Universidad de las Ciencias Informáticas

Facultad 3

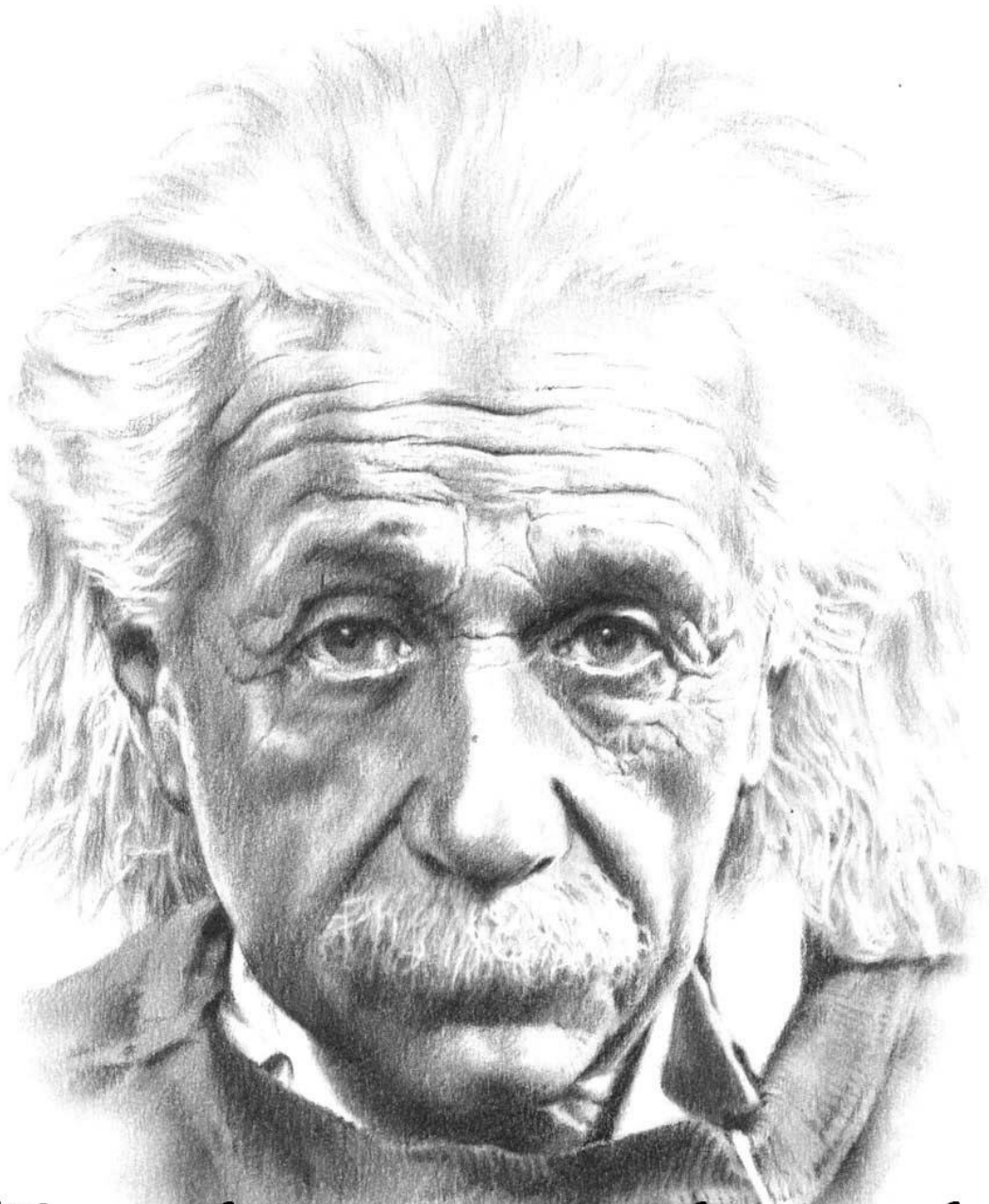
Sistema informático para la gestión de los procesos del PCC

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

Autores: Liset Gómez Chávez
Manuel González Flores

Tutores: Msc. Marieta Peña Abreu
Msc. Cealys Alvarez Trujillo

La Habana, Cuba
Junio 2015



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad".

Albert Einstein

Declaración de autoría

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

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

Autores:

_____	_____
Liset Gómez Chávez	Manuel González Flores

Tutores:

_____	_____
Msc. Marieta Peña Abreu	Msc. Cealys Alvarez Trujillo

Agradecimientos

A la persona que me ha enseñado que todo en la vida es posible, mami gracias por siempre estar cuando te necesité, por darme ánimo cuando pensé que no podía, por apoyarme en todas mis decisiones, por ser para mí un ejemplo de mujer luchadora.

A mi papa por ser el mejor padre del mundo, por ser la persona más maravillosa y especial, por quererme tanto, por siempre cuidar de mí y por haber formado la mujer que soy hoy.

A mi abuela Teresita y a mi abuelo Luis por todos sus sacrificios, por haber siempre cuidado de mí, por su amor y dedicación.

A mi hermano Yosvany por siempre estar a mi lado en todo momento, por apoyarme y por quererme tanto, gracias por todo Tata.

A mi madrasta Maday por ayudarme, por sus consejos, comprensión y por quererme como a una hija.

A mi compañero de tesis, por su comprensión, paciencia y apoyo, gracias Manu.

A Isabel, hermana de mi compañero de tesis, por sus consejos y dedicación así como a su familia.

A mis amigas y hermanas Anita y Leidy por sus consejos, su comprensión, su cariño y apoyo incondicional durante estos 5 años.

A Liannet, Marieta y Dayani por ser mis mejores amigas de la universidad, por siempre estar cuando más la necesite, por sus consejos y por todos los momentos compartidos en la universidad.

A Rubén por apoyarme en todo momento, por sus consejos, ayuda y dedicación desde el día que comencé lo que hoy he logrado terminar, mi sueño de graduarme, gracias por todo.

A Marieta y Carlos por su cariño, por ser tan especiales y por haber sido como unos padres para mí.

A mis amigos Liliana, Yadira, Michel, Alexey, Yariel, Ubesnel, los Alejandro, Orlando les agradezco por su ayuda y por sacarme una sonrisa cuando más lo necesité.

A Denia por su apoyo incondicional, por su cariño sincero, por su preocupación constante, por sus consejos de mujer, madre y amiga, gracias por formar parte hoy de mi sueño.

A mis tutoras Cealys y Marieta por su dedicación, y por habernos guiado durante todo este proceso.

A mis amigos del 3501 y a los que formaron parte un día del 3101.

Liset Gómez Chávez

Agradecimientos

A mi familia en especial a mis padres por el apoyo incondicional brindado durante estos 5 años de carrera.

A mi hermana por ser siempre un ejemplo de superación profesional constante, por su apoyo diario desde mi primer día en la universidad y por todos los consejos que me ha dado para ser un mejor estudiante y persona.

A Adrián (novio de mi hermana) y toda su familia por las atenciones que han tenido conmigo y brindarme su apoyo y cariño.

A mis compañeros de clase tanto los que hoy están con nosotros así como otros que por un motivo u otro no se encuentran presentes, muchas gracias a todos por su apoyo y amistad.

A mi compañera de tesis por la confianza depositada en mí desde los primeros días de tesis y compartir conmigo largas horas de trabajo para el día de hoy formarnos como ingenieros en ciencias informáticas.

A todas las amistades que logré hacer en la universidad desde el primer día que llegamos a la UCI que siempre me brindaron su ayuda sin pedir nada a cambio y por soportarme todos estos años. Especial saludo para mis amistades de la cancha, el piquete del Call of Dutty y al equipo de desarrollo del edificio 8 tercer paso de escalera José Javier Alemán, Alejandro Saborí y Roberto Jesús Rielo.

A mis compañeros de Guantánamo José Javier Figueras y Juan José Hernández por ser fieles amigos y siempre estar disponibles cuando necesité de su ayuda.

A mis compañeros de departamento que he tenido desde el primer año Miguel Ángel, José René, Regino, El Chambón, Pablito, Edmanuel y especialmente para los Alejandros que sin su ayuda no hubiese logrado estar aquí hoy.

A mis tutoras que fueron de gran ayuda en el desarrollo de la tesis y brindarme siempre su apoyo y atención incondicional.

A la UCI por permitirme formarme como profesional y darme la oportunidad de hacer grandes amistades durante los 5 años de carrera.

Muchas gracias a todos.

Manuel González Flores

Dedicatoria

A mis padres y abuelos por haber hecho de mí la mujer que soy, por sus sacrificios y porque sin ellos no hubiese logrado cumplir hoy este sueño.

A mi sobrino Hector Enmanuel por ser mi bebé bello y toda mi inspiración.

Liset Gómez Chávez

A toda mi familia en especial a mis padres y mi hermana por el amor y cariño que me han brindado.

Manuel González Flores

Resumen

La Universidad de las Ciencias Informáticas está estructurada por varias facultades. Cada una de estas, con el objetivo de mantener un correcto funcionamiento interno cuenta con la dirección del Partido Comunista de Cuba. Específicamente en la Facultad 3 se lleva a cabo un conjunto de procesos que son importantes para su funcionamiento, sin embargo en la ejecución de estos se han identificado una serie de dificultades en la gestión de la información, fundamentalmente para mantener la actualización, homologación y unificación de la información en cada uno de los niveles donde se utiliza. Es por ello que el presente trabajo de diploma tiene como objetivo realizar un sistema informático para la gestión de la información de los procesos del Partido Comunista de Cuba que contribuya a un mejor control y organización de los procesos de esta organización.

Para la construcción del sistema se realiza un estudio de la metodología a utilizar para guiar el proceso de desarrollo de software, así como de las principales herramientas y tecnologías utilizadas en la implementación de la solución. Además se describe la arquitectura que da soporte al sistema desarrollado. El diseño, la implementación y la solución propuesta fueron validados a través de métricas de diseño, pruebas de caja blanca, caja negra y aceptación. Con este trabajo se contribuye a mejorar la gestión de la información de los procesos del Partido en la Facultad 3.

Palabras clave: Sistema de gestión de la información, Partido Comunista de Cuba

Índice de contenido

Introducción 1

Capítulo 1: Fundamentación teórica 5

1.1 Introducción 5

1.2 Marco conceptual 5

1.3 Sistemas informáticos de gestión de información similares estudiados 6

1.3.1 Internacionales 6

1.3.2 Nacionales 6

1.4 Metodología de desarrollo 9

1.5 Plataforma de desarrollo 13

1.5.1 Lenguaje de modelado 13

1.5.2 Lenguaje de programación web 13

1.5.3 Marco de trabajo 14

1.5.4 Herramientas de desarrollo 15

1.5.5 Sistema gestor de base de datos 16

1.6 Patrones para el desarrollo de software 16

1.6.1 Patrones de diseño 16

1.6.2 Patrones de arquitectura 18

1.7 Validación del diseño 18

1.7.1 Tamaño Operacional de Clases 18

1.7.2 Responsabilidad entre Clases 19

1.8 Pruebas de software 20

1.8.1 Pruebas unitarias 20

1.8.2 Pruebas de caja negra 21

1.8.3 Pruebas de aceptación 21

1.9 Conclusiones parciales 21

Capítulo 2: Análisis y diseño de la solución 23

2.1 Introducción 23
2.2 Modelo de dominio 23
2.3 Roles relacionados con el sistema 24
2.4 Fase de planificación 24
2.4.1 Historias de usuario 25
2.4.2 Requisitos de software 27
2.4.2.1 Requisitos funcionales 27
2.4.2.2 Requisitos no funcionales 30
2.4.2.3 Validación de requisitos 31
2.4.3 Estimación del esfuerzo por Historia de usuario 33
2.4.4 Plan de iteraciones 34
2.4.5 Plan de entrega 34
2.5 Fase de diseño 35
2.5.1 Tarjetas CRC 35
2.5.2 Modelo de datos 36
2.5.3 Patrones de diseño utilizados 38
2.5.4 Arquitectura del sistema 41
2.5.5 Métricas para la validación del diseño 43
2.6 Conclusiones parciales 45
Capítulo 3: Implementación y prueba de la solución 46
3.1 Introducción 46
3.2 Fase de desarrollo 46
3.2.1 Tareas de ingeniería por iteraciones 46
3.2.2 Tareas de ingeniería detalladas 47
3.2.3 Estructura física de la solución 47
3.2.4 Estándares de codificación 48
3.2.5 Descripción de la seguridad del sistema 49

3.3 Fase de pruebas 50
3.3.1 Pruebas unitarias 50
3.3.2 Pruebas de caja negra 53
3.3.3 Pruebas de aceptación 54
3.3.4 Validación de la investigación 56
3.4 Conclusiones parciales 60
Conclusiones generales 61
Recomendaciones 62
Bibliografía 63
Anexos 68

Índice de tablas

Tabla 1. Resultados de los sistemas informáticos 7
Tabla 2. Funcionalidades que aportan a la solución 8
Tabla 3. Comparación entre metodologías ágiles y tradicionales 10
Tabla 4. Comparación entre las metodologías ágiles 11
Tabla 5. Fases y artefactos de XP 12
Tabla 6. Rango de valores para la métrica TOC 19
Tabla 7: Rango de valores para la métrica RC 20
Tabla 8. Roles 24
Tabla 9. Historia de usuario: Gestionar militante 25
Tabla 10. Historia de Usuario: Gestionar artículo 26
Tabla 11. Requisitos funcionales 27
Tabla 12. Requisitos no funcionales 31
Tabla 13. Representación de la matriz de trazabilidad 32
Tabla 14. Estimación de esfuerzos por HU 33
Tabla 15. Plan de duración de las iteraciones 34
Tabla 16. Plan de entrega de las iteraciones 35
Tabla 17. Tarjeta CRC: ArticuloController 35
Tabla 18. Tarjeta CRC: MilitanteController 36
Tabla 19. Tareas de ingeniería para la iteración 1 46
Tabla 20. Tarea de ingeniería detallada 4 47
Tabla 21. Tarea de ingeniería detallada 5 47
Tabla 22. Caso de prueba para el camino básico #4 53
Tabla 23. No conformidades detectadas 54
Tabla 24. Descripción del caso de prueba de aceptación Adicionar militante 54
Tabla 25. Descripción del caso de prueba de aceptación Modificar militante 55
Tabla 26. No conformidades detectadas 56
Tabla 27. Expertos involucrados en la validación de la encuesta 57

Índice de figuras

Figura 1. Modelo de dominio 23
Figura 2. Prototipo adicionar militante y Figura 3. Prototipo adicionar artículo 33
Figura 4. Modelo de Datos 37
Figura 5. Controlador del módulo Militante 38
Figura 6. Construcción de una vista 39
Figura 7. Creación de objetos y formularios 40
Figura 8. Arquitectura del sistema. Fuente: Elaboración propia 42
Figura 9. Componentes utilizados por terceros 42
Figura 10. Gráfica de la métrica Tamaño Operacional de Clases 44
Figura 11. Gráfica de la métrica Relaciones entre clases 44
Figura 12. Estructura física de la solución 48
Figura 13. Código del algoritmo ModificarReunionAdmision 51
Figura 14. Grafo de flujo asociado al algoritmo 52

Introducción

El surgimiento y desarrollo de las ciencias informáticas ha intervenido directamente en los procesos organizacionales, generando nuevos paradigmas y rompiendo nuevos existentes hasta el momento. El desarrollo acelerado de esta ciencia conlleva a cambios importantes en la forma en que se llevan a cabo las tareas rutinarias en el ámbito social y dentro de una organización (Padilla, 2008).

En Cuba existen varias organizaciones políticas y de masas entre la que se destaca el Partido Comunista de Cuba (PCC en lo adelante). Dicha organización es la vanguardia organizada de la nación cubana y la fuerza dirigente superior de la sociedad y el estado, que establece y orienta los esfuerzos comunes hacia los altos fines de la construcción del socialismo y el avance hacia la sociedad comunista. Entre sus objetivos fundamentales está la lucha por consolidar una nueva moral en la sociedad cubana, cimentada en la ideología de la Revolución, la solidaridad, la igualdad y la justicia social (Congreso, 2015).

Con el objetivo de mantener un correcto funcionamiento interno, la Universidad de las Ciencias Informáticas (UCI en lo adelante) cuenta con la dirección del PCC en cada una de sus facultades. Este juega un papel protagónico por tener la responsabilidad de forjar en sus militantes una ideología revolucionaria, además de persuadir en cada uno la capacidad de escuchar y de incorporar a la mayoría a la lucha por los objetivos de la Revolución. El PCC en la Facultad 3 está estructurado por un comité primario y dos núcleos, denominados La especialidad y CEIGE[1]-CEGEL[2].

Esta organización lleva a cabo varios procesos importantes para su funcionamiento interno como: el crecimiento del PCC, publicación de artículos políticos y de discusión, gestión de la información de las actas, cotización, entre otros. Sin embargo al aplicar entrevistas (Ver Anexo 1) a dirigentes de la Facultad 3 se identificaron las siguientes dificultades.

- Las tareas de organización, planeación y control de los procesos están parcialmente informatizadas utilizando herramientas de la suite Office. Esto trae consigo que no se encuentra organizada bajo criterios definidos y se hace difícil obtener reportes de la misma.
- El proceso de crecimiento es uno de los más importantes, pero actualmente se dificulta el conocimiento del estado en que se encuentra el mismo, así como el acceso a la información que se genera en cada paso.

[1] Centro de informatización de la gestión entidades
[2] Centro de gobierno electrónico

- El proceso relacionado con la cotización presenta deficiencias debido a que el activista invierte aproximadamente 90 minutos para gestionar el pago de la cotización de todos los militantes.

- La información almacenada de los años anteriores no está centralizada y se dificulta la confección de los reportes estadísticos necesarios para la toma de decisiones al demorarse el proceso de recopilación y consulta de la misma.

- Los niveles de dirección conformados por el comité primario y los núcleos de la Facultad 3 no pueden acceder fácilmente a información específica de las actas lo cual provoca demoras en la gestión de las mismas y dificulta la entrega de la reseña que solicita el organismo superior.

- Las personas que están interesadas en ingresar en las filas del PCC no cuentan con información digital centralizada que les permita conocer de manera precisa los requisitos y normas generales para el ingreso a la organización, así como su historia, noticias e informaciones referentes a las actividades convocadas por esta, lo cual limita el impacto que puede tener la misma en el trabajo político ideológico dentro de la facultad y la UCI en general.

A partir de la problemática descrita se define como **problema a resolver**: la gestión de la información de los procesos del PCC en la Facultad 3 dificulta la actualización, homologación y unificación de la información necesaria en cada uno de los niveles donde se requiere.

La investigación se centra en el **objeto de estudio**: Sistemas de gestión de información enmarcado en el **campo de acción**: Sistemas de gestión de información de los procesos del PCC en la Facultad 3.

Para dar solución al problema planteado se traza como **objetivo general:** desarrollar un sistema informático para la gestión de la información de los procesos del PCC en la Facultad 3 que facilite la actualización, homologación y unificación en cada uno de los niveles donde se requiere.

