Universidad de las Ciencias Informáticas

Facultad 3

Sistema Integral de Control Interno para el Vicedecanato de Administración y Servicios de la Facultad 3

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

Autores:

Yairon Vargas Águila

José Carlos Arencibia Pérez

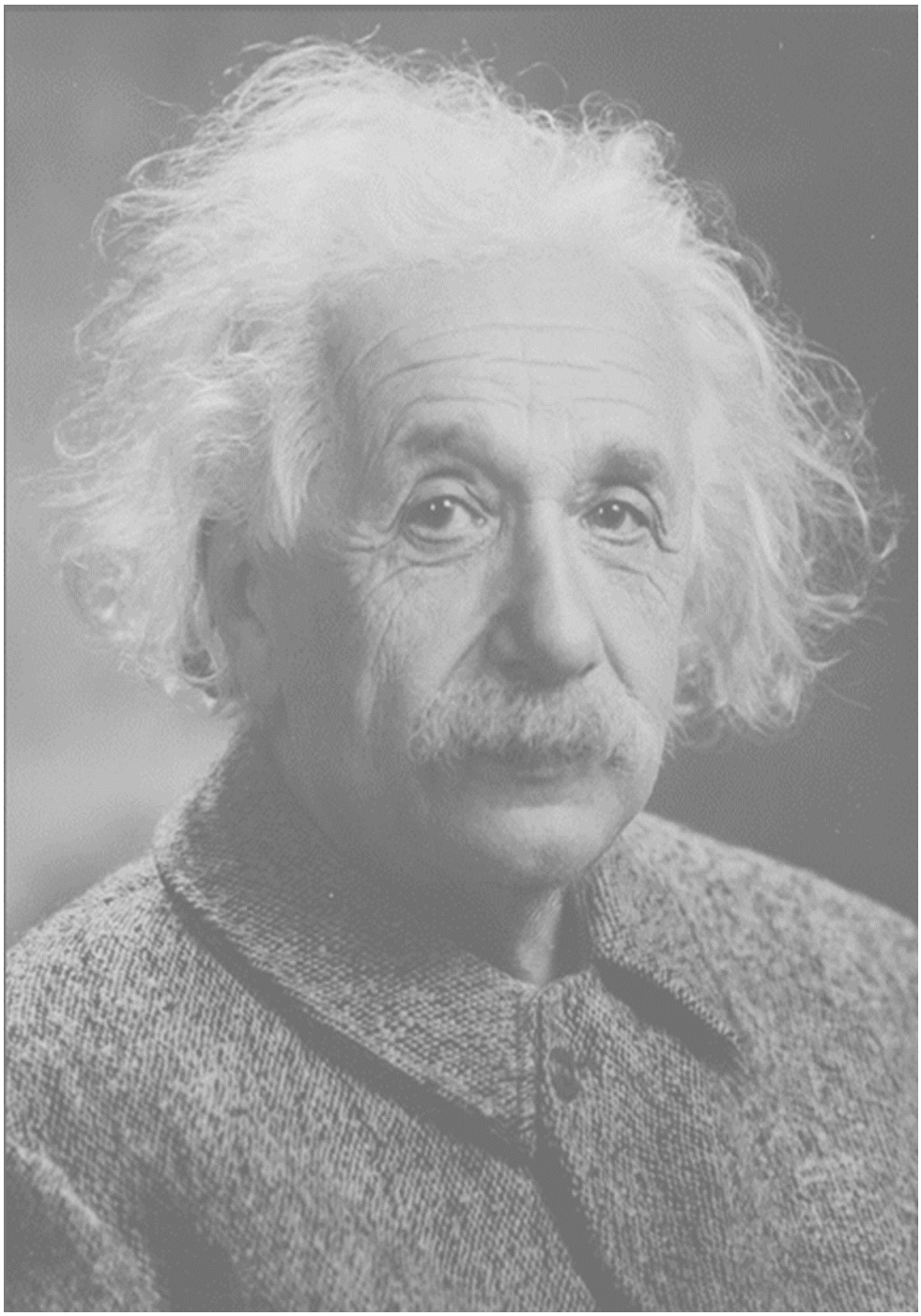
Tutoras:

MSc. Ana Marys Garcia Rodríguez

Ing. Dailét Manuela Soto Fumero

La Habana, Junio de 2017

Declaración de Autoría



La mente que se abre a una nueva idea nunca vuelve a su tamaño original.

Albert Einstein

Declaración de Autoría

DECLARACIÓN DE AUTORÍA JURADA

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

Yairon Vargas Águila	José Carlos Arencibia Pérez
_____	_____
Firma del Autor	Firma del Autor
MSc. Ana Marys Garcia Rodríguez	Ing. Dailét Manuela Soto Fumero
_____	_____
Firma del Tutor	Firma del Tutor

AGRADECIMIENTOS

A la Revolución por permitirme formarme como profesional comprometido con los principios y valores adquiridos en el transcurso de mi formación.

A mis padres por el apoyo constante durante mis estudios, y en circunstancias de la vida.

A mi familia, que siempre me dio apoyo en los momentos más difíciles.

A mis tutoras por el apoyo constante durante días y noches para que pudiera alcanzar la gran meta de ser ingeniero.

A mis compañeros de grupo que durante el transcurso de los 5 años fuimos inseparables para todo.

A todos los profesores que de alguna forma han incurrido en mi formación no sólo durante estos los años en la universidad, sino a todos desde el comienzo de mi educación.

A mis amistades que sé que siempre puedo contar con ellos para lo que necesite.

José Carlos

Agradecimientos

A mi mamá, por ser mi primera dama, por ser mi inspiración más grande y encontrar siempre en ella una consejera, una maestra y una amiga. Por preocuparse cuando yo lo hacía, cuando tenía pruebas, cuando no comía y me ponía flaco, o cuando me quedaba en la escuela porque sí y más que nada, darle gracias por siempre cuidarme.

A mi papá, por ser mi guía, por ser un ejemplo a seguir, por enseñarme todo lo que sé. Muchas veces aprendí dándome golpes por no hacer caso a algunos de sus consejos y después tuve que escuchar de su boca, más de una vez, un "te lo dije". Te quiero pipo.

A mi súper abuela, que siempre me está complaciendo en todo y que su mayor deseo siempre ha sido verme graduado.

A mi abuelito, el mayor de los Vargas, muchas gracias por darme tu apoyo, tu ejemplo y tu cariño.

A mis tías y tíos maternos y paternos, por todo el apoyo que me han brindado durante el transcurso de mi carrera.

A mi tía Yaremis, por siempre estar pendiente de todo lo que me puede hacer falta, por todo lo que ha hecho por mí y por todo su amor.

A mi tío Hanier, por toda la ayuda que me ha dado y por todo el amor de todos estos años.

A mis primos, Marlon y Jorgito, por ser más que primos, hermanos y vivir esta travesía junto conmigo en estos años.

A mis primitos Antuan y Antony porque sé que son grandes de corazón y de mente y que van a seguir mis pasos hacia la universidad.

A mi compañero de tesis, por vivir estos cinco años de amistad a mi lado como compañeros de aula y ahora, este último año, como un amigo indiscutible. Gracias por ser mi compañero de tesis, alguien no podría ocupar ese lugar mejor que tú.

A muchos más que no me alcanza la hoja para escribir, por favor no se sientan dolidos.

En general a toda mi familia por preocuparse por mí en todo este tiempo y por todo su cariño.

A todas mis amistades, Dayans, Reinier, Francisco, Lídice, Pablito y otros.

A mi gente de la cueva El Gabbo, Ernesto, Yara, Yoe y El teacher, por ser compañeros de apartamento.

A la gente de mi aula que de una forma u otra ayudaron a hacer este sueño realidad.

Agradecimientos

A la instructora Made por ser la mejor tía de la UCI, y por ponerme a limpiar y organizar más de una vez.

A los profesores que se esforzaron tanto porque aprendiéramos a ser buenos profesionales.

Al tribunal, por el apoyo brindado durante los talleres y fuera de ellos.

A mis tutoras Dailét y Ana Marys por tantas noches de desvelo a nuestro lado y todo el apoyo brindado.

Agradecimiento doble a Dailét por ser mi tutora, mi amiga y mi novia, nadie tiene tanta responsabilidad con esos tres títulos.

Por último y no por ser menos importante, a la revolución y al comandante eterno Fidel Castro Ruz por darme la oportunidad de estudiar en esta universidad.

Yairon

Dedicatoria

DEDICATORIA:

A nuestros padres, por creer en nosotros y acompañarnos en los momentos difíciles de este gran camino, dándonos ejemplos dignos de superación y entrega, porque gracias a ustedes, hoy podemos ver alcanzada nuestra meta, y porque el orgullo que sentimos nosotros de que sean nuestros padres, fue lo que nos hizo ir hasta el final. Va para ustedes, por lo que valen, porque admiramos su fortaleza y por lo que han hecho de nosotros.

A familiares y amigos. Gracias por haber fomentado en nosotros el deseo de superación y el anhelo de triunfo en la vida.

Mil palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles.

A todos, espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

José Carlos y Yairon

RESUMEN:

La Universidad de las Ciencias Informáticas se rige por la Resolución 60/2011 de la Contraloría General de la República para garantizar una correcta implementación del control interno. En la Facultad 3 el Vicedecanato de Administración y Servicios es el encargado de regir las pautas para su implementación. Debido a la gran documentación que se maneja de forma manual y en formato duro, se presentan problemas de pérdida, deterioro y desactualización de la información. Por esta razón, se hace necesario crear una solución informática que permita contribuir a elevar el control y disponibilidad de la información asociado al control interno en la Facultad 3. Como solución a la problemática, se desarrolló el Sistema Integral de Control Interno para el Vicedecanato de Administración y Servicios de la Facultad 3, bajo la metodología de desarrollo Proceso Unificado Ágil con la versión establecida por la universidad y siguiendo las políticas de soberanía tecnológica establecidas por el país. El sistema desarrollado fue probado funcionalmente mediante la aplicación de los métodos de pruebas de caja negra y caja blanca. Además, se realizó la validación de la investigación mediante la aplicación de la técnica de Iadov, donde se evidenció un alto nivel de satisfacción por parte del cliente.

Palabras claves: control interno, Resolución 60/2011, sistema de control interno.

ABSTRACT

The University of Informatics Sciences is governed by the Resolution 60/2011 of the General Comptroller of the Republic to guarantee a correct implementation of the internal control. The Vice Deanery for Administration and Services of School 3 is in charge of ruling the guidelines for its implementation. Due to the great amount of documents being managed manually and in hard format, the Vice Deanery is presenting problems with lost, deteriorated and outdated information. Therefore, creating an informatics solution that contributes to enhance the control and availability of the information associated to the internal control at School 3 is needed. As a solution to the problematic, the Integral System of Internal Control for the Vice Deanery for Administration and Services of School 3 was developed, under the guidance of the Agile Unified Process development methodology with the version established by the university and following the technological sovereignty policy. The system developed was verified by the application of the software testing methods Black box and White box. Moreover, the validation of the research was performed applying the Iadov technique, where a high level of satisfaction from the client was evidenced.

Key words: internal control, Resolution 60/2011, system of internal control.

ÍNDICE

Índice de contenidos

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El control interno (CI) ha adquirido en el mundo una gran importancia para la alta dirección de órganos, organismos, organizaciones y entidades. Brinda un enfoque de mejoramiento continuo extendido a todas las actividades inherentes a la gestión y efectuado por la dirección y el resto del personal. Se implementa mediante un sistema integrado de normas y procedimientos, que contribuyen a prever y limitar los riesgos internos y externos, proporciona una seguridad razonable al logro de los objetivos institucionales y una adecuada rendición de cuentas. (Contraloría General De La República, 2011). El CI determina además, el cumplimiento de los objetivos estratégicos en las entidades, a partir de identificar y esclarecer los riesgos asociados con cada actividad y proceso, sustentado dicho criterio en el cuidado de los activos, los intereses que se persiguen y previsión de fraudes y riesgos innecesarios (Cra. Daniela Biasco, 2008).

En Cuba el CI ha sufrido varias transformaciones desde los inicios del triunfo de la Revolución, debido a los diversos sistemas económicos aplicados en el país. En la derogada Resolución 297 del 2003 del Ministerio de Finanzas y Precios, se define el marco conceptual a aplicar en Cuba en lo que a CI respecta, la cual retoma y aplica el concepto sobre esta temática expuesto en el Informe COSO[1].

La Resolución 60 de fecha 1ro de marzo de 2011, “Sistema de Control Interno” de la Contraloría General de la República de Cuba, establece las normas y principios básicos de obligada observancia para los sujetos de las acciones de auditorías, supervisión y control de ese Órgano, constituyendo un modelo estándar del Sistema de Control Interno (SCI). El SCI es de preocupación para los directivos de las entidades, en mayor o menor grado, con diferentes enfoques y terminologías. Esto ha permitido que con el tiempo se hayan planteado diferentes concepciones acerca del mismo, sus principios, así como elementos que se deben conocer e instrumentar en la entidad cubana actual. Siendo necesario brindarle a los cuadros, dirigentes, funcionarios y demás trabajadores, un instrumento de trabajo que le permita implementar en sus entidades el SCI (Fuentes, y otros, 2012).

La Universidad de las Ciencias Informáticas (UCI), al igual que otras instituciones del país, diseña, armoniza, implementa y se autocontrola de forma sistemática de acuerdo con su misión, visión,

[1] **Committee of Sponsoring Organizations of the Treadway Commission.** Documento que contiene las principales directivas para la implantación, gestión y control de un sistema de control.

objetivos, estrategias fundamentales, características, competencias y atribuciones, en correspondencia con lo establecido en la Ley No. 107 y valida el SCI de acuerdo con su estructura.

El Vicedecanato de Administración y Servicios (VDA) de la Facultad 3 de la UCI está a cargo del proceso del CI de la facultad. En el mismo se controlan los cinco componentes definidos en la Resolución 60 de la Contraloría General de la República: Ambiente de control, Gestión y prevención de riesgos, Actividades de control, Información y comunicación y Supervisión y monitoreo. Para ello el VDA cuenta con el Sistema de Administración y Economía de la Facultad 3 (SAEF3), donde el Módulo Control Interno permite establecer un control sobre los procesos de adecuación, aplicación y seguimiento a la Guía de Autocontrol (GAC) en la Facultad 3. Sin embargo, estas funcionalidades solo responden a cinco aspectos del componente Supervisión y Monitoreo, de un total de 140 aspectos asociados a los cinco componentes.

De esta forma la gestión de los componentes restantes se hace de forma manual y se maneja un gran cúmulo de documentación, lo que trae como consecuencia la existencia de errores durante la gestión de los mismos al contar con información desactualizada. Por otra parte, se ha presentado deterioro y pérdida de la información, lo que puede llegar a presentar deficiencias en una correcta implementación del CI. Además, la gran cantidad de datos disponible solo en formato duro, dificulta el control y disponibilidad de la información.

Teniendo en cuenta lo anteriormente expuesto, el **problema científico** queda expresado de la siguiente manera: ¿cómo gestionar la información de asociada al control interno de la Facultad 3 de manera que se eleve la disponibilidad y el control de la información para el perfeccionamiento continuo?

A partir del problema planteado se define como **objeto de estudio** de la investigación: la informatización de los procesos asociados al control interno, enmarcado en el **campo de acción**: la informatización de los procesos asociados al control interno en la Facultad 3. Por tanto, se define la siguiente **idea a defender:** el desarrollo del Sistema Integral de Control Interno para la Facultad 3, contribuirá a elevar el control y disponibilidad de la información de dicho proceso.

Para dar solución al problema planteado se define como **objetivo general**: desarrollar el Sistema Integral de Control Interno para la Facultad 3, de manera que contribuya a elevar el control y disponibilidad de la información de los procesos asociados al control interno de la Facultad 3. El objetivo general se desglosa en los siguientes **objetivos específicos:**

1. Definir el marco teórico de la investigación mediante el estudio y el análisis de los principales referentes teóricos para el desarrollo de la solución.
2. Modelar el análisis y diseño de la solución para la posterior implementación de los procesos asociados al control interno de la Facultad 3.
3. Implementar los componentes de software de los procesos asociados al control interno de la Facultad 3.
4. Valorar la efectividad de la solución propuesta mediante la realización de pruebas de caja negra y caja blanca.
5. Validar las variables de la investigación mediante la aplicación de la técnica Iadov.

Para dar cumplimiento a los objetivos específicos se aplicaron los métodos de investigación:

Teóricos:

- **Análisis - Histórico – Lógico:** para la recopilación de información sobre cómo el CI ha evolucionado desde su surgimiento, tanto en el ámbito nacional como en el internacional, y a través de estos períodos cómo se evidencia su funcionamiento y desarrollo.
- **Analítico - Sintético:** para la realización del análisis de la bibliografía consultada referente al CI, con el objetivo de sintetizar los elementos relevantes para el desarrollo de la solución.

