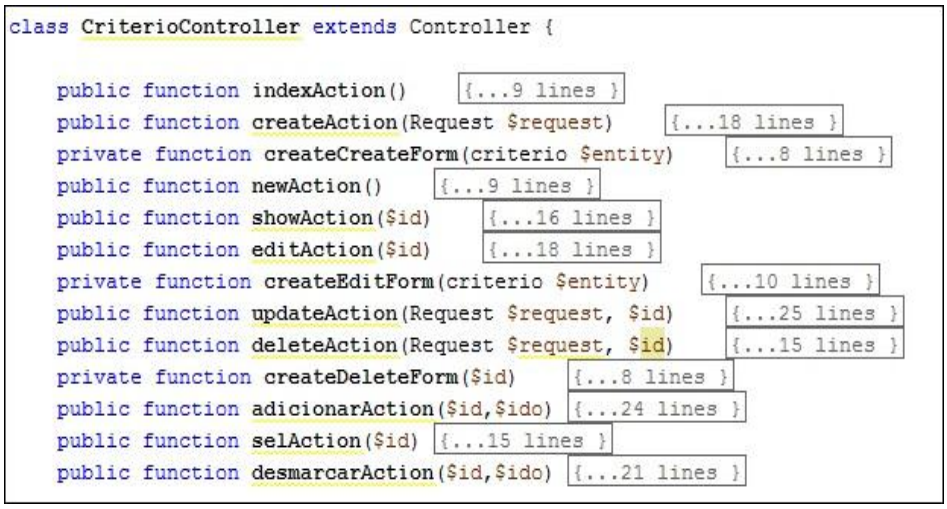

Figura 7: Uso del patrón Controlador en la clase CriterioController

Experto: Este patrón resulta de utilidad en las clases del modelo, las cuales contienen toda la información relacionada con los objetos persistentes que representan. Es empleado por Symfony2 puesto que este utiliza el ORM Doctrine 2 para realizar su capa de abstracción en el modelo, encapsulando toda la lógica de los datos. Doctrine genera las clases para la gestión de las entidades con las responsabilidades debidamente asignadas según este patrón, debido a que cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan.

Alta cohesión: El sistema está diseñado de forma tal que las clases tienen responsabilidades estrechamente relacionadas y no realizan un trabajo excesivo. Symfony2 permite asignar responsabilidades con una alta cohesión, los controladores definen las acciones para las plantillas y colaboran con otras clases para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, lo cual permite simplificar el mantenimiento, generar el bajo acoplamiento y aumentar la capacidad de reutilización de las mismas.

Bajo acoplamiento: Symfony2 utiliza el bajo acoplamiento pues cada clase depende lo menos posible de otra, de manera tal que una de ellas solo recurre a otra en caso de que exista referencia dentro de sus atributos, lo cual permite que el sistema sea mucho más robusto. Se evidencia claramente el uso de este patrón en las entidades, que son las clases más reutilizadas y no presentan ninguna asociación con la vista y con el controlador.

Patrones GoF

En el diseño de la solución propuesta fueron utilizados los siguientes patrones GoF:

Método de Fábrica: Symfony2 brinda una forma confiable para manejar la creación de objetos, así como una manera sencilla de especificar los parámetros necesarios para la creación de estos. El EntityManager (Administrador de entidades) implementado por Doctrine hace uso del Método de Fábrica para proporcionar una instancia de las clases repositorios asociadas a cada una de las entidades, que son las encargadas del proceso de búsqueda y filtrado relacionadas con la base de datos. En la Figura 8 se muestra el uso de dicho patrón.

```
public function adicionarAction($id,$ido) {
    $em = $this->getDoctrine()->getManager();
    $criterio=$em->getRepository('TesisBundle:criterio')->find($id);
    $despliegue=$em->getRepository('TesisBundle:despliegue')->find($ido);
    $pesocriterio=new Pesocriterio();
    $pesocriterio->setcriterio($criterio);
    $pesocriterio->setdespliegue($despliegue);
     $em->persist($pesocriterio);
     $em->flush();
```

Figura 8: Uso del patrón Método de Fábrica en la clase CriterioController

Decorador: En Symfony2 el motor de plantillas Twig implementa la herencia de plantillas, a través de la cual es posible utilizar el patrón decorador; permitiendo la declaración de manera general de la organización del sistema y definiendo los bloques que podrán ser redefinidos por sus descendientes. De esta forma puede ser definida la estructura del sistema en la plantilla Tesis/app/Resources/views/base.html.twig, dejando a las plantillas descendientes de esta la redefinición de los bloques que competen en cada nivel según la funcionalidad que se esté implementando en cada caso. A continuación en la Figura 9 se hace referencia al uso del patrón.

```
{% extends '::index1.html.twig' %}
{% block title %}Gestionar Organización{% endblock %}
 {% block panel %}
    <h4>Lista de Organizaciones</h4>
    {% endblock %}
{% block contenido -%}
```

Figura 9: Uso del patrón Decorador en la vista Organizacion/Index.html.twig

2.5.3 Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. A diferencia del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real (Larman 1999). A continuación en la Figura 10 se muestra el diagrama de clases del diseño perteneciente al requisito Gestionar despliegue donde se presenta la interfaz principal del sistema, la cual puede contener uno o varios formularios, que son los encargados de enviar la información a la clase controladora. Esta contiene los atributos comunes y es la responsable de darle solución a las peticiones a través de las funcionalidades, así como enviar una respuesta a la página cliente, además interactúa con el modelo para la obtención y registro de la información.

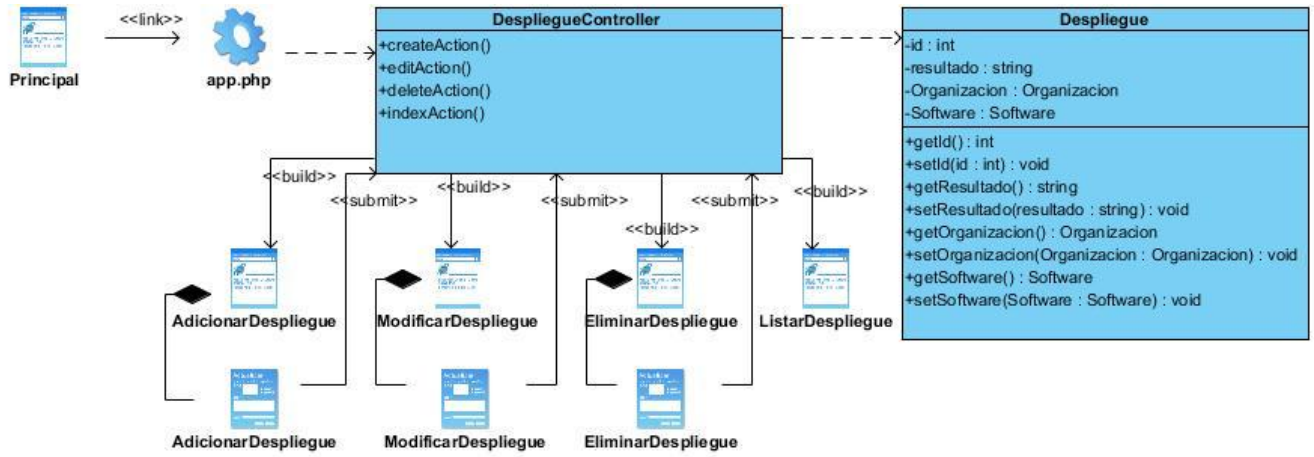

Figura 10: Diagrama de clases del diseño Gestionar despliegue

2.5.4 Diseño de la base de datos

Otra de las etapas del diseño es la elaboración del modelo de datos, con el propósito de garantizar que los datos persistentes sean almacenados coherentemente y definir el comportamiento que debe ser implementado en la base de datos (Jacobson *et al.* 2000).