Para dar cumplimiento al objetivo general se definen como **objetivos específicos:**

- Elaborar el marco teórico de la investigación haciendo uso de métodos empíricos y teóricos para determinar los fundamentos que serán utilizados como base para el desarrollo del sistema informático.
- Diseñar la solución para facilitar la implementación de la misma.
- Implementar la solución para la gestión de la información de los procesos del PCC.
- Validar la solución propuesta y el objetivo de la investigación.

Se propone como **idea a defender:** con el desarrollo y utilización de un sistema informático para la gestión de la información de los procesos del PCC en la Facultad 3 se facilitará la actualización, homologación y unificación de la información necesaria en cada uno de los niveles donde se requiere.

Posibles resultados:

- Aplicación web para la gestión de la información de los procesos del PCC.
- Documento de ayuda para la utilización del sistema.

Los principales **métodos teóricos** utilizados son:

- **Analítico-Sintético:** se utiliza para el análisis de la bibliografía relacionada con el tema de investigación, que permite realizar un resumen de los elementos más importantes de los procesos y Sistemas de gestión de información. Facilita la selección de las tecnologías y herramientas para el desarrollo de la propuesta de solución.
- **Histórico-Lógico:** se utiliza para realizar un estudio del estado del arte analizando sus principales funcionalidades, características y ventajas.
- **Inductivo-Deductivo:** a partir de este método se logra identificar las generalidades de los Sistemas de gestión de información a partir del estudio de un conjunto de soluciones informáticas y deducir las características particulares que debe poseer la propuesta de solución.

Los **métodos empíricos** utilizados son**:**

- **Entrevista:** se entrevista al Secretario general del PCC en la Facultad 3 Eliober Cleger Despaigne y al activista de cotización Gaspar Melchor González Font para obtener la información relacionada con los procesos del PCC.
- **Encuesta:** se aplica una encuesta a las personas más involucradas en los procesos del PCC con el objetivo de caracterizar la gestión de la información a partir de los indicadores: actualización, homologación y unificación de la información.

El presente trabajo de diploma posee la siguiente estructura:

Capítulo 1: Fundamentación teórica: en este capítulo se realiza el estudio de los Sistemas de gestión de información existentes que se considera que pueden aportar a la solución, la fundamentación de la metodología de desarrollo de software, las tecnologías y las herramientas a utilizar. Se precisan los patrones de diseño, la arquitectura del sistema, las métricas a utilizar para la validación del diseño y las pruebas a realizar al software.

Capítulo 2: Análisis y diseño de la solución: en este capítulo se realiza la descripción de la fase de planificación y diseño de acuerdo a la metodología de desarrollo de software seleccionada. Se

muestran las Historias de Usuario, artefactos generados en la fase de planificación. Se elaboran en la fase de diseño las Tarjetas CRC para guiar el proceso de implementación y además se aplican las métricas para realizar la validación del diseño.

Capítulo 3: Implementación y prueba de la solución: en este capítulo se presentan los principales aspectos de la fase de implementación. Se muestran los resultados obtenidos en la encuesta realizada a las personas más involucradas en los procesos del PCC de la Facultad 3 para validar la solución, así como los resultados de la validación del sistema a partir de las pruebas unitaria, funcionales y de aceptación del cliente.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se enuncian los conceptos relacionados con la gestión de información de los procesos del PCC en la Facultad 3 para un mejor entendimiento del presente trabajo. Se realiza un estudio de una representación de Sistemas de gestión de información existentes a nivel nacional e internacional. Se selecciona la metodología de desarrollo de software, las tecnologías y las herramientas para el desarrollo del sistema informático.

1.2 Marco conceptual

A continuación se enuncian los principales conceptos relacionados con la gestión de los procesos del PCC en la Facultad 3 para un mejor entendimiento y comprensión de la investigación.

Algunos autores como Moreiro González (1998) conceptualizan los Sistemas de gestión de información como el elemento integrador de todos los procesos vinculados al tratamiento de la información y como agente de desarrollo. Fairer-Wessels (1997) reconoce la gestión de la información que se genera en una organización mediante la aplicación de las tecnologías y técnicas con una integración sistémica para desarrollar estrategias y alcanzar las metas. Davis y Olson (1985) lo definen como un sistema integrado y automatizado para proveer la información que sostenga las funciones de operatividad, gestión y toma de decisiones en una organización.

Por tanto para el desarrollo de la investigación se asumirá el concepto de Davis y Olson ya que los definidos por los autores Moreiro y Fairer-Wessels son muy amplios en relación al alcance de la investigación realizada. El propuesto por Fairer-Wessels se ve más a un nivel estratégico mientras el propuesto por Davis y Olson está enfocado a un nivel más práctico.

Partido Comunista de Cuba: martiano y marxista-leninista, vanguardia organizada de la nación cubana, es la fuerza dirigente superior de la sociedad y el estado, que organiza y orienta los esfuerzos comunes hacia los altos fines de la construcción del socialismo y el avance hacia la sociedad comunista (Congreso, 2015).

Proceso de crecimiento: es el proceso que se le realiza a aquellas personas que aspiran ingresar al PCC. A estas se les realiza ciertas comprobaciones para analizar y determinar si cumplen con las condiciones y cualidades para ser militante (Despaigne, 2015).

Homologación: equiparar, poner en relación de igualdad dos cosas (Real Academia Española, 2015).

Unificación: hacer de muchas cosas una o un todo, uniéndolas, mezclándolas o reduciéndolas a una misma especie (Real Academia Española, 2015).

Actualización: hacer actual algo, darle actualidad (Real Academia Española, 2015).

1.3 Sistemas informáticos de gestión de información similares estudiados

En el presente epígrafe se muestra un resumen de los sistemas informáticos de gestión de información estudiados a nivel nacional e internacional con funcionalidades y características similares a las de la propuesta de solución.

1.3.1 Internacionales

Los Sistemas de gestión de información a nivel internacional estudiados se describen a continuación:

Aspel-COI 7.0: es un Sistema de gestión de información que procesa, integra y mantiene actualizada la información contable y fiscal de la empresa en forma segura y confiable. Posee funcionalidades que proporcionan la generación de diversos reportes, documentos de trabajo y gráficas que permiten soportar y evaluar el estado financiero de la organización. Mantiene interfaz con los sistemas Aspel y hojas de cálculo, lo que contribuye a lograr una eficiente administración de la empresa (Aspel, 2004).

Alfresco: es un software potente y escalable para la gestión de documentos. Es una aplicación web, fácil de usar y además permite soporte para la Gestión de Contenidos Empresariales, incluidas Gestión Documental y Gestión de Activos Digitales. Brinda funcionalidades que permiten almacenar, recuperar y compartir documentos y contenidos, editar documentos almacenados en la plataforma y publicar documentos. Este permite la gestión avanzada de usuarios, trámites y búsquedas avanzadas (Alfresco, 2015).

NUXEO DM: este sistema permite el flujo de documentos a través de los procesos de negocio, los cuales pueden ser gestionados y seguidos. Además permite visualizar documentos ofimáticos dentro del navegador web simplificando actividades rutinarias como la búsqueda rápida para un simple vistazo del contenido, sin perder tiempo en descargarlo y sin la necesidad de abrirlo en una aplicación de escritorio (como Microsoft Office) (Nuxeo, 2015).

1.3.2 Nacionales

Los Sistemas de gestión de información a nivel nacional estudiados se describen a continuación:

Análisis y diseño de una aplicación informática para la gestión de información de la UJC[3] en la UCI: en (Osorio, 2007) se propone el análisis y diseño de un sistema para gestionar la estructura

[3] Unión de Jóvenes Comunistas

organizativa de la UJC, para crear o modificar las estructuras, gestionar el expediente de cada militante y controlar la gestión de sanciones y evaluaciones para cada uno de estos.

Sistema de Gestión Fiscal (SIGEF): sistema creado para mejorar la calidad, rapidez y profesionalidad que brinda la Fiscalía General de la República en sus diferentes instancias, así como también para incrementar los niveles de gestión de la información, garantizando su adecuada organización y preservación. La tecnología concebida para el proyecto permite la réplica y el intercambio y flujo de información entre las diferentes instancias de la Fiscalía General de Republica de manera que los flujos de trabajo para el control de los procesos, los reportes estadísticos y el control jurisdiccional se realice con celeridad (CEGEL, 2014).

DataFeu: es un Sistema de gestión realizado para informatizar los principales procesos llevados a cabo por la Federación Estudiantil Universitaria en la UCI y apoyar a sus principales dirigentes en el proceso de toma de decisiones. Ofrece a los usuarios un listado con cada una de las actividades que se realizan en la universidad durante los años cursados y la búsqueda de las mismas a nivel de facultad, universidad, provincial o de carácter nacional (Gonzalez, 2014).

Resultados del estudio de los sistemas nacionales e internacionales:

Los sistemas estudiados resuelven un problema específico pero no se ajustan a las necesidades, funcionalidades y características de la investigación. Sin embargo pueden aportar elementos a la propuesta de solución de la investigación, pues se pretende realizar un Sistema de gestión de información para informatizar los procesos del PCC específicamente en la Facultad 3. En la Tabla 1 se muestran los indicadores por los cuáles se realizó el análisis de los sistemas seleccionados a partir de las necesidades del cliente y el estudio de otros desarrollos de investigación con objetivos similares. En la Tabla 2 se describen funcionalidades que presentan estos sistemas que se tuvieron en cuenta en la solución.

Tabla 1. Resultados de los sistemas informáticos

Sistemas estudiados / **Indicadores**	**Aspel-COI 7.0**	**Alfresco**	**NUXEO DM**	**SIGEF**	**DataFeu**
Sistema operativo	Windows	Windows, Unix Solaris, Linux	Windows, MAC OS, Solaris, Linux	Linux	Linux

Sistema gestor de base de datos	Firebird	Oracle, PostgreSQL, MySQL	PostgreSQL, Oracle, MS SQL Server y MySQL	PostgreSQL	MySQL
Libre de costo	x	x	x	✓	✓

X: No cumple

✓: Si cumple

Tabla 2. Funcionalidades que aportan a la solución

Nombre	**Funcionalidades**	**Aplicación en la solución**
Aspel-COI 7.0	- Generar reporte	La creación de reportes estadísticos a partir de la información registrada en las actas.
Alfresco	**Gestión avanzada de usuarios:** -Creación y búsqueda de usuarios -Creación y asignación de roles -Creación y asignación de permisos a usuarios **Gestión avanzada de trámites:** -Cargar documento	Gestionar usuarios y asignación de roles para establecer privilegios y niveles de acceso al sistema por los usuarios. Cargar documento para adjuntar todo tipo de documentos y almacenarlos en el sistema.
NUXEO DM	-Visualizar documento -Buscar documento	Para visualizar los documentos en el navegador sin necesidad de descargarlos. Buscar documento para tener acceso a los documentos registrados.

Análisis y diseño de una aplicación informática para la gestión de información de la UJC en la UCI.	- Crear, modificar y eliminar los usuarios - Autenticar usuario - Realizar Búsqueda - Realizar Crecimiento	Para crear, modificar y eliminar un militante, autenticarse para acceder al sistema y crear un crecimiento.
SIGEF	- Reportes - Búsquedas	Para realizar búsquedas de la información almacenada en el sistema.
DataFeu	- Mis dirigentes - Listado de actividades	Conocer la estructura de la organización y las actividades convocadas y organizadas por esta.

Con el estudio realizado teniendo en cuenta la estructura, diseño y funcionalidades principales de estos sistemas se pudo determinar las funcionalidades y características que son de utilidad para resolver el problema planteado en la investigación. Se analizaron las tecnologías que se utilizaron para su implementación obteniendo una visión general de cómo desarrollar el diseño de la propuesta de solución y la implementación del flujo de eventos o pasos a realizar en cada uno de los módulos y funcionalidades. Para ello se seleccionó el SIGEF como guía para la implementación de las funcionalidades: Reportes, Exportar documentos, Adjuntar documento así como el diseño del sistema.

1.4 Metodología de desarrollo

Según Mario Piattini (Piattini, 2007) una metodología de desarrollo de software es “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software”. Una metodología es la que define Quién debe hacer Qué, Cuándo y Cómo (Piattini, 2007).

La selección de la metodología apropiada, es un factor determinante para obtener un sistema con calidad. No existe una metodología que pueda ser aplicada en todos los proyectos y su selección depende de las características de cada uno de estos. Las metodologías de desarrollo se pueden enmarcar en dos grupos, las metodologías tradicionales y las metodologías ágiles.

Metodologías tradicionales: centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto. Presentan altos costes al implementar un cambio y tienen una falta de flexibilidad en proyectos donde el entorno es volátil. Las metodologías tradicionales se focalizan en la planificación, control del proyecto, especificación de requisitos y en el modelado (INTECO, 2009).Son eficaces y necesarias para proyectos de gran envergadura que requieren de

mucho tiempo y recursos, donde la organización es de vital importancia para su desarrollo. Algunas de las metodologías robustas más conocidas son: Rational Unified Process (RUP), Microsoft Solution Framework (MSF), entre otras.

Metodologías ágiles: son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan según se describe en (INTECO, 2009). Algunas de las metodologías ágiles son: SCRUM, Extreme Programming (XP), Crystal Clear, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), entre otras.

A continuación en la Tabla 3 se muestra una comparación entre estas dos clasificaciones de metodologías desarrollada por (INTECO, 2009).

Tabla 3. Comparación entre metodologías ágiles y tradicionales

Criterios	Metodologías ágiles	Metodologías tradicionales
Cambios en los requisitos	Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Interacción con el cliente	El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Cantidad de integrantes	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Artefactos generados	Pocos artefactos.	Muchos artefactos.

Se opta por usar una metodología ágil, ya que el equipo de desarrollo es pequeño, la investigación está dirigida a satisfacer al cliente mediante la entrega continua de software funcional, manteniendo una constante comunicación entre el cliente y el equipo de desarrollo. A continuación se describen tres de las metodologías de desarrollo de software más referenciadas en la literatura.

Scrum: es una metodología en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente en equipo, y obtener el mejor resultado posible de un proyecto. En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita tener resultados prontos, donde los requisitos son cambiantes y pocos definidos y la innovación, la competitividad, la flexibilidad y la productividad son fundamentales (Scrum, 2015).

Programación Extrema (XP): es una metodología ágil para pequeños y medianos equipos, desarrollando software cuando los requerimientos son ambiguos o rápidamente cambiantes. A diferencia de los procesos tradicionales para desarrollar software, XP asume el cambio como algo natural, y que en alguna etapa de un proyecto sucede. En XP se realiza el software que el cliente solicita y necesita, en el momento que lo precisa, alentando a los programadores a responder a los requerimientos cambiantes que plantea el cliente en cualquier momento. Esto es posible porque está diseñado para adaptarse en forma inmediata a los cambios, con bajos costos asociados, en cualquier etapa del ciclo de vida (Calabria, y otros, 2003).

Crystal Clear: es una metodología ágil pensada para aplicarse en grupos pequeños de 6 a 8 desarrolladores ubicados en el mismo sitio basándose en la comunicación osmótica entre los miembros del equipo de desarrollo. Ofrece poca resistencia a los cambios por lo que suele utilizarse cuando los requerimientos son ambiguos o rápidamente cambiantes. Se centra en la eficiencia y en las personas, no siendo así en los procesos y artefactos. Presenta como características la entrega frecuente de últimas versiones del software (INTECO, 2009).

A continuación en la Tabla 4 se presenta una tabla comparativa entre estas dos metodologías desarrollada por (Mendes Calo, y otros, 2009).

Tabla 4. Comparación entre las metodologías ágiles

Criterios	XP	Scrum	Crystal Clear
Fecha de entrega.	Las iteraciones de entrega son de una a tres semanas.	Las iteraciones de entrega son de dos a cuatro semanas.	Las iteraciones de entrega son diarias, semanal o mensual.

Interacción del equipo sobre el proceso.	Los miembros de XP trabajan en dúos.	Los miembros de SCRUM trabajan de forma independiente.	Los miembros de Crystal trabajan de forma individual utilizando la Comunicación Osmótica.
Respuesta ante los cambios.	Permite introducir cambios en la iteración en curso.	Permite la evolución y el cambio, pero no es recomendable en la iteración en curso.	Permite introducir cambios en la iteración en curso.
Seguimiento de un plan.	Define un plan detallado para cada iteración, que puede ser modificado.	Define un plan detallado de iteraciones, no acepta cambios durante una iteración.	Define un plan detallado para cada iteración, que puede ser modificado.
Desarrollo de software que funcione.	Generar entregable con prueba satisfactoria e integrada con el resto de las funciones al finalizar cada iteración.	Generar entregable con prueba satisfactoria al finalizar cada iteración.	Generar entregable con prueba satisfactoria e integrada con el resto de las funciones al finalizar cada iteración.

Debido a las características del equipo de trabajo, compuesto por dos desarrolladores, en constante colaboración con el cliente, admitiendo cambios de última hora en las iteraciones del sistema, con el objetivo de favorecer las entregas rápidas del software funcional en un periodo que no excede los seis meses, se opta por el uso de un enfoque de desarrollo ágil, escogiéndose la Programación Extrema como metodología de desarrollo.

XP consta de cuatro fases (Escribano, 2002), en cada una se generan los artefactos que a continuación se muestran en la Tabla 5:

Tabla 5. Fases y artefactos de XP

Fases	Artefactos
Planificación	Historias de Usuario
Diseño	Tarjetas CRC(Clase Responsabilidad Colaboración)
Desarrollo	Tareas de ingeniería por iteración y detalladas
Prueba	Casos de prueba de aceptación y de caja blanca

1.5 Plataforma de desarrollo

1.5.1 Lenguaje de modelado

Para realizar el modelado de la propuesta de solución se utilizan los lenguajes de modelado UML en su versión 2.1.

El Lenguaje Unificado de Modelado (UML): por sus siglas en inglés, es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. (Stevens, y otros, 2002).

Ventajas de UML (UML, 2015)

- Se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real.
- Es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos.

1.5.2 Lenguaje de programación web

Un lenguaje de programación es un sistema estructurado y diseñado principalmente para establecer la comunicación entre el programador y los dispositivos, ya sean hardware o software existentes. Contiene un conjunto de acciones consecutivas que el ordenador debe ejecutar (Tecnología, 2015).

Para el desarrollo de aplicaciones web, estos lenguajes tienden a clasificarse de dos formas: **del lado del servidor** los cuales son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él y **del lado del cliente** son los lenguajes que basan su procesamiento en el cliente web o se ejecutan en el navegador del usuario (Torre, 2006).

1.5.2.1 Programación del lado del servidor

PHP: es un lenguaje interpretado del lado del servidor, de propósito general, ampliamente usado y que está diseñado especialmente para desarrollo web. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Es multiplataforma con capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. (PHP, 2015).

1.5.2.2 Programación del lado del cliente

HTML[4] 5: es la quinta versión de este lenguaje. Posee un entorno de programación para aplicaciones en cualquier plataforma. Sus componentes vitales son las etiquetas, los atributos y los tipos de datos que estos admiten (HTML, 2012).

