Universidad de las Ciencias Informáticas

Facultad 3

Desarrollo de una aplicación informática para el proceso de gestión de las comisiones disciplinarias en la Universidad de las Ciencias Informáticas.

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

Autores:

Yiselly Azaharez Matos

Diana Rosa Martínez García

Tutores:

Ing. Liset González Polanco

Ing. Lazaro Heredia Maso

Ciudad de La Habana

Junio 2015

“Lo fundamental es que seamos capaces de hacer cada día algo que perfeccione lo que hicimos el día anterior.”

Declaración de autoría

Se declara que Yiselly Azaharez Matos y Diana Rosa Martínez García son los únicos autores del trabajo de diploma “Desarrollo de una aplicación informática para el proceso de gestión de las comisiones disciplinarias en la Universidad de las Ciencias Informáticas” y se le reconocen a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año 2015.

Firma del autor	**Firma del autor**
Yiselly Azaharez Matos	Diana Rosa Martínez García
Firma del Tutor	**Firma del Tutor**
Liset González Polanco	Lazaro Heredia Manso

Datos del contacto

Autor: Yiselly Azaharez matos

Correo: yazaharez@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Autor: Diana Rosa Martínez García

Correo: drmartinez@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Tutor: Ing. Liset González Polanco

Correo: lgpolanco@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Tutor: Ing. Lazaro Heredia Manso

Correo: lheredia@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Dedicatoria

Este trabajo de diploma está dedicado con todo el amor del mundo a mis padres y a mis abuelos.

Diana

Este trabajo de diploma está dedicado con todo el amor del mundo a mis padres.

Yiselly

Agradecimientos

Agradezco a mis padres por impulsarme a ser una mejor persona en la vida, a ser una profesional, a ser independiente, por creer en mí, y darme todo su amor, confianza, cariño y dedicación. A mi hermana por darme su cariño incondicional y a mi sobrina linda. A mi tutor Lazaro y a mi compañera de tesis porque sin su ayuda no hubiese sido posible este gran sueño. A mis abuelos por siempre apoyarme, a mi tío Ernesto por siempre estar ahí y a mis primos Ernesto y Ana Olga. A mis grandes amigas inolvidables Greter, Dayana, Laura, Lisandra, Wendy, Yesenia y Geydi por soportarme y estar ahí en los momentos buenos y malos en todos mis años de universidad, aunque este curso sea el último que estemos juntas, y quizás sea difícil volvernos a ver, sepan que nunca las voy a olvidar. Y a mis amigos Yandry y Arian. A todos mis compañeros de aula por todos los momentos que pasamos juntos y todos aquellos que de una forma u otra contribuyeron a lograr este sueño.

Diana

Agradezco a mi novio Alexander (Tato) por acompañarme a lo largo de toda mi carrera, por estar a mi lado en todos los momentos buenos o malos, y sobre todo por darme su amor incondicional. Agradezco a mis padres por impulsarme a ser una mejor persona en la vida, a ser una profesional, a ser independiente, por creer en mí, y darme todo su confianza, cariño y dedicación. A mi hermana Ismarais y a mi hermano Vinvi por darme su cariño incondicional y cuidar de mami todo este tiempo que no estuve allá. A mi tutor Lazaro y a mi compañera de tesis porque sin su ayuda no hubiese sido posible este gran sueño. A mis grandes amigos inolvidables Idelmis y Yorguy por soportarme y estar ahí en los momentos buenos y malos en todos mis años de universidad, sepan que nunca los voy a olvidar. A todos mis compañeros de aula por todos los momentos que pasamos juntos y todos aquellos que de una forma u otra contribuyeron a lograr este sueño.

Yiselly

Resumen

El presente trabajo de diploma tiene como propósito realizar una nueva versión del sistema “Aplicación informática para el proceso de gestión de las comisiones disciplinarias en la Facultad 3” (SIPCD), para que pueda ser utilizado en las diferentes facultades y en la universidad. Actualmente dicho sistema presenta insuficiencias que dificultan la oportunidad de la información atendiendo a tres factores de evaluación: disponibilidad, tiempo e integridad, además que no contempla los procesos que se realizan fuera de la facultad como son: revisión, apelación y suspensión condicional de la medida. Se realiza un estudio de los sistemas utilizados en el mundo para estos fines. En el desarrollo de la solución propuesta, se lleva a cabo un análisis comparativo de herramientas, lenguajes y metodologías de desarrollo, las cuales se seleccionan según las necesidades propias de la aplicación, de las tecnologías disponibles y del conocimiento y experiencia del equipo de desarrollo. Se hace uso de buenas prácticas de la programación mediante los beneficios de patrones y métricas que facilitan y simplifican la implementación de la solución propuesta. Con el desarrollo de esta aplicación informática se proveerá a la Universidad de las Ciencias Informáticas (UCI) de un sistema para el proceso de las comisiones disciplinarias contribuyendo a la oportunidad de su información.

Palabras claves:

Comisión disciplinaria, UCI, oportunidad, sistema de gestión, software.

Índice de contenidos

Introducción *1*
Capítulo 1: Fundamentación teórica *5*
1.1 Introducción 5
1.2 Conceptos fundamentales 5
1.3 Descripción del proceso 6
1.4 Sistemas informáticos analizados 8
1.5 Formulación de la propuesta de solución 10
1.5.1 Propuesta de solución 10
1.5.2 Metodología de desarrollo de software 10
1.5.3 Marco de trabajo 12
1.5.4 Gestor de base de datos 13
1.5.5 Servidores web 15
1.5.6 Lenguajes utilizados 15
1.5.7 Lenguajes de Programación 16
1.5.8 Herramientas informáticas 17
1.5.9 Patrón de arquitectura 19
1.5.10 Patrones de Diseño 20
1.6 Métricas 21
1.6.1 Métricas para requisitos 21
1.6.2 Métricas para el diseño 22
1.7 Pruebas 26
1.8 Conclusiones Parciales 26
Capítulo 2: Análisis y diseño *27*
2.1 Introducción 27
2.2 Requisitos del sistema 27
2.2.1 Requisitos funcionales 28
2.2.2 Requisitos no funcionales 30
2.3 Validación de requisitos 32
2.4 Fase de exploración 33
2.4.1 Historias de usuarios 33
2.5 Fase de Planificación 35
2.5.1 Estimación de esfuerzo 36
2.5.2 Plan de entrega 39
2.5.3 Plan de iteraciones 41

2.6 Fase de diseño 42
2.6.1 Tarjetas CRC (Clase–Responsabilidad-Colaborador) 42
2.6.2 Diagrama de Clases 43
2.7 Patrones de diseño 44
2.8 Validación del diseño 46
2.9 Conclusiones del capítulo 50
Capítulo 3: Implementación y prueba. *52*
3.1 Introducción 52
3.2 Fase de desarrollo 52
3.3 Estándares de codificación 52
3.4 Diagrama de componente 55
3.5 Diagrama de despliegue 57
3.6 Modelo de datos 58
3.7 Fase de pruebas 59
3.7.1 Prueba de caja negra 59
3.8 Validación de las variables 61
3.9 Conclusiones del capítulo 63
Conclusiones generales *64*
Recomendaciones *65*
Bibliografía *66*

Introducción

La Batalla de Ideas es una acción política de la Revolución Cubana donde se desarrolla un debate de carácter ético en defensa de sus avances en la justicia social, de la integridad nacional y el internacionalismo (1). Se acompaña por un conjunto de programas que se llevan a cabo en las diferentes esferas de la sociedad y que ha contribuido a la realización de importantes cambios en la calidad de vida de los cubanos.

La Universidad de las Ciencias Informáticas (UCI), es un centro de enseñanza superior de nuevo tipo, surgido en el fragor de la Batalla de Ideas y obra del pensamiento visionario del Comandante en Jefe Fidel Castro Ruz, que tiene como misión formar profesionales competentes y comprometidos con la Revolución. La misma cuenta con un modelo del profesional, que mediante la formación de estudiantes impulsa la informatización de la sociedad cubana, a través de la producción de aplicaciones y servicios (2). La UCI también realiza esfuerzos en la informatización de sus actividades, entre los que se puede mencionar el relacionado con los temas disciplinarios, que en el caso de los estudiantes está normado por el “Decreto Ley 240 del año 2007”.

El proceso de Comisiones Disciplinarias (CD) en la UCI se realiza en tres niveles de dirección (nivel de Facultad, de Universidad y Ministerio de Educación Superior) de forma manual y con herramientas informáticas muy básicas, por ejemplo: Microsoft Excel, Microsoft Word, entre otras, provocando afectaciones en la gestión de la información. Una CD comienza con una denuncia de indisciplina que llega a la facultad, una vez aprobada genera un expediente, el cual es archivado en las secretarías docentes. Al tener que trabajar con información totalmente manuscrita y archivada físicamente en un local, se afecta la disponibilidad de la misma, las consultas futuras y la obtención y precisión de los reportes. La obtención de reportes para los análisis que se requieren por los niveles de dirección se realiza de forma manual, lo que provoca atrasos y da margen a que el resultado no sea confiable, imposibilitando conocer los datos para cumplir en gran medida con lo que se dispone en el Capítulo XIV del ya citado reglamento, que se ocupa de la rehabilitación de los sancionados y de la mitigación de riesgos de aquellos eventos que ocasionan un mayor número de indisciplinas.

Para la realización de un correcto análisis estadístico, es necesario obtener datos en el menor tiempo posible y con la precisión requerida. Los indicadores estadísticos son valores numéricos que tienden a localizar, en algún sentido, la parte central de un conjunto de datos, el término medio se asocia a estas mediciones; los principales son: Media-Aritmética, Mediana y Moda. Los mismos están sujetos a margen de errores y el tiempo que hay que dedicar para la obtención es alto. Para mejorar la gestión de la información relacionada con las comisiones disciplinarias, en la Facultad 3 se realizó

un sistema que informatiza el proceso, pero el mismo no contempla las acciones que se llevan a cabo fuera de la facultad, como son, apelación, revisión y suspensión condicional de la medida.

Por lo antes planteado surge como **problema a resolver:** ¿cómo contribuir al proceso de comisiones disciplinarias para mejorar la oportunidad de su información en la UCI?, se plantea como **objeto de estudio:** proceso de las CD en la Enseñanza Superior Cubana, teniendo como **objetivo general:** desarrollar un sistema informático para el proceso de comisiones disciplinarias en la Universidad de las Ciencias Informáticas, de manera que se contribuya a la oportunidad de su información, enmarcándose en el **campo de acción:** informatización del proceso de las comisiones disciplinarias en la Universidad de las Ciencias Informáticas. Como **idea a defender** se define que: si se informatiza el proceso de las comisiones disciplinarias en la Universidad de las Ciencias Informáticas, se contribuye a la oportunidad de su información.