Empíricos:

- **Entrevista no estructurada o libre:** para la recopilación de información relevante para el desarrollo de la solución, mediante intercambios con el cliente. Las entrevistas fueron realizadas principalmente a los trabajadores del VDA y algunos cuadros de la facultad.
- **Observación:** para visualizar el estado actual del entorno del VDA, obteniendo la información a partir de la percepción propia con el objetivo de lograr una mejor comprensión de los procesos por parte de los desarrolladores.

Resultado esperado:

Sistema Integral de Control Interno para el Vicedecanato de Administración y Servicios de la Facultad 3.

La **estructura del trabajo** quedó constituida por: introducción, tres capítulos, conclusiones, recomendaciones y bibliografía. Se incluye un grupo de figuras y tablas, así como anexos que facilitan

la comprensión del documento. A continuación se describen los principales elementos abordados en los tres capítulos:

Capítulo 1. Fundamentación teórica: en el capítulo se presentan los elementos teóricos-conceptuales vinculados con la problemática. Se realiza un estudio del control interno en el ámbito nacional e internacional, así como de las soluciones informáticas desarrolladas para su gestión con vista a identificar posibles fortalezas a ser reutilizadas en la solución del problema. Para el análisis de los elementos aplicables a la solución, se abordan elementos técnicos como: lenguajes, metodología de desarrollo y herramientas utilizadas.

Capítulo 2. Características del Sistema Integral de Control Interno: en el capítulo, a partir del análisis del funcionamiento y necesidades actuales del VDA, se identifican las necesidades del cliente, las mismas se describen y se aplican técnicas para su validación. Se concibe la arquitectura mediante la aplicación de un patrón arquitectónico y se realiza la modelación del diseño haciendo uso de los patrones de diseño. Por último, para corroborar la calidad del diseño realizado, se aplican métricas de validación del diseño.

Capítulo 3. Implementación y análisis de los resultados: en el capítulo se muestran los resultados de la implementación de los componentes de la solución, para lo cual se emplearon estándares de codificación que facilitan la comprensión del código. Además, se describen los resultados de la aplicación de pruebas al sistema, así como de la satisfacción del cliente con la solución mediante la técnica Iadov.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción del capítulo

En el presente capítulo se precisan elementos teóricos que respaldan la investigación y el desarrollo del tema propuesto. Se realiza un análisis alrededor de los conceptos asociados al dominio del problema, así como la revisión, en distintas literaturas, acerca de sistemas informáticos del ámbito nacional e internacional. Se describen además, la metodología de desarrollo de software, herramientas y lenguajes de programación a utilizar en la solución.

1.2. Marco teórico. Conceptos y definiciones

A continuación se presentan los conceptos asociados al dominio del problema para un mejor entendimiento de todos los aspectos que se tratarán posteriormente.

1.2.1. Control Interno

Para Cuellar, Estupiñan y Fonseca (Cuellar Mejía 2008; Estupiñan 2006; Fonseca 2011), el CI comprende el plan de organización con los métodos y procedimientos que en forma coordinada se adoptan en un negocio para la protección de sus activos, la obtención de información financiera correcta y segura, la promoción de eficiencia de operación y la adhesión a políticas prescritas por la dirección.

En el entorno nacional y ajustado el término a las particularidades de Cuba, se define el CI por la Contraloría General de la República como “el proceso integrado a las operaciones con un enfoque de mejoramiento continuo, extendido a todas las actividades inherentes a la gestión, efectuado por la dirección y el resto del personal; se implementa mediante un sistema integrado de normas y procedimientos, que contribuyen a prever y limitar los riesgos internos y externos, proporciona una seguridad razonable al logro de los objetivos institucionales y una adecuada rendición de cuentas” (Contraloría-General-de-la-República 2011). Este concepto, atendiendo al marco de la investigación es el adoptado por los autores.

El SCI está formado por cinco componentes interrelacionados entre sí, en el marco de los principios básicos y las características generales; estos son los siguientes (Contraloría-General-de-la-República 2011): Ambiente de control, Gestión y prevención de riesgos, Actividades de control, Información y comunicación, y Supervisión y monitoreo.

1.2.2. Ambiente de control

El componente Ambiente de control se define como las pautas para el funcionamiento legal y armónico de los órganos, organismos, organizaciones y demás entidades, para el desarrollo de las acciones. Se considera la base de los demás componentes, pues conforma el conjunto de buenas prácticas y documentos referidos a la constitución de la organización, al marco legal de las operaciones aprobadas, a la creación de sus órganos de dirección y consultivos, a los procesos, sistemas, políticas, disposiciones legales y procedimientos; lo que tiene que ser del dominio de todos los implicados (Contraloría-General-de-la-República 2011).

1.2.3. Gestión y prevención de riesgos

El componente Gestión y prevención de riesgos define las bases para la identificación y análisis de los riesgos que enfrentan las organizaciones para alcanzar sus objetivos. Una vez clasificados los riesgos en internos y externos, por procesos, actividades y operaciones, y evaluadas las principales vulnerabilidades, se determinan los objetivos de control y se conforma el Plan de prevención de riesgos para definir el modo en que deberán gestionarse (Contraloría-General-de-la-República 2011).

1.2.4. Actividades de control

El componente Actividades de control establece las políticas, disposiciones legales y procedimientos de control para gestionar y verificar la calidad de la gestión para el cumplimiento de los objetivos y misión de las organizaciones. Las actividades de control son aplicables a las operaciones de todo tipo, las que tributan a la fiabilidad de la información financiera y al cumplimiento de las disposiciones legales correspondientes al marco de desarrollo de la actividad, así como a la comprobación de las transacciones u operaciones económicas que le dan cobertura a los objetivos y metas en cuanto a su exactitud, autorización y registro contable conforme a las normas cubanas establecidas al efecto, con un enfoque de mejoramiento continuo (Contraloría-General-de-la-República 2011).

1.2.5. Información y comunicación

El componente Información y comunicación precisa que las entidades deben disponer de información oportuna y fiable, así como definir el sistema de información adecuado a sus características. Genera datos, documentos y reportes que sustentan los resultados de las actividades operativas, financieras y relacionadas con el cumplimiento de los objetivos, metas y estrategias, con mecanismos de retroalimentación y la rendición transparente de cuentas. La información debe protegerse y conservarse según las disposiciones legales vigentes (Contraloría-General-de-la-República 2011).

1.2.6. Supervisión y monitoreo

El componente Supervisión y monitoreo está dirigido a la detección de errores e irregularidades que no fueron detectados con las actividades de control, permitiendo realizar las correcciones y modificaciones necesarias. Estas se realizan mediante dos modalidades de supervisión: actividades continuas, que son aquellas que incorporadas a las actividades normales, generan respuestas dinámicas, y evaluaciones puntuales que son ejecutadas por los responsables de las áreas, por auditorías internas y externas (Contraloría-General-de-la-República 2011).

1.3. Análisis de las soluciones existentes

Para el desarrollo de la propuesta de solución fue necesario realizar un análisis de los sistemas similares que existen a nivel nacional e internacional. El resultado del análisis se describe a continuación.

1.3.1. Meycor COSO AG

Esta herramienta contiene una guía metodológica que facilita la aplicación de la metodología COSO y lo asiste durante todo el proceso de revisión. Permite la creación de grupos de trabajo y de revisores, que facilitan la distribución de las tareas. Además posibilita la asignación a los revisores de privilegios de administración. Cuenta con informes inmediatos y la posibilidad de obtener datos históricos sobre las asignaciones de cada activo, además de realizar la depreciación fiscal de cada uno (Ma et al. 2012).

Este sistema es adoptado por diferentes países, sin embargo no se ajusta a las normas cubanas para el CI, por lo cual, no satisface las necesidades de la investigación.

1.3.2. Etecsa

Es un sistema utilizado por la Dirección Nacional de la Empresa de Telecomunicaciones de Cuba (ETECSA), para llevar el CI de las actividades en las unidades de dicha entidad. Permite la creación de una GAC que implemente el informe COSO. A los administradores del sistema se les permite revisar la GAC de cada unidad y modificarla (Larramendi Valdes and Jerez Camps 2011).

Este sistema no se rige por la Resolución 60 referente al CI. Además, no brinda la posibilidad a los usuarios de almacenar en el sistema evidencias por cada aspecto de la guía, por lo que hay que trasladarse hasta las empresas para verificar la veracidad del cumplimiento de los aspectos. Los administradores de cada empresa crean su GAC por separado y los usuarios de una empresa pueden consultar la GAC de otras, lo cual atenta contra la confidencialidad de la información. Por lo antes

expuesto, los autores consideran que el sistema en cuestión, no soluciona la problemática de la presente investigación.

1.3.3. Sistema Informático de gestión de Auditoría y Control (SIGAC)

El SIGAC responde a las necesidades de interoperabilidad entre los módulos Planificación y Acciones de Control, requeridos en el antes existente Ministerio de Auditoría y Control de Cuba. La integración de aplicaciones para SIGAC impide que los módulos Planificación y Acciones de Control realicen el tratamiento de la información de manera aislada, con lo cual el valor de los módulos individuales puede ser aprovechado al máximo (Díaz Estrada and González Morales 2009).

Como aspecto negativo se identificó que el sistema no gestiona los cinco componentes del CI por los que se rige la Resolución 60, solo se enmarca en la planificación y control de las auditorías. Por tal motivo, no es considerado para dar solución a la problemática.

1.3.4. Farola 297

Farola es un sistema diseñado para satisfacer todos los requerimientos de la Resolución 297 del Ministerio de Auditoría y Control referidos a la actividad Evaluación de Riesgos. Se basa en una plataforma distribuida que permite almacenar datos importantes referentes a la Gestión y Prevención de los Riesgos. Inicia en la detección de estos por cada proceso, teniendo en cuenta las actividades que se vinculan, así como los objetivos que estos pueden afectar. Brinda reportes de las variables relacionadas por diferentes criterios de filtrado, una estadística muy detallada de dichos reportes, un seguimiento de las acciones planificadas tanto preventivas, correctivas como de mejora con el objetivo de minimizar o erradicar los riesgos que pueden convertirse, con determinada probabilidad, en no conformidades futuras. Es una herramienta que se vincula de forma sistémica con la administración por objetivos con enfoque a procesos. Puede ser empleado para la implantación del Sistema de Gestión de la Calidad y del Perfeccionamiento Empresarial en las organizaciones (Trujillo González 2010).

Este sistema, a pesar de enmarcarse en el entorno nacional, tampoco se rige por la vigente Resolución 60/2011. Motivo por el cual no se considera como parte de la solución.

1.3.5. Sistema de Gestión de Control Interno (SIGCI)

SIGCI es un sistema web que informatiza la aplicación de la GAC según lo establecido en la Resolución 60/2011. SIGCI centra su alcance en las áreas de la UCI, aunque puede ajustarse a otras entidades. El principal objetivo que persigue es asegurar una coherencia lógica entre la GAC y el plan de medidas, garantizando la calidad y viabilidad de la aplicación de la GAC. Cuenta entre sus principales

funcionalidades con: aplicar la GAC, elaborar el plan de medidas, generar el consolidado de la GAC, mostrar estadísticas sobre la aplicación de una GAC y monitorear en tiempo real la ejecución de este proceso. Se desarrolló con tecnología web actualizada, novedosa y de gran uso por las empresas desarrolladoras de software (UCI 2015).

La relevancia de este sistema, radica en su adherencia a la actual Resolución 60/2011 y la posibilidad de establecer el control de las áreas subordinadas a la UCI para la aplicación de la GAC, sin embargo, solo se limita a gestionar este aspecto del componente Supervisión y Monitoreo. Además, SIGCI gestiona la GAC a nivel macro dentro de la universidad (tiene en cuenta las Facultades pero no las áreas dentro de las facultades). Teniendo en cuenta la necesidad de informatizar los cinco componente del CI y de llevar la gestión del CI a todas las áreas subordinadas a la Facultad 3, no constituye una solución aplicable al problema de la investigación.

1.3.6. Sistema para la Planificación de Actividades (SIPAC)

El sistema SIPAC fue desarrollado para dar cumplimiento a la Instrucción 1 de los Consejos de Estado y de Ministro, con el fin de garantizar una organización en la planificación de las actividades y objetivos de la entidad. Su primera versión es liberada en el 2010 y puesta en prueba experimental en las entidades de las FAR. Este sistema brinda todo un mecanismo de gestión de estructuras, dominios, roles, entre otros aspectos que constituyen la base para la compartimentación de la información. Presenta una arquitectura orientada a componentes, que posibilita, dentro de la misma plataforma tecnológica, generalizar la solución e ir agregando nuevos componentes según la necesidad que presente el cliente (Díaz Estrada and González Morales 2009).

El sistema gestiona la planificación de actividades dentro del componente Ambiente de Control y se encuentra institucionalizado, por lo que se determina (a petición del cliente) no implementar estas funcionalidades en la solución de la investigación, ya que SIPAC lo realiza satisfactoriamente. No obstante, no gestiona el resto de los aspectos del componente Ambiente de Control ni los componentes restantes del CI, por lo que se considera un aporte a la informatización del CI, pero no se adopta como parte de la investigación.

1.3.7. Módulo Control Interno del SAEF3

El sistema SAEF3 fue desarrollado para el VDA con el propósito de gestionar los procesos: económico, de guardia obrera-estudiantil y de CI en la Facultad 3. Específicamente el Módulo Control Interno, gestiona la aplicación de la GAC, la generación del informe y el plan de medidas correspondiente. Incluye las funcionalidades del SIGCI pero adaptado a las áreas de la Facultad 3. Además, permite

llevar un control sobre el seguimiento al plan de medidas de cada área y posibilita la generación de informes y planes de medida, no solo como resultado de la aplicación de la GAC, sino también por la realización de supervisiones a las áreas (Sánchez González et al. 2016).

Considerando que el módulo implementa las funcionalidades asociadas a la aplicación de la GAC y la realización de supervisiones a las áreas subordinadas a la Facultad 3, pero no gestiona todos los componentes, se determinó que no era necesario reimplementar sus funcionalidades, por lo que se considera un aporte a la informatización del CI, pero no se adopta como parte de la investigación.

A continuación se muestran las tablas comparativas correspondientes a indicadores asociados a la soberanía tecnológica por la cual abogan Cuba y la UCI y, teniendo también en consideración si los sistemas se rigen o la Resolución 60/11 y los procesos que se realizan en el VDA (Tablas 1 y 2).

Sistemas	**Exento de pago**	**Código Abierto**	**Multiplataforma**	**Resolución 60/2011**
Etecsa	✓	-	-	-
MEYCOR COSO AG	-	-	-	-
SIGAC	✓	✓	✓	✓
Farola 297	✓	-	✓	-
SIPAC	✓	✓	✓	✓
SIGCI	✓	-	✓	✓
Módulo Control Interno de SAEF3	✓	✓	✓	✓

Tabla 1. Comparación entre los sistemas

Fuente: (Elaboración propia)

Sistemas / Componentes	Etecsa	MEYCOR COSO AG	SIGAC	Farola	SIGCI	Módulo Control Interno de SAEF3	SIPAC
Ambiente de Control	-	-	-	-	-	-	-
Gestión y Prevención de Riesgos	-	-	-	✓	-	-	-
Actividades de Control	-	-	-	-	-	-	✓
Información y Comunicación	-	-	-	-	-	-	-
Supervisión y Monitoreo	-	-	-	-	✓	✓	-

Tabla 2. Componentes de la Resolución 60/11 que gestionan los sistemas

Fuente: (Elaboración propia)

Como se muestra en las tablas 1 y 2, la mayoría de los sistemas homólogos estudiados, no se rigen por la Resolución 60/2011 vigente en el país. En el caso de los sistemas SIPAC, SIGCI y el Módulo Control Interno del SAEF3, gestionan parcialmente al menos uno de los componentes del CI; SIPAC dentro de Ambiente de control gestiona la planeación, SIGCI y Módulo Control Interno del SAEF3 dentro de Supervisión y Monitoreo, gestionan la GAC. No obstante en todos los casos se visualizan carencias en la gestión de la totalidad de los componentes, por tanto no satisfacen en su totalidad las necesidades del VDA de la Facultad 3 y se decide desarrollar un sistema que integre la gestión de los cinco componentes, incorporando las funcionalidades del Módulo Control Interno de SAEF3.

1.4. Proceso de desarrollo de software

Un proceso de desarrollo de software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. Su misión es transformar los requerimientos del usuario en un producto de software (Jacobson et al. 2000). Para guiar el proceso de desarrollo de software se requiere de la aplicación de una metodología de desarrollo de software.

1.4.1. Metodología de desarrollo

Una metodología consiste en múltiples herramientas, modelos y métodos para asistir en el proceso de desarrollo de software donde se definen con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas (Figueroa et al. 2008).

Para la elección de una metodología se deben tener en consideración las características del proyecto de desarrollo de software. Atendiendo al programa de mejora de procesos de software en el que se encuentra enfrascada la UCI, se determinó escoger la metodología AUP con la variación establecida por la UCI en el cuarto escenario, debido a que el equipo de trabajo es pequeño, el tiempo de entrega del producto es relativamente corto, el cliente forma parte del equipo de desarrollo y la comunicación es activa.