[4] **HTML:** Acrónimo en inglés de Hyper Text Markup Language.

Bootstrap 3.0

Bootstrap 2.0 es un framework diseñando para simplificar el proceso de creación de diseños web. Ofrece una serie de plantillas CSS y de ficheros JavaScript, los cuales permiten conseguir (Cochran, 2012):

- Interfaces que funcionen en los navegadores actuales y correctamente en los no tan actuales.
- Un diseño que pueda ser visualizado de forma correcta en distintos dispositivos y a distintas escalas y resoluciones.
- Una mejor integración con las librerías que se suelen usar habitualmente, como por ejemplo jQuery
- Un diseño sólido basado en herramientas actuales y potentes como LESS o estándares como CSS3/HTML5.

JavaScript: Es un lenguaje de programación multiplataforma, interpretado y orientado a objetos, lo que permite a los desarrolladores añadir y crear interactividad en el desarrollo y diseño de sitios web, así como la validación de datos. Su característica principal es que ayuda a mejorar el aspecto y la funcionalidad de una página web. La forma de emplear JavaScript en una página web es directamente dentro de las etiquetas HTML (JavaScript, 2015).

JavaScript permite la programación de pequeños scripts y de programas más grandes orientados a objetos, con funciones y estructuras de datos complejas. Además, pone a disposición del programador todos los elementos que forman la página web para acceder a ellos y modificarlos dinámicamente (JavaScript, 2015).

1.5.3 Marco de trabajo

Un marco de trabajo o framework es un diseño reutilizable del sistema completo o de alguna de sus partes y se expresa mediante un conjunto de clases abstractas y la forma de interactuar de sus instancias (Almeira, 2007). A continuación se describe el framework que se utiliza para el desarrollo de la solución.

Symfony 2.2.6: es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación web. Proporciona herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Está desarrollado con PHP[5] 5 y es compatible con los sistemas gestores de bases de datos: MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas (Unix, Linux, etc.) como en plataformas Windows (LibrosWeb, 2015). Este marco de trabajo cuenta con las siguientes características (LibrosWeb, 2015).

[5] PHP: Pre procesador de Hipertexto 5 (conocido por sus siglas en inglés Hypertext Pre-processor)

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de las buenas prácticas y patrones de diseño para la web.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.
- Usa como ORM[6] Doctrine, siendo este un mapeador de objetos escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una librería muy completa y muy configurable. Puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidades extras a las clases autogeneradas.

Las características antes mencionadas se ajustan a las necesidades de la solución propuesta. Symfony2 es un marco de trabajo sencillo que permite al equipo de desarrollo adaptarse fácilmente a este. El mismo implementa el patrón arquitectónico Modelo Vista Controlador que es seleccionado para el desarrollo de la propuesta de solución. A partir de lo anteriormente expuesto se decide utilizar Symfony2 como marco de trabajo.

1.5.4 Herramientas de desarrollo

Para el desarrollo de la propuesta de solución se utilizarán las siguientes herramientas:

Visual Paradigm 8.0: ofrece un entorno de creación de diagramas para UML. El diseño es centrado en casos de usos y enfocado al negocio (Visual Paradigm, 2010). Para la etapa de diseño del sistema se seleccionó Visual Paradigm por ser una herramienta de modelado multiplataforma que no se inclina por ninguna metodología específica. Los diagramas que se van a utilizar en la propuesta de solución son: diagramas de proceso de negocio, el diagrama entidad, el modelo de datos, diagramas de clases del diseño y el diagrama de despliegue.

NetBeans 7.3: es un entorno de desarrollo integrado. Permite a los programadores escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso (Oracle, 2015). Posee las herramientas necesarias para crear aplicaciones web y de escritorio con los lenguajes de programación Java, C, C++ e incluso lenguajes dinámicos como: PHP, JavaScript, Groovy, y Ruby. Puede ser ejecutado en múltiples plataformas como Windows, Linux y Max, además de ser fácil de usar e instalar (WIELENGA, GEERTJAN, 2015). Dadas las ventajas y características antes mencionadas que brinda

[6] ORM: Mapeo de objeto relacional (en inglés Object Relational Mapping): es una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional utilizado en el desarrollo de la aplicación

este entorno de desarrollo, se selecciona para implementar el sistema que dará solución a la problemática de la investigación.

Apache 2.0: es un servidor web HTTP[7] de código abierto para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 (The Apache Software Foundation, 2015). Es el servidor web por excelencia por su usabilidad, robustez y estabilidad, esto hace que cada vez millones de servidores reiteren su confianza en este programa (APACHE, 2015).

1.5.5 Sistema gestor de base de datos

PostgreSQL: es un sistema de gestión de bases de datos relacional orientado a objetos, el cual incluye características como herencia, restricciones, tipos de datos, reglas e integridad transaccional. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, sub-consultas y joins. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Utiliza el modelo cliente servidor y es un manejador de base de datos de código abierto liberado bajo la licencia BSD[8]. PostgreSQL está diseñado para administrar grandes volúmenes de datos (PostgreSQL, 2015).

1.6 Patrones para el desarrollo de software

Un patrón es una descripción de un problema y la solución a la que se le da un nombre y que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (Larman, 2003). En la implementación del sistema se emplearon un grupo de patrones de diseño así como un patrón arquitectónico.

1.6.1 Patrones de diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos fundamentales de una estructura común de diseño que la hace útil para la creación de un diseño orientado a objeto reutilizable. Los patrones de diseño están orientados a describir el cómo se debe construir el software de manera tal que se utilice correctamente las clases y los objetos que las componen (Gamma, 2003). Estos patrones se dividen en dos grupos: GRASP[9] y GOF[10]. La utilización de estos patrones refleja un diseño sencillo, pensado para la creación de un sistema robusto y fácil de entender.

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. El nombre se eligió para sugerir la importancia de

[7] HTTP: Protocolo de transferencia de hipertexto (en inglés Hypertext Transfer Protocol)
[8] **BSD:** Acrónimo en inglés de Berkeley Software Distribution.
[9] GRASP: Patrones de asignación de responsabilidades (En inglés General Responsibility Assignment Software Patterns)
[10] GOF: Banda de los cuatro (en inglés Gang Of Four)

captar estos principios para diseñar con éxito el software orientado a objetos (Larman, 2003). Es necesario elegir cuidadosamente las clases adecuadas y decidir cómo estas deben interactuar, por esta razón resulta de vital importancia el uso de patrones GRASP (Larman, 2003). Los patrones utilizados para la asignación de responsabilidades son (Larman, 2003):

- Experto: se le asigna la responsabilidad a la clase que tiene la información necesaria para cumplirla.
- Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.
- Alta cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta, cada elemento del diseño debe realizar una labor única dentro del sistema.
- Bajo acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento. Lo que significa que el sistema debe contener pocas dependencias entre clases (principio básico para la creación un buen diseño).
- Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema a clases.

Los patrones GOF reciben este nombre en honor a los 4 autores del libro Design. En el libro se describen 23 patrones clasificados en tres grupos por su naturaleza (Gamma, 1994):

- Creacionales: abstrae el proceso de instanciación de objetos. Su misión es permitir construir sistemas independientes de la forma de creación, composición o representación de objetos.
- Estructurales: controla como se combinan las clases u objetos en la construcción de estructuras mayores. Un patrón estructural de clases utiliza la herencia para componer interfaces o implementaciones. Un patrón estructural de objetos describe la forma en que se componen objetos para obtener nueva funcionalidad.
- De comportamiento: se relaciona con algoritmos, la forma en la que interactúan las clases u objetos y la asignación de responsabilidades entre ellos. Por su parte los patrones de comportamiento de objetos cooperan con un grupo de objetos interconectados para realizar una tarea que un solo objeto no puede realizar por sí solos.

Los patrones utilizados dentro de este grupo son (Software, 2015)

Factory Method / Método de fábrica: define una interfaz para la creación de un objeto, pero permitiendo a las subclases decidir de qué clase instanciarlo. Permite que una clase difiera la instanciación en favor de sus subclases.

Decorator/ Decorador: añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades.

El uso de estos patrones permitió asignar a cada clase la responsabilidad que le corresponde con el objetivo de obtener el menor número de relaciones y dependencias entre clases, aumentando las posibilidades de reutilización de las mismas.

1.6.2 Patrones de arquitectura

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

El patrón utilizado para la implementación del sistema es el Modelo Vista Controlador (MVC). Este patrón divide una aplicación interactiva en tres componentes. El modelo contiene la funcionalidad central de los datos y los datos, las vistas despliegan la información al usuario y los controladores manejan la entrada (Cavenago, 2007).

1.7 Validación del diseño

Realizar una validación del diseño para verificar su flexibilidad y calidad garantiza una buena base para la fase de implementación. Para realizar la validación del diseño se utilizan un conjunto de métricas de software orientadas a determinar qué características del modelo de diseño se pueden estimar para comprobar que el sistema será fácil de implementar en cuanto a organización (Fornaris, 2010). Las métricas a utilizar son Tamaño operacional de Clases y Relaciones entre Clases.

1.7.1 Tamaño Operacional de Clases

Esta métrica está determinada por el número total de operaciones encapsuladas en una clase, es decir, por la cantidad de métodos y atributos asignados a esta. Para medir el diseño según la Relación entre clases (RC) se evalúan un conjunto de atributos que se describen a continuación (Fornaris, 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 implica un aumento de la responsabilidad asignada a una clase determinada. Si existen valores grandes de TOC significa que una clase tiene demasiada responsabilidad, lo cual reduciría la reutilización de la clase y hará complicada la implementación. De forma contraria sucede si los valores de TOC son pequeños.

Complejidad de implementación: consiste en el grado de dificultad que tiene implementar un diseño de clases determinado. Un aumento del TOC implica un aumento de la complejidad de implementación de una determinada 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 implica una disminución del grado de reutilización de una determinada clase. La Tabla 6 muestra el rango de valores para la evaluación técnica de calidad relacionados con la métrica TOC.

Tabla 6. Rango de valores para la métrica TOC

Atributos	Categorías	Criterios
Responsabilidad	Baja	TOC< = Promedio
	Madia	Promedio < TOC < 2* Promedio
	Alta	TOC> 2* Promedio
Complejidad de implementación	Baja	TOC< = Promedio
	Madia	Promedio < TOC < 2* Promedio
	Alta	TOC> 2* Promedio
Reutilización	Baja	TOC> 2* Promedio
	Madia	Promedio < TOC < 2* Promedio
	Alta	TOC< = Promedio

1.7.2 Responsabilidad entre Clases

Esta métrica está dada por el número de relaciones de uso de una clase con otra. Para medir el diseño según la Relación entre Clases (RC) se evalúan un conjunto de atributos que se describen a continuación (Fornaris, 2010):

Reutilización: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

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.

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 costes y la planificación del proyecto.

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, etc.) diseñado. La Tabla 7 muestra el rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

Tabla 7: Rango de valores para la métrica RC

Atributos	Categorías	Criterios
Acoplamiento	Ninguno	RC=0
	Baja	RC=1
	Madia	RC=2
	Alta	RC>2
Complejidad de mantenimiento	Baja	RC< =Promedio
	Madia	Promedio < RC < 2* Promedio
	Alta	RC> 2* Promedio
Reutilización	Baja	RC> 2* Promedio
	Madia	Promedio < RC < 2* Promedio
	Alta	RC<= Promedio
Cantidad de pruebas	Baja	RC<= Promedio
	Madia	Promedio < RC < 2* Promedio
	Alta	RC> 2* Promedio

1.8 Pruebas de software

Las pruebas de software son un elemento crítico para la garantía de la calidad de software y una revisión final de las especificaciones del diseño y de la codificación. El objetivo fundamental de estas pruebas es medir el grado en que el software cumple con los requerimientos definidos (Pressman, 2002).

XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (Beck, 1999). Para una mejor evaluación del sistema se aplicaron además pruebas de caja negra utilizando la técnica partición de equivalencia.

1.8.1 Pruebas unitarias

Las pruebas unitarias o pruebas de caja blanca se basan en efectuar pruebas al código del sistema. Para realizar esta tarea se comprueban los caminos lógicos de la aplicación mediante casos de prueba. No se pueden realizar a todo el código de la aplicación, ya que el número de caminos lógicos puede llegar a crecer de manera exponencial lo cual imposibilita realizar casos de pruebas para todo estos caminos y muchos menos se podrían procesar todos. Por este motivo las pruebas de caja blanca se realizan a los principales algoritmos o procedimientos (Pressman, 2002).

Entre las técnicas de pruebas de caja blanca se encuentra la técnica de camino básico**.** Esta técnica proporciona una medida cuantitativa de la complejidad lógica de un programa. Se basa en obtener una medida de la complejidad de un procedimiento o algoritmo y un conjunto básico de caminos de

ejecución de este, los cuales luego se utilizan para obtener los casos de prueba, asegura que durante la prueba se ejecute al menos una vez cada sentencia del código que se está probado. Entre las métricas del software para realizar pruebas unitarias se encuentra la complejidad ciclomática la que en este caso será utilizada en conjunto con la técnica mencionada anteriormente (Pressman, 2002).

1.8.2 Pruebas de caja negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. Permiten derivar conjuntos de condiciones de entrada que ejercitan por completo todos los requisitos funcionales del programa. Permite encontrar funciones incorrectas o ausentes, error de interfaz, errores en estructuras de datos o acceso a las bases de datos externas, errores de rendimiento, de inicialización y de terminación (Pressman, 2002).

Para la aplicación de la prueba de caja negra se propone utilizar la técnica Partición equivalente, que permite examinar los valores válidos e inválidos de las entradas existentes en el software (Pressman, 2002). Esta técnica propone la confección de casos de prueba del sistema. Tiene como objetivo ejecutar los casos de pruebas y encontrar la mayor cantidad de no conformidades, mientras exista alguna no conformidad el sistema no puede ser desplegado, por lo que es obligatorio realizar todas las iteraciones que sean necesarias hasta que el mismo se encuentre libre de no conformidades.

1.8.3 Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las HU en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como “pruebas de caja negra” donde cada prueba representa una salida esperada del sistema. Según (Joskowicz, 2008) se realizan con el objetivo de validar que un sistema cumple con el funcionamiento requerido y esperado por el cliente, permitiendo al mismo determinar su aprobación, teniendo en cuenta la funcionalidad y el rendimiento de la aplicación informática. Estas pruebas se caracterizan por (Juristo, 2005):

- Participación activa del usuario que debe ejecutar los casos de prueba ayudado por miembros del equipo de pruebas.
- Están enfocadas a probar los requisitos de usuario, o mejor dicho a demostrar que no se cumplen los requisitos, los criterios de aceptación o el contrato. Si no se consigue demostrar esto el cliente deberá aceptar el producto.

1.9 Conclusiones parciales

Según lo expuesto en el capítulo se arriba a las siguientes conclusiones:

- La definición del marco conceptual relacionado con la gestión de información de los procesos del PCC contribuyó a una mejor comprensión de la investigación.

- El estudio de los Sistemas de gestión de información a nivel nacional e internacional, fundamenta la necesidad de desarrollar un sistema que incluya las funcionalidades y procesos que se realizan en el PCC de la Facultad 3.
- La selección de las tecnologías, herramientas y metodología a utilizar permitió establecer el entorno de desarrollo en el que se implementa la propuesta de solución.

Capítulo 2: Análisis y diseño de la solución

2.1 Introducción

En el presente capítulo se define el negocio a través del modelo de dominio y se realiza la descripción de las fases: planificación y diseño propuestos por la metodología de desarrollo de software seleccionada. Se describe de forma general la propuesta de solución, los roles que interactúan con dicha propuesta, los requisitos funcionales y no funcionales, las Historias de Usuario (HU) y Tarjetas CRC.

2.2 Modelo de dominio

A partir del análisis realizado en el capítulo anterior, se propone el desarrollo de un sistema web que gestione los procesos del PCC en la Facultad 3. Para un mejor entendimiento del tema de investigación se decide confeccionar el modelo de dominio. Según (Larman, 2003) un modelo de dominio es una representación visual de las clases u objetos del mundo real en un dominio de interés.

Utilizando UML, un modelo de dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. A continuación en la Figura 1 se exponen las principales clases así como sus relaciones.

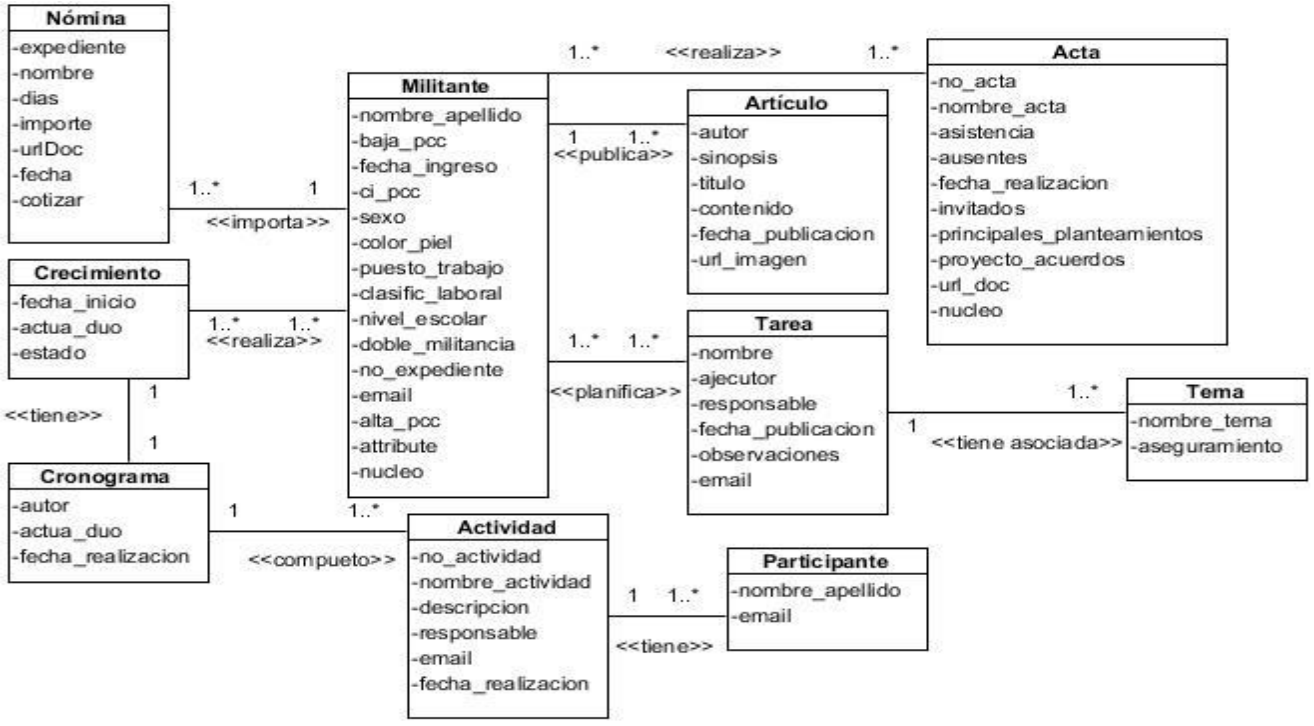


Figura 1. Modelo de dominio

2.3 Roles relacionados con el sistema

En la Tabla 8 se muestran los roles que interactúan con el sistema y la descripción de los módulos a los que tienen acceso.