Se plantean como **objetivos específicos:**

- Elaborar el marco teórico de la investigación mediante un estudio del estado del arte de los procesos relacionados con las comisiones disciplinarias.
- Desarrollar el análisis y diseño del sistema para el proceso de comisiones disciplinarias, mediante la metodología de desarrollo de software establecida.
- Implementar el código del sistema para el proceso de comisiones disciplinarias, mediante las herramientas y tecnologías seleccionadas.
- Validar la solución propuesta mediante pruebas de software y el cumplimiento de los objetivos de la investigación.

Para dar cumplimiento a los objetivos específicos se hace necesario definir como **tareas de la investigación:**

- Investigación de los sistemas enfocados a las sanciones por incumplimiento de las leyes para contribuir al estado del arte.
- Definición de la metodología de software, herramientas y tecnologías a utilizar.
- Análisis del proceso de CD en la UCI.
- Diseño del sistema de CD para la UCI.
- Implementación de las funcionalidades detectadas.
- Validación de los resultados obtenidos a través de pruebas de caja negra.

- Validación de la propuesta de solución.

Para realizar la investigación se utilizaron los siguientes métodos científicos:

Métodos teóricos:

- *Histórico-Lógico:* estudia la trayectoria real de los fenómenos y acontecimientos en el transcurso de su historia e investiga las leyes generales de su funcionamiento y desarrollo (3). Se empleó para realizar el análisis de los antecedentes del proceso de comisiones disciplinarias, a través de la documentación consultada y la evolución del mismo en la UCI.
- *Analítico-Sintético:* el análisis posibilita descomponer mentalmente un todo complejo en sus partes y cualidades; la síntesis establece la unión entre dichas partes y posibilita descubrir relaciones y características generales entre los elementos de la realidad (4). Se usó para realizar el análisis y síntesis de toda la documentación, tendencias mundiales de los sistemas informáticos de gestión de la información en el sector educativo existentes, metodologías, herramientas y lenguajes.
- *Modelación:* el modelo es un instrumento de la investigación de carácter material o teórico para reproducir el fenómeno que se está estudiando, es una reproducción simplificada de la realidad (4). Posibilitó la obtención de los principales artefactos de cada una de las fases de desarrollo del sistema, por ejemplo: diagrama de clases (lógico y físico), despliegue y componentes.

Método Empírico:

- *Entrevista:* es una conversación planificada entre el investigador y el entrevistado para obtener información (4). Permitió analizar las necesidades del cliente y obtener información acerca de los requisitos en el desarrollo del software, mediante encuentros realizados con los involucrados en el proceso de CD en la UCI.

El presente trabajo de diploma está compuesto por tres capítulos estructurados como se muestra a continuación:

Capítulo 1: Fundamentación Teórica: se realiza una búsqueda de los sistemas existentes, que en la actualidad contienen algún proceso de sanción en el mundo y en Cuba. Además se hace una investigación de las diferentes metodologías y herramientas que pueden ser empleadas para el desarrollo de una aplicación informática en el área de las comisiones disciplinarias.

Capítulo 2: Análisis y diseño: se especifican los distintos artefactos a generar por cada una de las etapas de la metodología de desarrollo escogida; además el diseño de la propuesta de solución y su

validación a través de distintas métricas. Se obtienen los artefactos según la metodología especificada.

Capítulo 3: Implementación y prueba: se implementa el sistema para darle solución a la problemática planteada. Se realizan las pruebas de aceptación para validar funcionalmente la propuesta de solución. Se valida que la propuesta de solución cumpla con el objetivo planteado.

Capítulo 1: Fundamentación teórica

1.1 Introducción

El presente capítulo abordará los conceptos fundamentales relacionados con el proceso de comisiones disciplinarias en la UCI, a los cuales se hará referencia en todo el desarrollo del trabajo. Se realizará un estudio de los sistemas informáticos existentes actualmente que contengan algún proceso de sanción, además de las herramientas, plataforma de desarrollo y metodología de desarrollo de software que se utilizarán para darle solución al problema a resolver durante la investigación. Se incluye una investigación acerca de los diferentes patrones empleados en el desarrollo de sistemas informáticos y las pruebas para evaluar la calidad del sistema desarrollado.

A continuación se relacionan los principales conceptos estudiados, para lograr un mayor entendimiento del proceso de gestión de las CD en la UCI.

1.2 Conceptos fundamentales

Reglamento: colección ordenada de reglas o preceptos, que por la autoridad competente en un estado o centro, se da para la ejecución de una ley o para el régimen de una corporación, una dependencia o un servicio (5).

Denuncia: acción y efecto de denunciar. Documento en que se da noticia a la autoridad competente, de la comisión de un delito o de una falta (5).

Comisión disciplinaria: conjunto de personas encargadas por una autoridad de velar por la buena conducta y disciplina (5).

Apelación: procedimiento judicial mediante el cual se solicita a un juez o tribunal que anule o enmiende la sentencia dictada por otro de inferior rango por considerarla injusta. Petición o llamada que hace una persona a otra para que la ayude en su propósito (5).

Opinión: idea, juicio o concepto que se tiene sobre alguien o algo (5).

Declaración: acción y efecto de declarar o declararse. Manifestación o explicación de lo que otro u otros dudan o ignoran. Manifestación formal que realiza una persona con efectos jurídicos, especialmente la que hacen las partes, testigos o peritos en un proceso (5).

Ley: regla y norma constante e invariable de las cosas. Precepto dictado por la autoridad competente, en que se manda o prohíbe algo en consonancia con la justicia (5).

1.3 Descripción del proceso

Una CD comienza con una denuncia de indisciplina que llega a la facultad, la cual es analizada por el Decano, que decide si procede o no, en caso de que proceda se la envía al Asesor de trabajo educativo el cual la asigna a una comisión disciplinaria. Las comisiones son creadas en la facultad en función de lo que establece el reglamento disciplinario para los estudiantes del MES, Decreto ley 240 del 2007, la misma está compuesta por: el presidente, el secretario y el vocal. Una vez asignada, la comisión debe contar con 30 días hábiles para todo el proceso de investigación; fecha que decide la Asesora de trabajo educativo ya que una denuncia puede tener a uno o varios estudiantes (solo en casos excepcionales se le otorga 10 días hábiles más, aprobados por el Decano). Cuando la comisión tiene la denuncia comienza a solicitar las declaraciones de: la persona denunciada, el denunciante, la FEU, la UJC y el tutor o profesor guía. Una vez que ya se cuente con todas esas opiniones, la comisión realiza una primera reunión donde discuten todas las declaraciones y llegan a un primer consenso con todos los implicados. Terminada la reunión, la comisión puede realizar entrevistas individuales, si no, se reúnen y tipifican la falta (faltas graves, menos graves y muy graves), o sea, verifican de acuerdo a las evidencias si el hecho que se denuncia procede. Si procede se da como un hecho probado y se propone una sanción al Decano. Luego se envía el expediente que contiene toda la documentación al Asesor de trabajo educativo. Cuando el Asesor revisa el expediente (está conformado por una denuncia, opiniones, declaraciones del denunciado y del que denuncia), si está conforme lo envía al Decano, este analiza si está de acuerdo o no con la sanción, en caso de que esté de acuerdo el expediente pasa a la secretaría docente donde se realiza el dictamen, luego la notificación de la resolución se le avisa al estudiante para que la firme y se le avisa el tiempo que tiene para apelar (10 días). Seguidamente se archiva dentro del expediente del estudiante el cual estará en dependencia de la tipificación de la falta. Los estudiantes sancionados por faltas “Graves” y “Menos Graves” tendrán derecho a apelar ante el Rector por conducto de quien le impuso la medida disciplinaria.

El proceso de apelación empieza cuando el estudiante sancionado hace entrega de un documento solicitando dicha apelación al Decano, que es el encargado de enviársela al Rector. Este último analiza dicha apelación y decide si procede o no. La apelación se tramitará en un término de 20 días hábiles posteriores a su recepción; salvo casos que por su masividad o complejidad requieran un tiempo superior. En caso de que proceda se le informa a la facultad (Decano), encargado de informar a todos los implicados los cambios realizados o no en la sanción. En los casos de faltas muy graves, el estudiante podrá solicitar procedimiento de revisión ante el Ministro de Educación Superior, a través del Rector. Una vez presentada la solicitud a través de la instancia correspondiente, este la

elevará a quien corresponda (Ministro de Educación Superior) junto con los expedientes académicos, disciplinario y demás avales que posea. Recibida la documentación, la autoridad facultada para conocer de la revisión contará con 30 días hábiles para resolverlo. En caso de que proceda se le informa al encargado y este le informará a todos los implicados.

Para mejor entendimiento véase la imagen 1 donde queda explicado el proceso de comisiones disciplinarias a través de un diagrama de proceso de negocio.