Metodología AUP versión de la UCI

Para el desarrollo de la aplicación se utilizó la metodología de desarrollo aprobada por la universidad para la actividad productiva de la misma, de tal forma que se adapte a su ciclo de vida definido. Esta metodología, es una variante realizada por la UCI a la metodología ágil AUP y está definida por la universidad como el documento rector de la actividad productiva. Queda estructurada en tres fases las cuales son detalladas a continuación (Rodríguez Sánchez 2015) (Ver tabla 3).

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 3: Ciclo de vida de la Metodología AUP-versión UCI

Fuente: (Rodríguez Sánchez 2015)

1.4.2. Patrones arquitectónicos

Los patrones arquitectónicos son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Permiten clasificar y describir formas de solucionar problemas que ocurren de forma frecuente en el desarrollo y están basados en la recopilación del conocimiento de los expertos en desarrollo de software (Pressman 2010).

Con la adopción de Symfony 2 como marco de trabajo para el desarrollo, se asume el patrón Modelo Vista Controlador (MVC) que es muy empleado para el desarrollo de aplicaciones web en la actualidad. MVC, consta de tres módulos: el Modelo, la Vista y el Controlador. Separa la lógica de negocio de la interfaz de usuario, facilita la evolución por separado de ambos aspectos e incrementa la reutilización y la flexibilidad (BAHIT 2011; González and Romero 2012).

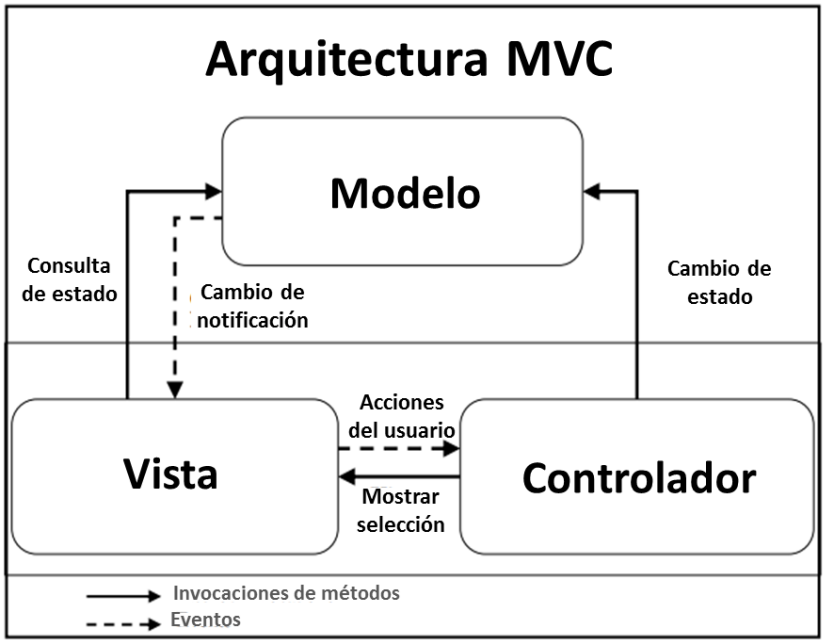


Figura 1. Modelo-Vista-Controlador

Fuente: (González and Romero 2012)

1.4.3. Patrones de diseño

El patrón es una descripción del problema y la esencia de su solución, de modo que la solución puede reutilizarse en diferentes configuraciones (Sommerville 2011). Entre los patrones de diseño más reconocidos y aplicados en el desarrollo de software se encuentran Patrones Generales de Software para Asignar Responsabilidades (GRASP por sus siglas en inglés) y Patrones Banda de los Cuatro (GoF por sus siglas en inglés).

Patrones GRASP

Para el desarrollo de la solución se estudia la aplicación de los patrones GRASP, los cuales describen los principios del diseño de objetos y la asignación de responsabilidades, expresados como patrones (Gamma 1995; Larman 2003; Larman 2004). Entre los más conocidos y empleados se encuentran:

- Experto: se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad.
- Creador: es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.
- Controlador: es el encargado de asignar la responsabilidad de controlar el flujo de eventos a una clase específica.
- Bajo acoplamiento: es el encargado de asignar una responsabilidad de manera que se minimicen las dependencias y se conserve bajo el acoplamiento.
- Alta cohesión: asigna una responsabilidad de forma tal que no exista sobrecarga innecesaria de funciones en las clases.

Patrones GoF

Los patrones GoF, conocidos así por las cuatro personas que lo propusieron, se clasifican en tres categorías: creacionales, estructurales y de comportamiento. Los patrones creacionales se ocupan del proceso de creación de clases y objetos; entre ellos se encuentran: solitario, método de fábrica, fábrica abstracta y constructor. Los patrones estructurales tratan la composición de clases y objetos, se ocupan de cómo estos se agrupan para formar estructuras más grandes, permitiendo que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos; entre ellos se encuentran: adaptador, fachada y decorador. Los patrones de comportamiento caracterizan la forma en que las clases interactúan y distribuyen la responsabilidad; entre ellos se encuentran: observador, iterador, comando, intérprete y mediador (Guerrero et al. 2013; Larman 2003; Pressman 2010).

1.4.4. Calidad de software

La calidad del software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente (Pressman 2010). Es el conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas (ISO 1994). La calidad es verificada y validada durante todo el proceso de desarrollo del software.

Métricas para la validación del diseño:

Chidamber y Kemerer realizan una propuesta para medir la calidad del diseño, entre las cuales se encuentran: Métodos Ponderados por Clases (MPC), Árbol de Profundidad de Herencia (APH), Número de Descendientes (ND), Acoplamiento entre Clases Objeto (ACO), Relaciones entre Clases (RC) y Carencia de Cohesión de los Métodos (CCM) (Chidamber and Kemerer 1994; Pressman 2010).

Lorenz y Kidd proponen un conjunto de métricas para la validación del diseño del software siguiendo el paradigma orientado a objetos, para ello las clasifican en cuatro amplios grupos: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño de las clases se centran en el recuento de atributos y operaciones para cada clase y los valores promedio para el sistema como un todo; las métricas basadas en la herencia se centran en la forma en que las operaciones se reutilizan en la jerarquía de clases; las métricas para valores internos examinan la cohesión y los aspectos orientados al código y las métricas orientadas a valores externos examinan el acoplamiento y reutilización entre las clases. Una de sus métricas más empleadas es Tamaño Operacional de Clases (TOC), pues permite medir el total de atributos y operaciones encapsulados en una clase para valorar la sobrecarga de responsabilidades asignadas (Lorenz and Kidd 1994; Pressman 2010).

Por la relevancia para la investigación se considera oportuno aplicar de Lorenz y Kidd la métrica TOC, pues permite visualizar si se distribuyen correctamente las asignaciones de responsabilidades entre las clases, verificándose así la cohesión y armonía entre las mismas, y de Chidamber y Kemerer la métrica RC para evaluar el grado de acoplamiento entre las clases.

En la siguiente tabla se muestra el resumen de las métricas a aplicar para validar el diseño de la solución.

Métrica	Atributos de calidad
RC	Acoplamiento, Complejidad de Mantenimiento, Cantidad de Pruebas, Reutilización.
TOC	Responsabilidad, Complejidad de Implementación, Reutilización

Tabla 4. Métricas para la validación del diseño

Fuente: (Chidamber and Kemerer 1994; Lorenz and Kidd 1994; Pressman 2010)

- Responsabilidad: consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta. Un aumento del TOC indica un aumento de la responsabilidad asignada a la clase.

- Complejidad de implementación: consiste en el grado de dificultad que tiene implementado un diseño de clases determinado. Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- Reutilización: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software. Un aumento del TOC o el RC implica una disminución del grado de reutilización de la clase.
- Acoplamiento: consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización. Un aumento del RC implica un aumento del acoplamiento de la clase.
- Complejidad del mantenimiento: consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto. Un aumento del RC implica un aumento de la complejidad de mantenimiento de la clase.
- Cantidad de pruebas: consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases) diseñado. Un aumento del RC implica un aumento de la cantidad de pruebas de la clase.

Atributo de calidad	Categoría	Criterio
Responsabilidad	Baja	CO < = Promedio
	Media	Promedio< CO < 2*Promedio
	Alta	CO > = 2* Promedio
Complejidad de implementación	Baja	CO < = Promedio
	Media	Promedio < CO < 2*Promedio
	Alta	CO > 2* Promedio
Reutilización (RC)	Baja	CRU > 2* Promedio
	Media	Promedio < CRU < 2*Promedio
	Alta	CRU < = Promedio
Reutilización (TOC)	Baja	CRU > 2* Promedio
	Media	CRU entre Promedio y 2*Promedio
	Alta	CRU < = Promedio
Acoplamiento	Ninguna	0
	Baja	1

	Media	2
	Alta	> 2
Complejidad de Mantenimiento	Baja	CRU < = Promedio
	Media	Promedio < CRU < 2*Promedio
	Alta	CRU > 2* Promedio
Cantidad de Pruebas	Baja	CRU < = Promedio
	Media	Promedio< CRU < 2*Promedio
	Alta	CRU > 2* Promedio

Tabla 5. Rango de valores para evaluar atributos de calidad de métricas TOC y RC

Fuente: Elaboración propia

Cantidad de relaciones de uso (CRU): Cantidad de relaciones de la clase con otras clases.

Cantidad de operaciones (CO): Cantidad de métodos de cada clase.

Cantidad de clases(N).

Promedio (RC): $\frac{\sum_1^N CRU}{N}$ (1)

Promedio (TOC): $\frac{\sum_1^N CO}{N}$ (2)

1.4.5. Pruebas de software

Para una mejor organización de la ejecución de las pruebas de software es conveniente desarrollar una estrategia de pruebas de software donde se especifiquen los niveles, tipos y métodos de pruebas a aplicar.

Niveles de prueba

Los niveles de prueba son aplicados en diferentes escenarios o niveles de trabajo (Pressman 2010):

- Prueba de desarrollador: es diseñada e implementada por el equipo de desarrollo. Se recomienda que estas pruebas cubran más que las pruebas de unidad.
- Pruebas independientes: es diseñada e implementada por alguien externo al grupo de desarrolladores. El objetivo de estas pruebas es proporcionar una perspectiva diferente y en un ambiente más rico que los desarrolladores.
- Pruebas de unidad: enfoca los esfuerzos de verificación en la unidad más pequeña del diseño de software: el componente o módulo de software. Es aplicable a componentes representados en el modelo de implementación para verificar que funcionen como se espera.

- Pruebas de integración: son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. Es el proceso de combinar y probar múltiples componentes juntos.
- Pruebas de sistema: se hacen cuando el software está funcionando como un todo, están dirigidas a verificar el programa final después que todos los componentes han sido integrados.
- Pruebas de aceptación: es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales.

Métodos de prueba

Pruebas de caja blanca: Se centran en evaluar la ejecución por lo menos una vez de cada sentencia del programa (Pressman 2010).

Pruebas de caja negra: se llevan a cabo en la interfaz del software y examinan aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del software (Pressman 2010).

Tipos de prueba

Para verificar de que el sistema cumpla con los atributos de calidad que se encuentran en correspondencia con los requisitos no funcionales se aplican diferentes tipos de pruebas (Grady and Caswell 1987; Pressman 2010):

- **Funcionalidad**: se califica de acuerdo con el conjunto de características y capacidades del programa, la generalidad de las funciones que se entregan y la seguridad general del sistema.
- **Usabilidad**: se evalúa teniendo en cuenta factores humanos, estética general, consistencia y documentación.
- **Confiabilidad**: se evalúa con la medición de la frecuencia y gravedad de las fallas, la exactitud de los resultados que salen, el tiempo medio para que ocurra una falla, la capacidad de recuperación ante esta y lo predecible del programa.
- **Rendimiento**: se mide con base en la velocidad de procesamiento, el tiempo de respuesta, el uso de recursos, el conjunto y la eficiencia.
- **Mantenibilidad**: combina la capacidad del programa para ser ampliable (extensibilidad), adaptable y servicial, y además que pueda probarse, ser compatible y configurable.

El principal objetivo del flujo de pruebas es evaluar la calidad del producto a través de la búsqueda de errores, la validación del cumplimiento de los requisitos y la validación del desempeño. Si bien es importante comprobar que se desarrolla un producto correctamente, también es esencial corroborar

que se desarrolló el producto correcto, para ello es aconsejable aplicar técnicas que permitan medir la satisfacción del cliente.

Técnica Iadov

La técnica Iadov permite medir la satisfacción del cliente con un producto, se compone de cinco preguntas: tres cerradas y dos abiertas, las cuales se reformulan en la investigación para valorar el grado de satisfacción de los clientes sobre un tema en específico. Una vez establecidas las preguntas se conforma el cuadro lógico de Iadov y el número resultante de la interrelación de las tres preguntas, indica la posición de los sujetos en la escala de satisfacción. La escala de satisfacción está dada por los criterios (Kuzmina 1970).

1. Máxima satisfacción.
2. Más satisfecho que insatisfecho.
3. No definida.
4. Más insatisfecho que satisfecho.
5. Máxima insatisfacción.
6. Contradictoria.

Pregunta cerrada 3	Pregunta cerrada 1								
	No			No sé			Sí		
	Pregunta cerrada 2								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

Tabla 6. Cuadro lógico de Iadov

Fuente: (Kuzmina 1970)

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y -1 de la siguiente forma:

Índice de satisfacción	Escala
Máxima satisfacción	+1
Más satisfecho que insatisfecho	0,5
No definido y contradictorio	0
Más insatisfecho que satisfecho	-0,5
Máxima insatisfacción	-1

Tabla 7. Índice de satisfacción de Iadov

Fuente: (Kuzmina 1970)

La satisfacción grupal (ISG) se calcula por la siguiente fórmula:

$$ISG = \frac{A(+1)+B(+0{,}5)+C(0)+D(-0{,}5)+E(-1)}{N} \quad (3)$$

Donde:

- A representa el número de sujetos con índice individual 1
- B representa el número de sujetos con índice individual 2
- C representa el número de sujetos con índice individual 3 ó 6
- D representa el número de sujetos con índice individual 4
- E representa el número de sujetos con índice individual 5
- N representa el número total de sujetos del grupo

A partir del análisis realizado se decide aplicar en la presente investigación: el nivel de pruebas de sistema; los tipos de prueba de funcionalidad, usabilidad y confiabilidad; los métodos de pruebas de caja negra y caja blanca; así como la técnica de Iadov para evaluar la satisfacción del cliente.

1.4.6. Estándares de codificación

Un estándar de codificación comprende aspectos de la generación de código que los programadores deben implementar de forma prudente. Es importante establecer un estándar de codificación al comenzar un proyecto para propiciar el trabajo de forma coordinada. Usar técnicas de codificación sólidas y aplicar buenas prácticas de programación con vistas a generar un código de alta calidad es relevante para obtener un buen rendimiento y contribuir a la calidad del software (Microsoft 2015).

1.4.7. Estudio de herramientas y tecnologías

A continuación se describen las herramientas y tecnologías para el marco de trabajo a utilizar en el desarrollo del sistema, teniendo en cuenta las características que debe cumplir y las necesidades del cliente.

1.4.8. Lenguajes de programación

Un lenguaje de programación es un lenguaje artificial para expresar programas de ordenador. Para definir un lenguaje de programación, se deben especificar: un conjunto de símbolos y palabras claves, así como las reglas gramaticales para construir sentencias sintáctica y semánticamente correctas (Rodríguez Sala et al. 2003). Para el desarrollo de la herramienta propuesta se emplearon los siguientes lenguajes:

HTML5

HTML5 es mantenido por la World Wide Web Consortium (W3C), es utilizado para la creación de páginas web estáticas. Esta versión incorpora algunas etiquetas nuevas para hacer que la estructura de la página web sea más lógica y funcional. El enfoque general ha cambiado bastante respecto a versiones anteriores de HTML, añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad. HTML5 está definido en base al Document Object Model (DOM), la representación interna de una web con la que trabaja un navegador, dejando de lado la representación real y definiendo a la vez un estándar HTML y XHTML (Anthes 2012; Gauchat 2012).

CSS 3.0 (Cascading Style Sheets, Hojas de Estilo en Cascada)

CSS es un lenguaje para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma de separar los contenidos y su presentación para la creación de páginas web complejas. Mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizarlo en dispositivos diferentes. Se utiliza para definir el aspecto de todos los contenidos. Esta forma de descripción de estilos ofrece a los desarrolladores el control sobre estilo y formato de sus documentos. Funciona a base de reglas para las declaraciones sobre el estilo de uno o más elementos (Gauchat 2012; Powell 2010).