A través del modelo de datos se definen los conceptos que se manejan en el sistema y que sirven para describir la estructura de la base de datos diseñada. En dicho modelo se representan los datos, sus atributos y tipos, sus relaciones y las restricciones que deben cumplirse sobre ellos. La Figura 11 representa el modelo de datos propuesto para el desarrollo de la solución. Este modelo cuenta con un total de 10 tablas, entre las más significativas se encuentran despliegue, criterioseleccionadoexperto y evaluación.

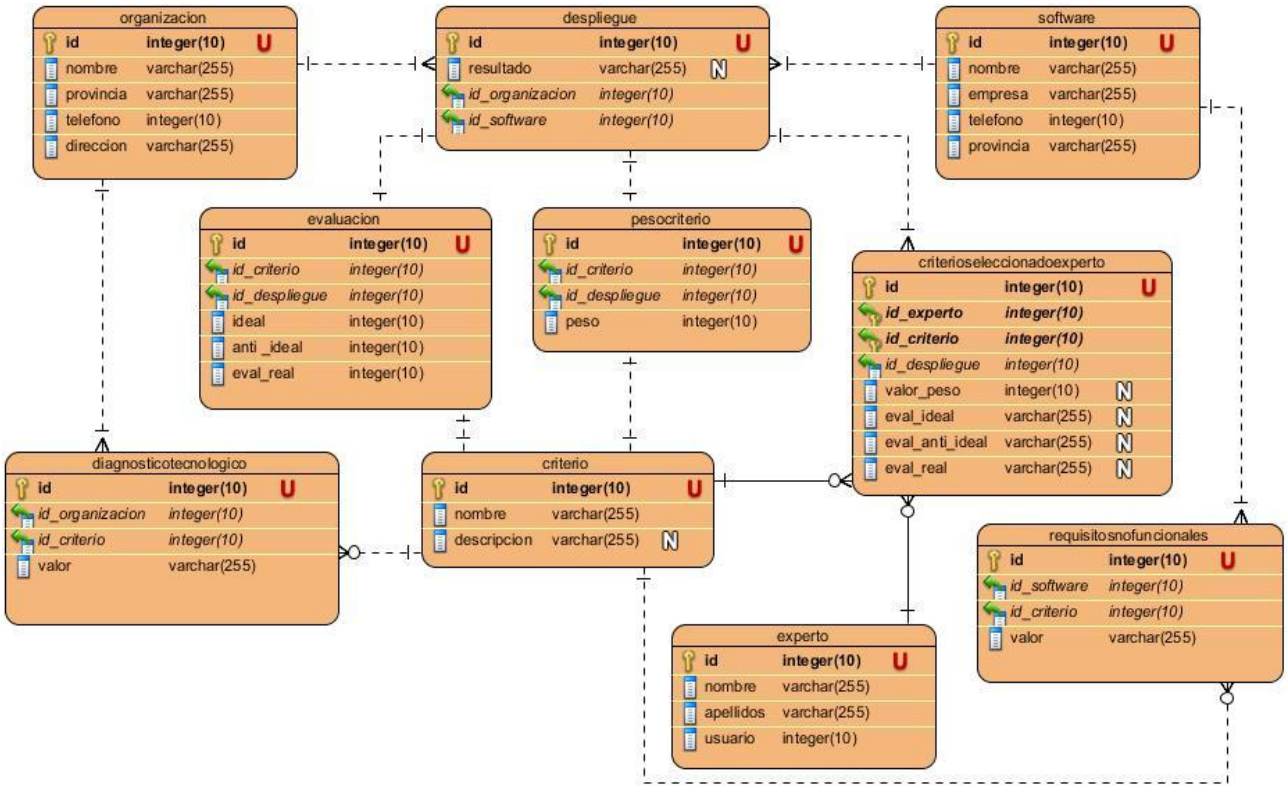

Figura 11: Diagrama Entidad- Relación

Descripción de las tablas

A continuación en la Tabla 4 y Tabla 5 se muestra la descripción de dos de las tablas correspondientes al modelo de datos representados anteriormente, el resto de la descripción se encuentra en el documento entregable ModeloDeDatos_DiagramaComponentes_DiagramaDespliegue.doc.

Tabla 4: Descripción de la tabla despliegue

Nombre: despliegue		
Descripción: Contiene los datos asociados al despliegue.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del despliegue.
resultado	varchar	Resultado obtenido en el despliegue.
id_organizacion	integer	Llave foránea que referencia a la tabla organización.
id_software	integer	Llave foránea que referencia a la tabla software.

Tabla 5: Descripción de la tabla evaluacion

Nombre: evaluacion		
Descripción: Contiene la evaluación final de un despliegue.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de cada evaluación.
id_criterio	integer	Llave foránea que referencia a la tabla criterio.
id_despliegue	integer	Llave foránea que referencia a la tabla despliegue.
eval_ideal	integer	Evaluación ideal emitida a cada criterio.
eval_anti_ideal	integer	Evaluación anti-ideal emitida a cada criterio.
eval_real	integer	Evaluación real emitida a cada criterio.

2.6 Descripción de la solución

La solución propuesta permite realizar la evaluación tecnológica en los proyectos de despliegue de SGE, esta presenta la siguiente estructura:

Menú superior: Muestra las opciones correspondientes a la gestión de los datos necesarios para el despliegue. Además muestra la administración del sistema y la ayuda del usuario.

Menú lateral: Muestra las opciones referentes a la evaluación del despliegue y los reportes generados.

El sistema comprende tres roles necesarios para garantizar su correcto uso, los cuales realizarán diferentes actividades. Dentro de los mismos se encuentra el administrador, el cual se encargará de gestionar los usuarios que utilizarán el sistema. El analista deberá ejecutar las actividades de definición de los requisitos no funcionales tecnológicos relacionados al software en despliegue y el diagnóstico tecnológico de la organización. Además de seleccionar los criterios a ser evaluados y por último realiza la evaluación del despliegue. El experto es el responsable de identificar el peso de los criterios que son seleccionados por el analista y luego emitir su evaluación ideal, anti ideal y real. Los criterios deberán ser evaluados por dos o más expertos.