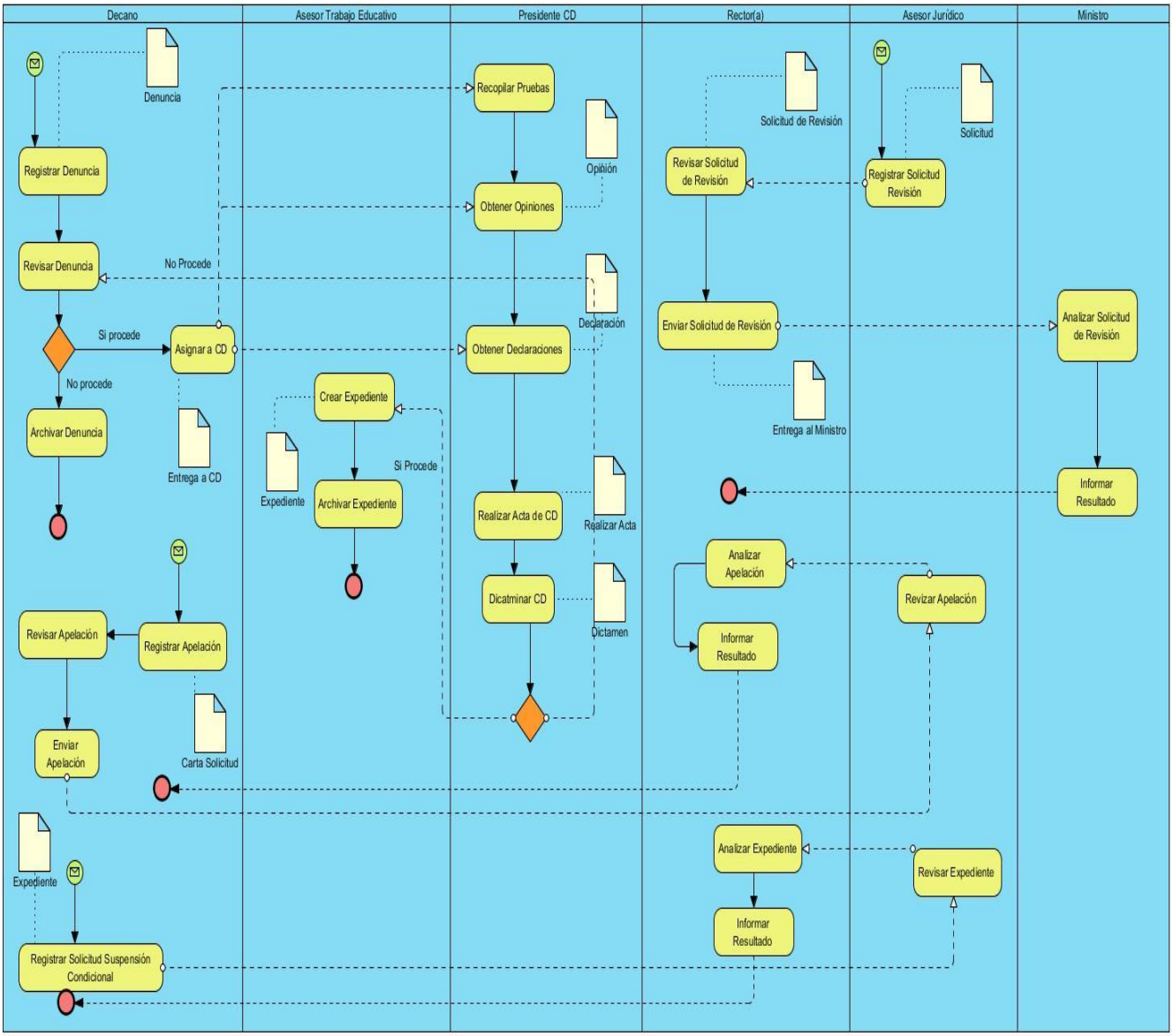

Imagen 1. Diagrama del proceso de negocio

1.4 Sistemas informáticos analizados

En la actualidad el uso de las tecnologías se ha convertido en un elemento fundamental para lograr efectividad y rapidez en todos los procesos que se realizan en las diferentes esferas. Para un mejor funcionamiento de la gestión de la información en el sector educativo, específicamente en la aplicación de los reglamentos, se han desarrollado aplicaciones que por sus características pudieran utilizarse, a continuación se muestran ejemplos:

- **BATUSI (Buzón de Atención al Usuario Informático)**

Es el sistema que tiene el Servicio de Informática de la Universidad de Extremadura (SIUE) de España para canalizar la comunicación entre los usuarios y el servicio de informática. A través de este sistema y siempre dentro de las competencias asignadas a este servicio, se analiza:

- Gestión de las incidencias que se producen en las instalaciones de hardware.
- Gestión de incidencias del funcionamiento de las aplicaciones corporativa de la Universidad: gestión académica, gestión económica, gestión de recursos humanos, gestión de investigación y portal de servicios.
- Gestión de incidencias del funcionamiento de las aplicaciones desarrolladas por el propio servicio.
- Gestión de incidencias del correo electrónico.
- Ayuda en la instalación de software oficial de la universidad.
- Gestión de incidencias relacionadas con infraestructura: redes y telefonía.

Inicialmente se habilitaron dos grandes grupos donde encuadrar las incidencias: las de los usuarios y las internas de SIUE. En estos grupos existen diferentes categorías que permitían especificar el origen de las incidencias: por ejemplo problemas relacionados con matrícula, con becas, con contabilidad, con equipos, etc.

Viendo el buen resultado de la herramienta se fueron creando grupos específicos para las grandes áreas de la gestión universitaria: gestión académica, gestión de recursos humanos, gestión contable y financiera, habilitándose también nuevas categorías tales como correo, antivirus, portal institucional, portal del personal, etc.

Al principio todas las incidencias se resolvían por el personal del SIUE, pero con la llegada del sistema centralizado de la gestión académica y su gran volumen de usuarios, se dio una nueva utilidad al sistema y es que los usuarios de dicho aplicativo, se pudieran poner en contacto con el

personal funcional a cargo de la gestión académica para que le pudieran resolver las dudas relacionadas con las funcionalidades de la herramienta y/o como efectuar determinadas operaciones (6).

- **Dumbo (Portal Integrado para la Gestión de Peticiones de Cambio e Incidencias de la Universidad de Murcia)**

Es una herramienta que integra de una manera ágil la gestión de incidencias y atención a usuarios en las dos vertientes que estas conllevan: los usuarios que las originan y los expertos que se encargan de resolverlas. Dumbo es un sistema web apoyado sobre una arquitectura de tres capas que garantiza fácil acceso e independencia de la plataforma cliente utilizada. Ofrece valiosa información a responsables de sistemas de información y a equipos directivos.

Este sistema es privativo y ajustado a las necesidades propias del cliente. El mismo no es multiplataforma, solo opera en el sistema operativo Windows. Gestiona varios procesos de los cuales se pueden tomar ideas a la hora de definir los requisitos como son:

- Solicitud de nuevas incidencias.
- Exploración de incidencias solicitadas.
- Atención de incidencias recibidas.
- Gestión de la cola genérica de incidencias.
- Administración del sistema.
- Alertas.
- Notificaciones.

El software de forma automatizada integra las principales funciones que se realizan en la Dirección de Recursos Humanos tales como: el inventario de personal, la selección y contratación del personal y el control de las sanciones y amonestaciones, pero en este caso se destacará esta última función debido a que realiza de una manera u otra una de las principales características que debe de cumplir el sistema. Con sus innumerables salidas, tanto estadísticas como gráficas hace posible que en mayor medida, los directivos relacionados con la gestión del personal, puedan conocer y prever las posibles promociones, necesidades de formación y capacitación, los reclutamientos futuros, el comportamiento de la disciplina laboral, el desempeño del personal, así como el desarrollo y un control efectivo de la asistencia a la empresa. El sistema fue programado en FoxPro para Windows versión 2.6 (7).

- **SIPCD (Aplicación informática para el proceso de gestión de las comisiones disciplinarias en la Facultad 3)**

Es un sistema de gestión de las comisiones disciplinarias de la Facultad 3, donde se registra todo el proceso a seguir en caso de que se incumpla con el reglamento “Decreto Ley 240 del año 2007”. El mismo comprende sólo lo que se realiza a nivel de facultad, dígase el proceso de denuncia y la realización de la comisión disciplinaria. A pesar de cubrir una gran gama de funcionalidades este no comprende lo que se realiza fuera de la facultad y los reportes que genera son muy limitados.

Luego de analizados los sistemas BATUSI y Dumbo se llega a la conclusión de que no pueden ser utilizados en la UCI porque son privativos y están implementados específicamente para las universidades donde fueron realizados. Con respecto al sistema desarrollado durante el curso 2013-2014 en la Facultad 3 no comprende todas las funcionalidades y requisitos que son necesarios por la universidad.

1.5 Formulación de la propuesta de solución

1.5.1 Propuesta de solución

Se propone el desarrollo del sistema SIPCD versión 2 (Aplicación informática para el proceso de gestión de las comisiones disciplinarias en la UCI), como continuación del ya desarrollado en la Facultad 3, por lo que se utilizarán algunas herramientas y tecnologías de dicha versión, aunque se establecerá una comparación para analizar si se mantienen, se pueden cambiar o actualizar. El sistema debe comprender un conjunto de procesos como son: revisión, apelación y suspensión condicional de la medida aplicada. La autenticación de los usuarios en el sistema se debe realizar introduciendo datos personales (usuario, contraseña); además debe permitir la realización de comisiones disciplinarias, generar reportes, análisis estadísticos y adicionalmente contar con un sistema de notificaciones basado en las responsabilidades asignadas según el rol autenticado.

1.5.2 Metodología de desarrollo de software

Una metodología de software se puede definir como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. La misma está formada por fases, las cuales se puede dividir en sub-fases (8). En la ingeniería de software existen dos clasificaciones de metodologías, las tradicionales y las ágiles. Las ágiles están dirigidas hacia equipos pequeños y orientadas a las necesidades del cliente, incorporándolo como parte del equipo de desarrollo, están preparadas para cambios durante el proyecto. Mientras que las tradicionales son todo lo contrario,

ya que están dirigidas a grandes equipos de desarrollo de software, están más orientadas al proceso de software que al cliente y son resistentes a los cambios (9).

- **Proceso Unificado de Rational (RUP):**

Forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo). Requiere de un grupo grande de programadores para trabajar con esta metodología. Es un marco del proyecto que describe una clase de los procesos que son iterativos e incrementales, define conjunto de actividades y de artefactos que se necesitan elegir para construir procesos individuales. Los procesos de RUP estiman tareas y horario del plan, midiendo la velocidad de iteraciones concerniente a sus estimaciones originales y los recursos necesarios. Divide el desarrollo del sistema en cuatro fases y nueve flujos de trabajos, de los cuales seis se conocen como flujos de ingeniería y los restantes como disciplinas de apoyo (10).

- **Microsoft Solution Framework (MSF):**

Es una metodología pesada que provee los principios, modelos y disciplinas para un correcto desarrollo de proyectos en cualquier plataforma (Linux, Microsoft, Unix). Es flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo. Teniendo como principales fases: visión y alcances, planificación, desarrollo, estabilización e implantación. Presentando como características su: adaptabilidad, escalabilidad y flexibilidad.

- **SCRUM:**

Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. Define un marco para la gestión de proyectos; está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto (12).

- **Programación Extrema (XP por sus siglas en inglés):**

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los

cambios (13). Constituye un proceso de diseño evolutivo que se basa en mejorar un sistema simple en cada iteración. Todo el diseño se centra en la iteración actual y no pensando necesidades futuras, resultando un proceso de diseño disciplinado y adaptable de manera que indiscutiblemente la hace la más utilizada de todas las metodologías ágiles (14).

Una vez identificadas las diferentes metodologías que se pueden utilizar en el proceso de desarrollo de software se selecciona XP, ya que permite la retroalimentación continua entre el cliente y el equipo de desarrollo, garantizando una comunicación fluida entre todos los participantes, brinda simplicidad en las soluciones implementadas siendo flexible a la hora de enfrentar los cambios y el equipo de desarrollo es pequeño. Además la propuesta de solución se enmarca en la Resolución 240 del año 2007, pero puede cambiar y XP es adecuado para proyectos con requisitos imprecisos y cambiantes, donde existe un alto riesgo técnico. También proporciona mayor control con las fechas de entrega y facilita el cumplimiento del cronograma del presente trabajo.

1.5.3 Marco de trabajo

El término framework se refiere a una estructura de software compuesta de componentes también conocidos como librerías. Provee plantillas o esqueletos que definen el funcionamiento de las aplicaciones que permiten manejar y controlar prácticamente toda la aplicación sin escribir mucho código. Estas plantillas que ofrece el framework se pueden adaptar a diferentes necesidades; además hace relativamente fácil entender otras aplicaciones hechas con el mismo framework, ya que comparten una estructura similar (15).

- **CodeIgniter**

Es un entorno de desarrollo de aplicaciones para desarrolladores que construyen sitios webs usando procesador de hipertexto (PHP). El objetivo es habilitar el desarrollo de proyectos de forma más rápida que si se escribe el código desde cero, provee un conjunto de librerías para tareas comúnmente necesarias, tanto como una simple interfaz y estructura lógica para acceder a librerías. CodeIgniter permite concentrarse creativamente en un proyecto, minimizando el volumen de código necesario para una tarea determinada (16).