Tabla 8. Roles

Roles	Descripción
Usuario	Usuarios que acceden al sistema para ver los artículos publicados y realizar comentarios sobre los mismos.
Administrador	Usuario encargado de la gestión de roles, usuarios, artículos, comentarios, así como la revisión de artículos y comentarios.
Secretario general del comité primario	Usuario con acceso a todos los procesos del PCC.
Secretario general del núcleo	Usuario con acceso a los módulos: Militante, Planificación, actas, Crecimiento, Reportes y Cotización.
Secretario ideológico	Usuario con acceso a todos los módulos pero con solo permisos de lectura.
Secretario de funcionamiento	Usuario con acceso a los módulos: Actas, Crecimiento, Militante y Cotización.
Activista de acta	Usuario con acceso a los módulos: Actas y Reportes.
Activista de cotización	Usuario con acceso a los módulos: Cotización y Reportes.
Dúos asignados para el crecimiento	Usuario con acceso a los módulos: Crecimiento y Reportes.

2.4 Fase de planificación

La metodología de desarrollo XP define como primera fase la planificación. Esta es una fase corta, en la que el cliente y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y asociadas a estas las entregas. El resultado de esta fase es un Plan de Entregas (Joskowicz, 2008).

Con el objetivo de definir y describir las HU se utilizaron las técnicas Entrevista, Tormenta de ideas y Estudio de documentación. A continuación se describen dichas técnicas.

Entrevista: permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de desarrollo se acerca al problema de una forma natural (Escalona, 2002).

Tormenta de ideas: son reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Es sencilla de usar y de aplicar y suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros (Escalona, 2002).

Estudio de documentación: consiste en realizar una lectura basada en documentos sobre el dominio del negocio de la organización o sus prácticas profesionales. Ejemplos de algunos documentos que se pueden consultar son: estatutos y reglamentos del PCC (Andalucía, 2013).

2.4.1 Historias de usuario

Las HU sustituyen a los documentos de especificación funcional y a los casos de uso. Estas historias son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido *(Joskowicz, 2008)*. Se describieron un total de 11 HU, continuación en la Tabla 9 y

Tabla 10 se muestran dos de las HU realizadas, las restantes pueden ser consultadas en el expediente de proyecto Artefactos con el nombre HistoriaUsuario.doc.

Tabla 9. Historia de usuario: Gestionar militante

Historia de usuario	
Número: 3	**Nombre:** Gestionar militante
Usuario: Secretario general, administrador	**Iteración asignada:** 1ra
Referencia: RF12, RF13, RF14, RF15, RF16, RF17	
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Descripción: La funcionalidad gestionar militante debe permitir adicionar, modificar, mostrar y eliminar un militante en el sistema. En una interfaz se listan los artículos que han sido adicionados, mostrando una breve información de los mismos: carné de identidad, nombre y apellidos, fecha de ingreso, carné del PCC, sexo, puesto de trabajo, número de expediente y las opciones para seleccionar las acciones que se pueden efectuar sobre estos. • **Adicionar:** en una interfaz se solicitan los datos: carné de identidad, nombre y apellidos, Fecha de alta, fecha de baja, fecha de ingreso, carné de identidad del PCC, sexo, color de piel, puesto de trabajo, clasificación laboral, nivel escolar, doble militancia, email, número de expediente y nombre del núcleo al que pertenece.	

• **Modificar:** cuando se selecciona el militante que se desea modificar se muestra la información que este contiene en una nueva interfaz, permitiendo realizar cambios en sus datos. • **Eliminar:** se elimina el militante seleccionado, antes de ejecutar esta acción el usuario debe confirmar la operación para evitar la pérdida de datos. • **Mostrar:** cuando se selecciona el militante que se desea mostrar se muestran sus datos. • **Listar:** se muestra una interfaz con los militantes adicionados. • **Buscar:** permite buscar un militante que se desea consultar.
Observaciones: El Secretario general del comité primario y el administrador del sistema son los únicos usuarios con acceso a las funcionalidades descritas.

Tabla 10. Historia de Usuario: Gestionar artículo

Historia de usuario	
Número: 7	**Nombre:** Gestionar artículo
Usuario: Administrador	**Iteración asignada:** 1ra
Referencia: RF123, RF124, RF125, RF126, RF127, RF128	
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alta	**Puntos reales:** 1
Descripción: La funcionalidad gestionar artículo debe permitir adicionar, modificar, mostrar, listar, buscar y eliminar un artículo en el sistema. En una interfaz se listan los artículos que han sido adicionados, mostrando una breve información de los mismos: título, autor, fecha de publicación, así como las opciones que se pueden ejecutar sobre éstos: • **Adicionar:** en una interfaz se solicitan los datos: título, autor, sinopsis, contenido, fecha de publicación y url de la imagen. • **Modificar:** cuando se selecciona el artículo que se desea modificar se muestra la información que este contiene en una nueva interfaz, permitiendo realizar cambios en sus datos. • **Mostrar:** cuando se selecciona el artículo que se desea mostrar se muestran los datos: título, autor, sinopsis, contenido, fecha de publicación y url de la imagen. • **Eliminar:** se elimina el artículo seleccionado, antes de ejecutar esta acción el usuario debe confirmar la operación para evitar la pérdida de datos. • **Listar:** se muestra una interfaz con los artículos adicionados. • **Buscar:** Permite buscar un artículo que se desea consultar.	

Observaciones: El administrador del sistema es el único usuario con acceso a las funcionalidades descritas.

Las HU permitieron describir los requisitos del usuario sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para esto, así como también responder rápidamente a los requisitos cambiantes.

2.4.2 Requisitos de software

La metodología XP permite que los requisitos se vayan definiendo a lo largo del proyecto, de una manera ordenada. De esta forma se posibilita que el cliente pueda ir cambiando de opinión sobre la marcha, pero a cambio han de estar siempre disponibles para solucionar las dudas del equipo de desarrollo (XP, 2015). El equipo de desarrollo fue definiendo los requisitos a medida que el cliente entregaba las HU para ir integrando cada uno de estos al sistema.

En (Somerville, 2005) Somerville define un requisito de software como una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. Los requisitos de software se clasifican en funcionales y no funcionales.

2.4.2.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Somerville, 2005). En la

Tabla *11* se enuncian los requisitos funcionales (RF en lo adelante) identificados a partir de las HU que el usuario describe con sus propias palabras las características del sistema:

Tabla 11. Requisitos funcionales

No.	Requisitos funcionales	Prioridad	No.	Requisitos funcionales	Prioridad
RF1	Generar reporte militantes	Alta	RF70	Buscar autobiografía	Media
RF2	Generar reporte reunión ordinaria	Alta	RF71	Listar autobiografía	Media
RF3	Generar reporte reunión extraordinaria	Alta	RF72	Descargar autobiografía	Media
RF4	Generar reporte reunión circulo de estudios políticos	Alta	RF73	Adjuntar documento de entrevista individual	Media
RF5	Adicionar tarea	Alta	RF74	Adicionar entrevista	Media
RF6	Mostrar tarea	Alta	RF75	Modificar entrevista	Media

RF7	Modificar tarea	Alta	RF76	Eliminar entrevista	Media
RF8	Eliminar tarea	Alta	RF77	Buscar entrevista	Media
RF9	Listar tarea	Alta	RF78	Listar entrevista	Media
RF10	Buscar tarea	Alta	RF79	Descargar entrevista	Media
RF11	Adicionar tema	Alta	RF80	Adjuntar comprobación	Media
RF12	Mostrar tema	Alta	RF90	Adicionar comprobación	Media
RF13	Modificar tema	Alta	RF91	Modificar comprobación	Media
RF14	Eliminar tema	Alta	RF92	Eliminar comprobación	Media
RF15	Listar tema	Alta	RF93	Buscar comprobación	Media
RF16	Buscar tema	Alta	RF94	Listar comprobación	Media
RF17	Adicionar militante	Alta	RF95	Descargar comprobación	Media
RF18	Modificar militante	Alta	RF96	Exportar comprobación	Media
RF19	Eliminar militante	Alta	RF97	Adjuntar documento resumen de trayectoria revolucionaria	Media
RF20	Listar militantes	Alta	RF98	Adicionar trayectoria revolucionaria	Media
RF21	Buscar militante	Alta	RF99	Modificar trayectoria revolucionaria	Media
RF22	Mostrar datos del militante	Alta	RF100	Eliminar trayectoria revolucionaria	Media
RF23	Adjuntar acta	Media	RF101	Buscar trayectoria revolucionaria	Media
RF24	Crear reseña del acta	Media	RF102	Listar trayectoria revolucionaria	Media
RF25	Listar reseñas	Media	RF103	Descargar trayectoria revolucionaria	Media
RF26	Buscar reseña	Media	RF104	Adjuntar documento reunión de admisión	Media
RF27	Exportar reseña del acta	Media	RF105	Adicionar reunión de admisión	Media

RF28	Modificar reseña del acta	Media	RF106	Modificar reunión de admisión	Media
RF29	Descargar reseña del acta	Media	RF107	Eliminar reunión de admisión	Media
RF30	Eliminar reseña del acta	Media	RF108	Buscar reunión de admisión	Media
RF31	Adicionar nómina de pago de cotización	Media	RF109	Listar reunión de admisión	Media
RF32	Eliminar nómina de pago de cotización	Media	RF110	Descargar reunión de admisión	Media
RF33	Buscar nómina de pago de cotización	Media	RF111	Adjuntar modelo de conclusión	Media
RF34	Adicionar crecimiento	Media	RF112	Adjuntar modelo de presentación	Media
RF35	Modificar crecimiento	Media	RF113	Adicionar acta de conclusión	Media
RF36	Eliminar crecimiento	Media	RF114	Modificar acta de conclusión	Media
RF37	Listar crecimiento	Media	RF115	Eliminar acta de conclusión	Media
RF38	Buscar crecimiento	Media	RF116	Buscar acta de conclusión	Media
RF39	Adicionar cronograma del crecimiento	Media	RF117	Listar acta de conclusión	Media
RF40	Modificar cronograma	Media	RF118	Descargar modelo de conclusión	Media
RF41	Eliminar cronograma	Media	RF119	Descargar modelo de presentación	Media
RF42	Listar cronogramas	Media	RF120	Adicionar artículo	Baja
RF43	Adicionar actividad al cronograma	Media	RF121	Modificar artículo	Baja
RF44	Modificar actividad	Media	RF122	Eliminar artículo	Baja
RF45	Eliminar actividad	Media	RF123	Mostrar artículo	Baja
RF46	Listar actividades	Media	RF124	Listar artículo	Baja
RF47	Buscar actividad	Media	RF125	Buscar artículo	Baja
RF48	Adicionar participantes a la actividad	Media	RF126	Modificar comentario	Baja

RF49	Modificar participantes	Media	RF127	Eliminar comentario	Baja
RF50	Eliminar participantes	Media	RF128	Buscar comentario	Baja
RF51	Listar participantes	Media	RF129	Mostrar comentario	Baja
RF52	Buscar participantes	Media	RF130	Listar comentario	Baja
RF53	Adjuntar documento de reunión (Consulta con las masas)	Media	RF131	Adicionar usuario	Baja
RF54	Adicionar reseña	Media	RF132	Modificar usuario	Baja
RF55	Modificar reseña	Media	RF133	Eliminar usuario	Baja
RF56	Eliminar reseña	Media	RF134	Buscar usuario	Baja
RF57	Listar reseñas	Media	RF135	Mostrar usuario	Baja
RF58	Buscar reseñas	Media	RF136	Adicionar noticia	Baja
RF59	Descargar documento de reunión (Consulta con las masas)	Media	RF137	Modificar noticia	Baja
RF60	Exportar reseña consulta con las masas	Media	RF138	Eliminar noticia`	Baja
RF61	Adicionar propuesto	Media	RF139	Listar noticia	Baja
RF62	Modificar propuesto	Media	RF140	Buscar noticia	Baja
RF63	Eliminar propuesto	Media	RF141	Mostrar noticia	Baja
RF64	Listar propuesto	Media	RF142	Adicionar nomenclador	Baja
RF65	Buscar propuesto	Media	RF143	Modificar nomenclador	Baja
RF66	Adjuntar autobiografía	Media	RF144	Eliminar nomenclador	Baja
RF67	Adicionar autobiografía	Media	RF145	Listar nomenclador	Baja
RF68	Modificar autobiografía	Media	RF146	Buscar nomenclador	Baja
RF69	Eliminar autobiografía	Media			

2.4.2.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares (Somerville, 2005). En la

Tabla 12 se enuncian los requisitos no funcionales (RNF) identificados a través de las técnicas aplicadas para la captura de estos.

Tabla 12. Requisitos no funcionales

Número	Requisitos no funcionales
Seguridad	
RNF1	Confiabilidad: La información manejada en el sistema debe de estar protegida de acceso no autorizado.
Usabilidad	
RNF2	Las funcionalidades y diseño del sistema deben ser intuitivas de forma tal que los usuarios puedan interactuar con el sistema sin necesidad de poseer conocimientos básicos de informática.
Rendimiento	
RNF3	Los tiempos de respuesta del sistema deben ser rápidos, no mayores de 5 segundos para cualquier operación realizada por el usuario.
Soporte	
RNF4	Se debe brindar al usuario la facilidad de instalación.
Sistema	
RNF5	La fecha y hora del sistema tienen que estar siempre actualizados.
Software	
RNF6	El servidor debe tener el sistema operativo con entorno de escritorio gráfico, preferentemente Windows o GNU/Linux en cualquiera de sus versiones.
RNF7	La estación de trabajo cliente debe poseer un navegador web para acceder al sistema informático, preferentemente el Mozilla Firefox o Chrome en su versión 34 o superior y 31 o superior respectivamente.
Hardware	
RNF8	Memoria RAM 512 MB o superior.
RNF9	Procesador a 1 GHz. de velocidad de procesamiento o superior.

2.4.2.3 Validación de requisitos

La validación de requisitos se hace necesaria para demostrar que la definición de estos describen realmente el sistema que el usuario necesita y para asegurar que el análisis realizado y los resultados

obtenidos de la etapa de definición de requisitos son correctos (Escalona, 2002). Las técnicas utilizadas se describen a continuación:

Matriz de trazabilidad: consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo los objetivos que cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos (Escalona, 2002).

Prototipos de interfaz: se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario (Escalona, 2002). Una vez identificados los requisitos, se diseñaron los prototipos pertenecientes a cada uno de estos utilizando la herramienta de modelado Visual Paradigm.

En la Tabla 13 se muestra una representación de la matriz de trazabilidad y en la Figura 2 y 3, 2 ejemplos de los prototipos de interfaz realizados. Los restantes pueden ser consultados en el expediente de proyecto Artefactos con el nombre Prototipos.doc.

Tabla 13. Representación de la matriz de trazabilidad

HU / Requisitos	HU1	HU2	HU3	HU4	HU5	HU6	HU7	HU8	HU9	HU10	HU11
RF1	X										
RF5		X									
RF17			X								
RF23				X							
RF31					X						
RF34						X					
RF120							X				
RF126								X			
RF131									X		
RF136										X	
RF142											X

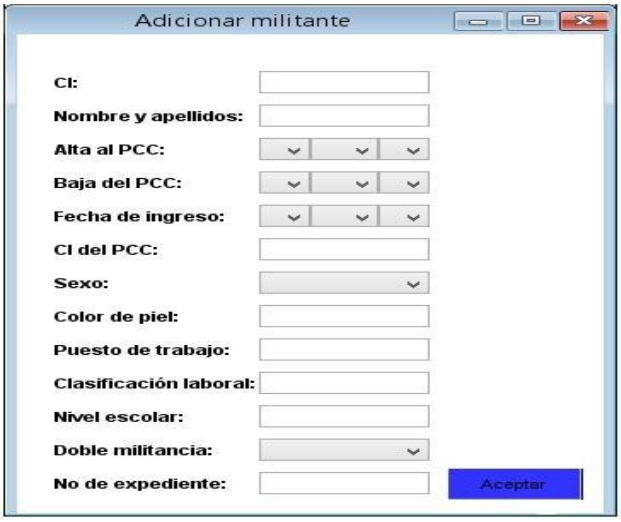


Figura 2. Prototipo adicionar militante

Figura 3. Prototipo adicionar artículo

Resultado de las técnicas aplicadas en la validación de los requisitos:

La matriz de trazabilidad permite detectar inconsistencias u objetivos no cubiertos al marcar los objetivos y chequearlos contra los requisitos del sistema, por lo que se le adicionaron al mismo los requisitos no identificados. Los prototipos fueron presentados al cliente, quien ratificó que los mismos cubrían sus necesidades.

2.4.3 Estimación del esfuerzo por Historia de usuario

Las HU deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menor de una semana, se debe combinar con otra (Joskowicz, 2008). Estas estimaciones permiten tener una medida de la velocidad del proyecto y ofrecen una guía a la cual ajustarse. Las realiza el equipo de desarrollo en conjunto con el cliente. Los resultados de la estimación realizada se muestran en la Tabla 14.

Tabla 14. Estimación de esfuerzos por HU

No	Historias de usuario	Puntos de estimación (semanales)
1	Generar reportes	2
2	Gestionar planificación	2
3	Gestionar militante	1
4	Gestionar acta	1
5	Realizar proceso de cotización	1
6	Gestionar crecimiento	3
7	Gestionar artículo	1

8	Gestionar comentario	1
9	Gestionar usuario	1
10	Gestionar noticia	1
11	Gestionar nomenclador	1

2.4.4 Plan de iteraciones

Después de identificar y describir las HU y de estimar el esfuerzo para la realización de cada uno de estos se puede comenzar la implementación del sistema. Se seleccionan las HU a implementar en cada iteración, de acuerdo a su nivel de prioridad, así como las posibles fechas de entrega. El plan de duración de las iteraciones tiene como objetivo mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU como se muestra en la Tabla 15.

Tabla 15. Plan de duración de las iteraciones

Iteraciones	Historias de usuario a implementar	Duración de las iteraciones (semanales)
Iteración 1	Generar reporte	5
	Gestionar planificación	
	Gestionar militante	
Iteración 2	Gestionar acta	5
	Proceso de cotización	
	Gestionar crecimiento	
Iteración 3	Gestionar artículo	5
	Gestionar comentario	
	Gestionar usuario	
	Gestionar noticia	
	Gestionar nomenclador	
Total en semanas		15

2.4.5 Plan de entrega

El cronograma de entregas establece las HU que serán agrupadas para conformar una entrega y el orden de las mismas. Se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores (Joskowicz, 2008). En la Tabla 16 se muestra como quedó conformado el plan de entrega desarrollado con el cliente.