La selección de los expertos no está comprendida como una de las actividades desarrolladas por la solución. Esta se realiza previamente mediante la aplicación de un cuestionario para medir el coeficiente de competencia experta que se obtiene a partir de la autovaloración realizada por la persona para determinar su conocimiento sobre el proceso de despliegue de software. Se pueden seleccionar tantos expertos como se desee, siempre que su coeficiente de competencia sea alto o en algunos casos medio. Los expertos seleccionados son gestionados como usuarios del sistema por el administrador.

Los datos generados por la solución durante la evaluación tecnológica deben ser manejados de manera discreta, pero no requieren de medidas rigurosas de seguridad. La sensibilidad de los mismos presenta un nivel medio, debido a que la información es protegida ante el acceso no autorizado, sin embargo el sistema no maneja información confidencial del software ni de la organización inmersa en el despliegue.

2.7 Conclusiones del capítulo

En el presente capítulo se enunciaron los aspectos fundamentales que se tuvieron en cuenta durante el proceso de análisis y diseño del sistema para la evaluación tecnológica de despliegues de SGE, en el mismo se concluye que:

- ✓ El modelado del negocio y la creación del modelo conceptual propiciaron un mayor entendimiento y compresión del dominio del problema.
- ✓ El empleo de las técnicas de obtención de requisitos posibilitó la definición de los requisitos del sistema, identificando a través de ellos los aspectos más importantes que definirán en la propuesta de solución.
- ✓ La elaboración del modelo de diseño sirvió como guía para la implementación a fin de materializar con precisión los requisitos del cliente.
- ✓ El uso de patrones de diseño permitió asignar correctamente las responsabilidades a las clases, lo que sirvió de apoyo a la creación de un código más fácil de comprender, mantener y extender.
- ✓ La construcción del modelo de datos proporcionó la representación lógica y física de los datos persistentes.

CAPÍTULO III: IMPLEMENTACIÓN Y VALIDACIÓN

3.1 Introducción

El presente capítulo comprende los aspectos referentes a la disciplina de implementación y validación de la solución. En este se expone la estructura física de la solución, los estándares de codificación utilizados y los diagramas de componentes y despliegue. Se muestra la validación del diseño mediante las métricas especificadas y los resultados alcanzados en las mismas. También se presenta y describe las pruebas realizadas al software y los resultados obtenidos con el fin de validar que el sistema desarrollado posea la mayor calidad posible. Por último se valida la investigación para demostrar que esta cumpla con el objetivo planteado.

3.2 Implementación de la solución

La implementación de la solución comienza con el resultado alcanzado en el diseño y proporcionará un producto que cumpla las exigencias y necesidades del cliente. En los próximos puntos se mostrará la estructura física de la solución, además de los estándares de codificación usados y los diagramas de componente y despliegue.

3.2.1 Estructura física de la solución

La estructura física de la solución está definida por el marco de trabajo Symfony2. Esta está conformada por un solo bundle llamado TesisBundle. Un bundle es un directorio que contiene todo tipo de archivos dentro una estructura jerarquizada de directorios. Los bundles de las aplicaciones Symfony2 suelen contener clases PHP y archivos web (JavaScript, CSS e imágenes) (Eguiluz 2013). TesisBundle posee todas las clases de dicho sistema (vistas, formularios, clases controladoras, entidades, repositorios y otras). A continuación se explica la estructura interna del bundle:

Controller/: Contiene los controladores del bundle, clases encargadas de gestionar las funcionalidades del sistema.

DependencyInjection/: Contiene elementos relacionados con el contenedor de inyección de dependencias, además contiene las extensiones para las clases de inyección de dependencias, la configuración que importan los servicios y registra uno o más pases del compilador.

Entity/: Contiene las clases entidades que representan la lógica del negocio.

Form/: Contiene todos los formularios definidos para las clases entidades.

Resourses/: Contiene la carpeta config/, donde se encuentran los archivos de enrutamiento del bundle y la carpeta views, la cual contiene las plantillas organizadas según el nombre del controlador.

Tests/: Contiene los tests unitarios y funcionales del bundle.

En la Figura 12 se muestra el conjunto de carpetas organizadas de forma jerárquica por la que se encuentra compuesto el bundle del sistema.

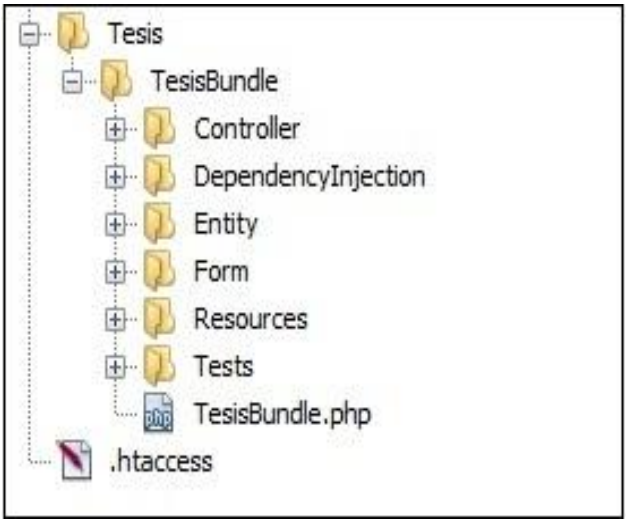

Figura 12: Estructura física de la solución

3.2.2 Estándares de codificación

En el desarrollo del sistema se hizo uso de varios estándares establecidos, dichos estándares se describen a continuación:

- **Definición de clases**

  Todos los nombres de las clases deben ser definidos haciendo uso del estilo de codificación UpperCamelCase, el cual establece que los nombres iniciarán con letra mayúscula y de poseer más de una palabra, la primera letra de cada una deberá ser también mayúscula. No se permiten

letras mayúsculas sucesivas a menos que se trate de siglas conocidas en el dominio del sistema. En la Figura 13 se muestra el uso del estándar.