- **Symfony 2**

Es un framework para construir aplicaciones web con PHP, es decir, posee un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web. Emplea el patrón de diseño Modelo Vista Controlador (MVC) para separar las distintas partes que forman una aplicación web. Fue diseñado para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Código fácil de leer que incluye comentarios que ayudan a un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo (17).

Después de una investigación de estos frameworks se escoge para la implementación de la aplicación el marco de trabajo “Symfony 2”, ya que contiene facilidades y ventajas que ayudan a que el trabajo sea menos complicado, debido a que libera a los desarrolladores de la tarea de crear funcionalidades menores. Las aplicaciones desarrolladas con Symfony son compatibles con la mayoría de las plataformas, bibliotecas e infraestructuras que existen. Promueve el uso de buenas prácticas de programación, genera código fácilmente comprensible por el desarrollador, y además la versión anterior del sistema SIPCD se encuentra desarrollada en este framework.

- **ORM Doctrine 2**

El marco de trabajo Symfony 2 utiliza Doctrine como mapeador de objetos relacionales[1], proporciona una capa de persistencia para objetos PHP, cuyo único objetivo es dotar de poderosas herramientas para facilitar el trabajo. Doctrine permite recuperar y persistir objetos completos desde y hacia la base de datos, respectivamente. Dispone de una gran cantidad de tipos de campos. Permite utilizar cualquier objeto PHP para almacenar los datos y se basa en la información de asignación de metadatos para asignar los datos de un objeto a una tabla particular de la base de datos. Una característica importante de Doctrine es la posibilidad de escribir consultas de base de datos utilizando un dialecto del lenguaje de consulta estructurado (SQL), denominado Lenguaje de Consulta de Doctrine (DQL), además mediante la parametrización de las mismas se protege de ataques como la inyección SQL (18).

1.5.4 Gestor de base de datos

Es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de manipulación de datos y

[1] Técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos y el sistema de base de datos relacional utilizado en el desarrollo de nuestra aplicación.

de un lenguaje de consulta. Un sistema gestor de base de datos (SGBD) permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Un SGBD debe permitir definir una base de datos, así como ser capaz de construirla y manipularla (19).

- **MySQL**

Es un sistema gestor de bases de datos, ampliamente usado por su simplicidad y notable rendimiento. Es una opción atractiva tanto para aplicaciones comerciales, como de entretenimiento precisamente por su facilidad de uso y tiempo reducido de puesta en marcha. Su libre distribución en Internet bajo licencia GPL le otorgan como beneficios adicionales contar con un alto grado de estabilidad y un rápido desarrollo. Está disponible para múltiples plataformas (20).

- **Microsoft SQL Server**

SQL Server es un sistema gestor de base de datos relacionales producido por Microsoft. Es un sistema cliente/servidor que funciona como una extensión natural del sistema operativo Windows. Entre otras características proporciona integridad de datos, optimización de consultas, control de concurrencia, salvas (backup) y recuperación. Es relativamente fácil de administrar a través de la utilización de un entorno gráfico para casi todas las tareas de sistema y administración de bases de datos. Utiliza servicios del sistema operativo Windows para ofrecer nuevas capacidades o ampliar la base de datos, tales como enviar y recibir mensajes y gestionar la seguridad de la conexión. Es fácil de usar y proporciona funciones de almacenamiento de datos que sólo estaban disponibles en Oracle y otros sistemas gestores de bases de datos más caros (21).

- **PostgreSQL**

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia Berkeley Software Distribution (BSD) y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y utiliza multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (19).

Se escoge utilizar como gestor de base de datos PostgreSQL ya que su código fuente está disponible libremente, es de código abierto, utiliza el modelo cliente/servidor, es altamente escalable tanto en la gran cantidad de datos que puede manejar como en el número de usuarios simultáneos que puede

soportar. Posee una interfaz gráfica y facilita enormemente la administración e incluso puede realizar cambios en el modelo de datos desde la herramienta de manera muy sencilla.

1.5.5 Servidores web

- **Servidor web local**

Es aquel servidor web que reside en una red local. Puede estar instalado en cualquiera de los equipos que forman parte de una red local (22). Es la tecnología que tiene implícito programas informáticos que procesan aplicaciones realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente.

- **Apache 2**

Es un servidor web de protocolos de transferencia de hipertexto (HTTP) gratuito, modular y extensible para la creación de páginas y servicios web. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente (23).

Se escoge este servidor debido a que es multiplataforma, lo que lo hace prácticamente universal. Es una tecnología de código abierto además de ser altamente configurable de diseño modular. Facilita personalizar las respuestas ante los posibles errores que se puedan dar en el servidor.

1.5.6 Lenguajes utilizados

- **Lenguaje Unificado de Modelado (UML)**

Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar elementos conceptuales, como son: procesos de negocio y funciones de sistema, además de características concretas como son: escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables (24). UML brinda muchos beneficios como son: establecer conceptos y artefactos ejecutables, mejor soporte a la planeación y al control de proyectos, alta reutilización y minimización de costos, el tiempo invertido en el desarrollo de la arquitectura se minimiza. El uso de lenguajes visuales facilita su asimilación y entendimiento por parte del equipo de desarrollo.

- **Notación para la Modelación de Procesos de Negocio (BPMN)**

Es un estándar de modelado de procesos de negocio, en donde se presentan gráficamente las diferentes etapas del proceso del mismo. La notación ha sido diseñada específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre ellos (25). BPMN es independiente de cualquier metodología de modelado de procesos; crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos, permite modelar los procesos de una manera unificada y estandarizada, garantizando un entendimiento a todas las personas de una organización. Se hizo uso de esta notación ya que logra ajustarse a las necesidades del proyecto para proveer un modelado de procesos que sea fácil y entendible para todos los usuarios y los clientes que estarán manejando el proceso de negocio (25).

1.5.7 Lenguajes de Programación

Lenguajes de programación del lado del servidor:

- **Procesador de Hipertexto (PHP*)* 5.4**

Es un lenguaje de código abierto muy popular, adecuado para desarrollo web que puede ser incrustado en Lenguaje de Marcas de Hipertexto (HTML). Se utiliza para generar páginas web dinámicas (26). Su diseño elegante lo hace perceptiblemente más fácil de mantener y ponerse al día. Por ser de código abierto[2], goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente (27). Es el lenguaje utilizado por el framework seleccionado, puesto que vinculado a sus ventajas se hace necesaria su utilización.

Twig 1.2

Es un motor de plantillas desarrollado para el lenguaje de programación PHP y nace con el objetivo de facilitar a los desarrolladores de aplicaciones web que utilizan la arquitectura MVC el trabajo con la parte de las vistas, gracias a que se trata de un sistema que resulta muy sencillo de aprender y capaz de generar plantillas con un código preciso y fácil de leer. Actualmente el código se distribuye bajo licencia BSD y es utilizado por el framework Symfony 2. Twig tiene tres características principales: rápido, seguro y flexible (28).

[2] Código abierto es un software que pone a disposición de cualquier usuario su código fuente.

Lenguajes de programación del lado del cliente:

- **JavaScript 1.2**

Es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web, puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. No requiere de compilación ya que el lenguaje funciona del lado del cliente; tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. Es un lenguaje basado en acciones que posee menos restricciones. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros (29). JQuery 1.7.2 es la librería que se utilizará para JavaScript ya que es pequeña, rápida, ligera y con cientos de utilidades para crear fácilmente las páginas web de las aplicaciones dinámicas complejas (30). Está diseñado para modificar la forma en que se escribe JavaScript, además posibilita visualizar toda la información en tablas dinámicas.

- **Lenguaje de Marcas de Hipertexto** (**HTML, por sus siglas en inglés) 5**

Es un lenguaje que se utiliza fundamentalmente en el desarrollo de páginas web. Funciona por medio de etiquetas que describen la apariencia o función del texto enmarcado, es un lenguaje de marcación de elementos para la creación de documentos hipertexto. Este lenguaje puede llegar a incluir un script o código que tenga incidencia en el comportamiento del navegador web de elección, tales como CSS, Javascript o PHP (31).

- **Hoja de Estilo en Cascada (CSS, por sus siglas en inglés) 3**

Es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación. Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla. Utilizado para dar estilos a html y Lenguaje de Marcas Extensible (XML), separando el contenido de la presentación (31).

1.5.8 Herramientas informáticas

Se hizo necesario utilizar determinadas herramientas para facilitar el trabajo y la modelación del sistema de comisiones disciplinarias. A continuación se detallan las características significativas que definieron su uso para el desarrollo.

- **Subversion 1.5**

Es un sistema de control de versiones libres, el cual maneja ficheros y directorios a través del tiempo. Este sistema puede acceder al repositorio a través de redes, lo que permite ser usado por personas desde distintos ordenadores. Subversion proporciona las facilidades de borrar, añadir, copiar, o renombrar ficheros y directorios, creando así, un nuevo historial para cada fichero agregado. Este sistema puede conectarse al servidor Apache mediante el protocolo HTTP dándole gran ventaja e interoperabilidad así como el acceso instantáneo a las características existentes que ofrece este servidor (32).

Esta herramienta permitió el acceso al repositorio a través de redes, garantizando a los desarrolladores la posibilidad de conectarse para leer o escribir archivos ya existentes. Facilitó además la recuperación de versiones antiguas de datos y la realización de consultas históricas de cómo cambiaron los mismos desde la primera hasta la última versión.

- **NetBeans 7.4**

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) gratuito, de código abierto y multiplataforma. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Brinda soporte a varias tecnologías como: Java, PHP, HTML5, entre otras. Entre sus principales características se encuentran: editor de código, multilenguaje, simplifica la gestión de grandes proyectos, herramientas para depurado de errores, optimización de código (33). Esta herramienta permitió que en un solo entorno se editara el código de la aplicación, así como depurarlo. Además, permitió la integración con “Subversión” para el control de versiones.

- **PgAdmin 3**

Es una herramienta de código abierto para la administración del gestor de bases de datos PostgreSQL, fue diseñada para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos (34).

- **Visual Paradigm for UML 8.0**

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código

desde diagramas y generar documentación. La herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML. Una de las características más importantes es que es multiplataforma (35). Permitió modelar el diagrama de clase, componente, despliegue y modelo de datos.

1.5.9 Patrón de arquitectura

Un patrón arquitectónico es un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño. La arquitectura del patrón **Modelo-Vista-Controlador** (MVC) es un paradigma de programación bien conocido para el desarrollo de aplicaciones con interfaz gráfica. El principal objetivo de este patrón es aislar tanto los datos de la aplicación como el estado (modelo) de la misma, del mecanismo utilizado para representar (vista) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (controlador) (36). Por un lado define componentes para la representación de la información, y por otro, la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

- **Modelo**

Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos se tendrán en una base de datos, así los modelos tendrán todas las funciones para acceder a las tablas y hacer las consultas (36).

- **Vista**

Las vistas, como su nombre hace entender, contienen el código de la aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que permitirá renderizar los estados de la aplicación en HTML. En las vistas nada más se tienen los códigos HTML y PHP que permiten mostrar la salida (36).