Tabla 16. Plan de entrega de las iteraciones

Artefacto	Iteración	Entrega
HU 1	1	13 de marzo
HU 2		
HU 3		
HU 4	2	17 de abril
HU 5		
HU 6		
HU 7	3	15 de mayo
HU 8		
HU 9		
HU 10		
HU 11		

2.5 Fase de diseño

En esta fase se confeccionan las tarjetas CRC para guiar el proceso de implementación de la solución, además se describen los patrones de diseño utilizados. Esta fase aporta una representación del software mediante el establecimiento de la arquitectura y el modelo de datos.

2.5.1 Tarjetas CRC

Las tarjetas CRC son utilizadas para representar las responsabilidades de las clases y sus interacciones. El nombre de la clase se coloca a modo de título de la tarjeta, las responsabilidades se colocan a la izquierda y las clases que se implican en cada responsabilidad a la derecha. Las tarjetas determinan el comportamiento de cada actividad. En las Tabla 17 y Tabla 18 se muestran las tarjetas CRC de las clases DefaultController y MilitanteController, las restantes pueden ser consultadas en el expediente de proyecto Artefactos con el nombre TarjetasCRC.doc.

Tabla 17. Tarjeta CRC: ArticuloController

Nombre de la clase: ArticuloController	
Responsabilidades	**Colaboradores**
Mostrar listado de los artículos creados	Artículo
Adicionar artículo	
Mostrar la información del artículo	

Modificar artículo	
Eliminar artículo	
Buscar artículo	
Crear formulario de adicionar artículo	
Crear formulario para modificar el artículo	

Tabla 18. Tarjeta CRC: MilitanteController

Nombre de la clase: MilitanteController	
Responsabilidades	**Colaboradores**
Mostrar listado de militantes	Militante
Adicionar militante	
Mostrar la información del militante	
Modificar militante	
Eliminar militante	
Buscar militante	
Crear formulario de militante	
Crear formulario para modificar el militante	

Con la realización de las tarjetas CRC se logra realizar una descripción de las clases utilizadas en la implantación del sistema y la forma en que van a interactuar. Estas facilitan el análisis y discusión de las mismas por parte del equipo de desarrollo con el objeto de que el diseño fuese lo más simple posible verificando las especificaciones del sistema.

2.5.2 Modelo de datos

El modelo de datos describe de forma abstracta las entidades, sus características y relaciones entre estas dentro de la base de datos. Las entidades son objetos que guardan información necesaria para el sistema. Los atributos son características de una entidad, se clasifican en obligatorios, opcionales, claves foráneas y claves primarias. Se tuvo en cuenta el patrón de diseño: llave subrogada, a partir de la generación de una llave primaria única para cada entidad en vez de usar un atributo identificador en el contexto dado, permitiendo que las tablas sean más fáciles de consultar por el identificador, dado que se conoce el mismo en cada una de ellas. El modelo de datos que se muestra en la Figura 4 está conformado por 25 tablas.

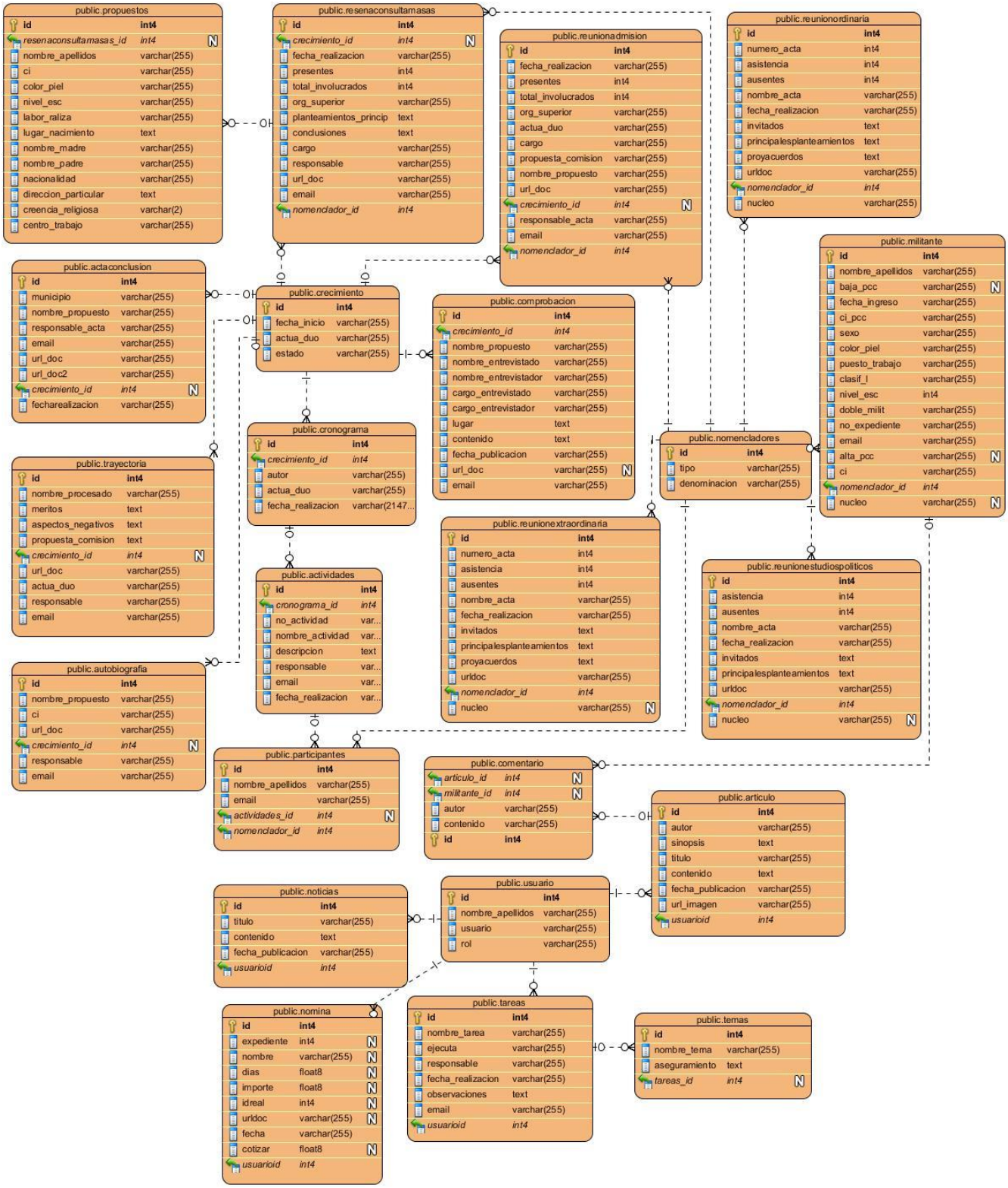


Figura 4. Modelo de Datos

2.5.3 Patrones de diseño utilizados

En la implementación del sistema se utilizaron varios patrones de diseño, estos se definieron en el capítulo anterior y a continuación se explica el modo en que fueron utilizados.

- **Patrones GRASP**

Controlador: este patrón está presente en los controladores frontales de Symfony2. Todas las peticiones web realizadas por el usuario son manejadas por los archivos app.php y app_dev.php y conforman los únicos puntos de entrada a la aplicación en los diferentes entornos que maneja Symfony2 "desarrollo y producción". Una vez que este recibe la petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL introducida por el usuario. Después de finalizada la ejecución, le devuelve al usuario la página creada a partir de la plantilla. El controlador frontal se localiza en "Tesis/src/UCI/TesisBundle/web/".

Dicho patrón también se encuentra presente en todas las clases controladoras y sirve de intermediario entre las interfaces y el algoritmo que las implementa. Es el encargado de recibir los datos del usuario y enviarlos a las distintas clases según el método solicitado. Su filosofía principal se basa en que la lógica de negocios debe estar separada de la capa de presentación, con el propósito de aumentar la reutilización de código y tener un mayor control sobre los cambios. En la Figura 5 se muestra un ejemplo de este patrón.

```
<?php

namespace UCI\TesisBundle\Controller;

use ...

/** Militante controller. ...*/
class MilitanteController extends Controller
{
    public function indexAction(){...}

    public function createAction(Request $request){...}

    public function newAction(){...}

    public function showAction($id){...}

    public function editAction($id){...}

    public function updateAction(Request $request, $id){...}

    public function deleteAction(Request $request, $id){...}
```

Figura 5. Controlador del módulo Militante

Experto: este patrón es uno de los que se emplean cuando se hace uso del marco de trabajo Symfony2. En la solución propuesta se puede apreciar un ejemplo de su inclusión con el uso de Doctrine como ORM a la hora de construir los módulos. Doctrine genera las clases para la gestión de

las entidades con las responsabilidades debidamente asignadas debido a que cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan. Además este patrón se utiliza en las clases controladoras cuando se construye una vista y en las clases gestoras debido a que estas contienen toda la información correspondiente a las entidades. En la Figura 6 se muestra un ejemplo de este patrón.

```php
<?php

namespace UCI\TesisBundle\Controller;

use ...

/** Militante controller. ...*/
class MilitanteController extends Controller
{
    /** Lists all Militante entities. ...*/
    public function indexAction()
    {
        $em = $this->getDoctrine()->getManager();

        $entities = $em->getRepository('TesisBundle:Militante')->findAll();

        return $this->render('TesisBundle:Militante:index.html.twig', array(
            'entities' => $entities,
        ));
    }
```

Figura 6. Construcción de una vista

Creador: la utilización de este patrón se evidencia en las clases gestoras, las cuales se encargan de crear los objetos de las clases que representan las entidades, evidenciando de esta manera que las clases gestoras son las creadoras de las entidades. Esto siempre ocurre cuando un controlador necesita dicha operación. También se evidencia el uso de este patrón en las clases controladoras a la hora de crear los formularios. En la Figura 7 se muestra un ejemplo de la utilización de dicho patrón.

```
public function createAction(Request $request)
{
    $entity  = new Militante();
    $form = $this->createForm(new MilitanteType(), $entity);
    $form->bind($request);

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

        return $this->redirect($this->generateUrl('militante_show', array('id' => $entity->getId())));
    }

    return $this->render('TesisBundle:Militante:new.html.twig', array(
        'entity' => $entity,
        'form'   => $form->createView(),
    ));
}
```

Figura 7. Creación de objetos y formularios

Bajo acoplamiento: este patrón está estrechamente relacionado con los patrones Experto y Alta Cohesión y plantea la baja dependencia que debe existir entre las clases. Symfony2 favorece ampliamente el bajo acoplamiento de las clases en el sistema, esto ocurre porque a cada clase se le asignan solamente las responsabilidades necesarias de manera que no dependan en gran medida de otras. Esto favorece la escalabilidad del sistema, la reutilización y el mantenimiento futuro de la aplicación. Se evidencia claramente el uso de este patrón en las entidades, que son las clases más reutilizadas y no presentan ninguna asociación con la vista ni con el controlador.

Alta cohesión: este patrón tiene como propósito asignar responsabilidades de manera que la cohesión siga siendo alta. El marco de trabajo Symfony favorece la alta cohesión asignando responsabilidades a las clases de tal manera que estas se encuentren estrechamente relacionadas entre sí y no lleguen a realizar un trabajo excesivo.

- **Patrones GOF**

Factory Method: los contenedores de servicios de Symfony2 proporcionan una forma eficaz de controlar la creación de objetos, lo cual permite especificar los argumentos pasados al constructor, así como llamar a los métodos y establecer parámetros. Sin embargo, en algunas ocasiones esto no proporcionará todo lo necesario para construir objetos. Es en esta situación, donde se puede utilizar una fábrica para crear el objeto y decirle al contenedor de servicios que llame a un método en la fábrica y no directamente una instancia del objeto.

Esto se puede apreciar cuando se crean los servicios de las clases Repository en “Tesis/src/UCI/TesisBundle/Entity/EntityRepository”. Estas clases son las que contienen las consultas a la base de datos, una vez que sea necesario hacer uso de una consulta en alguna clase gestora,

simplemente se llama al servicio del Repository donde dicha consulta se encuentra y se accede a esta, sin necesidad de crear una instancia de la clase que la contiene.

Patrón Decorador: este patrón viene implícito en el marco de trabajo Symfony2 al hacer uso del motor de plantillas Twig. La parte más poderosa de Twig es la herencia entre plantillas, que permite crear un "esqueleto" de plantilla base que contenga todos los elementos comunes del sitio y define los bloques que las plantillas descendientes pueden sustituir. Esta plantilla es el archivo base.html.twig, que define un simple documento donde se almacena el código HTML que es común a todas las páginas de sistema, para no tener que repetirlo en cada una de ellas. Es trabajo de las plantillas "descendientes" rellenar los bloques vacíos con contenido.

2.5.4 Arquitectura del sistema

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

- La vista: es la capa que permite al usuario interactuar directamente con la aplicación, dentro de esta se ubica lo referente a la visualización de la información en las páginas de la aplicación. En synfomy2 se evidencia esta capa a través de la biblioteca jQuery en conjunto con los archivos CCS, JavaScript y HTML.
- El controlador: responde a eventos que se traducen en solicitudes de servicios para el modelo o la vista, gestionando la interacción entre el usuario y los datos. Esta capa queda evidenciada a través del Controlador Frontal, mediante los archivos app.php para el entorno de producción y app_dev.php para el entorno de desarrollo. Los archivos de enrutamiento también forman parte de esta capa, los cuales permiten determinar el controlador que está asociado con la página que se solicita. El motor de plantillas Twig es el encargado de transformar cada archivo Twig en HTML y las clases Controller controlarán el flujo de información que se recibe y se envía hacia la vista.
- El modelo: administra el comportamiento y los datos del dominio de la aplicación. Esta capa se divide en dos subcapas, la capa de negocio y la capa de acceso a datos. La subcapa de negocio contiene las clases gestoras encargadas del manejo de la lógica del negocio. La subcapa de acceso datos es la encargada de establecer la conexión con la base de datos. Dentro de esta subcapa se encuentra ubicado el ORM Doctrine para la comunicación con la base de datos.
- Datos: en esta capa se encuentra el sistema gestor de base de datos PostgreSQL, este permite que persista la información con la que trabaja la aplicación y gestiona su almacenamiento.

En la Figura 8 se muestra la estructura de la arquitectura descrita anteriormente:

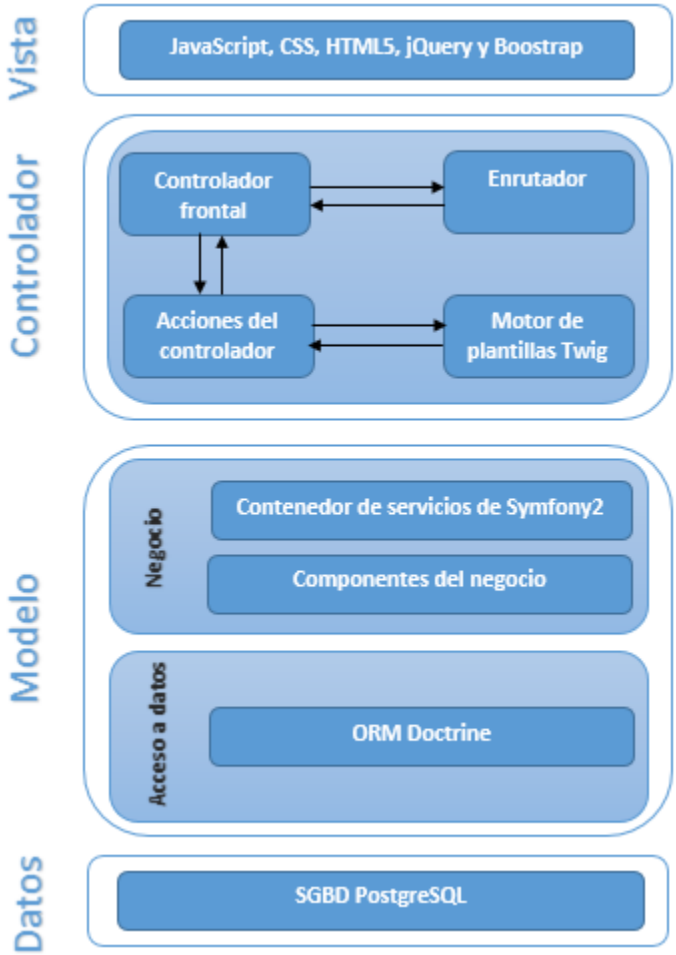


Figura 8. Arquitectura del sistema. Fuente: Elaboración propia

Conforman además la arquitectura varios componentes que satisfacen el conjunto de requisitos definidos en la solución, estos permiten utilizar funcionalidades construidas por terceros o empaquetar funcionalidades para distribuirlas y reutilizarlas. En la Figura 9 se muestran seleccionados los componentes utilizados:

Figura 9. Componentes utilizados por terceros

Liuggio: es un bundle que permite la lectura y escritura de archivos Excel a través de funciones que brinda el lenguaje php, gracias a este componente el sistema permite importar la nómina de pago de la cotización utilizando un fichero en formato .xls adjuntado por el usuario.

Spraed: es un bundle que permite convertir paginas html en documentos PDF empleados en la generación de diversos reportes gracias a las funcionalidades de Java que ejecuta este bundle en su funcionamiento interno.

Swiftmailer: es un bundle que provee al sistema un servicio de mensajería a través de diversas notificaciones enviadas a los usuarios garantizando así el cumplimiento de las actividades y procesos definidos en el sistema.

Ztec bundle: es utilizado para garantizar la seguridad del sistema, haciendo uso de la autenticación por el servicio LDAP, lográndose así que solamente puedan acceder al sistema únicamente usuarios que posean una cuenta del dominio UCI.

2.5.5 Métricas para la validación del diseño

Para la validación del diseño se utilizaron las métricas Tamaño operacional de clase (TOC) y Relación entre Clases (RC). A continuación se describe como son aplicadas al diseño de la propuesta de solución:

Tamaño operacional de clase

En la Figura 10 se observa la representación de los resultados obtenidos al realizar la evaluación de la métrica TOC en los atributos reutilización, acoplamiento, complejidad de mantenimiento y cantidad de pruebas. Para determinar el valor de los atributos se calcula el promedio de relaciones entre clases, en este caso se obtuvo 1,68.

Tras un análisis de los resultados arrojados por dicha métrica se demuestra que se alcanzaron buenos resultados para cada uno de los atributos de calidad evaluados, puesto que, como se observa el 42% de las clases poseen una baja responsabilidad, lo que permite que la complejidad de implementación también sea baja en un 42% de las clases y por esto se tornan mucho más reutilizables.