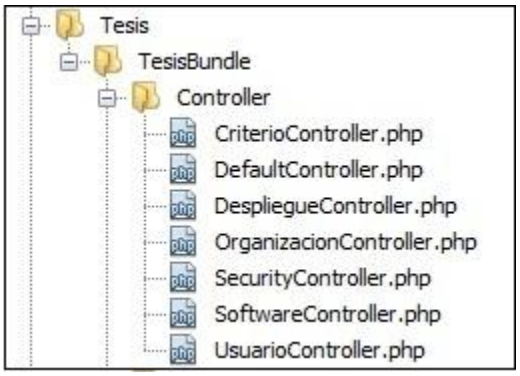

Figura 13: Estándar para la definición de clases

- **Definición de variables y constantes**

Los nombres de las variables deben expresar claramente el contenido de las mismas y serán definidos según la notación lowerCamelCase. Los nombres iniciarán con letra minúscula y cada nueva palabra debe iniciar con mayúscula. Se excluyen de esta nomenclatura las variables generadas por el propio marco de trabajo. En la Figura 14 se muestra el uso del estándar.

```
/**
 * @var integer
 *
 * @ORM\Column(name="id_criterio", type="integer"
 */
private $idCriterio;

/**
 * @var integer
 *
 * @ORM\Column(name="id_experto", type="integer")
 */
private $idExperto;
```

Figura 14: Estándar para la definición de variables

- **Definición de funciones**

Todos los nombres de las funciones serán definidos haciendo uso de la notación lowerCamelCase. Los nombres de las funciones deben dar una idea clara del objetivo con el que fueron concebidas. En la Figura 15 se muestra el uso del estándar.

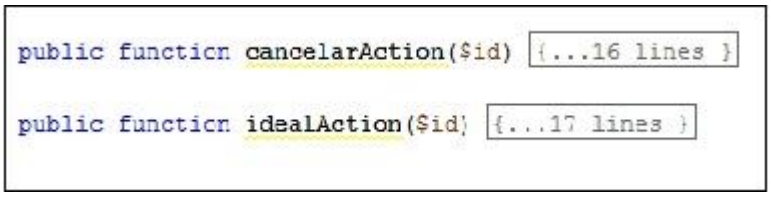

Figura 15: Estándar para la definición de funciones

- **Posición de las llaves en bloques de instrucciones.**

  Las llaves de apertura se colocarán al final de la sentencia que delimita el bloque de instrucciones, las de cierre se alinean con el inicio de la sentencia en una nueva línea. En la Figura 16 se muestra el uso del estándar.

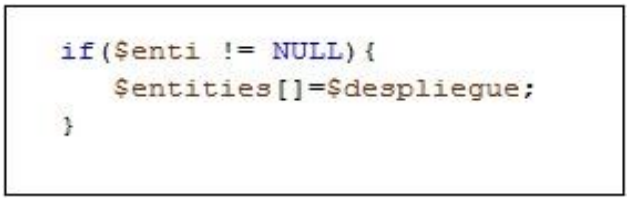

Figura 16: Posición de las llaves en bloques de instrucciones

3.2.3 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y las dependencias entre tipos de componentes. Representa las dependencias entre componentes de software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables (Rumbaugh *et al.* 2000). El diagrama obtenido permitió modelar el sistema y la interacción entre sus principales componentes. A continuación en la Figura 17 se muestra el diagrama de componentes elaborado y una breve descripción del mismo.

El diagrama está compuesto por un componente principal llamado TesisBundle, y a su vez él está asociado a otros componentes. Se puede apreciar el componente Modelo donde se encuentran las clases Repositorios que hace uso de ORM Doctrine y las clases Entidades. La Vista hace uso del componente de plantillas Twig y es usada por el Controlador. Este también hace uso del componente Enrutado, Entidades, SwiftMailer y de SecurityController.

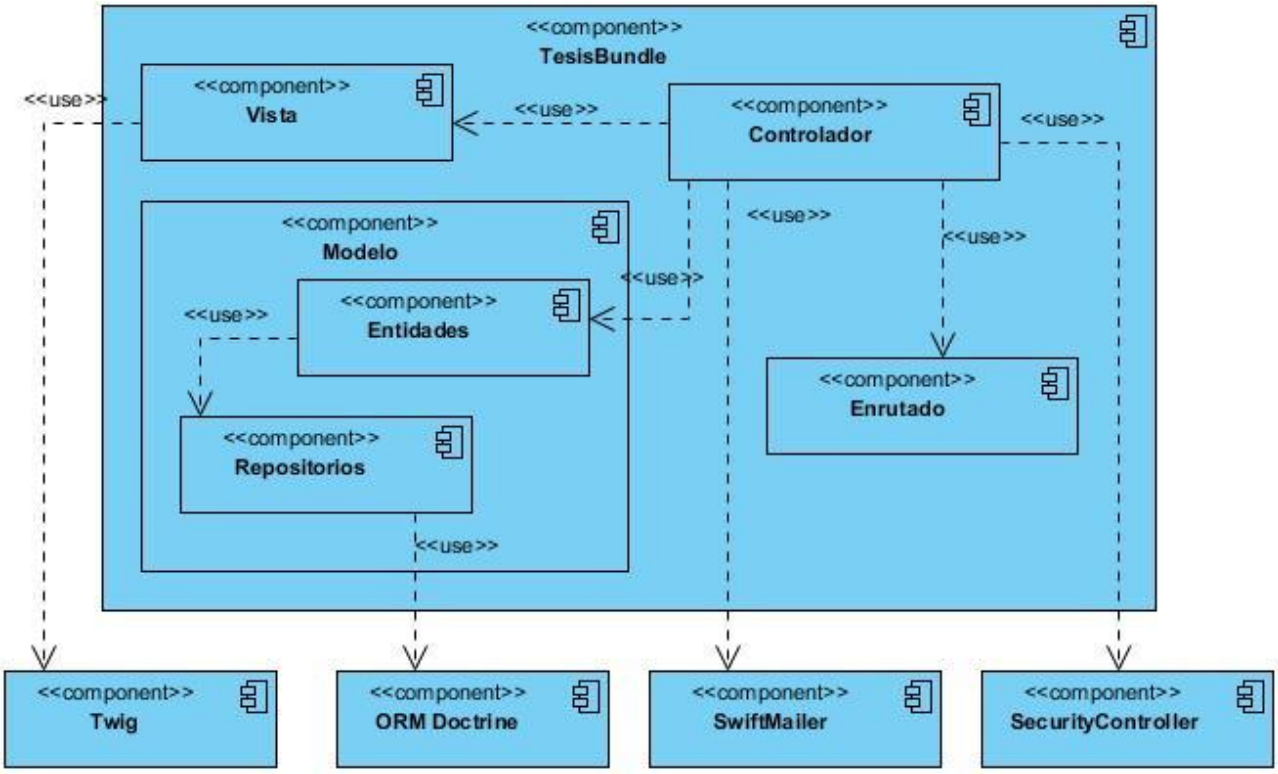

Figura 17: Diagrama de componentes

3.2.4 Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El modelo de despliegue elaborado para la propuesta de solución está compuesto por los siguientes nodos:

PC Cliente: Computadora en el cual el sistema será ejecutado mediante el navegador Mozilla Firefox versión 32.

Servidor Aplicaciones Web: El servidor de aplicaciones empleado donde radica la lógica de negocio de la aplicación. Servidor Web Apache 2.4 utilizando el lenguaje de programación del lado del servidor PHP.