- **Controlador**

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento o realizar una búsqueda de información (36).

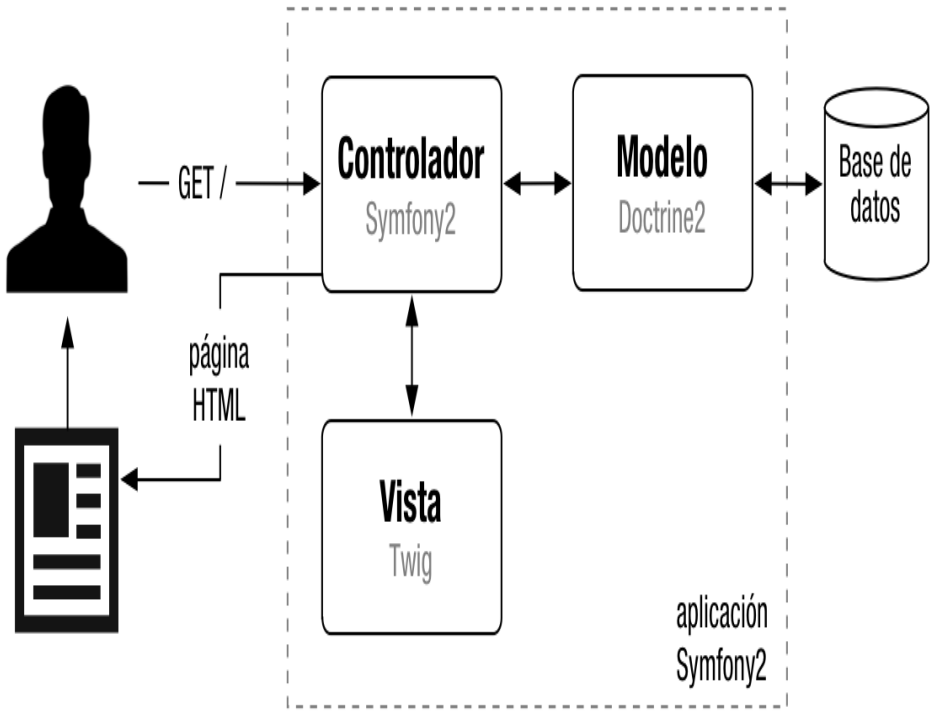

Imagen 2. Arquitectura MVC

1.5.10 Patrones de Diseño

"Los patrones de diseño son descripciones estructurales y funcionales de cómo resolver, de forma concreta un determinado problema mediante orientación a objetos" (37). Permiten que el diseño sea más flexible, elegante y reutilizable.

- **GRASP (Patrones Generales de Software para Asignación de Responsabilidades)**

Son patrones que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento al diseño de una forma sistemática, racional y explicable (38). Dentro de sus patrones se encuentran experto, creador, controlador, alta cohesión y bajo acoplamiento.

- **GoF (Gang of Four)**

Es la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Los patrones pretenden proporcionar catálogos de elementos reusables en el diseño de sistema de software y estandarizar el modo en que se realiza el diseño. Se clasifican en creacionales, estructurales y de comportamiento.

Tabla 1.Patrones GoF según su clasificación

Creacionales	Estructurales	De comportamiento
Fábrica abstracta	Adaptador	Orden
Constructor	Proxy	Cadena de responsabilidades
Método de fabricación	Objeto compuesto	Estado
Prototipo	Decorador	Estrategia
Instancia única	Fachada	Intérprete
	Peso ligero	Iterador
	Puente	Mediador
		Método plantilla
		Observador
		Recuerdo
		Visitante

1.6 Métricas

Las métricas de software son utilizadas para propósitos estratégicos, permitiendo minimizar la planificación del desarrollo haciendo los ajustes necesarios que eviten retrasos y reduzcan problemas y riesgos potenciales. Son utilizadas también para evaluar la calidad de los productos en el momento actual y cuando sea necesario, modificando el enfoque técnico que mejore la calidad, para establecer objetivos de mejora durante el proceso de desarrollo de software (10).

1.6.1 Métricas para requisitos

Los requisitos del software son la base de las medidas de la calidad. En la disciplina de requisitos se tuvieron en cuenta las métricas estabilidad y especificidad.

- **Estabilidad de los requisitos**

El correcto funcionamiento de los flujos de trabajo depende del logro de una estabilidad en los requisitos. Se considera que los requisitos son estables cuando no existan adiciones o supresiones que impliquen modificaciones en las funcionalidades principales de la aplicación.

$$ETR = \left[\frac{RT - RM}{RT}\right] * 100$$

Fórmula para calcular la estabilidad de requisitos

ETR: valor de la estabilidad de los requisitos.

RT: total de requisitos definidos.

RM: número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados.

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100, ya que mostrará que no se están realizando cambios sobre los requisitos, son estables y por tanto es confiable trabajar el diseño sobre ellos (10).

- **Especificidad**

La especificidad es la ausencia de ambigüedad de los requisitos. Esta métrica está basada en la consistencia de la interpretación de los revisores para cada requisito.

$$Q_1 = \frac{n_{ui}}{n_r}$$

Fórmula para calcular la especificidad de requisitos

n_{ui}**:** número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

n_r**:** número de requisitos en una especificación.

Cuanto más cerca de 1 esté el valor de Q1, menor será la ambigüedad de la especificación (10).

1.6.2 Métricas para el diseño

Las métricas para el diseño evalúan distintos parámetros tales como: responsabilidad de las clases, reutilización y complejidad de implementación y mantenimiento. Las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC) permiten evaluar estos parámetros y adicionalmente el bajo acoplamiento y la cantidad de pruebas unitarias que deben realizarse.

Tamaño Operacional de Clase:

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 2. Atributos de calidad y su definición

Atributos de calidad	Definición
Responsabilidad	Consisten en asignarle una responsabilidad a una clase en un marco de modelado de dominio.
Complejidad de implementación	Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado
Reutilización	Consiste en el grado de reutilización presente en una clase.

Las métricas referentes al tamaño para las clases orientadas a objetos se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema orientadas a objetos como un todo. El tamaño general de una clase puede medirse determinando las siguientes medidas:

- El total de operaciones (heredadas y privadas de la instancia), que se encapsulan dentro de la clase.
- El número de atributos (heredados y privados de la instancia), encapsulados por la clase.

Estos dos valores son sumados de acuerdo a la clase que se analiza y los resultados son tomados como cantidad de procedimientos (CP) que luego son comparados en la tabla 3 para determinar el TOC de cada clase. A continuación se muestran tres tablas encaminadas a un mejor entendimiento de la utilización de esta métrica.

Tabla 3.Clasificación de las clases según el tamaño

Clasificación	Valores
Pequeño	CP <= 20
Medio	20 < CP <= 30
Grande	CP > 30

Tabla 4.Modo en que afecta el TOC a los atributos de calidad

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC indica que la clase puede tener demasiada responsabilidad.

Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC reduce la reutilización de la clase.

Para determinar el valor de los atributos de calidad, se calcula el promedio de la columna cantidad de procedimientos y se emplea en la tabla 5 en la columna criterio (10).

Tabla 5.Rango de valores para la evaluación de los atributos de calidad

Atributos de calidad	Clasificación	Criterio
Responsabilidad	Baja	CP <= Promedio
	Media	Promedio <= CP <= 2* Promedio
	Alta	CP > 2* promedio
Complejidad de implementación	Baja	CP <= Promedio
	Media	Promedio <= CP < = 2* promedio
	Alta	CP > 2* promedio
Reutilización	Baja	CP > 2* promedio
	Media	Promedio <= CP < = 2* promedio
	Alta	CP <= Promedio

Relaciones entre clases:

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- **Acoplamiento:** un aumento de las RC implica un aumento del acoplamiento de la clase.
- **Complejidad de mantenimiento:** un aumento de las RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Cantidad de pruebas:** un aumento de las RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

A continuación se muestran las tablas encaminadas a un mejor entendimiento de la utilización de esta métrica.

Tabla 6.Atributos de calidad de la métrica RC

Atributos de calidad	Definición
Acoplamiento	Consiste en el grado de dependencia o interconexión de una clase con otras.
Complejidad de 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.
Cantidad de pruebas	Consiste en el número o grado de esfuerzo necesario para realizar las pruebas.

Para determinar el grado de afectación de los atributos de calidad que mide la métrica RC es necesario determinar la cantidad de relaciones de uso (CRU) que posee cada una de las clases a medir. Una vez determinada la CRU, se procede a calcular el promedio de las mismas y luego teniendo ambos valores se clasifican según los criterios expuestos en la tabla 7 (10).

Tabla 7.Rango de valores para la evaluación de los atributos de calidad

Atributos de calidad	Clasificación	Criterio
Acoplamiento	Ninguna	CRU = 0
	Baja	CRU = 1
	Media	CRU = 2
	Alta	CRU > 2
Complejidad de mantenimiento	Baja	CRU < = Promedio
	Media	Promedio < CRU < = 2 * Promedio
	Alta	CRU > 2 * Promedio
Cantidad de pruebas	Baja	CRC <= Promedio
	Media	Promedio <= CRC <= 2 * Promedio
	Alta	CRC > 2 * Promedio

1.7 Pruebas

Son las investigaciones empíricas y técnicas cuyo fin es proporcionar información objetiva e independiente sobre la calidad del producto. Son además un conjunto de actividades dentro del desarrollo de software y dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento del proceso de desarrollo (39). Las pruebas de software son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Para validar la calidad de la aplicación se utilizó la prueba de caja negra.

Prueba de caja negra

Permite obtener un conjunto de condiciones de entradas que ejecuten todos los requisitos funcionales de un programa. Las pruebas de caja negra intentan hallar errores tales como:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en la estructura de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación (40).

Técnica partición de equivalencia: se basa en una evaluación de las clases de equivalencia para una condición de entrada, donde una clase de equivalencia representa un conjunto de estados válidos, no válidos o que no aplican. Las condiciones de entrada son valores numéricos específicos, un rango de valores, un conjunto de valores relacionados o una condición lógica (41).

1.8 Conclusiones Parciales

Luego de lo analizado en el presente capítulo se llegan a las siguientes conclusiones:

- El análisis de los sistemas informáticos desarrollados en otros países y en Cuba mostró que no pueden ser utilizados debido a las particularidades del proceso de comisiones disciplinarias en la UCI, por lo que se hace necesario desarrollar una aplicación que cumpla con las características propias del proceso.
- La selección de la metodología de desarrollo XP permitirá lograr mayor organización, planificación y ejecución de las actividades e hitos a cumplir.
- El empleo de las herramientas seleccionadas favorecerá el logro de los objetivos trazados e influyen positivamente en el incremento de la productividad.

Capítulo 2: Análisis y diseño

2.1 Introducción

En el presente capítulo se detallará la propuesta de solución presentando sus principales características. Se llevará a cabo el proceso de desarrollo especificando cada uno de los procesos que intervienen en las distintas fases de la metodología seleccionada. Se obtendrán los artefactos Historias de Usuarios, Tarjetas CRC, Plan de Iteraciones y Plan de Entrega pertenecientes a la metodología seleccionada.