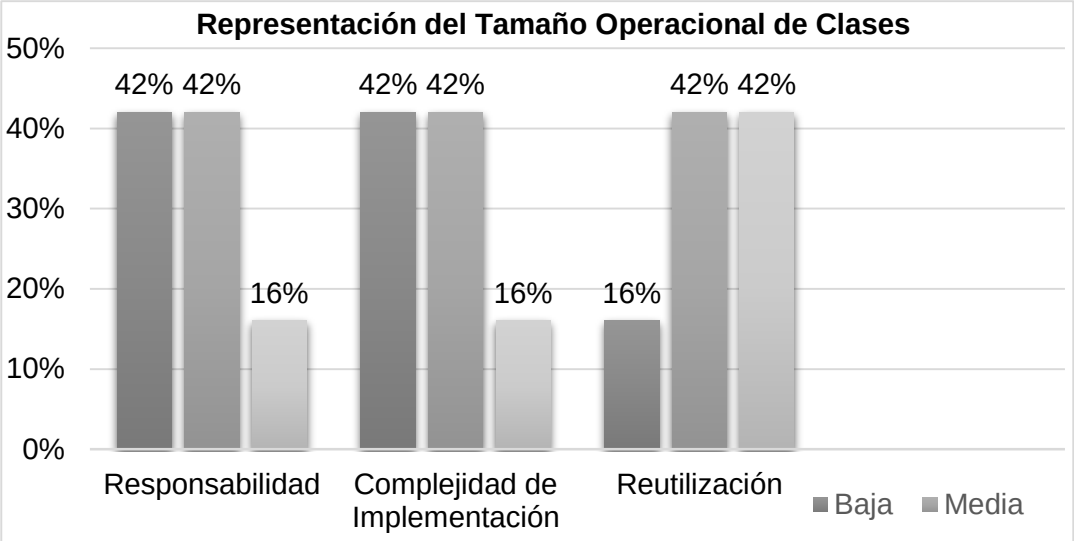


Figura 10. Gráfica de la métrica Tamaño Operacional de Clases

Relaciones entre clases

En la Figura 11 se observa la representación de los resultados obtenidos al realizar la evaluación de la métrica RC en los atributos responsabilidad, complejidad de implementación y reutilización. Para determinar el valor de los atributos se calcula el promedio de procedimientos por clases, en este caso se obtuvo 9,63.

Tras un análisis de los resultados arrojados se puede observar que el 61% de las clases posee una alta reutilización, además se evidencia el bajo acoplamiento en el 60% de las clases. Por otra parte no existe ninguna clase con complejidad de mantenimiento ni cantidad de pruebas alta.

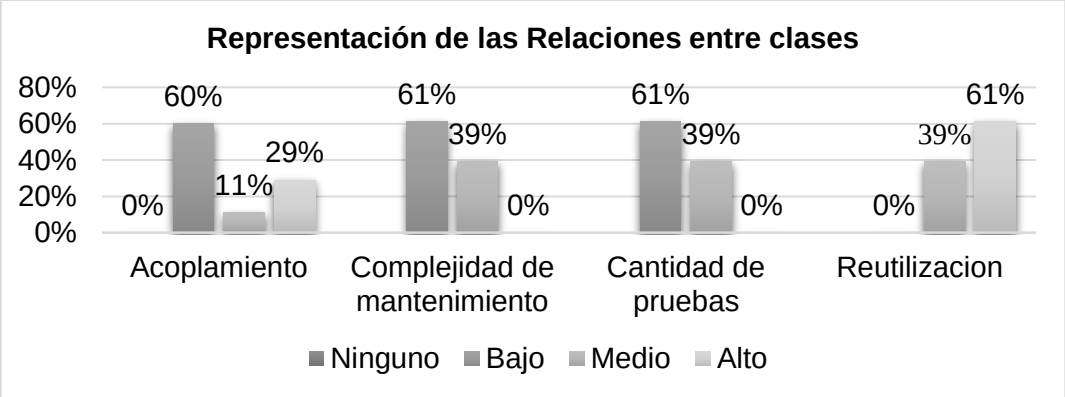


Figura 11. Gráfica de la métrica Relaciones entre clases

2.6 Conclusiones parciales

Según lo expuesto en el capítulo se arriba a las siguientes conclusiones:

- Realizar las HU permitieron determinar las funcionalidades básicas que se realizan en los procesos que lleva a cabo el PCC en la Facultad 3 de la UCI.
- La estimación del esfuerzo y el plan de iteraciones permitió crear un cronograma que facilita la implementación teniendo en cuenta los compromisos con el cliente.
- Con la creación de las tarjetas CRC se representaron las diferentes clases que intervienen en el sistema con sus responsabilidades y relaciones.
- El uso de patrones de diseño contribuyó a asignar correctamente las responsabilidades a las clases.
- Con la validación del diseño a través de métricas se determinó que la mayor parte de las clases son reutilizables y poco dependientes entre sí.

Capítulo 3: Implementación y prueba de la solución

3.1 Introducción

En este capítulo se abordan los elementos de la fase de desarrollo y prueba propuestos por la metodología que guía la investigación. Se define el estándar de codificación utilizado, aspectos de la seguridad del sistema, así como las pruebas realizadas: caja blanca, caja negra, y de aceptación para la validación de la propuesta de solución.

3.2 Fase de desarrollo

En la fase de desarrollo se obtiene una primera versión del producto deseado por el cliente. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones, como parte de este plan se crean tareas de ingeniería para ayudar a organizar la implementación exitosa de las HU.

3.2.1 Tareas de ingeniería por iteraciones

Cada HU está compuesta por una o varias tareas de ingeniería, estas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. En la Tabla 19 se listan las tareas a desarrollar en la iteración 1 para cada HU, las restantes pueden ser consultadas en el expediente de proyecto Artefactos con el nombre TareasDeIngieneria.doc.

Tabla 19. Tareas de ingeniería para la iteración 1

Historias de usuario	Tareas de ingeniería por Historia de usuario
Generar reporte	• Generar reportes de las reuniones dado un rango de fechas.
Gestionar planificación	• Adicionar, modificar, eliminar y mostrar tarea. • Adicionar, modificar, eliminar y mostrar tema.
Gestionar Militante	• Adicionar, modificar y eliminar militante. • Listar los militantes que se encuentren almacenados y buscar un militante dado un parámetro determinado. • Mostrar interfaz con los datos del militante seleccionado.

3.2.2 Tareas de ingeniería detalladas

A continuación en las Tabla 20 y 21 se muestran dos de las tareas de ingeniería detalladas para la iteración 1, las restantes pueden ser consultadas en el expediente de proyecto Artefactos con el nombre TareasDeIngieneria.doc

Tabla 20. Tarea de ingeniería detallada 4

Tarea de Ingeniería	
Número de Tarea: 4	**Nombre Historia de usuario:** Gestionar militante
Nombre de la tarea: Adicionar, modificar y eliminar militante.	
Tipo de tarea: Desarrollo	**Puntos Estimados(días):** 2
Fecha inicio: 9/3/2015	**Fecha fin:** 10/3/2015
Programador responsable: Manuel González Flores y Liset Gómez Chávez	
Descripción: Permite mostrar interfaces que permitan adicionar, modificar y eliminar un militante.	

Tabla 21. Tarea de ingeniería detallada 5

Tarea de Ingeniería	
Número de Tarea: 5	**Nombre Historia de usuario:** Gestionar militante
Nombre de la tarea: Listar los militantes que se encuentren almacenados y buscar un militante dado un parámetro determinado.	
Tipo de tarea: Desarrollo	**Puntos Estimados(días):** 2
Fecha inicio: 11/3/2015	**Fecha fin:** 12/3/2015
Programador responsable: Manuel González Flores y Liset Gómez Chávez	
Descripción: Permite mostrar una interfaz con un listado de los militantes y además buscar un militante dado un parámetro determinado.	

3.2.3 Estructura física de la solución

La estructura física de la solución es la definida por el marco de trabajo Symfony2. Esta está conformada por un solo bundle llamado TesisBundle. Un bundle es un directorio que contiene todo tipo de archivos dentro una estructura jerarquizada de directorios. Los bundles de las aplicaciones Symfony2 suelen contener clases PHP y archivos web (JavaScript, CSS e imágenes) (Eguiluz, 2013).

TesisBundle posee todas las clases de dicho sistema (vistas, formularios, clases controladoras, entidades, repositorios y otras). A continuación se explica la estructura interna del bundle:

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

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

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

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

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

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

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

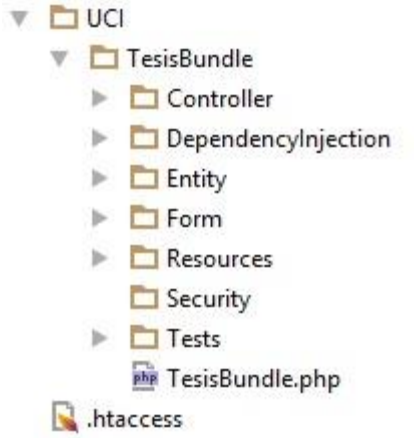


Figura 12. Estructura física de la solución

3.2.4 Estándares de codificación

Los estándares de codificación garantizan la comunicación fluida y directa entre los programadores, permitiendo el aumento de la reutilización y el mantenimiento de los sistemas. Con el objetivo de lograr un buen entendimiento del código, durante el desarrollo de la propuesta de solución se adopta utilizar el estándar CamelCase[11] para la definición de variables y funciones.

[11] CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula. Si comienza con mayúscula, se denomina UpperCamelCase y, si no, lowerCamelCase

- **Comentarios en las funciones**

Todas las funciones deben tener un comentario, antes de su declaración, explicando su funcionamiento. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben ser suficiente para

```
/*Crear un nuevo Militante.*/
public function createAction(Request $request)
```

entender el código.

- **Nombres de las variables**

Los nombres deben ser descriptivos y concisos. No usar grandes frases ni pequeñas abreviaciones para las variables. Siempre es mejor saber lo que hace una variable con solo conocer su nombre. Esto se aplica para los nombres de variables, funciones, argumentos de funciones y clases. Los nombres de las variables se definen utilizando el estilo lowerCamelCase.

```
private $ciPcc;
```

- **Nombre de las funciones**

Los nombres de las funciones pueden contener solo caracteres alfanuméricos y son definidos utilizando el estilo lowerCamelCase.

```
public function indexAction()
```

- **Nombre de las clases**

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará el estilo UpperCamelCase.

```
class ReunionadmisionController extends Controller
```

3.2.5 Descripción de la seguridad del sistema

El sistema de seguridad de Symfony2 se basa en identificar primero al usuario (autenticación) y comprueba después si ese usuario tiene acceso al recurso solicitado (autorización). Una vez realizada la autenticación del usuario en el sistema y este acceda a las funcionalidades del mismo se comprueba

si la URL de la petición realizada está protegida por un firewall[12] en la configuración del patrón (pattern) establecida. Comprobando si el usuario necesita estar autenticado.

Para autorizar el acceso o no de los usuarios a las distintas funcionalidades del sistema se hace uso del control de acceso, donde se comprueba que la ruta establecida para cada funcionalidad concuerda con el valor de su opción de configuración pattern. Chequeando que para el acceso a esta URL por el usuario este posea el rol definido en la configuración del access_control (control de acceso) establecido para el patrón de esta ruta. Además provee un nivel de seguridad mayor al ofrecer la opción de cifrado y descifrado del usuario y contraseña con potentes algoritmos de cifrado como es el caso del MD5, evitando que el valor de estos campos introducidos en el formulario de autenticación del sistema sean accedidos por atacantes.

3.3 Fase de pruebas

Pressman plantea que una prueba de software es la ejecución de programas de software con el objetivo de detectar defectos y fallas. Estas permiten comprobar y mostrar la calidad de un producto de software (Pressman, 2002). A continuación de describe como fueron aplicadas las pruebas unitarias, de aceptación y de caja negra en el sistema.

3.3.1 Pruebas unitarias

Para realizar las pruebas unitarias o de caja blanca se utiliza la técnica camino básico para obtener una medida de la complejidad de un algoritmo y un conjunto básico[13] de caminos de ejecución de este, utilizados luego para obtener los casos de prueba. La técnica de Camino Básico se realiza calculando la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se muestra en la Figura 13 el código de la funcionalidad AdicionarReunionAdmision encargada de adicionar los datos de la reunión de admisión.

[12] Un **cortafuego** (***firewall***) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas.

[13] Conjunto básico: es el conjunto de caminos independientes.

```
public function AdicionarReunionAdmision(Request $request) {
    $entity = new Reunionadmision(); //1
    $form = $this->createForm(new ReunionadmisionType(), $entity); //1
    $form->bind($request); //1
    $em = $this->getDoctrine()->getManager(); //1
    $name = $em->getRepository('TesisBundle:Reunionadmision')
            ->findByNombrePropuesto($entity->getNombrePropuesto()); //1
    if ($name != null) {//2
        $this->get('session')->getFlashBag()->add('reunionadmi', null); //3
        $var = $this->redirect($this->generateUrl('reunionadmision_new')); //3
    } else { //4
        if ($form->isValid()) {//5
            $em = $this->getDoctrine()->getManager(); //6
            $entity->setIdCrecimiento($this->get('session')->get('id')); //6
            $entity->setActuaDuo($this->get('session')->get('actuaDuo')); //6
            $entidad = $em->getRepository('TesisBundle:Crecimiento')
                    ->find($this->get('session')->get('id')); //6
            $entity->setCrecimiento($entidad); //6
            $entity->subirDoc(
                    $this->container->getParameter('cupon.directorio.documentos')
            ); //6

            $message = \Swift_Message::newInstance()
         ->setSubject('Entrega del acta de la Reunión de admisión')
         ->setFrom('mflores@estudiantes.uci.cu')
         ->setTo($entity->getEmail())
  ->setBody("Estimado(a) " . $entity->getResponsableActa() . " se le notifica que debe
                    ". Tiene 72 horas para la enrtega del documento a parti de hoy.")
            ->attach(Swift_Attachment::fromPath('uploads/documentos/'
                    . $entity->getUrlDoc())); //6
    $this->get('mailer')->send($message); //6
if ($entity->getPropuestaComision() == 'Aprobado') {//7
        $message = \Swift_Message::newInstance()
                ->setSubject('Entrega del acta de la Reunión de admisión')
                ->setFrom('mflores@estudiantes.uci.cu')
                ->setTo('mflores@estudiantes.uci.cu')
                ->setBody("Estimado(a) manuel gonzalez flores puede adicionar al propu
                       $entity->getNombrePropuesto() . " a las filas de militantes del
                ->attach(Swift_Attachment::fromPath('uploads/documentos/'
                        . $entity->getUrlDoc())); //7
        $this->get('mailer')->send($message); //8
    }
    $em->persist($entity); //9
    $em->flush(); //9
    $var = $this->redirect($this->generateUrl('reunionadmision_show',
            array('id' => $entity->getId()))); //9
 } else { //10
    $var = $this->render('TesisBundle:Reunionadmision:new.html.twig', array(
        'entity' => $entity,
        'form' => $form->createView(),)); //11
 }}return $var; //12
```

Figura 13. Código del algoritmo ModificarReunionAdmision

Para el cálculo de la complejidad ciclomática se representa el grafo del flujo asociado al código antes presentado a través de nodos, aristas y regiones, quedando como se muestra en la Figura 14:

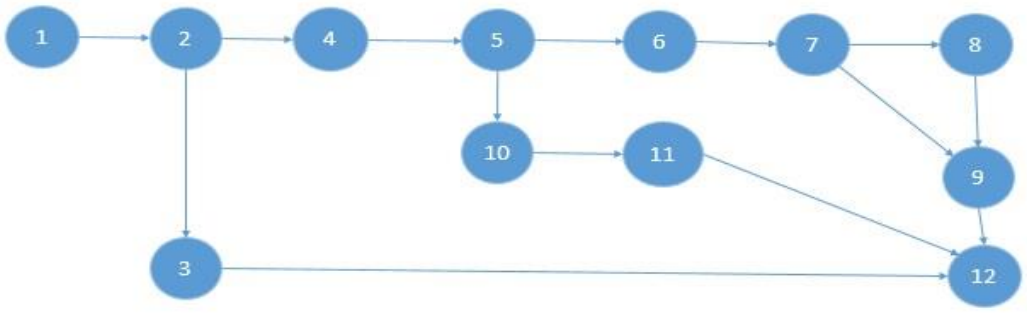


Figura 14. Grafo de flujo asociado al algoritmo

Una vez construido el grafo de flujo asociado al algoritmo anterior se calcula la complejidad, el cálculo es necesario efectuarlo mediante tres fórmulas utilizando el mismo grafo en cada caso.

- **V (G) = (A - N) + 2**

Donde **A** es la cantidad total de aristas y **N** la cantidad total de nodos.

Resultado: V (G) = (14-12) + 2

V (G) = 4

- **V (G) = P + 1**

Donde **P** es la cantidad de nodos predicados[14]

Resultado: V (G) = 3 + 1

V (G) = 4

- **V (G) = R**

Donde **R** es la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

Resultado: V (G) = 4

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática es de 4, lo que significa que existen 4 posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de prueba para el procedimiento tratado. Seguidamente se especifican los caminos básicos que puede tomar el algoritmo durante su ejecución.

Camino básico #1: 1, 2, 3, 12

Camino básico #2: 1, 2, 4, 5, 10, 11, 12

Camino básico #3: 1, 2, 4, 5, 6, 7, 9, 12

[14] Nodos predicados: nodos de los cuales parten dos o más aristas.

Camino básico #4: 1, 2, 4, 5, 6, 7, 8, 9, 12

Luego de obtenidos los caminos básicos, se realizan los casos de prueba para cada uno de ellos. La Tabla 22 muestra el caso de prueba generado para uno de los caminos de ejecución, específicamente para el camino 1, 2, 4, 5, 6, 7, 8, 9, 12. Los restantes pueden ser consultados en el expediente de proyecto Artefactos con el nombre CasosDePrueba.doc.

Tabla 22. Caso de prueba para el camino básico #4

Camino básico #1: 4: 1, 2, 4, 5, 6, 7, 8, 9, 12	
Descripción	A partir de la entrada de los datos de la reunión de admisión, se verifican si estos son válidos para posteriormente adicionarlos.
Condición de ejecución	if ($ form -> isValid()) if($ entity->getPropuestaComision() == 'Aprobado')
Entrada	Fecha de realización, cantidad de presentes, total de involucrados, organismo superior, cargo, núcleo del partido, propuesta de la comisión, nombre del propuesto, responsable del acta, email y el documento adjuntado.
Resultado esperado	Se asocia la reunión de admisión al crecimiento que pertenece. Se envía un correo notificando que ya se puede adicionar al propuesto como militante y se inserta la reunión de admisión correctamente.

Luego de haber ejecutado el método de Camino básico a las funcionalidades más complejas en el desarrollo del sistema, se pudo comprobar que el flujo de trabajo de las funcionalidades es correcto, pues se probó que cada sentencia es ejecutada al menos una vez con los parámetros de entrada y los resultados esperados, que se ejerciten todas las decisiones lógicas en las vertientes verdaderas y falsas, así como las estructuras internas de datos para asegurar su validez.

3.3.2 Pruebas de caja negra

La técnica Partición de equivalencia fue ejecutada por los revisores del grupo de calidad del centro CEIGE, en dos iteraciones, detectando un conjunto de no conformidades se clasifican en:

- No conformidades detectadas en la documentación.
- No conformidades detectadas en la aplicación.

En la Tabla 23 se muestra la cantidad de no conformidades encontradas en la documentación y la aplicación en las iteraciones realizadas, que fueron resueltas en su totalidad para que el sistema quedara libre de errores, obteniéndose el Acta de liberación del producto.