PHP 5.6.15

PHP es el acrónimo recursivo del inglés Hypertext Pre-processor (Pre-procesador de hipertextos). Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. El cliente solamente recibe una página con el código HTML resultante de la ejecución del PHP. Como la página resultante contiene únicamente

código HTML, es compatible con todos los navegadores. Es necesario señalar que respecto a la seguridad, en muchas ocasiones PHP se encuentra instalado sobre servidores Unix o Linux, que brindan un elevado nivel de seguridad, presentan una gran estabilidad y permiten auditar la seguridad y privacidad de los datos. Este lenguaje de programación cuenta con una extensa librería de funciones (Cobo 2005; Welling and Thomson 2005).

Javascript 1.6

Javascript es un lenguaje de programación web en el lado del cliente que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlos, es decir, los programas escritos con Javascript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Eguíluz Pérez 2012; Gauchat 2012).

1.4.9. Marcos de trabajo (Framework por sus siglas en inglés)

Symfony 2.8.4

Symfony es un marco de trabajo diseñado para agilizar el desarrollo de las aplicaciones web. Se basa en el patrón MVC para separar la lógica de negocio, la lógica de servidor y la presentación. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Es multiplataforma y compatible con la mayoría de los gestores de bases de datos (Potencier and Zaninotto 2008). Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto (Aguiluz, 2013).

JQuery 1.9.0

JQuery es una biblioteca gratuita de JavaScript, cuyo objetivo principal es simplificar las tareas de creación de páginas web responsivas, acordes a lo estipulado en la Web 2.0, la cual funciona en todos los navegadores modernos. Otra de las grandes ventajas de JQuery es que se enfoca en simplificar los scripts y en acceder/modificar el contenido de una página web. Además, JQuery agrega una cantidad impresionante de efectos nuevos a JavaScript[2] (Bibeault and Kats 2008; Osmani 2012).

Bootstrap 3.1

Bootstrap es una colección de varios elementos web personalizables y funciones, empaquetados en una sola herramienta. Cuando se diseña una web con Bootstrap, los desarrolladores pueden elegir qué elementos utilizar. Los elementos personalizables de Bootstrap son una combinación de HTML, CSS y JavaScript y se le han añadido una variedad de funcionalidades tales como una selección amplia de complementos JQuery (Firdaus 2013; Hall 2013).

1.4.10. Entorno de Desarrollo Integrado (IDE por sus siglas en inglés)

Un IDE, es una aplicación de software compuesto por herramientas informáticas que proporciona servicios integrales para que los programadores de software desarrollen aplicaciones. Los IDE están diseñados para maximizar la productividad de los desarrolladores, proporcionando componentes muy unidos con interfaces de usuario similares. Presentan un único programa en el que se lleva a cabo todo el desarrollo de una aplicación y poseen múltiples características para la creación, modificación, compilación, implementación y depuración del software. Generalmente se componen de un editor de código, un compilador, un depurador y una interfaz gráfica (Netbeans, 2015).

PhpStorm 8.0

PhpStorm es un IDE inteligente para desarrollar aplicaciones en PHP, proporcionando herramientas esenciales como análisis de código y comprobación de errores. Las principales novedades en PhpStorm 8.0 incluyen: Soporte para PHP 5.6, depuración sin necesidad de configuraciones en los navegadores, compatibilidad con el marco de trabajo Symfony, editores de SQL con resultados editables y soporte para HTML5 (Chaudhary and Kumar 2014).

Heramientas de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Está compuesto por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, además de garantizar la seguridad e integridad de los mismos (Ramos Martín et al. 2006).

- PostgreSQL 9.1.14:

Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia de Distribución de Software Berkeley[3]. Da la posibilidad de que mientras un proceso es escrito en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases. Implementa el uso de retrocesos, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz. Posee la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos (Lockhart 1998; Narváez Coello 2014).

- PgAdmin 1.14.2-2:

Es una herramienta de código abierto para la administración de bases de datos PostgreSQL incluye: interfaz administrativa gráfica, herramienta de consulta SQL y editor de código procedural. PgAdmin se diseña para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos (pgAdmin 2016).

- Mapeo de Objeto Relacional (ORM por sus siglas en inglés)

ORM, es una técnica de programación que permite convertir datos entre el lenguaje de programación orientado a objetos y el sistema de base de datos relacional. Posibilita convertir automáticamente los objetos de datos de las aplicaciones orientadas a objeto en registros de datos primitivos de las bases de datos relacionales y viceversa (Enriquez and del Busto 2011; Keith and Schnicariol 2010). En el caso específico de Symfony, utiliza Doctrine como ORM.

- Doctrine:

Doctrine es un ORM para PHP 5.3.0 (y otras versiones superiores) que proporciona persistencia transparente de objetos PHP. Brinda una capa de abstracción de la base de datos y es una librería muy completa y configurable. Permite generar de forma automática la base de datos basándose en el modelo relacional entre las clases. Una de sus características más importantes es la posibilidad de acceder a la base de datos a través de un lenguaje integrado y propio llamado Lenguaje de Consultas Doctrine (DQL por sus siglas en ingles), permitiendo escribir consultas de una manera sencilla y flexible (Dunglas 2013).

[3] Sistema operativo derivado del sistema Unix realizado por la Universidad de California en Berkeley.

Apache 2.2.22-13:

Apache es el servidor web por excelencia, su facilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, de código abierto, altamente configurable y de diseño modular por lo que resulta muy sencillo ampliar sus capacidades. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda (Kabir 2003; Kew 2008).

1.4.11. Lenguaje Unificado de Modelado (UML)

Este lenguaje prescribe un conjunto de notaciones y diagramas estándares para modelar sistemas orientados a objetos y describe la semántica esencial de lo que estos diagramas y símbolos significan; posibilitando así visualizar, especificar y documentar los artefactos generados en el proceso de desarrollo del software (Larman 2003; Schmuller and Marín 2000).

1.4.12. Herramientas para el modelado del sistema

La ingeniería de Software Asistida por Computadoras (CASE por sus siglas en inglés) brinda soporte para desarrollar y mantener software. Además, se considera de gran ayuda al desarrollador en las fases del desarrollo de software, entre las herramientas CASE más utilizadas en la actualidad se encuentra Visual Paradigm para UML.

Visual Paradigm para UML 8.0

Es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: Análisis y diseño orientados a objetos, Construcción, Pruebas y Despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, haciéndolas mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Paradigm 2013; Sierra 2008).

Las herramientas y tecnologías antes mencionadas fueron seleccionadas debido a las potencialidades que presentan, además de que al sistema a desarrollar se le debe integrar un módulo, el cual ya tenía definidas estas herramientas y tecnologías para su implementación y que el equipo de desarrollo tiene experiencia en el uso de las mismas.

1.5. Conclusiones parciales

- El estudio de los sistemas existentes que gestionan el CI, constató que no abordan en su totalidad los aspectos asociados a los cinco componentes de la Resolución 60, lo que constituyó motivación para la investigación considerándose necesario incorporar al Módulo Control Interno de SAEF3, como parte de la solución integral.
- El empleo de la metodología AUP-UCI se considera oportuna para el desarrollo de la solución atendiendo a su estandarización de procesos para el cumplimiento de CMMI-Dev nivel 2 y su institucionalización en la actividad productiva de la UCI.
- Después de un estudio de las herramientas y tecnologías a utilizar se concluye que tendrán como característica más importante que son de código abierto garantizando la soberanía tecnológica en la que se encuentra inmensa el país.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción del capítulo

En el capítulo se realiza una breve descripción de la solución propuesta, se identifican las necesidades del cliente mediante las técnicas tormenta de ideas y entrevista con el cliente, y los requisitos no funcionales para el sistema. Se explica la concepción de la arquitectura y diseño del software mediante los artefactos generados, haciendo uso de patrones y aplicando métricas para la validación del diseño propuesto.

2.2 Descripción de la solución

El Sistema Integral de Control Interno es la solución propuesta que ofrece funcionalidades de apoyo a los procesos asociados al CI de la Facultad 3. Fue concebido para brindar un entorno de control que le permita a la facultad efectuar y controlar los procesos vinculados al CI de un modo sencillo y fácil, que facilita al proceso en general, la disminución de errores humanos y de pérdidas de información. Permite manejar los procesos que actualmente se realizan manualmente, a través de una aplicación web con una interfaz visual fácil de manipular.

Este sistema es el encargado de gestionar los aspectos asociados al CI en la Facultad 3, con el objetivo de elevar la disponibilidad y control de la información generada en este proceso. Para ello, se implementó un conjunto de funcionalidades que ofrecen un medio donde se facilita el acceso a la documentación por parte de los involucrados y se propicia un mayor control sobre el expediente de CI en las áreas de la facultad. El sistema además, proporciona la gestión de usuarios y el establecimiento de roles para garantizar acceso apropiado y seguridad a cada una de las funcionalidades, así como un registro de las trazas de los usuarios que requiera el sistema.

2.2.1. Técnicas para la recopilación de la Información

Para identificar las necesidades del cliente se utilizaron las técnicas: entrevista y tormenta de ideas, con el objetivo de obtener un acercamiento al problema en el menor tiempo posible. Para ello se planificaron varias sesiones de trabajo con el cliente logrando paulatinamente un mayor acercamiento a la solución requerida. Entre las nuevas funcionalidades a incluir en la solución se identificaron:

Componente Ambiente de control:

1. Gestión de los cuadros nombrados en la Facultad.
2. Gestión de las reservas de los cuadros nombrados en la Facultad.
3. Gestión de los planes de superación de los cuadros y sus reservas.

4. Control de firma del código de ética de los cuadros (con alerta de nuevos nombramientos que deben realizar la firma del código de ética).
5. Gestión de código de ética específico por actividad.
6. Gestión de medidas disciplinarias por áreas.
7. Gestión y seguimiento del plan anual de preparación de cuadros (actividades, fechas, responsables y cumplimiento).
8. Gestión de comité de expertos.
9. Gestión de actas de reuniones de comité de expertos.
10. Gestión de perfiles de competencia y funciones por cargos específicos.
11. Gestión del Plan Anual de Capacitación.
12. Gestión de procesos de la Facultad (fichas de procesos).
13. Gestión de procedimientos de la Facultad (manual de procedimientos).
14. Políticas y prácticas en la gestión de los recursos humanos
15. Gestión de procesos de la Facultad (fichas de procesos).
16. Gestión del Plan de Salud de los trabajadores (por áreas).

Componente Gestión y prevención de riesgos:

17. Gestión de riesgos (por cada área).
18. Gestión de objetivos de control (por cada área).
19. Gestión de actas de análisis de riesgos y objetivos de control (por cada área).
20. Gestionar Plan de Prevención de Riesgos (por cada área).
21. Controlar cumplimiento de las acciones del Plan de Prevención de Riesgos.

Componente Actividades de control:

22. Gestión de relaciones de familiaridad (especificando si se afecta la contrapartida).
23. Gestión de planes de acciones para dar solución a las relaciones de familiaridad cuando se afecta la contrapartida.
24. Gestión de firmas autorizadas por operaciones y áreas.
25. Gestionar Programa de Ahorro Energético.
26. Gestión de incidencias en la jornada laboral.
27. Gestión de nómina de trabajadores.
28. Gestión de Plan de inspecciones a la seguridad informática.
29. Gestión de vulnerabilidades en la seguridad informática.
30. Registro de resultados obtenidos en inspecciones a la seguridad informática.

Componente Información y comunicación:

31. Gestión de datos de la información relevante (canales, emisor, receptor, frecuencia, formato, almacenamiento, soporte, clasificación, períodos de conservación, responsables – por cada área).
32. Generar flujo de información (por cada área).
33. Gestión de Cronograma de rendiciones de cuentas de los cuadros.
34. Gestión de rendiciones de cuentas de los cuadros.

Componente Supervisión y monitoreo:

35. Gestión de Plan de supervisión al CI (AFT, útiles, RRHH, entre otros).
36. Gestión de miembros del Grupo de Prevención y Control (GPC).
37. Registro de actas de reuniones del GPC.

Para la documentación de las necesidades del cliente, se emplearon las historias de usuario, siguiendo el escenario 4 de AUP-UCI.

2.2.2. Historias de usuario (HU)

Las HU son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las HU permiten responder rápidamente a los requisitos cambiantes (Cohn 2004). A continuación se muestra la descripción de uno de los requisitos funcionales mediante una historia de usuario.

HU-Gestionar Ficha de Reserva.

Número: 12	**Nombre:** Adicionar plan de preparación y superación de la reserva
Programador: José Carlos Arencibia Pérez	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado**: 5 días
Riesgo en Desarrollo: alta	**Tiempo real:**
Descripción: el usuario presiona el botón adicionar en el listar planes de preparación y superación de la reserva, luego despliega un componente y presiona el botón adicionar en el listado de las acciones del mismo, se muestra una formulario con los	

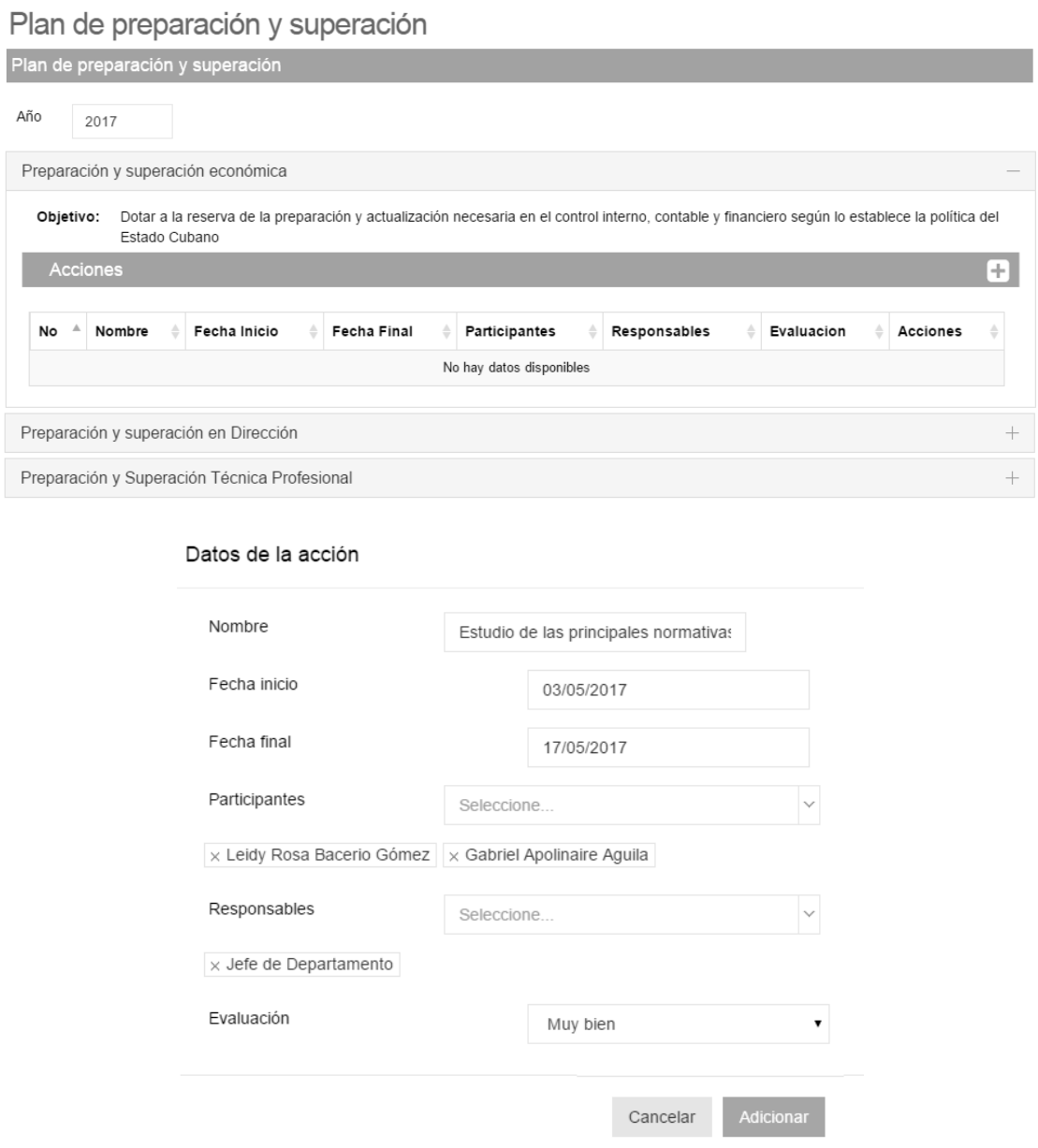
campos a llenar, se introduce los datos correspondientes, finalmente presiona el botón aceptar para guardar los cambios.
Observaciones: no debe de existir campos vacíos, los campos deben de tener el formato establecido. Debe de existir al menos un componente en el nomenclador.
Prototipo de interfaz: 

Tabla 8.Historias de usuario- Adicionar plan de preparación y superación de la reserva

Fuente: Elaboración propia

A continuación se muestran los requisitos no funcionales, propiedades o cualidades que el sistema debe cumplir.

2.2.3. Requisitos no funcionales del sistema

Los Requisitos No Funcionales son difíciles de verificar y testear, y por ello son evaluados subjetivamente (Dorfman, y otros, 1990).