2.2 Requisitos del sistema

De la palabra "requisito" existen numerosas definiciones, según el glosario de la IEEE[3]:

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

Según los autores un requisito es una necesidad de un usuario que debe ser resuelta por un sistema. Existen dos tipos de requisitos, los funcionales (capacidades o condiciones que el sistema debe cumplir) y los no funcionales (propiedades o cualidades que el producto debe tener, como la fiabilidad, el tiempo de respuesta, capacidad de almacenamiento, facilidad de mantenimiento, entre otras) (42).

Captura y análisis de los requisitos

La captura de requisitos es el proceso para extraer, de cualquier fuente de información disponible, las necesidades que debe cubrir el sistema, en esta actividad los ingenieros de software trabajan con los clientes y los usuarios finales. Existen varias técnicas de descubrimiento de requisitos, como son: las entrevistas, los escenarios, los prototipos y las tormentas de ideas (39).

Para llevar a cabo el proceso de captura de requisitos se aplicaron las técnicas que a continuación se exponen:

Entrevistas: se realizaron varias entrevistas al asesor de trabajo educativo, asesor jurídico, sustentado en una serie de preguntas, con el objetivo de obtener la mayor cantidad posible de información y comprender los objetivos generales de la solución buscada.

[3] Del inglés: *Institute of Electrical and Electronics Engineers.*

- **Tormenta de ideas:** el equipo de desarrollo realizó varias reuniones para acumular ideas e información, que proporcionaron una visión general del proceso en cuestión.

2.2.1 Requisitos funcionales

Los requisitos funcionales "definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas y especifican los servicios que debe proporcionar la aplicación" (43).

Los requisitos son la base para un desarrollo exitoso así como para una plena conformidad con el entregable final. A continuación se muestra el listado de requisitos definidos para el sistema informático propuesto.

Tabla 8.Requisitos funcionales

Requisitos HU 1 Gestionar usuario.	
RF_Autenticar_Usuario	RF_Actualizar_Usuario
RF_Listar_Usuarios	RF_Adicionar_Usuario
Requisitos HU 2 Gestionar denuncias	
RF_Adicionar_Datos_Persona	RF_Actualizar_Datos_Persona
RF_Revisar_Denuncia	RF_Asociar_Personas_Denuncia
RF_Asociar_Adjuntos_Denuncia	RF_Asignar_Denuncia_CD
RF_Actualizar_Denuncia	RF_Adicionar_Denuncia
RF_Listar_Denuncias	
Requisitos HU 3 Gestionar comisión disciplinarias	
RF_Actualizar_Opinión	RF_Adicionar_Dictamen
RF_Actualizar_Acta	RF_Adicionar_Declaración
RF_Adicionar_Decisión	RF_Listar_CD
RF_Adicionar_Opinión	RF_Asociar_Adjuntos_CD
RF_Listar_Declaraciones	RF_Listar_Declaraciones
RF_Adicionar_Acta	RF_Listar_Opiniones

RF_Asociar_Persona_Opinión	RF_Actualizar_Dictamen
RF_Actualizar_Decisión	RF_Actualizar_Declaración
Requisitos HU 4 Gestionar notificaciones	
RF_Listar_Notificaciones	RF_Adicionar_Notificaciones
Requisitos HU 5 Gestionar reporte	
RF_Generar_Reporte_Denuncias	RF_Generar_Reporte_CD
RF_Generar_Reporte_Apelación	RF_Generar_Reporte_Revisión
RF_Generar_Reporte_Suspención	
Requisitos HU 6 Gestionar términos de tiempo	
RF_Listar_Término	RF_Notificar_Término
Requisitos HU 7 Gestionar comisiones	
RF_Adicionar_Comisión	RF_Actualizar_Comisión
RF_Listar_Comisión	RF_Asociar_Integrantes_Comisión
Requisitos HU 8 Gestionar plantillas	
RF_Mostrar_Vista_Previa_Plantilla	RF_Generar_pdf
RF_Actualizar_Plantillas	
Requisitos HU 9 Gestionar apelación	
RF_Adicionar_Apelación	RF_Actualizar_Apelación
RF_Adicionar_Acta_Apelación	RF_Listar_Apelación
RF_Actualizar_Acta_Apelación	RF_Adicionar_Dictamen_Apelación
RF_Actualizar_Dictamen_Apelación	
Requisitos HU 10 Gestionar nomencladores	
RF_Adicionar_Actualizar_Facultad	RF_Adicionar_Actualizar_Origen_Opinion
RF_Adicionar_Actualizar_Origen_Denuncia	RF_Adicionar_Actualizar_Tipificación_Falta
Requisitos HU 11 Gestionar revisión	

RF_Adicionar_Revisión	RF_Listar_Revisión
RF_Actualizar_Revisión	RF_Actualizar_Respuesta_Revisión
Requisitos HU 12 Gestionar suspensión condicional de la medida	
RF_Adicionar_Apelación	RF_Actualizar_Apelación
RF_Adicionar_Acta_Apelación	RF_Listar_Apelación
RF_Actualizar_Acta_Apelación	RF_Adicionar_Dictamen_Apelación
RF_Actualizar_Dictamen_Apelación	
Requisitos HU 13 Gestionar reporte estadístico	
RF_Obtener_Reporte_Estadístico_del_proceso	
Requisitos HU 14 Gestionar trazas	
RF_Buscar_Trazas	
Requisitos HU 15 Gestionar búsquedas	
RF_Buscar_Revisiones	RF_Buscar_CD
RF_Buscar_Denuncia	RF_Buscar_Apelación
RF_Buscar_Suspención	

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces. Se refieren a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, entre otras (43).

Requisitos de apariencia o interfaz externa

- Las interfaces tienen que mostrar un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios su utilización.
- La interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma.

Requisitos de usabilidad

- La aplicación propuesta será usada por personas que tengan o no conocimientos básicos de informática.
- El sistema estará disponible las 24 horas del día.
- A los administradores del sistema se les dará un adiestramiento básico en el uso de la aplicación.
- Los mensajes de error deben ser reportados por la propia aplicación. Los mensajes del sistema deben estar en el idioma español.
- El sistema será multiplataforma.

Requisitos de soporte

- Se necesita un servidor de bases de datos que soporte volúmenes de datos (PostgreSQL 9.1).
- Se necesita un servidor web (Apache).

Requisitos de seguridad

- La información manejada por el sistema debe estar protegida de acceso no autorizado.
- La aplicación deberá estar disponible en todo momento para aquellas personas con acceso a la información.
- La seguridad se establecerá por roles, que se le asignarán a los usuarios que interactúen con el sistema por el administrador del mismo.
- La comunicación entre el servidor Web y las máquinas clientes será mediante el protocolo HTTPS.

Requisitos de software

- Un servidor web con los módulos: php5-xsl, php5-pgsql, php5-mencache, php5-cli, phpapc, php-soap, el php5-intl y php5-ldap.
- Un servidor de base de datos Postgres 9.1.
- Computadoras clientes con un navegador web (Mozilla Firefox 20.0 o superior).

Requisitos de hardware

- En el cliente se requiere una máquina con 256 MB de RAM como mínimo, procesador Intel® a 1 GHz de velocidad de procesamiento o superior, tarjeta de red Ethernet.
- Todas las máquinas implicadas en la funcionalidad de la aplicación deben estar conectadas a la red.
- El servidor requiere mínimo 1024 MB de RAM, procesador a 3 GHz de velocidad y tarjeta de red Ethernet.

2.3 Validación de requisitos

La validación de requisitos tiene como misión demostrar que la definición de los mismos se corresponde realmente con el sistema que el usuario necesita. Esta es una actividad fundamental, pues un levantamiento de requisitos con errores que no se detecten a tiempo y que conduzcan a resultados inesperados, provocan costos excesivos y gran pérdida de tiempo. Para la validación de estos se aplicaron las técnicas de estabilidad y especificidad.

Estabilidad

Es necesario lograr una estabilidad en los requisitos para el correcto funcionamiento de los demás flujos de trabajo. El objetivo de esta métrica es medir la estabilidad para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación.

Se aplicó la métrica estabilidad como se muestra a continuación:

$$ETR = [(72–36) /72] * 100 = 50$$

Total de requisitos funcionales (RT)=72

Total de requisitos modificados (RM)=36

Se definió por el grupo de desarrollo que los requisitos se consideran estables si mediante la aplicación de la métrica se obtiene un valor mayor que 90, por lo que los resultados obtenidos en la primera iteración demuestran que los requisitos no fueron lo suficientemente estables, debido a las modificaciones que ocurrieron en su definición.

Luego se procedió a realizar una segunda iteración, como se muestra a continuación:

$$ETR = [(72– 3) /72] * 100 = 95,83$$

Total de requisitos funcionales (RT)=72

Total de requisitos modificados (RM)=3

Dicha cifra demuestra que los requisitos son estables y por tanto es confiable trabajar el diseño sobre ellos.

Especificidad

Para medir la ausencia de ambigüedad de los requisitos, se aplicó la métrica especificidad como se muestra a continuación:

$$Q1 = 72/ 72= 1$$

Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas $\mathbf{n_{ui}}$= 72.

Número de requisitos en una especificación $\mathbf{n_r}$= 72.

Como resultado de la aplicación de la métrica se definió que de 72 requisitos, 72 tuvieron interpretaciones idénticas por el equipo de expertos obteniéndose un valor de 1. Dicha cifra demuestra que los requisitos son entendibles, específicos y que no poseen ambigüedades.

Fases de la metodología de desarrollo

El ciclo de vida de la metodología XP se divide en 5 fases: exploración, planificación, diseño, implementación y prueba (13). A continuación se explicará cada una de ellas, especificando los artefactos que se generan por cada una.

2.4 Fase de exploración

"En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema" (13).

2.4.1 Historias de usuarios

Las Historias de Usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. En cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en pocas semanas. Estas fichas quedan estructuradas de la siguiente forma:

Número: a cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Usuario: es el usuario del sistema que realizó la historia de usuario.

Nombre: nombre descriptivo de la HU.

Programador: nombre del programador encargado de desarrollar esa historia.

Prioridad en negocio: grado de prioridad que le asigna el cliente a la HU en dependencia del valor en el negocio. Los valores que puede tomar son (Alta, Media o Baja).

Riesgo en desarrollo: grado de complejidad que le asigna el equipo de desarrollo a la HU luego de analizarla (Alto, Medio o Bajo).

Puntos estimados: esfuerzo estimado por el equipo de desarrollo para darle cumplimiento a la HU.

Puntos reales: días o semanas que realmente demora el programador en realizar una HU.

Iteración asignada: número de la iteración en la cual será implementada la HU.

Descripción: descripción simple sobre lo que debe hacer la funcionalidad a la que se hace referencia.

Se confeccionaron un total de 15 HU, de las cuales 9 fueron adicionadas y 6 modificadas. A continuación se muestra un ejemplo de estas fichas.