Tabla 23. No conformidades detectadas

Iteración	Documentación	Aplicación
1	**1**	**7**
2	**0**	**0**

3.3.3 Pruebas de aceptación

Las pruebas de aceptación son una herramienta para validar que el sistema cumple con las funcionalidades requeridas y esperadas por el cliente. A continuación se muestran en la Tabla 24 y Tabla 25 dos de los casos de prueba de aceptación (CPA) aplicados para validar el sistema a partir de las HU descritas por el cliente, los restantes pueden ser consultados en el expediente de proyecto Artefactos con el nombre CPA.doc:

Tabla 24. Descripción del caso de prueba de aceptación Adicionar militante

Caso de prueba de aceptación		
Código: CP_17	**Historia de usuario (Nro. y nombre):** HU #4 Gestionar militante	
Nombre de la funcionalidad: Adicionar militante		
Condiciones de ejecución: El usuario autenticado en el sistema debe tener los permisos para realizar esta funcionalidad.		
Acción	**Datos de entrada**	**Resultados esperados**
Se selecciona en el panel superior de la página de inicio del sistema la opción “Militante”		El sistema debe mostrar un listado con todos los militantes que han sido adicionados hasta el momento.
Se selecciona la opción “Adicionar militante”		El sistema debe mostrar un formulario para introducir los datos del nuevo militante.
Se introducen los datos para adicionar un nuevo militante.		
Se guardan los datos introducidos en el formulario.	El nuevo militante adicionado.	El sistema muestra nuevamente el listado de militante incluyendo el último que se acaba de adicionar.

Resultado de la prueba: Satisfactoria

Tabla 25. Descripción del caso de prueba de aceptación Modificar militante

Caso de prueba de aceptación		
Código: CP_18	**Historia de usuario (Nro. y nombre):** HU #4 Gestionar militante	
Nombre de la funcionalidad: Modificar militante		
Condiciones de ejecución: El usuario autenticado en el sistema debe tener los permisos para realizar esta funcionalidad.		
Acción	**Datos de entrada**	**Resultados esperados**
Se selecciona en el panel superior de la página de inicio del sistema la opción “Militante ”		El sistema debe mostrar un listado con todos los militantes que han sido adicionados hasta el momento.
Se selecciona el militante que se desea modificar, mostrándose los datos del mismo.		Se muestran los datos del militante.
Se selecciona la opción “Modificar”		El sistema debe mostrar un formulario con los datos actuales para modificarlos con los nuevos valores.
Se introducen los nuevos datos de los campos que se desean modificar.	Los datos de los campos que se modifican.	
Se guardan los datos modificados.		Se muestra el listado de militantes incluyendo el último militante que ha sido modificado.
Resultado de la prueba: Satisfactoria		

Al sistema se le aplicaron las pruebas de aceptación en tres iteraciones obteniendo en las dos primeras no conformidades de aplicación y de documentación. Finalmente en la tercera el sistema quedó libre de no conformidades. En la Tabla 26 se muestra la cantidad de no conformidades obtenidas en cada iteración.

Tabla 26. No conformidades detectadas

Iteración	Documentación	Aplicación
1	6	15
2	3	5
3	0	0

Terminadas estas pruebas y corregidas las no conformidades el cliente Eliober Cleger Despaigne, actualmente Secretario general del PCC en la Facultad 3 avaló la solución con la entrega del Certificado del Producto (Ver Anexo 3). Con estas pruebas se pudo determinar que el sistema realiza lo que el cliente necesita.

3.3.4 Validación de la investigación

Dado que no existe actualmente un sistema para la gestión de la información de los procesos del PCC en la Facultad 3, no se tienen datos históricos que permitan realizar una comparación entre los resultados obtenidos antes del desarrollo de la propuesta de solución y luego de desarrollada la misma. Por ello se decide aplicar el método Delphi con el objetivo de seleccionar un grupo de expertos a fin de validar el objetivo de la investigación además que se desea mantener la heterogeneidad de los participantes con el propósito de asegurar la validez de los resultados.

El método se llevó a cabo en dos rondas y en ambas se contó con la participación de 12 militantes pertenecientes a la Facultad 3, con al menos 2 años de experiencia en la organización.

En la primera ronda se aplicó una encuesta totalmente abierta, con el propósito de que cada uno tributara los criterios que debían considerarse para seleccionar a un militante como experto en la temática estudiada. El listado inicial de criterios se conformó eliminando las repeticiones o similitudes.

La segunda ronda se realizó con el objetivo de buscar un consenso en las propuestas enunciadas en la primera ronda. Al concluir esta ronda, se conformó la matriz de criterios definitiva con nivel de concordancia (NC) superior al 60%. De esta etapa quedaron como criterios para seleccionar a los expertos los siguientes:

- Tener dos años o más años de experiencia en la organización.
- Poseer conocimientos sobre los procesos del PCC.
- Haber desempeñado algún cargo en la dirección del PCC.
- Tener dominio de los reglamentos y estatutos.
- Haber realizado 2 o más crecimientos.

Finalmente fueron seleccionados 5 expertos los cuales se muestran en la Tabla 27. A estos se les aplicó la encuesta para validar la solución (Ver Anexo 2) con el objetivo de caracterizar la gestión de la información de los procesos del PCC en la actualidad y después de desarrollada la propuesta de solución, a partir de los indicadores: actualización, homologación y unificación.

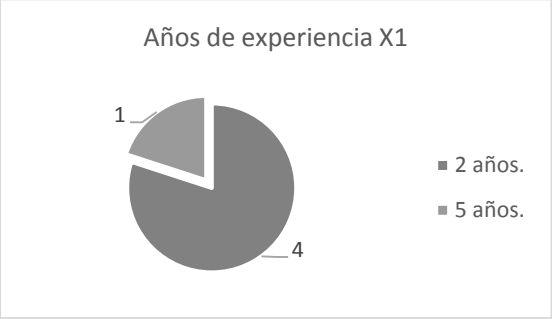
Tabla 27. Expertos involucrados en la validación de la encuesta

Nombre y apellidos
Eliober Cleger Despaigne
Darián González Ochoa
Yoansy López Reyes
Yenier Figueroa Machado
Gaspar Melchor González Font

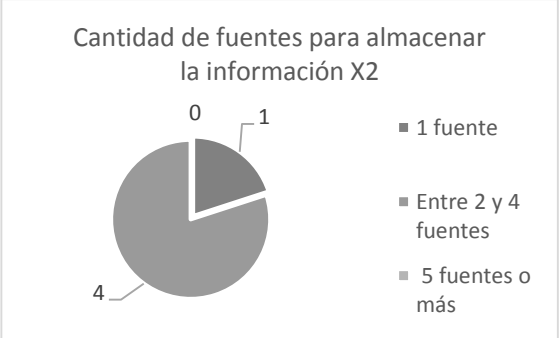
Análisis de los resultados

A continuación en la se muestran los resultados del análisis, apoyado de gráficos de pastel y de barra aplicado a las 3 preguntas realizadas en la encuesta para caracterizar la gestión de la información a partir de los indicadores: actualización, homologación y unificación de la información.

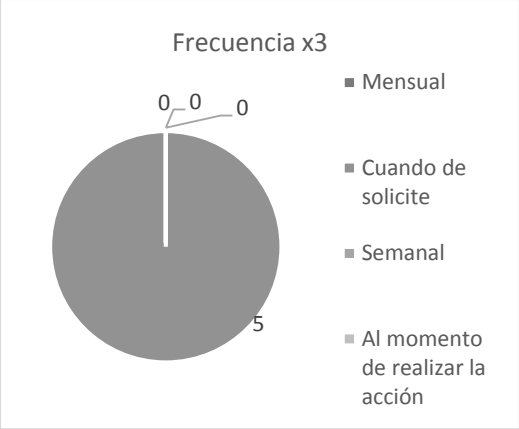
Antes:



La varible X1 refleja los años de experiencia que tienen los dirigentes del PCC. Estos se encuentran entre 2 y 5 años y se distribuyen de la siguiente mandera: 4 encuestados tienen 2 años de experiencia y 1 uno tiene 5 años de experiencia. El promedio de experiencia es de 2,75.

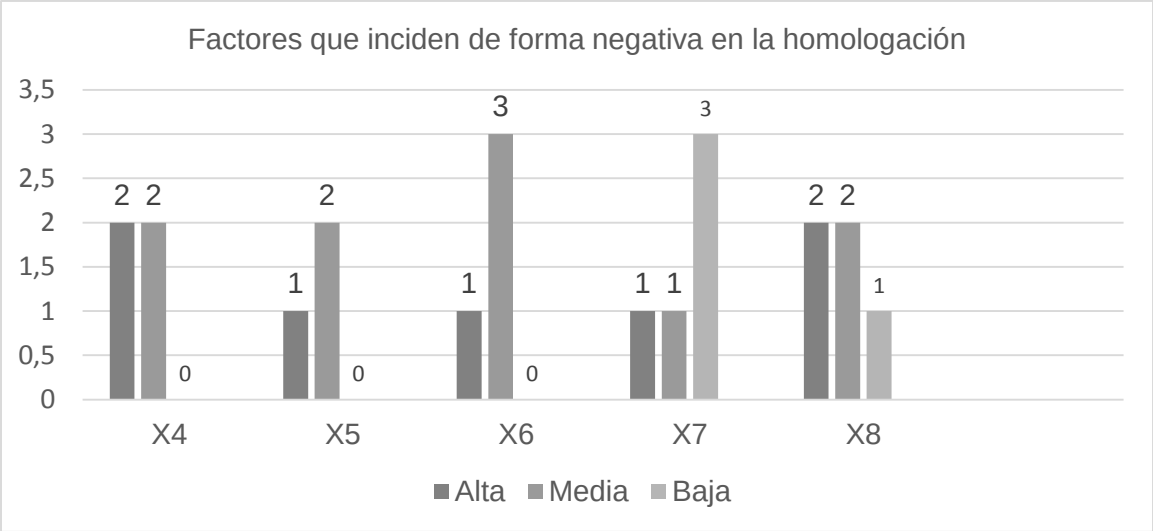


En esta pregunta, 4 encuestados afirman que almacenan la información entre 2 y cuatro fuentes y 1 persona la almacena en solo una. A partir de las respuesta de los encustados se puede concluir que la información de la misma no se encuentra centralizada en una única fuente.

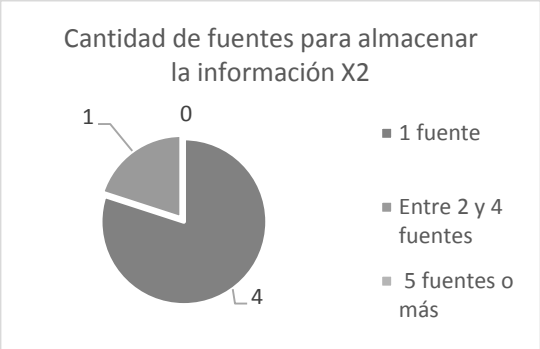


En esta pregunta todos los encuestados plantean que la información se actualiza solamente cuando se les solicita. Lo que trae consigo que la nueva información no estará registrada y actualizada hasta que se solicite nuevamente.

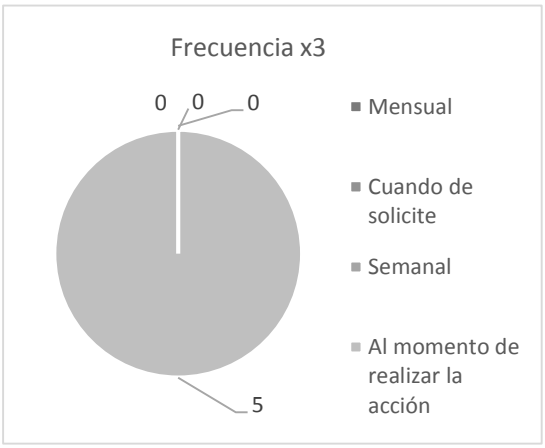
En la siguiente gráfica se muestran los factores que imposibilitan que la información que se maneja en los núcleos del PCC no coincida con la de los niveles superiores. Los encuestados plantean que las variables (desde X4 hasta X8) están influyendo de forma negativa en la correcta homologación. Las variables que influyen son: X4 (Los canales que se utilizan para el flujo de información), X5 (Formatos y plantillas en que solicita la información), X6 (Cantidad de involucrados en la gestión de información) y X7 (Errores humanos por no establecer estándares seguros para los nomencladores que se manejan). La variable X8 (no se notifica) no fue seleccionada por ningún encuestado.



Después:

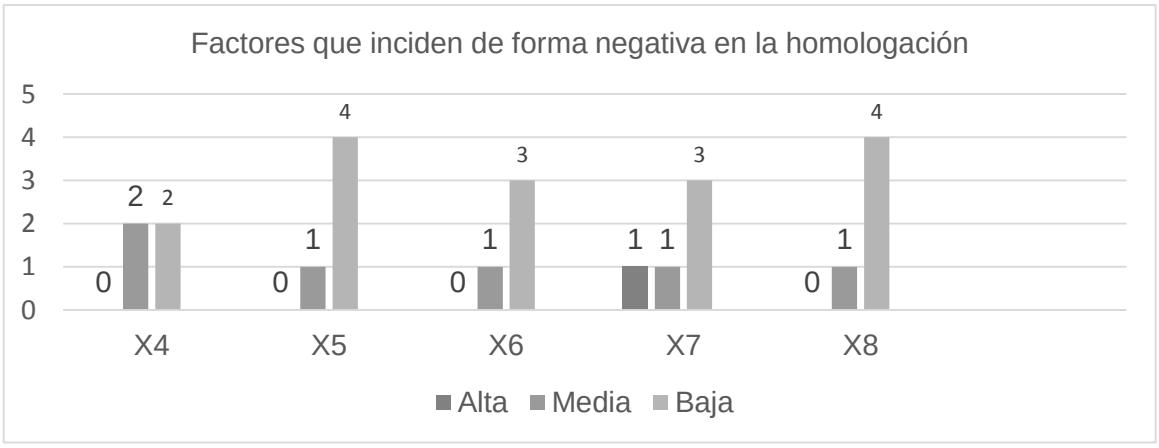


En esta pregunta, 4 encuestados afirman que con la solución se almacena la información en una fuente (base de datos del sistema). Mientras que 1 de los encuestados la almacena entre 2 y 4 fuentes. A partir de las respuesta de los encuestados se puede concluir que la solución facilita la unificación de la información.



En esta pregunta todos los encuestados plantean que con la solución la información se actualiza la momento de realizar la acción. A partir de las respuesta de los encuestados se puede concluir que la solución facilita la actualización de la información.

En la siguiente gráfica se muestran los factores que imposibilitan que la información que se maneja en los núcleos del PCC no coincida con la de los niveles superiores. Los encuestados plantean que las variables (desde X4 hasta X8) están influyendo de forma negativa en la correcta homologación. Las variables que influyen son: X4 (Los canales que se utilizan para el flujo de información), X5 (Formatos y plantillas en que solicita la información), X6 (Cantidad de involucrados en la gestión de información) y X7 (Errores humanos por no establecer estándares seguros para los nomencladores que se manejan). La variable X8 (no se notifica) no fue seleccionada por ningún encuestado.



A partir de los resultados obtenidos en la encuesta realizada antes y después del desarrollo de la solución se logró validar que la solución dio cumplimiento a los objetivos planteados en la investigación facilitando la unificación, actualización y homologación de la información en cada uno de los niveles donde se utiliza.

3.4 Conclusiones parciales

Según lo expuesto en el capítulo se arriba a las siguientes conclusiones:

- Establecer los estándares de codificación a utilizar permitió entender y analizar el código, durante el desarrollo de la propuesta de solución.
- Definir el manejo de la seguridad del sistema contribuyó a controlar el acceso de cada usuario al sistema.
- Realizar las pruebas de caja blanca, caja negra y aceptación permitió determinar que tanto el código como las funcionalidades del sistema cumplen con las necesidades del cliente.
- Con la validación de la solución se determina que el desarrollo de la propuesta de solución contribuye a una mejor unificación, homologación y actualización de la información.

Conclusiones generales

Con la realización y culminación del presente trabajo, se llegaron a las siguientes conclusiones:

- Luego de realizar un análisis de sistemas de gestión de información a nivel nacional e internacional, se determinó que no era factible utilizar ninguno de los sistemas estudiados debido a que no se ajustaban a las características requeridas. Por ello se evidenció la necesidad de desarrollar un sistema que incluyera las funcionalidades y procesos que se realizan en el funcionamiento del PCC en la Facultad 3.
- El estudio de las herramientas, tecnologías y metodologías permitió determinar cuáles se ajustaban a las características del sistema a desarrollar.
- El diseño permitió obtener una vista de la estructura estática del sistema teniendo en cuenta distintos patrones de diseño como el bajo acoplamiento, experto, creador y la alta cohesión. También se contribuyó a aumentar la escalabilidad del producto mediante el uso de estilos arquitectónicos como el MVC, lo cual propició la alta reutilización de las clases diseñadas.
- La fase de implementación permitió desarrollar un sistema a partir del diseño elaborado, cumpliendo con todos las HU descritas por el cliente. El uso de estándares de codificación contribuyó a la legibilidad del código y la comunicación fluida entre los programadores, lo cual permitirá proveer una guía para futuros mantenimientos del sistema.
- El proceso de prueba permite garantizar una mayor calidad del sistema desarrollado y el cumplimiento de los objetivos de la investigación.

Recomendaciones

Se recomienda:

- Sistematizar el uso de la solución en el PCC de la Facultad 3 e incorporarle mejoras que surjan como resultado de la explotación de la misma.
- Desplegar el sistema en otras facultades de la Universidad.

Bibliografía

2015. [En línea] 2015. [Citado el: 7 de enero de 2015.] http://pyme.lavoztx.com/qu-es-un-sistema-de-gestin-de-la-informacin-7690.html.

2015. [En línea] 2015. [Citado el: 7 de febrero de 2015.] http://www.wmo.int/pages/themes/wis/index_es.html.

2015. [En línea] 2015. [Citado el: 7 de febrero de 2015.] http://www.wmo.int/pages/themes/wis/index_es.html.

2015. about en español. [En línea] 2015. [Citado el: 2015 de febrero de 6.] http://aprenderinternet.about.com/od/Glosario/g/Que-Es-Html-5.htm.

Alfresco. 2015. Alfresco. [En línea] 2015. [Citado el: 6 de febrero de 2015.] http://www.athento.com/caracteristicas-tecnicas-athento-eadministracion/.

Almeira, Adriana Sandra y Perez Cavenago, Vanina. 2007. *Arquitectura de software: Astilos y patrones.* Argentina : s.n., 2007.

Andalucía. 2013. Marco de desarrollo de la Junta de 2013. *Junta de Andalucía.* 2013.

APACHE. 2015. What is the Apache HTTP Server Project? [En línea] 2015. [Citado el: 20 de enero de 2015.] http://whatis.techtarget.com/definition/Apache-HTTP-server-project.