Usabilidad

Capacidad para ser entendido:

- El tiempo de entrenamiento requerido para que los usuarios potenciales sean productivo operando el sistema es de 30 días.
- El sistema notifica a los usuarios los errores y sugiere cómo corregirlos.
- El sistema valida automáticamente la información contenida en los formularios de ingreso. En el proceso de validación de la información, se tienen en cuenta aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo, manejo de tipos de datos, el sistema no permitirá la entrada de datos incorrectos.

Capacidad para ser aprendido:

- Los botones siempre aparecen del lado derecho de la interfaz.
- La forma de llenar los formularios es de arriba hacia abajo.
- Los menús aparecen en el orden en que se desarrolla el proceso.

Capacidad de atracción:

- Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para que los usuarios puedan utilizar el sistema.

Capacidad de operabilidad:

- El sistema ofrece una interfaz fácil de operar para el cliente

Funcionalidad

Seguridad:

- Todo uso de las funcionalidades del sistema requiere la autenticación de los usuarios.
- El sistema concederá acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema.

Eficiencia

Utilización de recursos:

- El sistema permite imprimir los diferentes documentos que genere la aplicación como respuesta a las funcionalidades.

Servidor de aplicación:

- Procesador: 3.00 GHZ
- RAM: 4GB
- Disco duro: 1TB
- Tarjeta de Red: 1

Servidor de base de datos:

- RAM: 2GB
- Disco duro: 256 GB

PC_Cliente:

- Procesador: 2.0 GHZ
- RAM: 1 GB
- Tarjeta de Red: 1

Portabilidad

Adaptabilidad:

- El sistema puede ser utiizado desde diferentes entornos.

Facilidad de instalación:

- El sistema podrá ser instalado en el ambiente especificado en los requisitos tecnológicos para servidores.

1.2.7. Diseño de la solución

En el presente epígrafe se especifican los patrones del diseño aplicados a la solución, el diagrama de clases del diseño y el modelo de datos obtenido. El diseño que se realiza debe buscar ante todo satisfacer los requerimientos que inciden en la arquitectura definida para el Sistema Integral de Control Interno.

Arquitectura del sistema

Para el desarrollo del sistema se utiliza el marco de trabajo Symfony siguiendo el patrón MVC. La figura 2 muestra cómo funciona esta arquitectura ejemplificada en la solución.

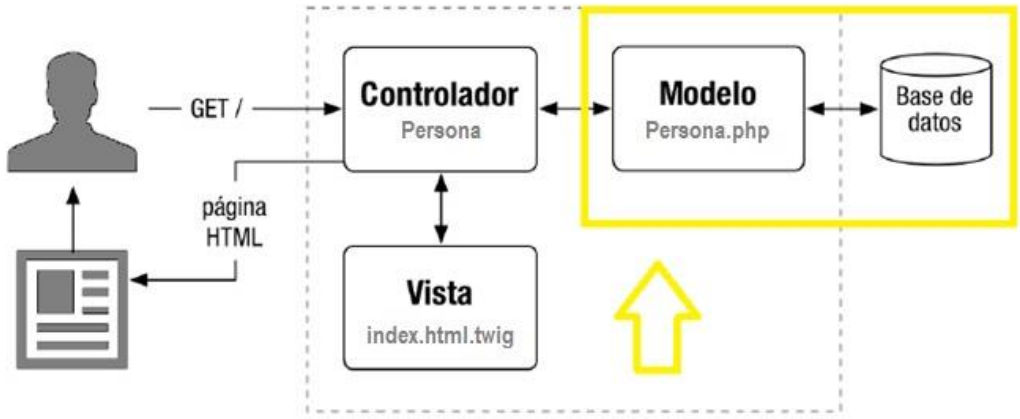


Figura 2. Arquitectura interna en el marco de trabajo symfony 2

Fuente: (Elaboración propia)

Cuando el usuario solicita ver la portada del sitio, internamente sucede lo siguiente:

- El sistema de enrutamiento determina qué *Controlador* está asociado con la página de la portada.
- Symfony2 ejecuta el *Controlador* asociado a la portada. Un *controlador* no es más que una clase PHP en la que se puede ejecutar todo el código que se necesite.
- El *Controlador* solicita al *Modelo* los datos. El *modelo* no es más que una clase PHP especializada en obtener información, normalmente de una base de datos.
- Con los datos devueltos por el *Modelo*, el *Controlador* solicita a la *Vista* que cree una página mediante una plantilla y que inserte los datos del *Modelo*. La Vista es la página HTML que utilizan los usuarios para interactuar con la aplicación.
- El *Controlador* entrega al servidor la página creada por la Vista. (Aguiluz, 2013).

Patrones de diseño

A continuación se muestran evidencias de la aplicación de los patrones de diseño en la solución desarrollada.

Patrones GRASP:

En la figura 3 se evidencia la utilización de los patrones GRASP, como es el caso del experto en la clase *entidad Competencia* y *CompetenciaRepository*, las mismas contienen toda la información y la

lógica del negocio que comprenden a ese proceso. Los patrones controlador y creador se reflejan en la clase controladoras CompetenciaController ya que es la encargada de atender todas las peticiones de las vistas y pasar los datos a las clases del modelo, así como la responsabilidad de identificar la creación de los nuevos objetos por las clases que contienen la información necesaria para realizarla. En todas las clases de la lógica del negocio se muestra una alta cohesión y un bajo acoplamiento ya que cada una de ellas posee el trabajo de realizar las responsabilidades que solo les competen sin asociaciones con la vista, favoreciendo que el nivel de dependencia sea bajo.

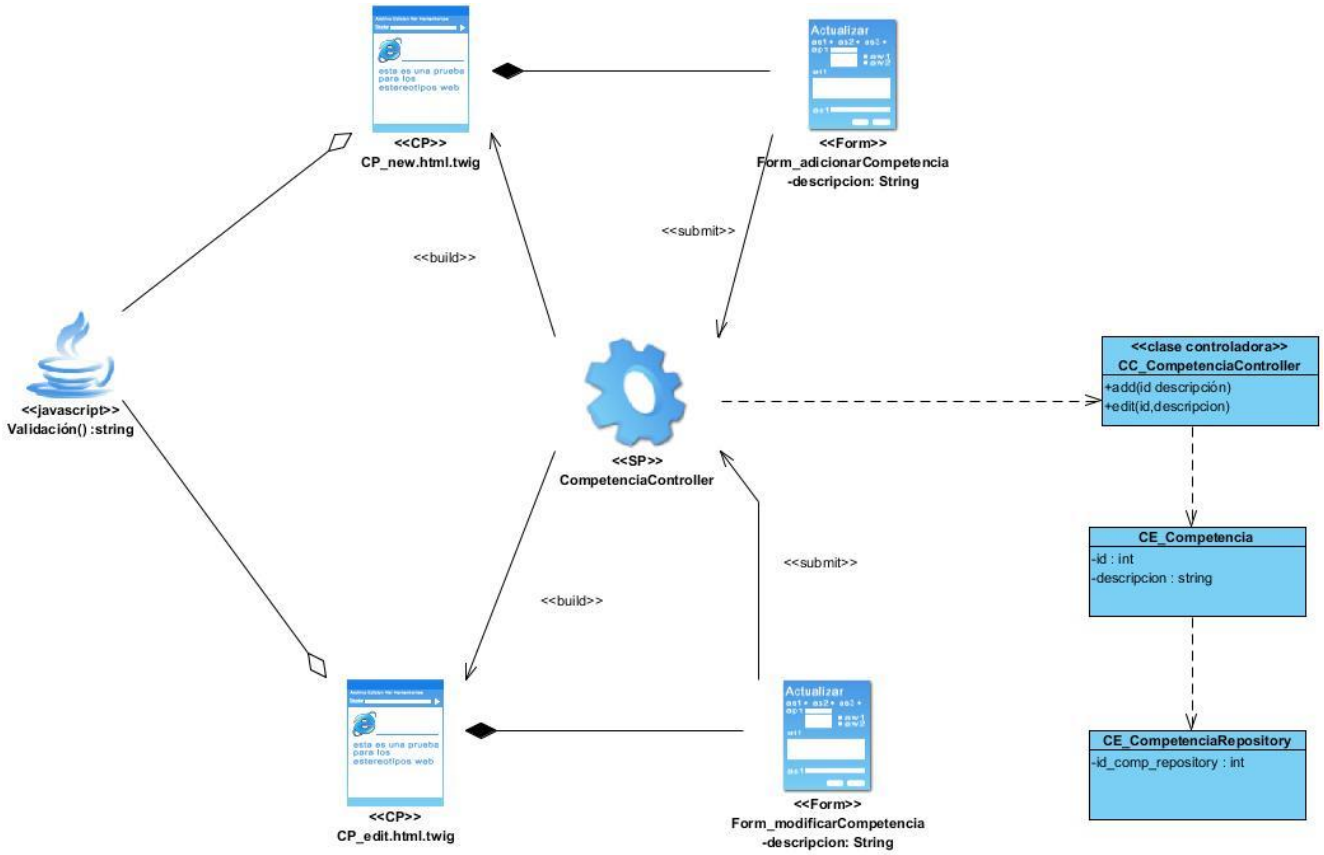


Figura 3. Diagrama de clases del diseño con estereotipos web “Perfiles de competencia”

Fuente: (Elaboración propia)

Patrones GoF:

El marco de trabajo Symfony emplea patrones GoF en las capas de Modelo y Controlador del patrón arquitectónico MVC. A continuación se ejemplifican los patrones presentes en la implementación de la solución:

- Patrón Singleton

Clase sfRouting – método getInstance Esta clase la utiliza el controlador frontal (sfWebFrontController) y se encarga de enrutar todas las peticiones que se hagan a la aplicación. El singleton sfRouting precisa otros métodos muy útiles para la gestión manual de las rutas: ClearRoutes (), hasRoutes (), getRoutesByName () (Potencier, y otros, 2009).

- Patrón Command

Este patrón se observa en la clase sfWebFrontController, en el método dispatch (). Esta clase está por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Este patrón se aplica además en la clase sfRouting, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el framework, la cual se puede activar o desactivar. En este método es parseada la URL con el objetivo de precisar los parámetros de la misma y de esta forma saber el Actions que debe responder a la petición (Potencier, y otros, 2009).

- Patrón Registry

Este patrón es muy útil para los desarrolladores en la Programación Orientada a Objetos. Este patrón es un medio sencillo y eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales. Este patrón se aplica en la clase sfConfig, que es la encargada de acumular todas las variables de uso global en el sistema. Symfony aplica además el patrón “Front Controller” (Controlador frontal), por lo que posee una estructura bien organizada de controladores, que comienza desde el “index.php” del ambiente y termina en los “Actions”. Cada clase de esta capa tiene su responsabilidad y es única, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, y otros que se encargan de identificar mediante algunos datos las clases que deben realizar determinadas tareas (Patrón GoF Command, clase sfRouting) y las clases relacionadas con la configuración del sistema (sfConfig, y sfConfigHandler) (Potencier, y otros, 2009).

- Patrón Decorador

Se observa en la clase abstracta *View* de *Symfony* que es utilizada en la creación de plantillas *html.twig*. Posibilita la creación dinámica de plantillas a partir de una plantilla base con el código HTML común para todas las vistas de la aplicación, de la cual heredan las demás plantillas, permitiendo además redefinir código en caso de ser necesario.

En la figura 4 se muestra la utilización del patrón Decorador implementado en la vista base *template.html.twig*, y en la vista de gestionar objetivos de control *index.html.twig*, que hereda el código definido en la plantilla base.

```
<!--[if !IE]><!-->
<html lang="en" class="no-js">
{%block html%}
<!--<![endif]-->
<!-- BEGIN HEAD -->
<head>
<meta charset="utf-8"/>
<title> {% block title %} {% endblock title %} | X3US </title>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<meta content="" name="description"/>
<meta content="" name="author"/>
<!-- BEGIN GLOBAL MANDATORY STYLES -->

{%block style%}
<link href="{{asset("bundles/assets/css/xeus.css")}}" rel="stylesheet" type="text/css"/>
```

Figura 4. Patrón Decorador implementado en la vista base template.htlm.twig

Fuente: Elaboración propia

```
{% extends 'AplicacionBundle:templates:template.html.twig' %}
   {%block content%}

   {% include 'AplicacionBundle:Nomencladores:categoria/encabezado.html.twig'%}
```

Figura 5. Patrón decorator implementado en la vista index.html.twig

Fuente: Elaboración propia

Modelo de datos

Para la representación de los datos en el sistema se diseñó el modelo de datos, compuesto por 81 entidades, de ellas 17 son nomencladores y el resto están destinadas al almacenamiento de la información correspondiente al dominio del problema. En la figura 4 se muestran las entidades del modelo de datos correspondientes al componente Información y comunicaciones:

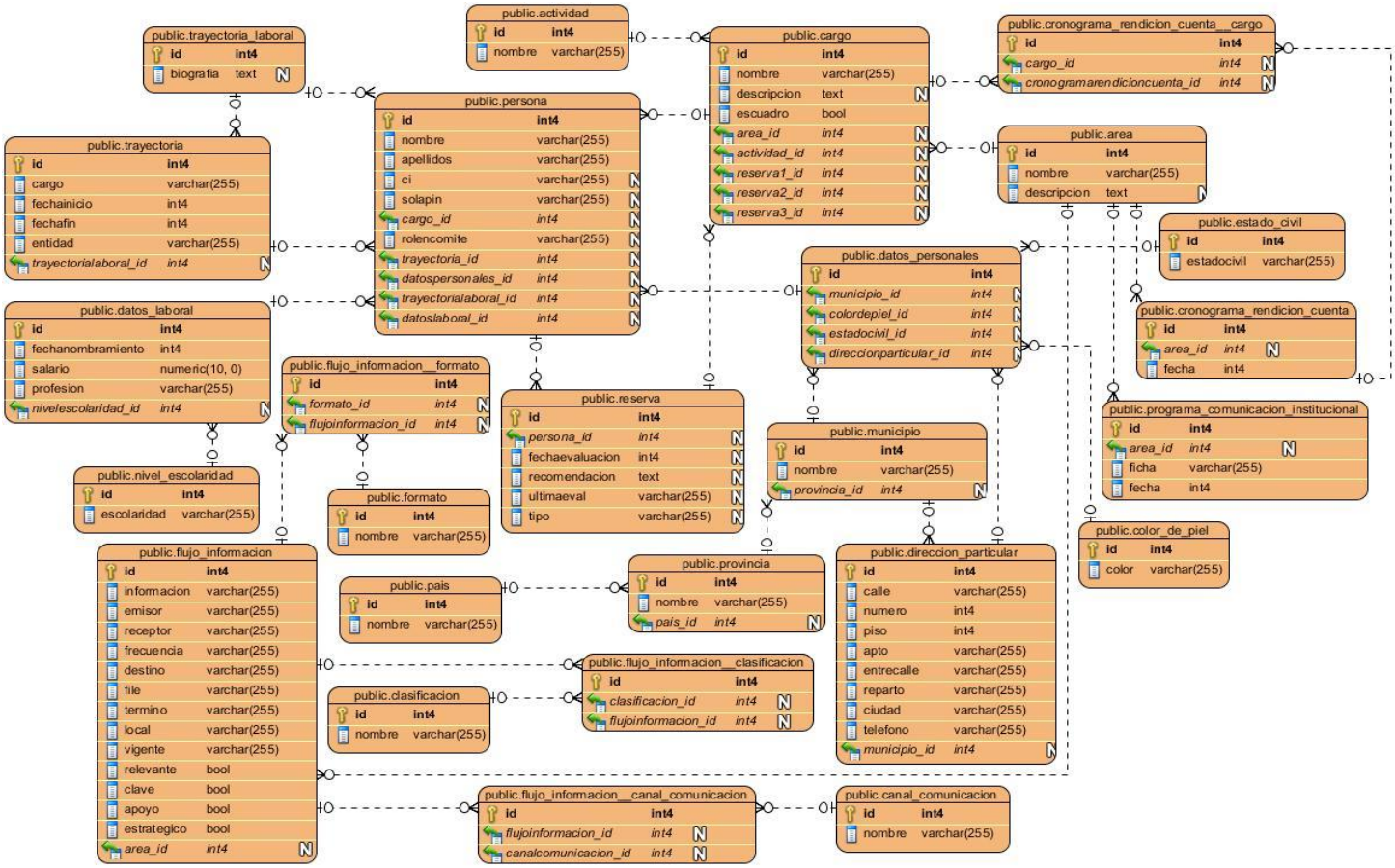


Figura 6. Modelo de datos

Fuente: (Elaboración propia)

Normalización del modelo de datos:

El modelo obtenido por el Sistema Integral de Control Interno cumple con la primera forma normal, ya que no existen campos multivaluados ni compuestos, además de tener un único identificador. También cumple con la segunda forma normal, pues los atributos que no son claves en la tabla dependen de la llave primaria. Por último, se evidencia que está en tercera forma normal al cumplir con las dos primeras y no tener dependencias transitivas los atributos no primos en esa misma relación.