Servidor de Base de datos: Sistema Gestor de Base de Datos: PostgreSQL 9.2 donde se encuentra la base de datos que utiliza el sistema, este puede estar instalado en la misma computadora donde se

encuentra el servidor web. A continuación en la Figura 18 se muestra el diagrama de despliegue realizado a la solución.

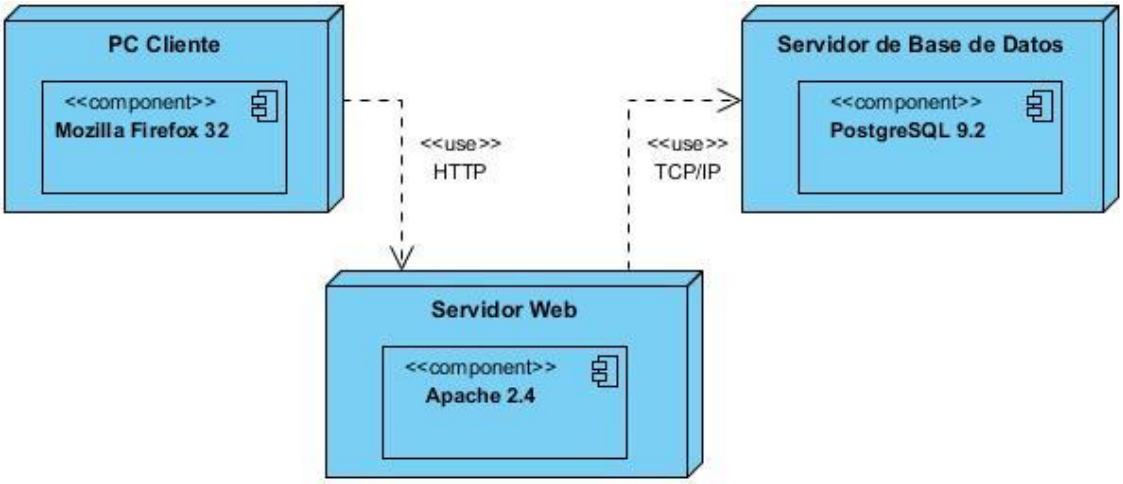

Figura 18: Diagrama de despliegue

3.3 Validación del diseño mediante métricas

Para realizar la validación del diseño de la solución propuesta se estudiaron varias métricas de diseño y sus características. La finalidad del uso de métricas es evaluar el sistema para conseguir alta calidad y robustez en su diseño. Para realizar la validación a las clases del diseño se seleccionaron las métricas Tamaño Operacional de clases (TOC) y Relación entre Clases (RC). La aplicación de estas métricas contiene medidas de los siguientes atributos de calidad:

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

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

- **Complejidad del mantenimiento:** nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costos y la planificación del proyecto.

3.3.1 Métrica Tamaño Operacional de Clase (TOC)

La métrica TOC describe el número de procedimientos (métodos) asignados a una clase. Además establece una relación entre los atributos responsabilidad y complejidad de implementación, sin embargo determina una relación inversa entre estos últimos y el atributo reutilización.

3.3.2 Resultados de la aplicación de la métrica TOC

En la Figura 19 se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad. Este gráfico muestra un resultado satisfactorio pues el 56% de las clases poseen una baja responsabilidad. Esta característica permite que en caso de fallos, como la responsabilidad está distribuida de forma equilibrada, ninguna clase es demasiado crítica como para dejar al sistema fuera de servicio.

Figura 19: Resultados de la evaluación de la métrica TOC para el atributo responsabilidad

En la Figura 20 se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo complejidad de implementación. Este gráfico muestra un resultado satisfactorio pues el 56% de las clases poseen una baja complejidad de implementación. Esta característica permite mejorar el mantenimiento y soporte de estas clases.

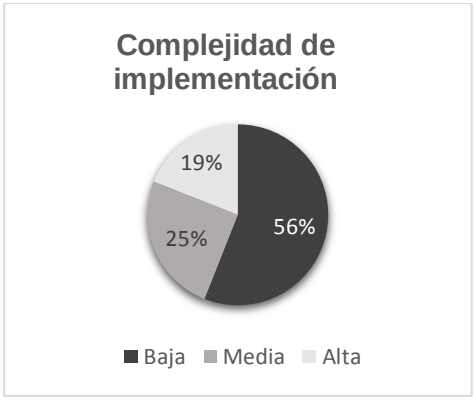

Figura 20: Resultados de la evaluación de la métrica TOC para el atributo complejidad de implementación

En la Figura 21 se muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización. El diseño del sistema tiene un grado de eficiencia aceptable pues solamente el 19% del total de las clases poseen una baja reutilización.

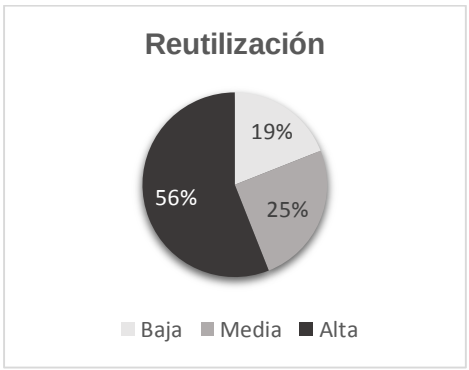

Figura 21: Resultados de la evaluación de la métrica TOC para el atributo reutilización

Luego de realizada la evaluación de la métrica TOC se concluye que los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio, evidenciándose un alto por ciento de posibilidad de reutilización de clases, dado por un 56% de baja responsabilidad de clases y de complejidad de implementación. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

3.3.3 Métrica Relaciones entre Clases (RC)

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

3.3.4 Resultados de la aplicación de la métrica RC

La Figura 22 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo acoplamiento. Se evidencia un bajo acoplamiento entre las clases pues el 69% de estas presentan una dependencia con otra. Este resultado es muy favorable para el diseño del componente pues al existir poca dependencia entre las clases aumenta el grado de reutilización del sistema.

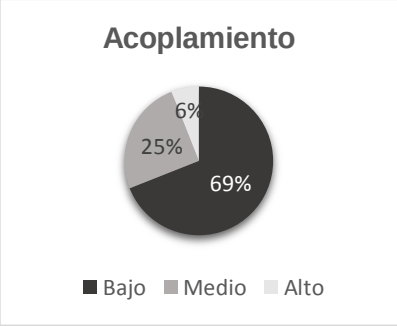

Figura 22: Resultados de la evaluación de la métrica RC para el atributo acoplamiento

La Figura 23 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento. El gráfico refleja un resultado aceptable del atributo pues el 69% de las clases presentan una baja complejidad de mantenimiento.