Tabla 9.Historia de Usuario número 9. Gestionar Apelación

Historia de usuario	
Número: 9	**Nombre historia de usuario**: Gestionar apelación
Modificación de historia de usuario número: ninguna	
Usuario: Diana Rosa Martínez García Yiselly Azaharez Matos	**Iteración asignada**: 2
Prioridad en negocio: alto	**Puntos estimados**: 0.86 semanas (6 días)
Riesgo en desarrollo: medio	**Puntos reales**: 0.86 semanas (6 días)

Descripción:

- Permite asociar adjuntos a una apelación.

El sistema debe de ser capaz de asociar adjuntos a la apelación correspondiente.

- Permite adicionar/actualizar una apelación.

El sistema debe permitir la opción de adicionar/actualizar una apelación donde debe seleccionar el implicado en la apelación que la solicita, la fecha de solicitud y las observaciones. Luego se envía al Rector.

- Permite adicionar/actualizar una decisión.

El sistema debe permitir la opción de adicionar/actualizar una decisión de la apelación, si al analizar el expediente se detecta que el mismo procede, se pasa a continuar con el proceso, sino se archiva y se notifica al Decano. Los datos recogidos son: decisión y consideraciones.

- Permite listar una apelación.

El sistema debe de ser capaz de mostrar un listado con todas las apelaciones mostrando la información de: fecha de solicitud e involucrado que solicita.

- Permite buscar una apelación.

El sistema debe ser capaz de mostrar un listado con las apelaciones según parámetros de búsquedas.

Observaciones:

La historia de usuario mostrada pertenece a la iteración número 2 debido a la importancia que tiene tanto para el cliente como para el equipo de desarrollo.

2.5 Fase de Planificación

Se puede destacar que las historias de usuario juegan un papel importante en la estimación de los tiempos requeridos para el desarrollo del proyecto. La velocidad del proyecto es utilizada para establecer cuantas historias se pueden implementar antes de una fecha determinada. Para ello se realizan acuerdos sobre el material a entregar en la primera iteración y en correspondencia se genera un cronograma junto al cliente (13).

2.5.1 Estimación de esfuerzo

La medida utilizada para la estimación del esfuerzo asociado a la implementación es el punto. Un punto equivale a una semana ideal de programación. Esta medida generalmente toma valores de 1 a 3.

Tabla 10.Plan de esfuerzo por HU

No	Historia de usuario	Punto de estimación
1	RF_Autenticar_Usuario	1
2	RF_Listar_Usuarios	1
3	RF_Actualizar_Usuario	1
4	RF_Adicionar_Usuario	1
5	RF_Adicionar_Datos_Persona	1
6	RF_Revisar_Denuncia	1
7	RF_Asociar_Adjuntos_Denuncia	2
8	RF_Actualizar_Denuncia	1
9	RF_Listar_Denuncias	1
10	RF_Actualizar_Datos_Persona	1
11	RF_Asociar_Personas_Denuncia	2
12	RF_Asignar_Denuncia_CD	1
13	RF_Adicionar_Denuncia	1
14	RF_Actualizar_Opinión	1
15	RF_Actualizar_Acta	1
16	RF_Adicionar_Decisión	1
17	RF_Adicionar_Opinión	1
18	RF_Listar_Declaraciones	1
19	RF_Adicionar_Acta	1
20	RF_Asociar_Persona_Opinión	1

21	RF_Actualizar_Decisión	1
22	RF_Adicionar_Dictamen	1
23	RF_Adicionar_Declaración	1
24	RF_Listar_CD	1
25	RF_Asociar_Adjuntos_CD	1
26	RF_Listar_Declaraciones	1
27	RF_Listar_Opiniones	1
28	RF_Actualizar_Dictamen	1
29	RF_Actualizar_Declaración	1
30	RF_Listar_Notificaciones	1
31	RF_Adicionar_Notificaciones	1
32	RF_Generar_Reporte_Denuncias	1
33	RF_Generar_Reporte_Apelación	1
34	RF_Generar_Reporte_Suspención	1
35	RF_Generar_Reporte_CD	1
36	RF_Generar_Reporte_Revisión	1
37	RF_Notificar_Término	1
38	RF_Adicionar_Comisión	1
39	RF_Listar_Comisión	1
40	RF_Actualizar_Comisión	1
41	RF_Asociar_Integrantes_Comisión	1
42	RF_Mostrar_Vista_Previa_Plantilla	1
43	RF_Actualizar_Plantillas	1
44	RF_Adicionar_Apelación	1
45	RF_Adicionar_Acta_Apelación	1

46	RF_Actualizar_Acta_Apelación	1
47	RF_Actualizar_Dictamen_Apelación	1
48	RF_Actualizar_Apelación	1
49	RF_Listar_Apelación	1
50	RF_Adicionar_Dictamen_Apelación	1
51	RF_Adicionar_Actualizar_Facultad	1
52	RF_Adicionar_Actualizar_Origen_Denuncia	1
53	RF_Adicionar_Actualizar_Origen_Opinion	1
54	RF_Adicionar_Actualizar_Tipificación_Falta	1
55	RF_Adicionar_Revisión	1
56	RF_Actualizar_Revisión	1
57	RF_Listar_Revisión	1
58	RF_Actualizar_Respuesta_Revisión	1
59	RF_Adicionar_Apelación	1
60	RF_Adicionar_Acta_Apelación	1
61	RF_Actualizar_Acta_Apelación	1
62	RF_Actualizar_Dictamen_Apelación	1
63	RF_Actualizar_Apelación	1
64	RF_Listar_Apelación	1
65	RF_Adicionar_Dictamen_Apelación	1
66	RF_Obtener_Reporte_Estadístico_del_proceso	2
67	RF_Buscar_Trazas	1
68	RF_Buscar_Revisiones	1
69	RF_Buscar_Denuncia	1
70	RF_Buscar_Suspención	1

71	RF_Buscar_CD	1
72	RF_Buscar_Apelación	1

Los puntos de estimación dieron como resultado un total de 75 semanas, pero teniendo en cuenta que el equipo de desarrollo está conformado por dos personas y que más de un requisito se realiza de forma paralela se acorta el tiempo de desarrollo a 14 semanas. En este tiempo influye que ya se encuentran implementadas funcionalidades a las que se le realizaron modificaciones.

2.5.2 Plan de entrega

En el momento en que culmina la elaboración de las HU, se inicia el proceso de creación de un plan de entrega. El cual tiene como objetivo fundamental la obtención por parte de los programadores de una estimación detallada del período de tiempo que deben tener en cuenta para la implementación.

Tabla 11.Plan de entrega de las iteraciones

Artefacto		**Iteración**	**Entrega**
Req 1	Req 6	1	20 de enero
Req 2	Req 7		
Req 3	Req 8		
Req 4	Req 9		
Req 5	Req 10		
Req 11	Req 23	2	27 de febrero
Req 12	Req 24		
Req 13	Req 25		
Req 14	Req 26		
Req 15	Req 27		
Req 16	Req 28		
Req 17	Req 29		
Req 18	Req 30		

Req 19	Req 31		
Req 20	Req 32		
Req 21	Req 33		
Req 22	Req 34		
Req 35	Req 36		
Req 37	Req 49	3	25 de marzo
Req 38	Req 50		
Req 39	Req 51		
Req 40	Req 52		
Req 41	Req 53		
Req 42	Req 54		
Req 43	Req 55		
Req 44	Req 56		
Req 45	Req 57		
Req 46	Req 58		
Req 47	Req 59		
Req 48			
Req 49	Req 58	4	30 de abril
Req 50	Req 59		
Req 51	Req 60		
Req 52	Req 61		
Req 53	Req 62		
Req 54	Req 63		

Req 55	Req 64		
Req 56	Req 65		
Req 57	Req 66		

Según la planificación del desarrollo para la entrega de las iteraciones del sistema se obtiene que el mismo estará culminado el 30 de abril del 2015.

2.5.3 Plan de iteraciones

Para cada entrega fueron escogidas algunas HU, teniendo en cuenta el orden definido. Este plan define las historias de usuario que deben ser implementadas en cada iteración y las fechas de liberación. A continuación se muestra una iteración.

Iteración: número de la iteración.

Requisitos: nombre de los requisitos.

Duración total: duración en semanas de la implementación de los requisitos.

Tabla 12.Plan de iteraciones

Iteración	Requisitos	Duración total (semanas)
4	RF_Listar_Apelación	3.4
	RF_Adicionar_Dictamen_Apelación	
	RF_Adicionar_Actualizar_Facultad	
	RF_Adicionar_Actualizar_Origen_Denuncia	
	RF_Adicionar_Actualizar_Origen_Opinion	
	RF_Adicionar_Actualizar_Tipificación_Falta	
	RF_Adicionar_Revisión	
	RF_Actualizar_Revisión	
	RF_Listar_Revisión	
	RF_Actualizar_Respuesta_Revisión	
	RF_Adicionar_Apelación	
	RF_Adicionar_Acta_Apelación	

	RF_Actualizar_Acta_Apelación	
	RF_Actualizar_Dictamen_Apelación	
	RF_Actualizar_Apelación	
	RF_Listar_Apelación	
	RF_Adicionar_Dictamen_Apelación	
	RF_Obtener_Reporte_Estadístico_del_proceso	

El plan de iteraciones se realizó para organizar la implementación, se obtuvieron un total de 4 iteraciones, el resto de las iteraciones se encuentra en el Anexo 45.

2.6 Fase de diseño

El diseño es el proceso para definir la arquitectura, los componentes, las interfaces, y otras características de un sistema o un componente. Es muy importante, ya que se establecen los mecanismos, para que este sea revisado y mejorado de manera continua a lo largo del proyecto, según se van añadiendo funcionalidades al mismo. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto (10)**.**

2.6.1 Tarjetas CRC (Clase–Responsabilidad-Colaborador)

Las tarjetas CRC fueron la base para la obtención del modelo entidad relación. Cada tarjeta se convirtió en un objeto, sus responsabilidades en métodos públicos y sus colaboradores en llamados a otras clases. Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. En la imagen 3 se muestra cómo se distribuye esta información.

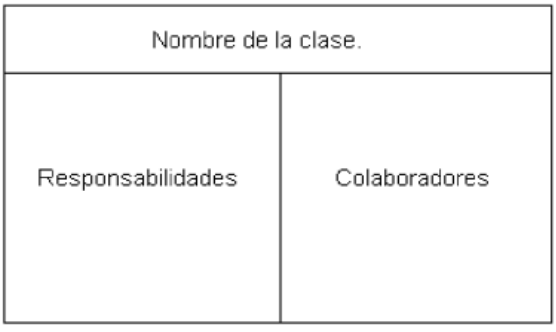

Imagen 3.Tarjeta CRC

A continuación se muestran dos ejemplos de tarjetas CRC confeccionadas durante la fase de diseño, el resto podrá verse en los Anexos del 2 al 30.