Verificación del diseño

Para la evaluación de la calidad del diseño se emplearon las métricas RC y TOC aplicadas a los indicadores: Acoplamiento, Complejidad de mantenimiento, Cantidad de pruebas, Reutilización, Responsabilidad y Complejidad de implementación (ver Anexo #3).

Las gráficas que corresponden a los resultados obtenidos se presentan en las figuras 5-13.

Resultados de la aplicación de la métrica TOC:

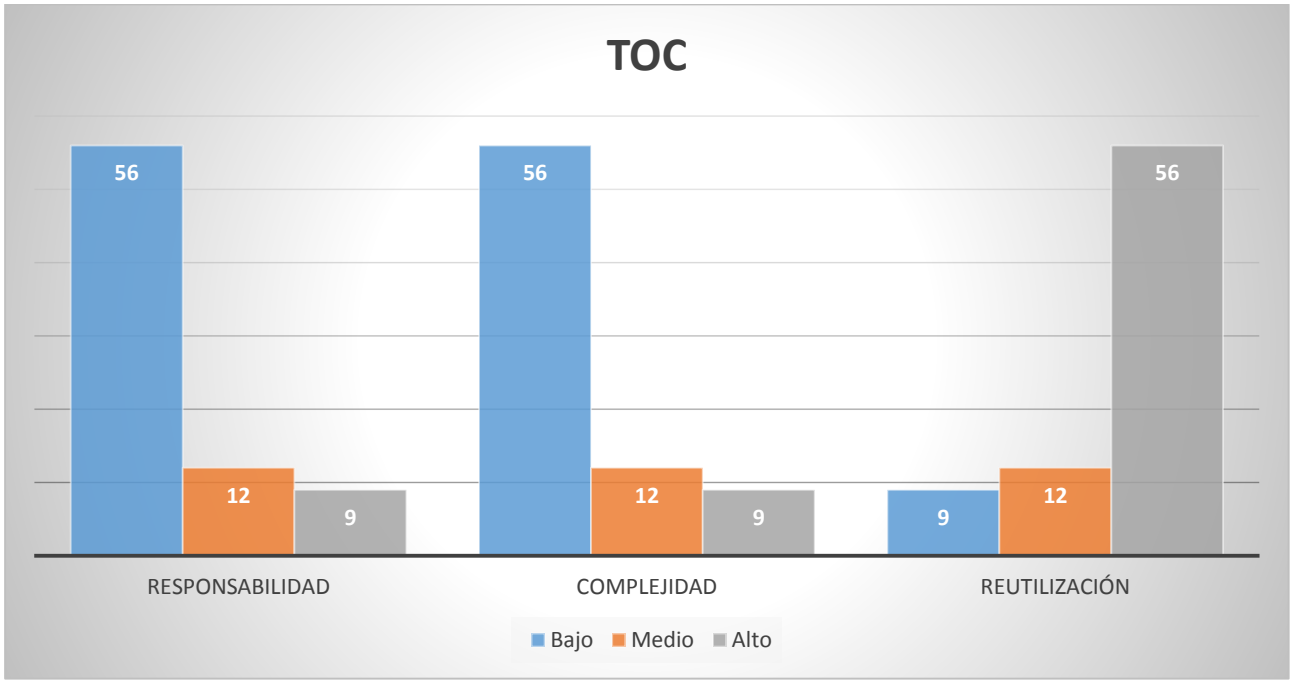


Figura 7. Representación de los resultados de la métrica TOC

Fuente: (Elaboración propia)

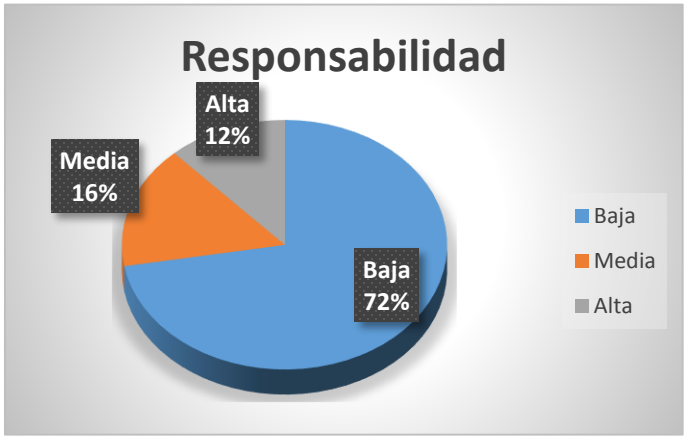


Figura 8. Representación de los resultados del atributo Responsabilidad (TOC)

Fuente: (Elaboración propia)

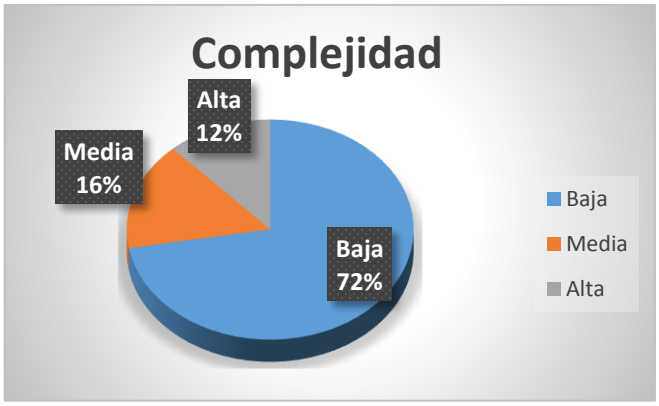


Figura 9. Representación de los resultados del atributo Complejidad (TOC)

Fuente: (Elaboración propia)

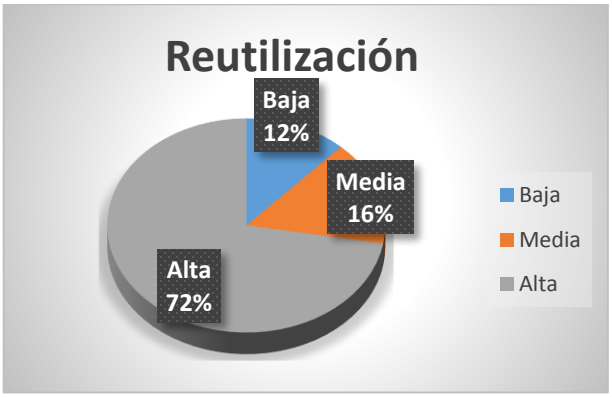


Figura 10. Representación de los resultados del atributo Reutilización (TOC)

Fuente: (Elaboración propia)

Resultados de la aplicación de la métrica RC:

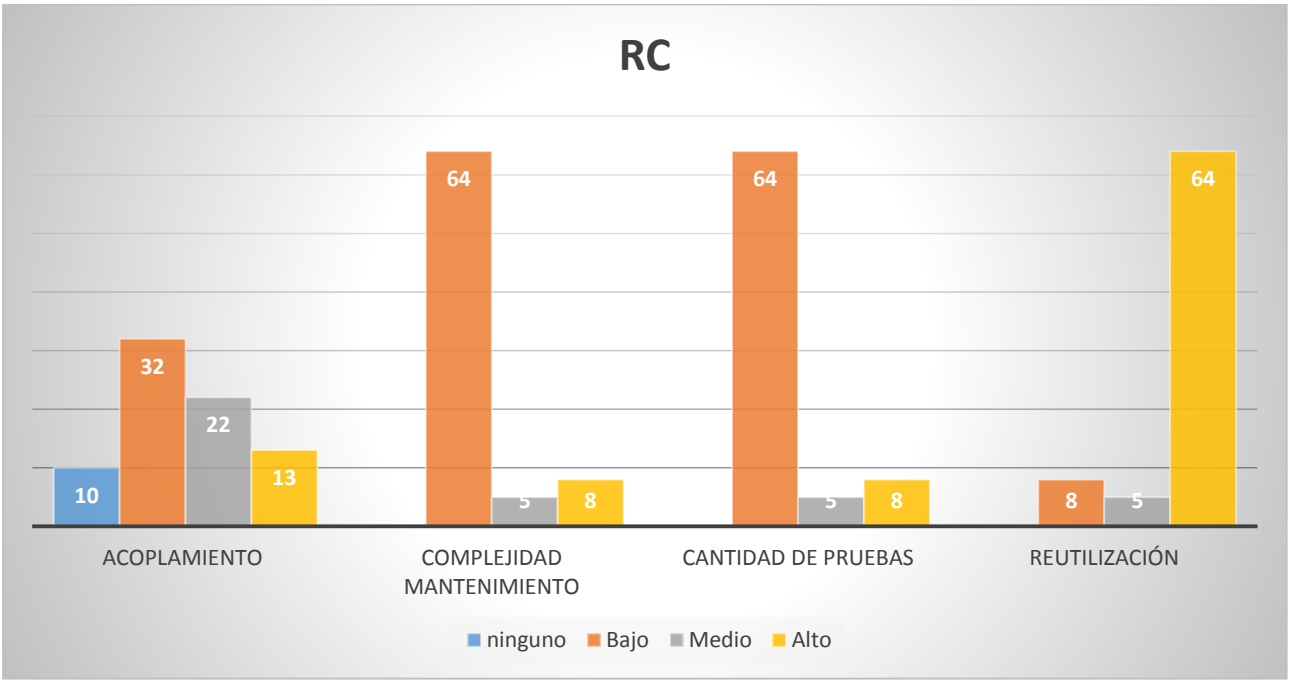


Figura 11. Representación de los resultados de la métrica RC

Fuente: (Elaboración propia)

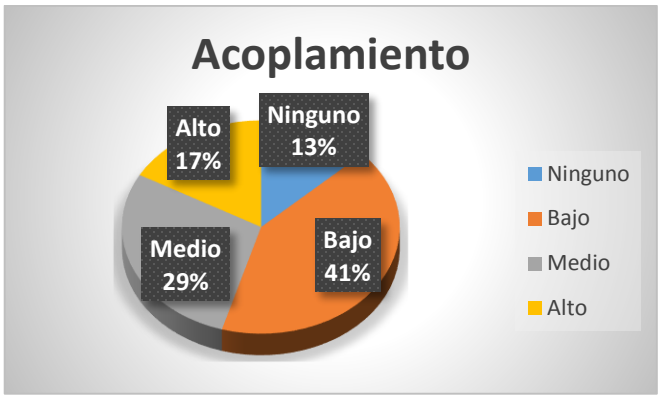


Figura 12. Representación de los resultados del atributo Acoplamiento (RC)

Fuente: (Elaboración propia)

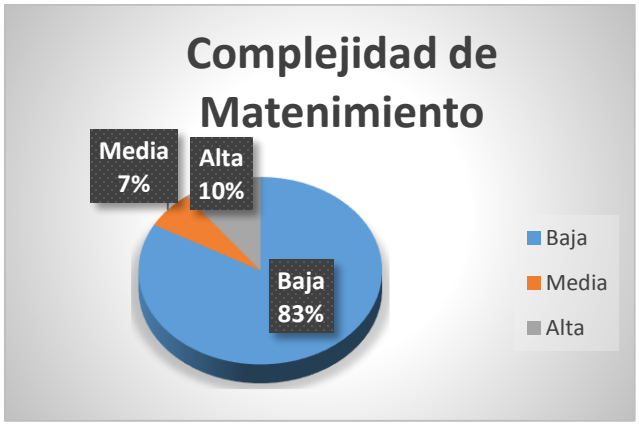


Figura 13. Representación de los resultados del atributo Complejidad de mantenimiento (RC)

Fuente: (Elaboración propia)

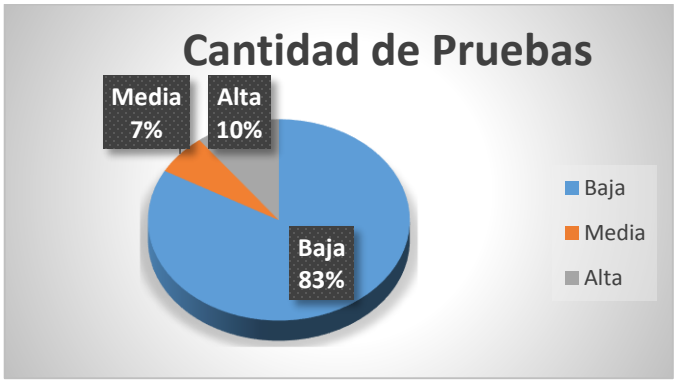


Figura 14. Representación de los resultados del atributo Cantidad de pruebas (RC)

Fuente: (Elaboración propia)

Figura 15. Representación de los resultados del atributo Reutilización (RC)

Fuente: (Elaboración propia)

Métrica	Resultados del análisis
TOC	Luego de la aplicación de la métrica los resultados obtenidos fueron satisfactorios pues más del 72% de las clases presentó baja responsabilidad y complejidad de implementación y solo el 12% presentó bajo grado de reutilización de un total de 77 clases.
RC	La aplicación de la métrica arrojo resultados favorables, el 42% de las clases presentan un bajo acoplamiento y un 84% de baja complejidad de mantenimiento, en el caso de la reutilización se alcanzó un 83 % y la cantidad de prueba arrojó un resultado de 83% para las pruebas bajas lo que conlleva a la conclusión de que la calidad del diseño escogido es Bueno.

Tabla 9. Resultado de la aplicación de las métricas del diseño.

Fuente: (Elaboración propia)

2.3 Conclusiones parciales

- La aplicación de las técnicas tormenta de ideas y entrevistas al cliente permitieron la identificación de las necesidades del cliente, las cuales fueron descritas mediante 77 HU con un total de 361 funcionalidades. Las revisiones técnicas formales y la validación de prototipos, corroboraron la correspondencia entre la descripción y el prototipado, acorde con las necesidades del cliente.
- La aplicación de los patrones GRASP y GoF contribuyeron a la obtención de un diseño con calidad, lo cual se constató en los resultados de la aplicación de las métricas TOC y RC, donde se evidenció una baja responsabilidad y complejidad, favoreciendo la reutilización, la alta cohesión y el bajo acoplamiento. Lo que se traduce en un diseño que contiene pocos errores, es adaptable a cambios y reduce el esfuerzo en la implementación de nuevas funcionalidades y la aplicación de las pruebas unitarias.

CAPÍTULO 3. IMPLEMENTACIÓN Y ANÁLISIS DE LOS RESULTADOS

3.1. Introducción del capítulo

En el capítulo se describen los resultados de la implementación de la solución, reflejados en el diagrama de componentes y en el uso de los estándares de codificación, así como el tratamiento de errores. Se muestran los resultados de las pruebas aplicadas al sistema y de la validación de la solución mediante la técnica Iadov.

3.2. Aspectos relevantes de la implementación

A continuación se muestran: el diagrama de componentes desarrollado, los estándares de codificación empleados y el tratamiento de errores.