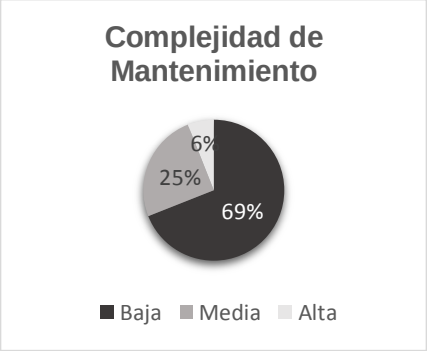

Figura 23: Resultados de la evaluación de la métrica RC para el atributo complejidad de mantenimiento

La Figura 24 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas.

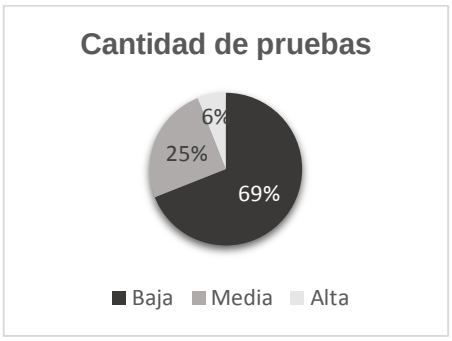

Figura 24: Resultados de la evaluación de la métrica RC para el atributo cantidad de pruebas

La Figura 25 muestra la representación de la incidencia de los resultados de la evaluación del atributo reutilización. Esto evidencia que el 69% de las clases poseen una alta reutilización lo que es un factor fundamental que debe ser tenido en cuenta en el desarrollo de software.

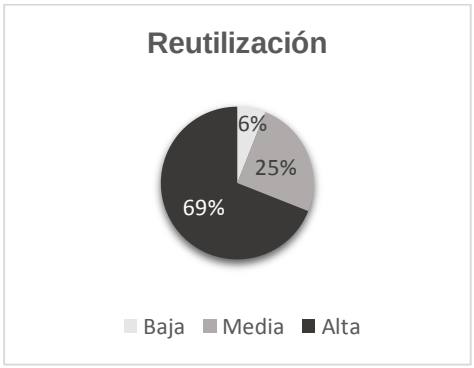

Figura 25: Resultados de la evaluación de la métrica RC para el atributo reutilización

Luego de analizar los resultados obtenidos de la métrica RC, se puede concluir que el diseño del sistema tiene una calidad aceptable. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 69% de las clases el nivel de acoplamiento es mínimo, la complejidad de mantenimiento y la cantidad de pruebas son bajas a un 69%, mientras que, el atributo reutilización cuenta con igual por ciento en la categoría alta, lo que representa valores favorables para el diseño realizado.

3.4 Validación de la solución

Se hace necesaria la validación de la solución propuesta con el objetivo de comprobar que cumple con las especificaciones y necesidades requeridas. A continuación se muestran las pruebas empleados para realizar la validación de la solución. Se especifican las pruebas realizadas, las cuales fueron hechas con el objetivo de revelar y corregir el máximo de errores posibles.

3.4.1 Pruebas de caja negra

El principal objetivo de las pruebas de caja negra es comprobar que las funcionalidades de la aplicación se realizan de forma correcta y responden a las necesidades del cliente, apoyándose en los casos de pruebas diseñados para cada funcionalidad. El uso de las pruebas de caja negra permitirá detectar no conformidades, así como mal funcionamiento en el sistema durante la implementación. Se realizaron tantas iteraciones como fueron necesarias, con el fin de obtener un sistema estable y acorde a las necesidades del cliente.

Para validar la implementación de la solución propuesta se aplicaron pruebas funcionales internas por parte del grupo de calidad del Centro de Informatización de Entidades CEIGE, haciendo uso de la técnica de partición de equivalencia como parte de las pruebas de caja negra. Para ello fueron diseñados casos de prueba para cada requisito permitiendo de esta manera, a partir de un conjunto de condiciones de entrada, encontrar posibles errores que pueda presentar el sistema. A continuación en la Tabla 6 y Tabla 7 se presenta el diseño de caso de prueba para el requisito Adicionar despliegue y sus la descripción de sus variables.

Condiciones de ejecución:

- Se debe identificar y autenticar ante el sistema, además debe tener el rol analista para ejecutar esta acción.
- Se debe seleccionar la opción del menú superior: Despliegue/Gestionar Despliegue.
- Se debe seleccionar la opción Adicionar Despliegue que aparece en el panel principal.
- Se debe haber adicionado al menos una organización y un software.

Tabla 6: Descripción del caso de prueba para el requisito Adicionar despliegue

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar despliegue.	El sistema debe permitir adicionar un nuevo despliegue.	EP 1.1: Adicionar un despliegue, introduciendo datos válidos.	– Se introducen y/o seleccionan los datos del despliegue correctamente. – Se presiona el botón **Adicionar**. – Se muestra un mensaje de confirmación.
		EP 1.2: Adicionar un despliegue, introduciendo datos inválidos.	– Se seleccionan y/o introducen datos inválidos del despliegue. – Se presiona el botón **Adicionar**. – El sistema muestra un mensaje informando el error.
		EP 1.3: Adicionar un despliegue dejando campos vacíos.	– Se seleccionan y/o introducen los datos del despliegue dejando algún campo vacío. – Se presiona el botón **Adicionar**. – El sistema muestra un mensaje informando el error.
		EP 1.4: Cancelar	– Se introducen o no los datos del despliegue. – Se presiona el botón **Cancelar**.

Tabla 7: Descripción de las variables a probar del requisito Adicionar despliegue

No	Nombre de campo	Tipo	Válido	Inválido	Inválido
1	Software	Campo de selección (No editable).	Letras	Números	Vacío
2	Organización	Campo de selección (No editable).	Letras	Números	Vacío

3.4.2 Resultados en las pruebas aplicadas

Los casos de prueba diseñados fueron aplicados en 2 iteraciones de pruebas, realizadas por el grupo de calidad de CEIGE, en las cuales se detectaron un conjunto de no conformidades. En la primera iteración se encontraron un total de 10 no conformidades, de las cuales 2 fueron de validación, 3 de interfaz, 2 de funcionalidad y 3 de ortografía. En la segunda iteración no se encontraron no conformidades, quedando así listo el sistema para ser utilizado por el cliente y con un grado de calidad aceptable. A continuación, en la Figura 26, se muestra el gráfico que representa el total de no conformidades detectadas y resueltas en cada iteración de pruebas realizadas.