2015. aprenderaprogramar.com. [En línea] 2015. [Citado el: 6 de febrero de 2015.] http://www.aprenderaprogramar.com/index.php?option=com_content&id=546:que-es-y-para-que-sirve-el-lenguaje-css-cascading-style-sheets-hojas-de-estilo&Itemid=163.

Architectural Styles. 2011. Architectural Styles. 2011.

Aspel. 2004. Aspel. [En línea] 2004. [Citado el: 6 de febrero de 2015.] http://www.aspel.com.mx/mx/homsistas.exe?id_menu=1&id_opcion=2&nopest=0&idsa=.

Beck, Kent. 1999. *Extreme Programming Explained.* 1999. 0201616416..

—. 1999. *Extreme Programming Explained.* 1999. ISSN: 0201616416.

—. 2002. *Una explicación de la Programación extrema: aceptar el cambio.* 2002. 8478290559..

Beck, Kent y Flower, Martin. 2002. *Planning Extreme Programming.* 2002. 0201710919.

Booch, Grady. 2002. *The Unified Modeling Language User Guide.* s.l. : Pearson Education, 2002.

BPMN. 2015. BPMN. [En línea] 2015. [Citado el: 20 de febrero de 2015.] http://www.bizagi.com/esp/descargas/BPMNbyExample.pdf.

BURBECK, STEVE. 2015. How to use Model-View-Controller. [En línea] 2015. [Citado el: 18 de febrero de 2015.] http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.

Calabria, Luis y Píriz, Pablo. 2003. Metodología XP. *Metodología XP.* Uruguay : s.n., 2003.

Cavenago, Adriana Sandra Almeira y Vanina Perez. 2007. Arquitectura de Software: Estilos y Patrones. Argentina : s.n., 2007.

CEGEL. 2014. Sistema de gestion fiscal.Gespliegue curso de capacitación. La Habana : s.n., 2014.

Cochran, David. 2012. *Twitter Bootstrap Web Development How-To.* 2012. ISBN: 978-1849518826.

Congreso. 2015. Con todos y para el bien de todos. Congreso del PCC. [En línea] 2015. [Citado el: 7 de febrero de 2015.] http://congresopcc.cip.cu/%C2%BFque-es-el-pcc.

Davis, Olson y. 1985. *Management Information Systems: Conceptual foundations, Structure and Development.* 1985.

Despaigne, Eliober Cleger. 2015. Habana, 25 de febrero de 2015.

Eguiluz, J. 2013. Desarrollo web ágil con Symfony2. 2013.

Encinosa, Lazaro J Blanco. 2011. *La informática en la dirección de empresas.* Habana : s.n., 2011. ISBN 978-959-07-1629-4.

Escalona, Maria Jose. 2002. Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo. Sevilla : s.n., 2002.

Escribano, Gerardo Fernandez. 2002. Introduccion a Extreme Programming. 2002.

Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. 2010. *Metodologías tradicionales vs. metodologías ágiles.* 2010.

Flower, Martin y Scot, Kendall. 1999. *UML gota a gota.* s.l. : Pearson Education, 1999.

Fornaris, Maite Sánchez y Rabí, Dayana Alcantara. 2010. *Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica.* 2010.

Gamma, Erich. 1994. *Design Patterns.* 1994. (ISBN 0-201-63361-2.

Gamma, Helm y. 2003. Patrones de Diseño. Elementos de software orientado a objetos. Madrid : Pearson Educación, 2003.

Gonzalez, Victor. 2014. dataFeu. *Tesis de diploma.* La Habana : s.n., 2014.

HTML. 2012. W3C. [En línea] 17 de diciembre de 2012. [Citado el: 6 de febrero de 2015.] http://www.w3c.es/Prensa/2012/nota20121217_html5.

INTECO. 2009. INGENIERÍA DEL SOFTWARE:METODOLOGÍAS Y CICLOS DE VIDA. *Artículo.* España : s.n., 2009. ISBN: 9788478290963.

JavaScript. 2015. Aprender a programar. *¿Qué es y para qué sirve JavaScript? .* [En línea] 2015. http://aprenderaprogramar.com.

Joskowicz, Ing. José. 2008. Reglas y Prácticas en eXtreme Programming. España : s.n., 2008.

Juristo, Ana M. Moreno, Sira Vegas. 2005. *Técnicas de evaluación de software.* 2005.

Larman, Craig. 2003. *UML y patrones. Una introducción al análisis y diseño a objetos y el proceso unificado.2da Edicion.* 2003.

LibrosWeb. 2015. [En línea] 2015. [Citado el: 5 de febrero de 2015.] http://librosweb.es/symfony/capitulo_1/symfony_en_pocas_palabras.html.

Mendes Calo, Karla, Estevez, Elsa y Fillottrani, Pablo. 2009. Un Framework para Evaluación de Metodologías Ágiles. *Un Framework para Evaluación de Metodologías Ágiles.* Argentina : s.n., 2009.

Nuxeo. 2015. athento. *Nuxeo DM: Características.* [En línea] 2015. [Citado el: 6 de febrero de 2015.] http://www.athento.com/nuxeo/caracteristicas/.

Oracle. 2015. NetBeans. [En línea] 2015. [Citado el: 5 de Febrero de 2015.] https://netbeans.org/index_es.html.

Osorio, Yeilyn Abad. 2007. Análisis y diseño de una aplicación informática. *Tesis de doctorado.* Habana : s.n., 2007.

Padilla, Ariday Ballesteros. 2008. Importancia de la gestión de la información y el conocimiento en el proceso de cambio organizacional. *Seminario evaluativo.* 2008.

PCC. 2015. Partido Comunista de Cuba. [En línea] 8 de Abril de 2015. [Citado el: 26 de marzo de 2015.] http://www.pcc.cu/i_historia3.php.

PHP. 2015. What is PHP? [En línea] 2015. [Citado el: 4 de Febrero de 2015.] http://php.net.

Piattini, Mario. 2007. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* Madrid : RA-MA, 2007. 9788478977765..

—. 1996. Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión. 1996.

2006. Political Database of de Americas. [En línea] 2006. [Citado el: 4 de Febrero de 2015.] http://pdba.georgetown.edu/Constitutions/Cuba/cuba1976.html#mozTocId637562.

Ponjuán, G. 2000. *Aplicaciones de Gestión de información en las organizaciones. El profecional de la informacion y su dominio de las tecnicas y herramientas de la gestion.* La Habana : s.n., 2000.

—. 2000. *Aplicaciones de Gestión de información en las organizaciones. El profecional de la informacion y su dominio de las tecnicas y herramientas de la gestion.* La Habana : s.n., 2000.

PostgreSQL. 2015. PostgreSQL. [En línea] 2015. [Citado el: 6 de febrero de 2015.] http://www.postgresql.org/about/.

Pressman, Roger. 2002. *Ingeniería de sofware, un enfoque practico. 5ta Edición.* 2002. 8448132149.

Raghavan, S., Zelesnik, G. y Ford, G. 1994. Lecture Notes on Requirements Elicitation. s.l. : Educational Materials CMU/SEI–94–EM–10, 1994.

Ramírez, Ing.Danay Pérez. 2002. METODOLOGÍAS ÁGILES.¿CÓMO DESARROLLO ? Habana : s.n., 2002.

Real Academia Española. 2015. Real Academia Española. [En línea] 2015. http://lema.rae.es/drae.

Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. 2002. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. 2002.

Scrum. 2015. proyectos agiles. *cómo gestionar proyectos con Scrum.* [En línea] 2015. http://www.proyectosagiles.org/que-es-scrum.

Software, Unidad Docente de Ingeniería del. 2015. Patrones del Gang of Four. Madrid : s.n., 2015.

Somerville, Ian. 2005. *Ingienería del software.* 2005. ISBN:84-7829-074-5.

Stevens, Perdita y Pooley, Rob. 2002. *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* s.l. : Addison-Wesley Publishing Company, 2002.

Suárez, Armijos y Naufredo, Javier. 2012. *Estudio de las características, funcionamiento, ventajas y técnicas .* Universidad Politécnica Salesiana : s.n., 2012.

TCP. 2012. Kioskea.net. [En línea] 2012. [Citado el: 11 de abril de 2015.] http://es.kioskea.net/contents/281-protocolo-tcp.

Tecnología. 2015. Tecnología. *Lenguaje de programación.* [En línea] 2015. [Citado el: 19 de Mayo de 2015.] http://www.areatecnologia.com/informatica/lenguajes-de-programacion.html.

The Apache Software Foundation. [En línea] [Citado el: 10 de Diciembre de 2014.] http://www.apache.org/licenses/..

The Apache Software Foundation. 2015. Configuration Files. [En línea] 2015. [Citado el: 10 de Diciembre de 2014.] http://httpd.apache.org/docs/current/configuring.html.

Toro, Amador Durán. 2000. Metodología para la Elicitación de Requisitos de . Sevilla : s.n., 2000.

Torre, Aníbal de la. 2006. Lenguajes del lado servidor o cliente. *adelat.* [En línea] 2006. [Citado el: 20 de mayo de 2015.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

UML. 2013. UML. [En línea] 12 de abril de 2013. [Citado el: 11 de abril de 2015.] http://umldiagramadespliegue.blogspot.com/.

—. 2015. UML. [En línea] 2015. [Citado el: 8 de febrero de 2015.] http://es.scribd.com/doc/2080534/UML#scribd.

Visual Paradigm. Round-trip Code Engineering. [En línea] [Citado el: 24 de Noviembre de 2014.] http://www.visualparadigm.com/product/vpuml/features/roundtripcodeengine.jsp#cpproundtrip.

—. 2010. Visual Paradigm for UML 8.0 Released. [En línea] 16 de Agosto de 2010. [Citado el: 24 de Noviembre de 2014.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

WIELENGA, GEERTJAN. 2015. Top 10 New Features in NetBeans IDE 7.3. [En línea] 3 de febrero de 2015. [Citado el: 2015 de febrero de 5.] https://blogs.oracle.com/geertjan/entry/top_10_new_features_in..

XP. 2015. Procesos de software. *Metodología XP.* [En línea] 2015. [Citado el: 16 de junio de 2015.] http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

Anexos

Anexo 1

Entrevista:

La siguiente entrevista tiene como objetivo conocer el funcionamiento de la gestión de la información de los procesos del PCC en la Facultad 3. Por ello le pedimos su contribución y que sea lo más sincero posible en sus planteamientos. Le garantizamos confidencialidad y anonimato. Gracias de antemano por su colaboración.

Entrevista a Eliober Cleger Despaigne (Secretario general del PCC):

¿Cuáles son los procesos que lleva a cabo el Partido Comunista de Cuba?

¿Cuáles son los pasos para realizar el proceso de crecimiento?

¿Dónde se almacena la información de cada militante?

¿Cuál es el formato y la estructura de las actas?

¿Qué tipo de reuniones realizan?

¿En cada una de estas reuniones las actas tienes igual o diferente estructura?

¿Cómo realizan las consultas de las actas para analizar los acuerdos, las inquietudes como también la cantidad de militantes ausentes y presentes?

¿De qué forma informan a sus militantes el día que se realizaran las reuniones, actividades convocadas, información de interés referente a condiciones para ingresar al PCC, Estatutos, Reglamentos, etc.?

Entrevista a Gaspar Melchor González Font (Activista de cotización):

¿Cómo realizan el proceso de la cotización?

¿Qué datos necesitan para calcular la cuota de la cotización?

¿De dónde obtienen datos para el cálculo de la cuota de la cotización?

¿Cómo acceden a las nóminas para conocer el salario de cada militante y determinar cuánto tienen que pagar en cada caso?

¿Qué tiempo invierte en el proceso de calcular la cuota de cotización?

Anexo 2

Encuesta aplicada:

La siguiente encuesta tiene como objetivo caracterizar la gestión de la información de los procesos del PCC en la Facultad 3. Por ello le pedimos su contribución y que sea lo más sincero posible en sus planteamientos. Le garantizamos confidencialidad y anonimato.

Gracias de antemano por su colaboración.

Datos de interés:

Tiempo como dirigente del PCC: ___ años. Cargo_____

1. Indique cuántas **fuentes** usted utiliza para gestionar la información de los procesos del PCC.

Antes (sin usar herramienta informática)	Después (con el uso de la solución)
( ) 1 fuente. Especifíquela: _____	( ) 1 fuente. Especifíquela: _____
( ) Entre 2 y 4 fuentes	( ) Entre 2 y 4 fuentes
( ) 5 fuentes o más. Especifique cuantas: ___	( ) 5 fuentes o más. Especifique cuantas: ___

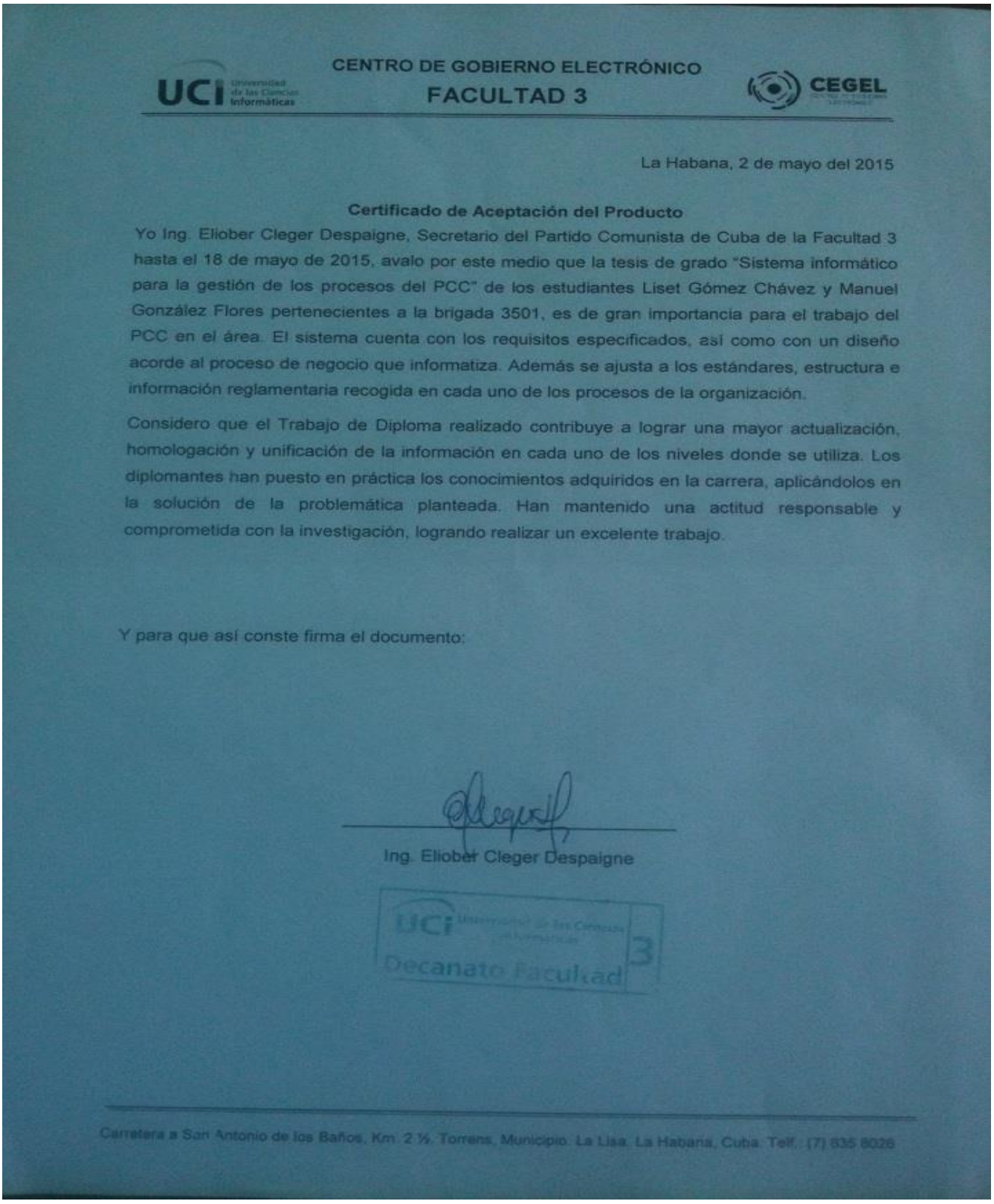
2. Indique con qué frecuencia se **actualiza** la información de los procesos del PCC.

Antes (sin usar herramienta informática)	Después (con el uso de la solución)
( ) Al momento de realizar la acción	( ) Al momento de realizar la acción
( ) Semanal	( ) Semanal
( ) Mensual	( ) Mensual
( ) Cuando la soliciten	( ) Cuando la soliciten

3. Indique cuáles de los siguientes factores considera que influyen en la **homologación** de la información y en qué medida.

Antes (sin usar herramienta informática)	Después (con el uso de la solución)
( ) Los canales para el flujo de la información: Alta___ Media___ Baja___	( ) Los canales para el flujo de la información: Alta___ Media___ Baja___
( ) Formatos y plantillas en que se solicita la información: Alta___ Media___ Baja___	( ) Formatos y plantillas en que se solicita la información: Alta___ Media___ Baja___
( ) Cantidad de involucrados en la gestión de la información: Alta___ Media___ Baja___	( ) Cantidad de involucrados en la gestión de la información: Alta___ Media___ Baja___
( ) Errores humanos por no establecer estándares seguros para los nomencladores que se manejan: Alta___ Media___ Baja___	( ) Errores humanos por no establecer estándares seguros para los nomencladores que se manejan: Alta___ Media___ Baja___
( ) No se notifica a los involucrados o afectados por los cambios en la información: Alta___ Media___ Baja___	( ) No se notifica a los involucrados o afectados por los cambios en la información: Alta___ Media___ Baja___

Anexo 3



CENTRO DE GOBIERNO ELECTRÓNICO
FACULTAD 3

La Habana, 2 de mayo del 2015

Certificado de Aceptación del Producto

Yo Ing. Eliober Cleger Despaigne, Secretario del Partido Comunista de Cuba de la Facultad 3 hasta el 18 de mayo de 2015, avalo por este medio que la tesis de grado "Sistema informático para la gestión de los procesos del PCC" de los estudiantes Liset Gómez Chávez y Manuel González Flores pertenecientes a la brigada 3501, es de gran importancia para el trabajo del PCC en el área. El sistema cuenta con los requisitos especificados, así como con un diseño acorde al proceso de negocio que informatiza. Además se ajusta a los estándares, estructura e información reglamentaria recogida en cada uno de los procesos de la organización.

Considero que el Trabajo de Diploma realizado contribuye a lograr una mayor actualización, homologación y unificación de la información en cada uno de los niveles donde se utiliza. Los diplomantes han puesto en práctica los conocimientos adquiridos en la carrera, aplicándolos en la solución de la problemática planteada. Han mantenido una actitud responsable y comprometida con la investigación, logrando realizar un excelente trabajo.

Y para que así conste firma el documento:

Ing. Eliober Cleger Despaigne

Decanato Facultad 3

Carretera a San Antonio de los Baños, Km. 2 ½, Torrens, Municipio. La Lisa. La Habana, Cuba. Telf.: (7) 835 8026