3.2.1 Diagrama de componentes

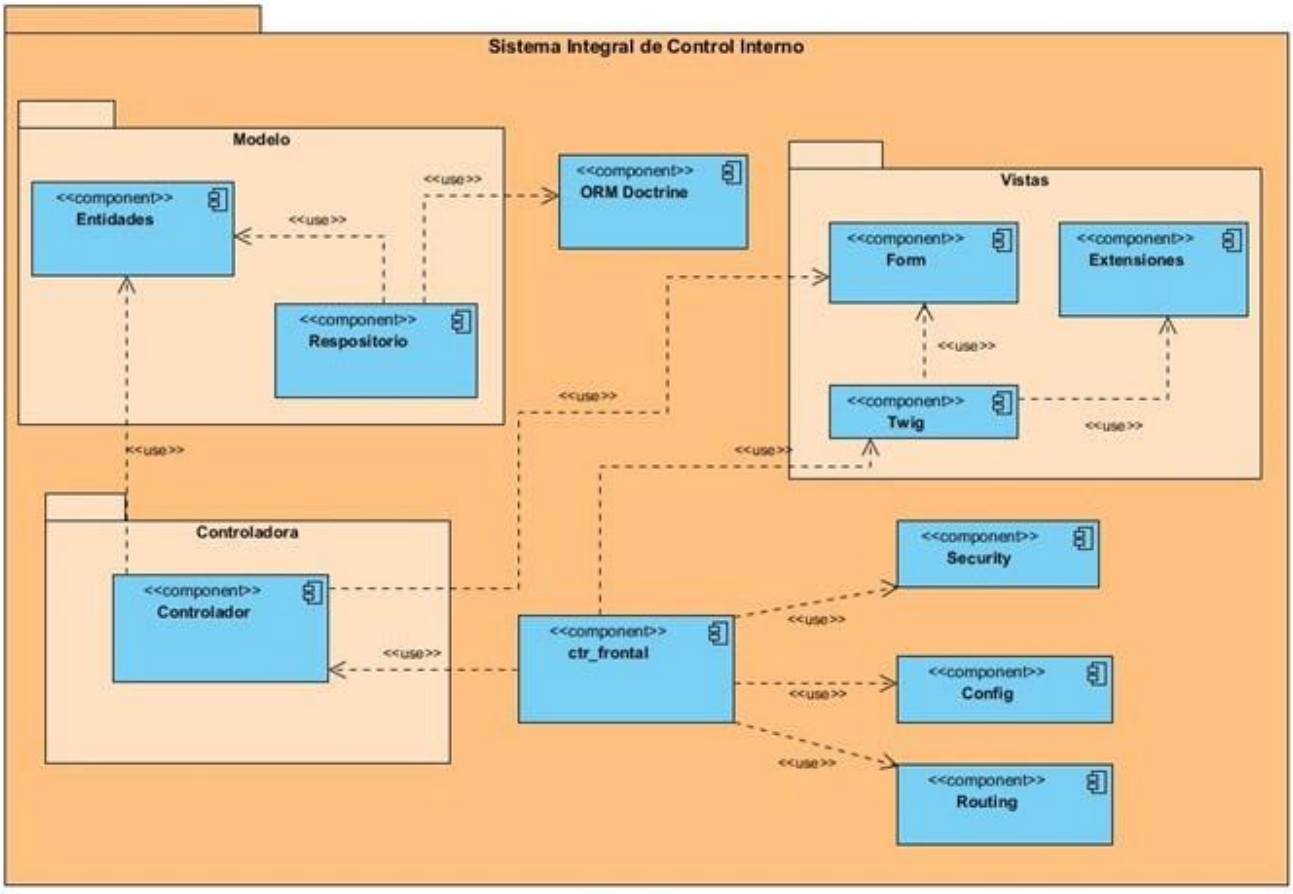


Figura 16. Diagrama de componentes del Sistema Integral de Control Interno

Fuente: (Elaboración propia)

3.2.2 Estándares de codificación

Estilos de escritura empleados en estándares de codificación *Camel Case*, específicamente la variante *Lower Camel Case:*

- El nombre de las clases controladoras terminan con la palabra Controller.
- Se exceptúan el uso de las tildes y la letra ñ, que es sustituida por nn.
- En todo momento se utilizan nombres claros, concretos y libres de ambigüedades. Ejemplo: "idpersona" y no solamente "id".
- El nombre de las funcionalidades públicas terminan con la palabra Action.
- Las variables booleanas deben tener nombres que sugieran respuestas o contenidos de tipo true/false, por ejemplo: “esGrande”.
- Las variables que poseen nombres de palabras compuestas se escriben juntas.
- Se hizo uso de los comentarios para la explicación del código siempre que fue necesario.

A continuación se brindan ejemplos de fragmentos de código donde se pondrán de manifiesto los estándares antes mencionados.


```
/**
 * codigo controller.
 *
 */
class CodigoEticaCuadroController extends Controller
{

    public function indexAction()
    {

        return $this->render('AmbienteControlBundle:codigoEticaCuadro:index.html.twig', array(

        ));

    }


    /**
     * Finds and displays a codigos entity.
     *
     */
    public function descargarCodigoEticaCuadroAction()
    {
        $em = $this->getDoctrine()->getEntityManager();

        $documento = $em->getRepository('AplicacionBundle:Documento')->findBy(array(
            'nombre' => 'CodigoEticaCuadro'
        ));
```

Figura 17. Uso de los estandares de codificación

Fuente: (Elaboración propia)

3.3. Validación de la solución

A continuación se muestran los resultados obtenidos en la aplicación de las pruebas aplicadas al sistema.

3.3.1 Método Prueba de caja negra

Para la aplicación de este método se seleccionó la técnica de Particiones de equivalencia, la cual divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de pruebas. En la tabla 10 se ejemplifica la aplicación con el caso de prueba Adicionar Datos del Riesgo.

Escenario	Descripción	Riesgo	Interno	Externo	Actividad	Frecuencia	Impacto	Nivel de detección	Respuesta del sistema	Flujo central
EC 1.1 Adicionar Datos del Riesgo	*Se adiciona un nuevo riesgo*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*El sistema muestra un mensaje indicando que debe llenar los campos*	*1. Seleccionar la opción Mapa de Riegos en el componente Gestión y Prevención de Riesgo.* *2. Seleccionar un Área.* *3. Seleccionar el botón "Adicionar"* *4.Se muestra una pantalla con los siguientes datos a llenar:* *Riesgo, interno, externo, actividad, frecuencia, impacto, nivel de detección.* *5. Una vez introducidos los valores se selecciona la opción "Adicionar"* *6. Se actualiza el listado de las Adecuaciones*
		Despilfarro de energía	*si*	*no*	*vacío*	*V*	*Alto*	*Bajo*		
		vacío	*vacío*	*vacío*	*vacío*	*vacío*	*vacío*	*vacío*		
		V	*vacío*	*vacío*	*V*	*I*	*V*	*I*		
		Fraudes o indisciplinas dentro del aula en examen	*si*	*no*	*Proceso docente educativo*	*V*	*Alto*	*Bajo*	*El sistema adiciona un nuevo riesgo*	
		V	*V*	*V*	*V*	*V*	*V*	*V*		
		Despilfarro de energía	*V*	*vacío*	*Contribución al ahorro energético*	*Moderada*	*bajo*	*Alto*		
		Despilfarro de energía	*V*	*vacío*	*Contribución al ahorro energético*	*Moderada*	*bajo*	*Alto*		

Tabla 10. Caso de Prueba Adicionar Datos de Riesgo

Fuente: (Elaboración propia).

Para comprobar la calidad del sistema se realizaron cuatro iteraciones por el Grupo de Calidad del Centro de Gobierno Electrónico (CEGEL) y en 361 funcionalidades se detectaron 152 no conformidades en la primera iteración, de las cuales 10 fueron no conformidades del sistema mientras que las 142 restantes fueron no conformidades de ortografía y diseño. Estas no conformidades fueron solucionadas y en una segunda iteración se detectaron 48 no conformidades, de las cuales una no conformidad fue del sistema y 47 de ortografía y diseño; estas también fueron corregidas y posteriormente en una tercera iteración se detectaron 3 no conformidades de ortografía que se le dieron solución. Por último, en una cuarta iteración no se detectaron no conformidades.

En la figura 16 se muestran las iteraciones realizadas y la cantidad de NC encontradas en cada iteración. En el anexo 4 se muestra el Acta de validación del producto.

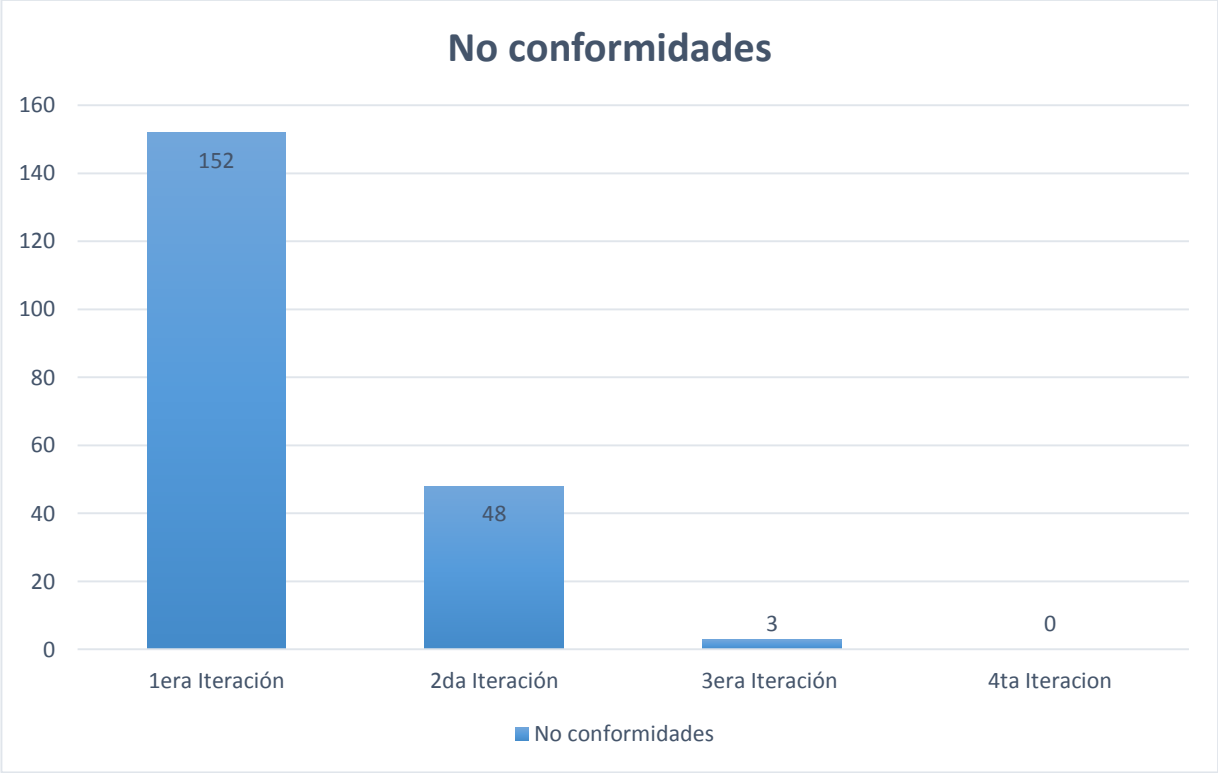


Figura 18. Resultado de la aplicación de las pruebas unitarias

Fuente: (Elaboración propia)

3.3.2 Métodos de prueba de caja blanca

A continuación se relacionan los pasos de la aplicación de la prueba del camino básico para el método newPorAreaAction () de la funcionalidad “Adicionar Medida Disciplinaria”.

Pasos en la aplicación de la prueba del camino básico:

1. Realizar la notación del grafo de flujo

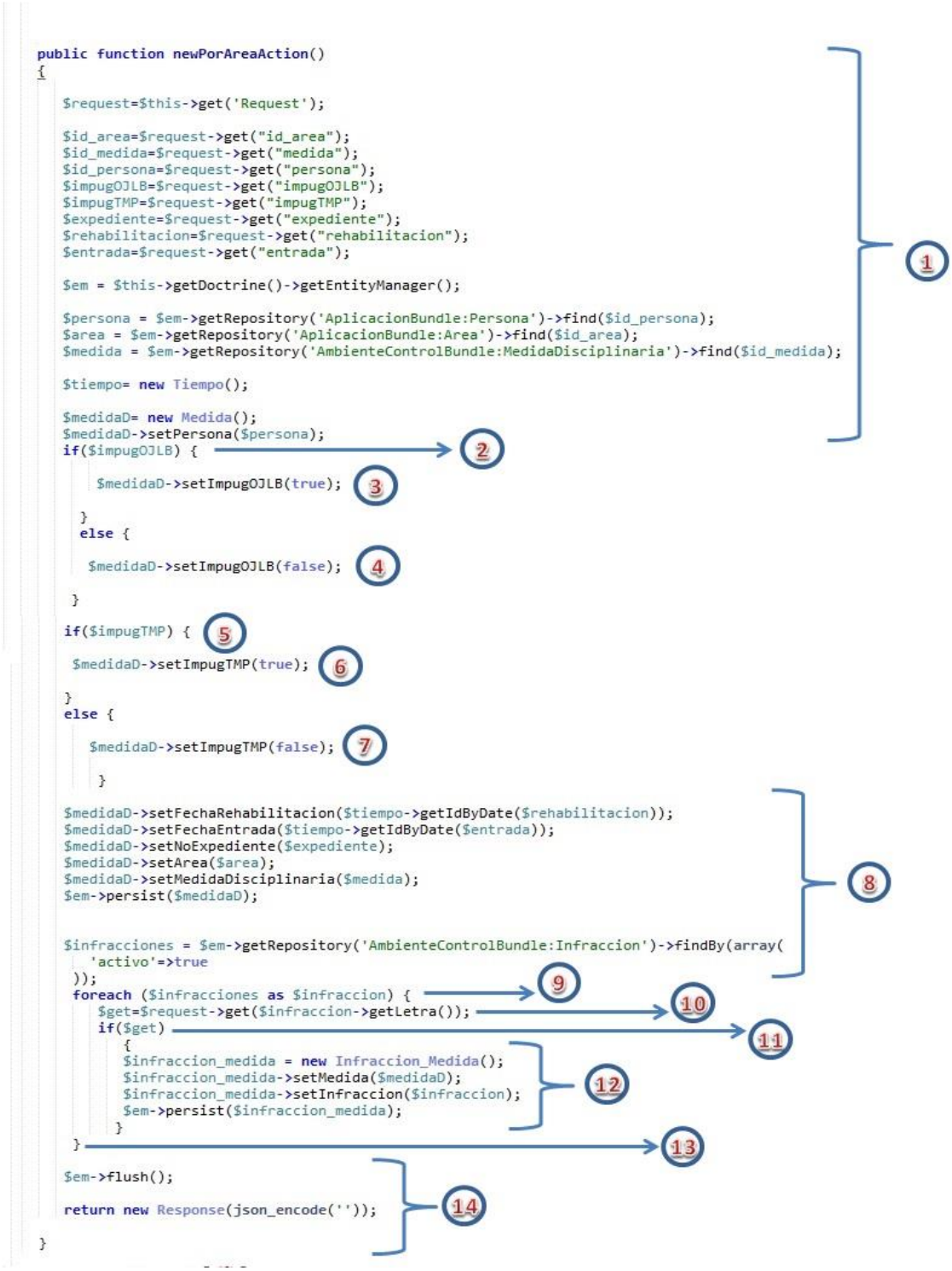


Figura 19. Código de la implementación del método newPorAreaAction ()

Fuente: (Elaboración propia)

La figura 18 presenta el grafo de flujo obtenido del código representado en la figura 17

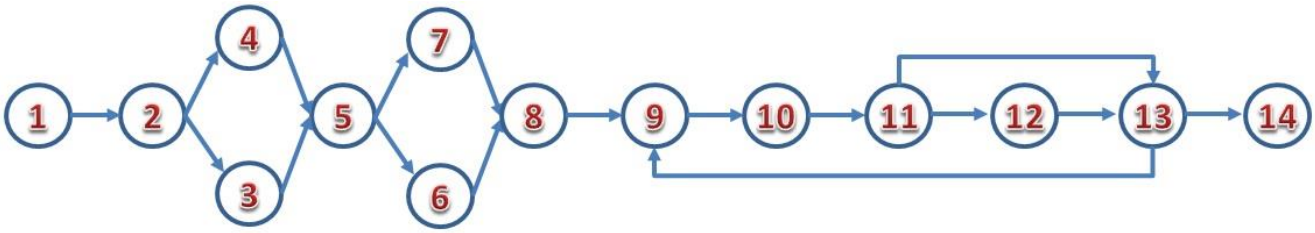


Figura 20. Grafo de flujo del código de implementación del método newPorAreaAction ()

Fuente: (Elaboración propia)

2. Determinar la complejidad ciclomática del grafo de flujo resultante.

Fórmula 1: $V(G) = (A - N) + 2$.

Resultado: $V(G) = (17 - 14) + 2$, $V(G) = 5$

Fórmula 2: $\mathrm{V}\,(\mathrm{G}) = \mathrm{P} + 1$.

Resultado: $\mathrm{V}\,(\mathrm{G}) = (4 + 1), \mathrm{V}\,(\mathrm{G}) = 5$

Fórmula 3: $V(G) = R$.

Resultado: $V(G) = 5$

A partir de los resultados obtenidos, se determina que la complejidad ciclomática es 5, que a su vez es el número mínimo de caminos independientes y de casos de prueba que se deben aplicar al código.

3. Determinar el conjunto básico de caminos linealmente independientes.
   - camino 1: 1,2,3,5,6,8,9,10,11,12,13,14
   - camino 2: 1,2,4,5,7,8,9,10,11,12,13,14
   - camino 3: 1,2,4,5,7,8,9,10,11,13,14
   - camino 4: 1,2,3,5,6,8,9,10,11,13,14
   - camino 5: 1,2,3,5,7,8,9,10,11,12,13,14

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas, realizando al menos un caso de prueba por cada camino.

4. Definir los casos de prueba para comprobar la ejecución de cada camino del conjunto básico.

En la tabla 11 se muestra el diseño del caso de prueba realizado al primer camino básico.

Caso de prueba para el camino básico 1:

Camino: 1,2,3,5,6,8,9,10,11,12,13,14	
Descripción	Los datos de entrada son correctos y cumplen con el formato indicado.
Condición de ejecución	**Campos válidos:** El campo expediente permite la entrada de un expediente y no admite caracteres especiales y es una cadena de caracteres no nula El campo persona permite la selección de un persona y es una cadena de caracteres no nula. El campo infracciones permite la entrada de un carácter que no sea ni un número ni caracteres especiales. El campo impugOJLB y impugTMP permite seleccionar si es verdadero true y si es falso false. El campo medidas permite la selección de una medida y es una cadena de caracteres no nula. El campo fecha de rehabilitación y fecha de entrada permite la selección de una fecha y es no nula.
Entrada	**Expediente:** EH13274 **Persona:** Yairon Vargas Aguila. **Infracciones:** a;b;f. **Medida disciplinaria:** amonestación pública. **Fecha de rehabilitación:** 22/05/2017 **Impug OJLB:** si **Impug TMP:** si **Fecha de entrada:** 21/04/2017
Resultados esperados	Se adiciona la nueva medida disciplinaria de un área en específico para una persona.