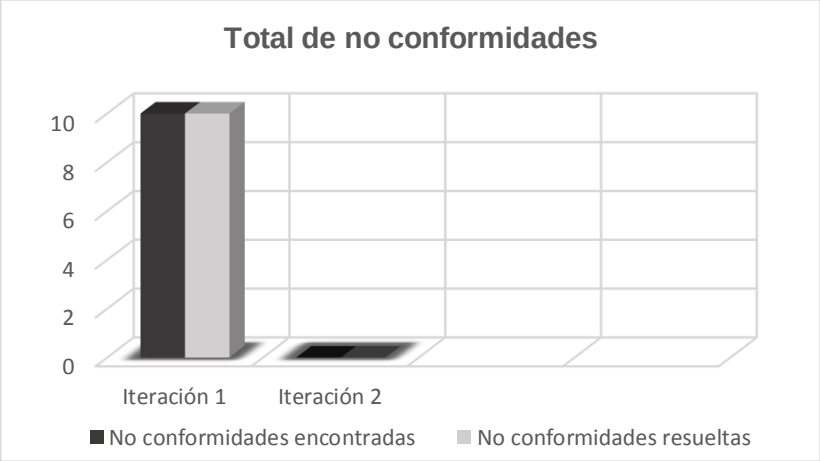

Figura 26: No conformidades encontradas por iteración

3.5 Validación de la investigación

Con el objetivo de demostrar la reducción del tiempo destinado a la evaluación tecnológica en el despliegue de SGE, se valida la investigación a través del diseño pre experimento pretest - postest de un solo grupo. En una primera prueba se midió el tiempo de aplicación del método de forma manual, luego se procedió a medir el tiempo del método utilizando la propuesta de solución para posteriormente comparar los resultados obtenidos y verificar si se registró el cambio esperado.

Se seleccionó como muestra para el pre experimento un proyecto de despliegue perteneciente a CEIGE. Para la aplicación del método tanto manual como a través de la solución se tuvo en cuenta el trabajo de una persona y dos expertos de manera no concurrente para emitir la evaluación, considerando que esta es la cantidad mínima de expertos para la evaluación del despliegue. En la

Tabla 8 se muestra el tiempo aproximado que fue obtenido para cada una de las actividades del método, teniendo en cuenta que las actividades selección de expertos e identificación de los criterios de evaluación demoran el mismo tiempo en ambos casos, debido a que la solución propuesta no incide directamente en estas.

Tabla 8: Tiempo total de aplicación del método

Actividades del método	**Tiempo de aplicar el método manual (hh:mm:ss)**	**Tiempo de aplicar el método utilizando la solución (hh:mm:ss)**
1. Selección de expertos.	00:18:00	00:18:00
2. Definición de RNF tecnológicos.	00:06:00	00:04:00
3. Diagnóstico tecnológico.	00:07:00	00:04:00
4. Identificación de los criterios de evaluación.	00:25:00	00:25:00
5. Identificación del peso de los criterios.	01:03:00	00:04:00
6. Determinación de la ITI ideal y anti-ideal.	01:04:00	00:05:00
7. Determinación de la ITI real.	00:31:00	00:03:00
8. Cálculo de la distancia.	02:35:00	00:01:00
9. Cálculo de proximidad relativa.	00:33:00	
10. Salida de valores lingüísticos.	00:05:00	
Tiempo total	**06:47:00**	**01:04:00**

3.5.1 Resultados de la investigación

Como resultado del pre experimento aplicado, se afirma que la utilización de la solución muestra una reducción de 5 horas y 43 minutos en relación a su aplicación manual del método. Por tanto, se puede

concluir que el sistema para la evaluación tecnológica en el despliegue de SGE permitió disminuir el tiempo de desarrollo de las actividades del método y sus cálculos asociados, logrando un menor tiempo destinado a la evaluación tecnológica del despliegue y una rápida toma de decisiones. En la Figura 27 se ve representada la disminución del tiempo asociada a la aplicación del método.

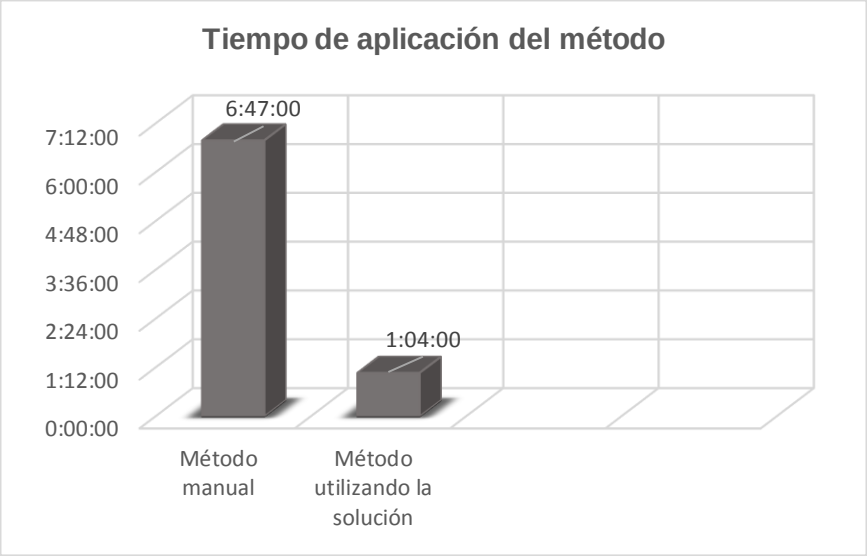

Figura 27: Tiempo de aplicación del método

3.6 Conclusiones del capítulo

Una vez finalizado el presente capítulo se arriba a las siguientes conclusiones:

- ✓ La aplicación de las métricas de diseño demostró que las clases generalmente presentan una baja complejidad, un acoplamiento relativamente bajo y un alto grado de reutilización, lo que facilitó en gran medida la implementación.
- ✓ A partir de las dos iteraciones de pruebas funcionales realizadas por el grupo de calidad de CEIGE, se obtuvo un correcto funcionamiento de las funcionalidades implementadas logrando que estas dieran respuesta a los requisitos funcionales.
- ✓ A través de los resultados obtenidos mediante la validación de los objetivos de la investigación se demostró una disminución del tiempo destinado al proceso.

CONCLUSIONES

Con la realización del presente trabajo de diploma se puede afirmar que se logró alcanzar el objetivo general propuesto, luego de realizar todas las tareas trazadas al inicio de la investigación. El desempeño de los objetivos específicos permitió arribar a las siguientes conclusiones:

- ✓ Se elaboró el marco teórico de la investigación, lo que permitió conceptualizar los elementos fundamentales que fueron utilizados y se evidenció la ausencia de un sistema para la evaluación tecnológica en el despliegue de SGE, por lo que se pone de manifiesto la necesidad de la creación de la solución.
- ✓ Mediante la generación de los artefactos definidos en la metodología escogida se realizó el análisis y diseño de la solución, sirviendo de base para la su correcta implementación.
- ✓ Se implementó una solución informática que facilitará la evaluación tecnológica en el despliegue de SGE.
- ✓ Los resultados arrojados en las pruebas de caja negra validaron que las funcionalidades desarrolladas cumplen con las especificaciones requeridas.
- ✓ A través del pre experimento desarrollado se comprobó que la solución propuesta reduce el tiempo del método para la evaluación tecnológica en el despliegue de SGE.

RECOMENDACIONES

Con el objetivo de mejorar la solución se proponen las siguientes recomendaciones:

- ✓ Continuar el desarrollo de la solución implementando la actividad selección de experto, en la cual el sistema no incide directamente.
- ✓ Utilizar la propuesta de solución en los proyectos de despliegue de SGE de CEIGE.

REFERENCIAS BIBLIOGRÁFICAS

ALMEIRA A. S. y PÉREZ V. *Arquitectura de Software: Estilos y Patrones*. Facultad de Ingeniería. Argentina, Universidad Nacional de la Patagonia San Juan Bosco, 2007. 114. p.

APACHE. *The Apache Software Foundation*, 2012. [Disponible en: http://www.apache.org/foundation/how-it-works.html

BAKOUROS Y. *TECHNOLOGY EVALUATION*. University of Thessaly, 2000. 1-37.

BIZAGI S. *BPMN 2.0*, 2014. 1-24.

CAPEÁNS C. *Método para la evaluación tecnológica de proyectos de despliegue de Sistemas de Gestión Empresarial*. Centro de Informatización de la Gestión de Entidades. La Habana, Universidad de las Ciencias Informáticas, 2014. 82. p.

DAI Q.; KAUFFAMAN R. J.*, et al.* *VALUING INFORMATION TECHNOLOGY INFRASTRUCTURES: A GROWTH OPTIONS APPROACH*, 2005. 1-36.

DEARLE A. *Software Deployment, Past, Present and Future*. School of Computer Science, University of St Andrews, Scotland, 2007. 7.

DESOFT. *Metodología para implementación de productos de gestión empresarial*. La Habana, 2007.

DÍAZ A.; GONZÁLEZ J. C.*, et al.* *Implantación de un sistema ERP en una organización*. Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, 2005. 2: 30-37.

EGUILUZ J. *Desarrollo web ágil con Symfony2*, 2013.

ESCALONA M. J. y KOCH N. *Ingeniería de Requisitos en Aplicaciones para la Web –Un estudio comparativo* Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, Sevilla, 2002. 4-25.

FIGUEREDO V. *Propuesta de modelo para la Gestión del Capital Humano en la implantación del Sistema Integral de Gestión Cedrux*. La Habana, Cuba, Universidad de las Ciencias Informáticas, 2009. p.

FINAZZI P. A. *Introducción a los sistemas ERP, despliegue y configuración de OpenERP.*, 2013. 3-44.

FINK L. y NEUMANN S. Exploring the perceived business value of the flexibility enabled by information technology infrastructure *Information & Management*, 2009, 46: 90-99.

GAMMA E.; HELM R., *et al.* *Patrones de Diseño. Elementos de software orientado a objetos reutilizable*. Madrid, Pearson Educación. S.A, 2003. p. 84-7829-059-1

HARDCASTLE E. Business Information Systems, 2008.

HOHENEGGER J.; BUFARDI A., *et al.* A new concept of compatibility structure in new product development *Advanced Engineering Informatics*, 2007, 21: 101-116.

ISASI-GENIX A. y GÓMEZ-ACOSTA M. I. Diseño del proceso de implementación de software en Desoft Habana *Ingeniería Industrial*, 2012, XXXIII: 60-68.

JACOBSON I.; BOOCH G., *et al.* *El proceso unificado de desarrollo de software*. Madrid, Pearson Educación, S. A., 2000. 464 p. 84-7829-036-2

LARMAN C. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. México, 1999. 536 p. 970-17-0261-1

MEGAL J. *Metodología clave en la implantación de un Sistema de Gestión Empresarial (ERP), IBdos*, 2004.

NETBEANS. *Bienvenido a NetBeans*, 2015. [Disponible en: https://netbeans.org/index_es.html

PARADIGM V. *Visual Paradigm - Software Design Tools for Agile Teams*, 2015. [Disponible en: http://www.visual-paradigm.com/

PHP T. P. G. *Manual de php*, 2015. [Disponible en: http://www.php.net/manual/es

POSTGRESQL. *PostgreSQL-es*, 2009. [Disponible en: http://www.postgresql.org.es/

PRESSMAN R. *Ingeniería del Software: Un enfoque práctico*. 6ta edición 2006. 953 p.

RIVERA J. *Método para despliegues de sistemas de gestión.* . Centro de Informatización de la Gestión de Entidades La Habana Universidad de las Ciencias Informáticas UCI, 2011. 80. p.

RODRÍGUEZ T. *Metodología de desarrollo para la Actividad productiva de la UCI.* Universidad de las Ciencias Informáticas, 2014. 1.2: 1-16.

RUMBAUGH J.; JACOBSON I.*, et al.* *El Lenguaje Unificado de Modelado. Manual de referencia* Madrid, Pearson Educación, S. A., 2000. 424 p. 84-7829-037-0

SOMMERVILLE I. *Ingeniería del software*. 7ma edición. Madrid, Pearson Educación, S.A, 2005. 712 p. 84-7829-074-5

SPARKS G. Introducción al UML. El modelo de Proceso de Negocio, 2000: 1-10.

TOMÉ F. *Implantación de soluciones SAP para el mercado de telecomunicaciones siguiendo la metodología ASAP.* Ingeniería informática, Universidad Carlos III de Madrid, 2009.

ANEXOS

Anexo 1: Entrevista realizada sobre evaluación tecnológica en despliegue de SGE

Datos del entrevistado

Nombre y apellidos: _____

Graduado: ____Ingeniero(a) en Ciencias Informáticas, ____Ingeniero(a) Informático(a), ____Licenciado(a) en Ciencias de la Computación, otros: _____

Categoría docente: _____

Categoría científica: _____

Años de experiencia en la gestión de proyectos de software: _____

Roles desempeñado en los proyectos en los que ha participado: _____

Proyecto en el que trabaja actualmente: _____

Rol que desempeña en el proyecto actual: _____

Experiencia en la realización de despliegues de software: _____

Proyectos de despliegues de software en los ha participado: _____

Preguntas

¿Dentro del método para la evaluación de proyectos de despliegue de Software de Gestión Empresarial, cuáles son las actividades que poseen alta prioridad?

¿Cómo se realiza la asignación de los roles utilizados en el método?

¿Cuál es la cantidad mínima de expertos que puede ser utilizada para evaluar el despliegue?

¿Pueden existir criterios de evaluación que sean obviados o ignorados en el diagnóstico tecnológico de la organización?

¿Pueden existir criterios de evaluación que sean obviados o ignorados en los requisitos no funcionales tecnológicos del software?

¿Cómo es obtenida la matriz asociada al peso de los criterios?

¿Cuáles son los datos persistentes durante toda la evaluación tecnológica?

¿Cómo se traducen las etiquetas lingüísticas emitida por los expertos a números difusos?

¿Cómo se realiza el cálculo entre números difusos y entre un escalar y un número difuso?