Tabla 11. Caso de prueba del camino básico 1

Fuente: Elaboración propia

Luego de aplicar los casos de pruebas, se pudo comprobar que el flujo del método analizado está correctamente implementado pues, según las condiciones de entrada especificadas, se obtienen los resultados esperados.

3.3.3 Validación de las variables de la investigación

En la tabla 12 se explica cómo se evalúan las variables disponibilidad y control de la información con el uso del Sistema Integral de Control Interno mediante las dimensiones de estas variables.

La **disponibilidad de la información** asegura que el acceso a los datos o a los recursos de información por personal autorizado se produzca correctamente. Es decir, la disponibilidad garantiza que los sistemas funcionan cuando se les necesita.

Se entiende por **control**, medir y corregir las actividades de subordinados para asegurarse que los eventos se ajustan a los planes como proceso de verificar si se están cumpliendo los planes o no, y si existe un progreso en el estado del CI. Constituye esencial además, el monitoreo de las acciones realizadas por las áreas.

Variables	Dimensiones	Proceso actual	Proceso con la utilización del sistema implementado
Disponibilidad de la información	Acceso a los datos	La documentación en su mayoría, se encuentra almacenada en estantes, lo que dificulta su búsqueda para la consulta de su información.	La documentación se encuentra disponible en el sistema, tanto durante la ejecución de los procesos como una vez culminado los mismos.
		La documentación en su mayoría, se encuentra en formato duro y se maneja gran cantidad de datos, lo que puede provocar desactualización y pérdida de la información.	La documentación está almacenada de forma digital en la herramienta, evitando la desactualización. La información guardada persiste en una base de datos, evitándose la pérdida de información.
		El acceso a los datos se realiza a través del personal del VDA.	El acceso a los datos puede ser realizado por los usuarios según el permiso que tengan en el sistema, en cualquier momento.

Control de la información	Supervisión	Es engorroso supervisar eficientemente a las áreas subordinadas considerando que son varias y las supervisiones se realizan manualmente.	Con el sistema desarrollado se facilita la supervisión a la información gestionada por las áreas de un modo más ágil.
		Es necesario visitar personalmente las áreas y ocupar la documentación.	La documentación puede ser consultada remotamente y realizar los controles sobre la información desde el sistema en cualquier momento.
	Monitoreo	Las áreas realizan actualizaciones sobre la documentación y la notificación de los cambios no se conoce hasta tanto no se visite el área o se notifique al VDA.	Las actualizaciones de la documentación pueden ser monitoreadas en cualquier momento.

Tabla 12. Validación de las variables de la investigación

Fuente: (Elaboración propia)

3.3.4 Aplicación de la técnica de Iadov

Para una valoración del grado de satisfacción del cliente con la solución desarrollada respecto al control y disponibilidad de la información, se aplicó la técnica Iadov que permite el estudio del grado de satisfacción del personal involucrado en un proceso objeto de análisis. Durante la valoración se contó con la participación de clientes como, VDAs, personal del VDA Facultad 3 y responsables del CI en áreas de la UCI, para un total de nueve personas. Como resultado se obtuvo:

<table>
<tr><td rowspan="3">¿Le gustaría hacer uso del Sistema Integral de Control Interno propuesto, para desarrollar los procesos de control interno en el VDEA?</td><td colspan="3">¿Considera usted oportuno continuar ejecutando los procesos asociados al control interno en el área del VDEA manualmente, a pesar del gran volumen de información que se genera?</td></tr>
<tr><td>No</td><td>No sé</td><td>Sí</td></tr>
<tr><td colspan="3">¿El Sistema Integral de Control Interno contribuye a mejorar el control y disponibilidad de la información del VDEA acorde a sus necesidades?</td></tr>
</table>

	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

Tabla 13. Cuadro lógico de Iadov

Fuente: (Elaboración propia)

$$ISG = \frac{7*1+2*0,5}{9} \quad (5)$$

De manera que el ISG = 0,89

Los resultados de la satisfacción individual según las categorías empleadas fueron los siguientes:

Nivel de satisfacción	**Cantidad**	**%**
Máxima satisfacción	7	77,78
Más satisfecho que insatisfecho	2	22,22
No definida	0	0

Tabla 14. Resultados de aplicación de la técnica Iadov

Fuente: (Elaboración propia)

Al procesar las respuestas a las encuestas en el cuadro lógico de Iadov, se obtiene un grado de satisfacción grupal de 0,89; lo cual se traduce en una clara satisfacción con el uso del Sistema Integral de Control Interno respecto a la disponibilidad y control de la información.

En el criterio respecto al control y disponibilidad de la información en el VDA a través del uso de la solución propuesta, hubo una concordancia de un 100% en que contribuye a su mejora. De igual manera el 100% de los participantes manifestó que le gustaría mucho hacer uso del Sistema Integral de Control Interno para desarrollar los procesos asociados al CI.

Las preguntas abiertas que se formularon fueron:

- ¿Qué elemento(s) usted adicionaría a la solución que se propone?
- ¿Qué valoraciones le sugiere sistema respecto al control y disponibilidad de la información asociada al VDA?

Entre las valoraciones positivas obtenidas como respuestas a las preguntas abiertas, se recopilaron criterios como los siguientes:

- El sistema garantiza a los jefes de áreas rapidez para acceder a la información.
- El sistema permite un control más eficiente de los procesos asociados al CI que se realiza a cada una de las áreas de la facultad.
- El sistema garantiza que exista un control sobre el personal que accede a la información.
- El sistema permite generar un expediente de CI adaptado a las características de cualquier área.
- El sistema permite la gestión de aspectos asociados a todos los componentes del CI.

La aplicación de la técnica de Iadov aportó información significativa respecto al grado de satisfacción del cliente. Los resultados obtenidos y los criterios emitidos validan la fortaleza de la propuesta, reflejándose una valoración muy positiva con la solución.

3.4. Conclusiones parciales

- El diagrama de componentes permitió representar la estructura de la aplicación, mientras que los estándares de codificación empleados en la implementación permitieron ganar en legibilidad, claridad y mayor entendimiento del código por parte de los desarrolladores.
- La aplicación de pruebas de caja negra permitió verificar el correcto funcionamiento del sistema, se realizaron cuatro iteraciones por el Grupo de Calidad de CEGEL, detectándose cero no conformidades en la cuarta iteración. Mientras que la aplicación de las pruebas de caja blanca corroboraron que los flujos de los métodos analizados están correctamente implementados.
- Mediante la validación de las variables de la investigación y la aplicación de la técnica Iadov, se comprobó que el sistema contribuye a elevar la disponibilidad y control de la información, evidenciándose en el caso de la técnica Iadov un índice de satisfacción grupal de 0,89 traducido en una clara satisfacción del cliente.

CONCLUSIONES GENERALES

- En el análisis de la literatura para identificar los elementos que constituyen aporte a la solución, se constató que los sistemas estudiados que se rigen por la Resolución 60/2011, presentan carencias en la gestión de los aspectos asociados a los cinco componentes del CI, lo cual constituyó motivación para el desarrollo de la solución. Para el desarrollo del sistema se consideró oportuno utilizar herramientas y tecnologías que se distinguen por ser de código abierto, garantizando la soberanía tecnológica en la que se encuentra inmersa el país.
- Las revisiones técnicas formales y la validación de prototipos corroboraron la correcta identificación de las necesidades del cliente. Esto propició el desarrollo de un diseño cuya calidad refleja baja complejidad de implementación, favoreciendo la reutilización y la correcta asignación de responsabilidades.
- La aplicación de los estándares de codificación permitió una implementación responsable y organizada facilitando la obtención del Sistema Integral de Control Interno. El correcto funcionamiento del sistema se comprobó mediante los resultados de las pruebas de caja negra y caja blanca, lográndose la liberación del producto.
- La validación de las variables de la investigación y la aplicación de la técnica Iadov permitió medir la satisfacción del cliente respecto al control y disponibilidad de la información con el Sistema Integral de Control Interno, donde los resultados obtenidos se traducen en un elevado agrado por parte del cliente con la solución.

RECOMENDACIONES

- Integrar las funcionalidades del Módulo de Control Interno del SAEF3 con el Sistema Integral de Control Interno.
- Generar el expediente de CI a partir de la información almacenada en el sistema.

REFERENCIAS BIBLIOGRÁFICAS

ANTHES, G. HTML5 leads a web revolution. Communications of the ACM, 2012, 55(7), 16-17.

BAHIT, E. El paradigma de la Programación Orientada a Objetos en PHP con el patrón arquitectónico MVC.[En línea]. In.: Bubok Publishing SL, 2011.

BIBEAULT, B. AND Y. KATS *jQuery in Action*. Edtion ed.: Dreamtech Press, 2008. ISBN 8177228870.

COBO, Á. *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*. Edtion ed.: Ediciones Díaz de Santos, 2005. ISBN 8479787066.

COHN, M. *User stories applied: For agile software development*. Edtion ed.: Addison-Wesley Professional, 2004. ISBN 0321205685.

CONTRALORÍA-GENERAL-DE-LA-REPÚBLICA. Resolución No. 60/2011. In.: Gaceta Oficial de la República de Cuba La Habana, 2011.

CUELLAR MEJÍA, G. A. *Teoría General de la Auditoría y Revisoría Fiscal*. Edtion ed., 2008.

CHAUDHARY, M. AND A. KUMAR *PhpStorm Cookbook*. Edtion ed.: Packt Publishing Ltd, 2014. ISBN 1782173889.

CHIDAMBER, S. R. AND C. F. KEMERER A metrics suite for object oriented design. IEEE Transactions on software engineering, 1994, 20(6), 476-493.

DÍAZ ESTRADA, A. AND A. B. GONZÁLEZ MORALES. Propuesta de integración de los subsistemas del Sistema Informático de Gestión de Auditoría y Control (SIGAC). Universidad de las Ciencias Informáticas, 2009.

DUNGLAS, K. *Persistence in PHP with the Doctrine ORM*. Edtion ed.: Packt Publishing Ltd, 2013. ISBN 1782164111.

EGUÍLUZ PÉREZ, J. Introducción a javascript. In., 2012.

ENRIQUEZ, O. Y. AND H. G. DEL BUSTO Mapeo Objeto/Relacional (ORM). Revista Telem@ tica. Vol, 2011, 10(3), 1-7.

ESTUPIÑAN, R. Control Interno y Fraudes, 450p, Editorial Ecoe Ediciones Ltda., 2ª. Edición, Bogotá–Colombia, 2006.

FIGUEROA, R. G., C. J. SOLÍS AND A. A. CABRERA Metodologías tradicionales vs. Metodologías ágiles. Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.(En línea), Disponible en: http://adonisnet. files. wordpress. com/2008/06/articulo-metodologia-de-sw-formato. doc, 2008.

FIRDAUS, T. *Responsive Web Design by Example Beginner's Guide*. Edtion ed.: Packt Publishing Ltd, 2013. ISBN 1849695431.

FONSECA, O. Sistemas de control interno para organizaciones. Lima: IICO, 2011.

GAMMA, E. *Design patterns: elements of reusable object-oriented software*. Edtion ed.: Pearson Education India, 1995. ISBN 8131700070.

GAUCHAT, J. D. *El gran libro de HTML5, CSS3 y Javascript*. Edtion ed.: Marcombo, 2012. ISBN 8426717829.

GONZÁLEZ, Y. D. AND Y. F. ROMERO Patrón Modelo-Vista-Controlador. Revista Telem@ tica, 2012, 11(1), 47-57.

GRADY, R. B. AND D. L. CASWELL Software metrics: establishing a company-wide program 1987.

GUERRERO, C. A., J. M. SUÁREZ AND L. E. GUTIÉRREZ Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. Información tecnológica, 2013, 24(3), 103-114.

HALL, P. *The bootstrap and Edgeworth expansion*. Edtion ed.: Springer Science & Business Media, 2013. ISBN 146124384X.

ISO. 8402: 1994. In *Gestión de la calidad y aseguramiento de la calidad. Vocabulario.* 1994.

JACOBSON, I., G. BOOCH AND J. RUMBAUGH El proceso unificado de desarrollo de software. Madrid, Editorial Addison Wesley, Traducción: Salvador Snánchez y Otros, 2000.

KABIR, M. J. *Servidor Apache 2*. Edtion ed., 2003. ISBN 8441514682.

KEITH, M. AND M. SCHNICARIOL Object-relational mapping. Pro JPA 2, 2010, 69-106.

KEW, N. *Desarrollo de módulos y aplicaciones con Apache*. Edtion ed., 2008. ISBN 8441523282.

KUZMINA, N. Metódicas investigativas de la actividad pedagógica. Moscú, Rusia: Editorial Leningrado, 1970.

LARMAN, C. *UML y Patrones. Segunda Edición*. Edtion ed.: Pearson Educación. SA Madrid, 2003. ISBN 84-205-348-2.

LARMAN, C. GRASP: Más patrones para asignar responsabilidades. L. Hernández MONTENEGRO, RODRÍGUEZ Y SALAZAR: Uso de patrones de diseño de software, 2004, 59.

LARRAMENDI VALDES, D. AND J. C. JEREZ CAMPS. Automatizacion de la gestion del control interno en el sector empresarial (ETECSA). 2011.

LOCKHART, T. PostgreSQL Tutorial. 1998.

LORENZ, M. AND J. KIDD *Object-oriented software metrics: a practical guide*. Edtion ed.: Prentice-Hall, Inc., 1994. ISBN 013179292X.

MA, L. A. A., R. A. Q. GARCÍA, M. J. FAJARDO AND D. E. M. MEDINA La supervisión, su impacto en la rentabilidad financiera de las PyMes. Sector Manufacturero. ISSN 1931-0285 CD ISSN 1941-9589 ONLINE, 2012, 870.

MICROSOFT. Revisiones de código y estándares de codificación - MSDN - Microsoft. In.: MSDN Microsoft, 2015, vol. 2017.

NARVÁEZ COELLO, J. I. Guía de las mejores prácticas administrativas, seguridad y alta disponibilidad, caso de estudio: PostgreSQL. Pontificia Universidad Católica del Ecuador, 2014.

OSMANI, A. *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*. Edtion ed.: " O'Reilly Media, Inc.", 2012. ISBN 1449334873.

PARADIGM, V. Visual paradigm for uml. Visual Paradigm for UML-UML tool for software application development, 2013, 72.

PGADMIN. pgAdmin - PostgreSQL Tools. In., 2016, vol. 2016.

POTENCIER, F. AND F. ZANINOTTO Symfony, la guía definitiva. Symfony, la guia definitiva.[En línea][Citado el: 20 de noviembre de 2015.] http://librosweb. es/libro/symfony_1_4, 2008.

PRESSMAN, R. S. Software Engineering: A Practitioner's Approach, 7/e, RS Pressman & Associates. Inc., McGraw-Hill, ISBN, 2010, 73375977.

RAMOS MARTÍN, M. J., A. RAMOS MARTÍN AND F. MONTERO RODRÍGUEZ *Sistemas gestores de bases de datos*. Edtion ed.: McGrawHill, 2006. 457 p. ISBN 84-481-4879-7.

RODRÍGUEZ SALA, J. J., L. SANTAMARÍA ARANA, A. REBASA DOLADO AND O. MARTÍNEZ BONASTRE *Introducción a la programación, teoría y práctica*. Edtion ed. Alicante: Editorial Club Universitario, 2003. ISBN 84-8454-274-2.

RODRÍGUEZ SÁNCHEZ, T. Metodología de desarrollo para la Actividad productiva de la UCI. In. La Habana. Cuba: Universidad de las Ciencias Informáticas, 2015.

SÁNCHEZ GONZÁLEZ, I. J., Y. PÉREZ RIVERA AND A. M. GARCIA RODRÍGUEZ Módulo Control Interno para el Sistema de Administración y Economía de la Facultad 3. Serie Científica de la Universidad de las Ciencias Informáticas, 2016, 9(6), 91-109.

SCHMULLER, J. AND A. D. G. MARÍN *Aprendiendo UML en 24 horas*. Edtion ed.: Pearson educación, 2000. ISBN 968444463X.

SIERRA, M. Trabajando con Visual Paradigm for UML. In.: Online, 2008.

SOMMERVILLE, I. *Software engineering*. Edtion ed.: Pearson, 2011. ISBN 0137053460.

TRUJILLO GONZÁLEZ, N. Propuesta de un procedimiento para la Administración de los Riesgos Empresariales en Desoft SA Validación en la División Desoft Villa Clara. Universidad Central" Marta Abreu" de Las Villas, 2010.

UCI. Sistema de Gestión del Control Interno (SIGCI). In. La Habana: Universidad de las Ciencias Informáticas, 2015, vol. 2017.

WELLING, L. AND L. THOMSON *Desarrollo web con PHP y MySQL*. Edtion ed., 2005. ISBN 8441518181.