Tabla 13.Tarjeta CRC número 31

Apelación	
Responsabilidades Atributos Descripción. Observaciones. Fecha entrega.	Colaboradores Persona. Comisión. Dictamen.

Tabla 14.Tarjeta CRC número 27

Dictamen	
Responsabilidades Atributos 1. Fecha de creación. 2. Medida. 3. Atenuantes. 4. Agravantes. 5. Texto tipificación.	Colaboradores 1. Persona. 2. Comisión. 3. Decisión. 4. Apelación. 5. Tipificación.

2.6.2 Diagrama de Clases

La imagen 4 muestra un total de 35 clases y las relaciones entre ellas. Además se obtuvieron seis clases nomencladores que responderán a determinadas características del negocio. Cada clase representada es experta en realizar las tareas para las cuales se definieron sus responsabilidades. Además en el diagrama se puede ver un bajo acoplamiento y una alta cohesión debido a que las clases realizan labores únicas y existen pocas relaciones entre ellas.

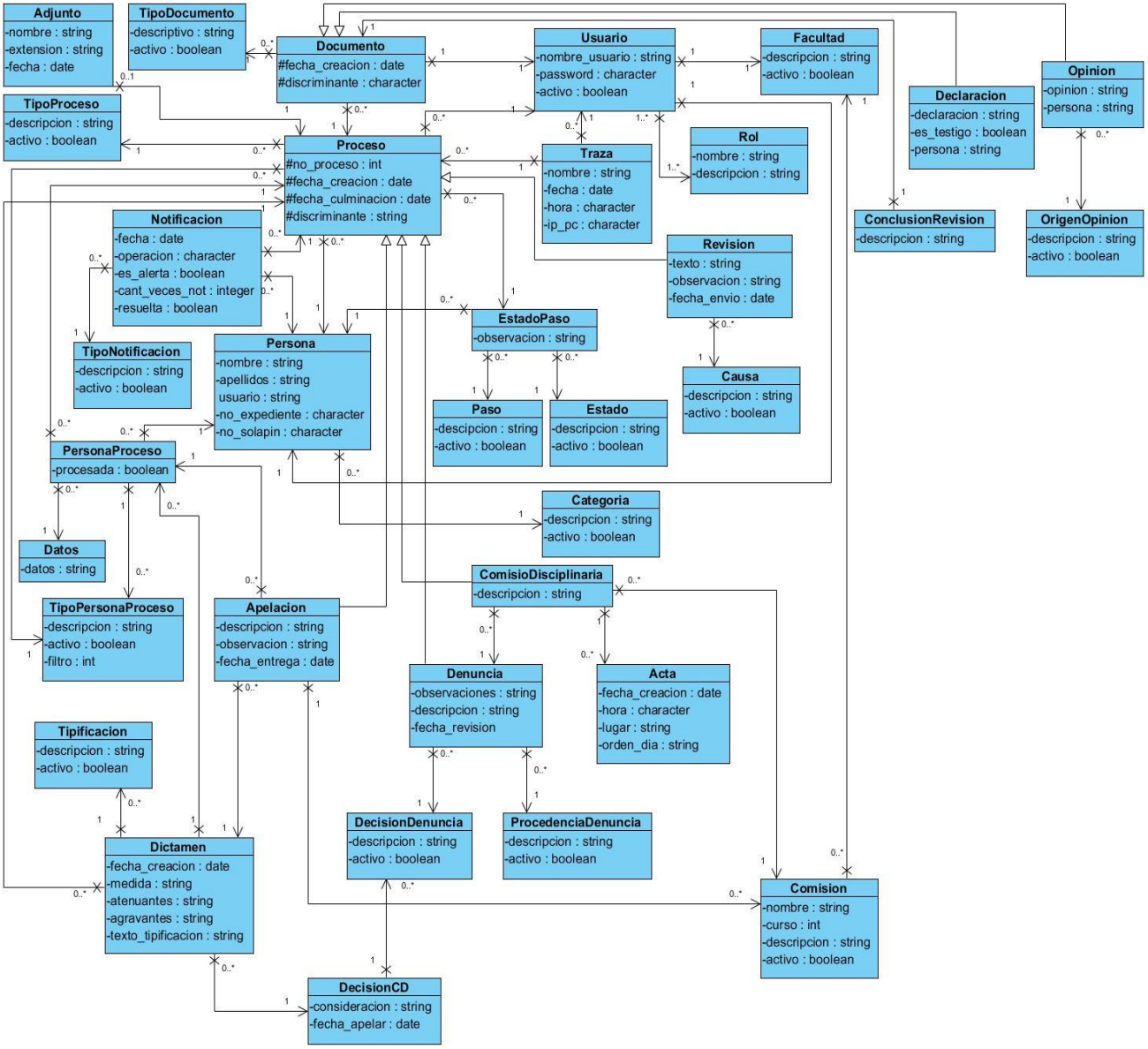

Imagen 4.Diagrama de clase entidades del sistema SIPCDII

2.7 Patrones de diseño

Patrones de diseño GRASP

- **Experto:** se pone en práctica en las clases controladoras, modelos y repositorios, que contienen toda la información necesaria para llevar a cabo las responsabilidades que le son asignadas.
- **Creador:** se refleja en las clases controladoras, donde se encuentran las acciones definidas para las operaciones lógicas del negocio en cuestión y se ejecutan cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades del negocio,

lo que evidencia que las clases controladoras son creadoras de dichas entidades, por ejemplo, en la acción *salvarActualizarApelación* se evidencia el uso de este patrón mediante la creación de la instancia de la clase entidad que contiene los datos del proceso de Apelación (clase *Apelación*).

- **Alta cohesión:** se evidencia en cada clase del diseño propuesto que realiza una labor única dentro del sistema y que colabore con las otras para llevar a cabo una tarea.
- **Bajo acoplamiento:** se pone en práctica, puesto que las clases poseen pocas relaciones entre sí, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás. Esta característica permitió potenciar la reutilización y disminuyó la dependencia entre las clases.
- **Controlador:** asigna la responsabilidad de controlar el flujo de eventos del sistema, a una clase específica, en este caso el controlador frontal (*app.php*), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Patrones GoF

Creacionales: Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de qué clase crear o cómo crearla (44).

- Fábrica abstracta: proporciona una interfaz para la creación de objetos, pero delega la responsabilidad de instanciarlo a sus subclases y promueve el encapsulamiento de las partes más variables del sistema. Se utiliza en las clases controladoras para la creación de formularios, mediante el método *createform()*, pasándole como parámetros el formulario a crear y el objeto de la entidad correspondiente.

Estructurales: Son los que plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos (44).

- Fachada: es una interfaz que actúa como mediadora entre dos capas en la aplicación. Este patrón se empleó en las clases controladoras como intermediarias entre el modelo y las restantes capas.

- Decorador: aplicado a la generación de vistas, la solución que ofrece dicho patrón es la de añadir funcionalidad adicional a las plantillas. Por ejemplo, añadir el menú y el pie de página a las plantillas que lo requieran, se trata de decorar las plantillas con elementos adicionales reutilizables. El sistema de plantillas twig, está provisto de un mecanismo de herencia, el cual se observa en el archivo denominado *layoutApelaciónhtml.twig* que contiene el Layout (plantilla global) de todas las páginas del registro. El mismo almacena el código HTML que es común a todas las páginas del registro, para no repetirlo en cada página, por lo que el Layout decora la plantilla.

Comportamiento: Plantea la interacción y cooperación entre las clases. Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal (44).

- Observador: este patrón define una dependencia “uno-a-muchos” entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente. Este patrón consiste en un objeto llamado “container”, que tiene una lista de observadores, llamados “listeners” y cuando se llama a un método del objeto se notifica a los observadores de que se ha producido ese evento. Se utilizó para la creación de trazas y la autenticación de los usuarios en la aplicación.
- Método plantilla: este patrón define el esqueleto de un algoritmo dejando las especificidades a las subclases y permitiendo que estas redefinan ciertos pasos del algoritmo.

2.8 Validación del diseño

Métrica tamaño operacional de clase (TOC)

A continuación se muestra el resultado de la aplicación de esta métrica al diseño.

Tabla 15.Responsabilidad

Responsabilidad	Cantidad de clases	Promedio
Baja	17	48,57142857
Media	11	31,42857143
Alta	3	8,571428571

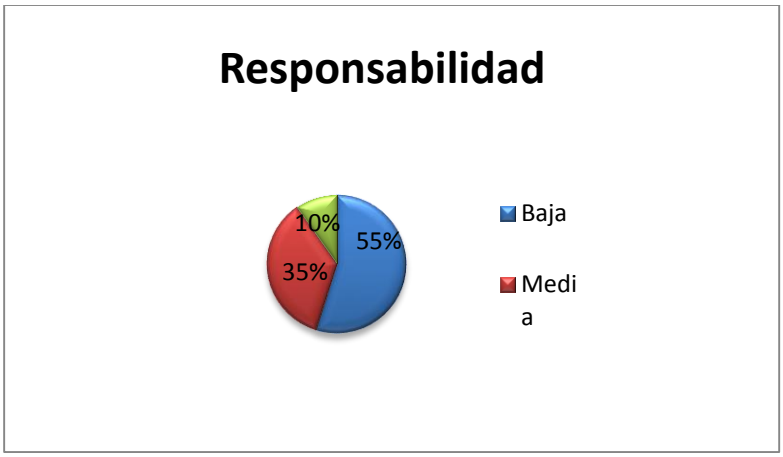

Imagen 5.Representación de la evaluación del atributo responsabilidad

Tabla 16.Complejidad

Complejidad	Cantidad de clases	Promedio
Baja	17	48,57142857
Media	11	31,42857143
Alta	3	8,571428571

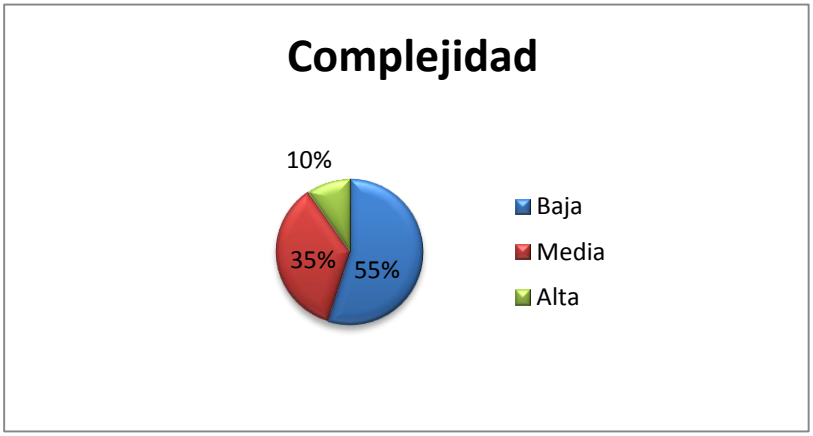

Imagen 6.Representación de la evaluación del atributo complejidad

Tabla 17.Reutilización

Reutilización	Cantidad de clases	Promedio
Alta	17	48,57142857

Media	8	22,85714286
Baja	2	5,714285714

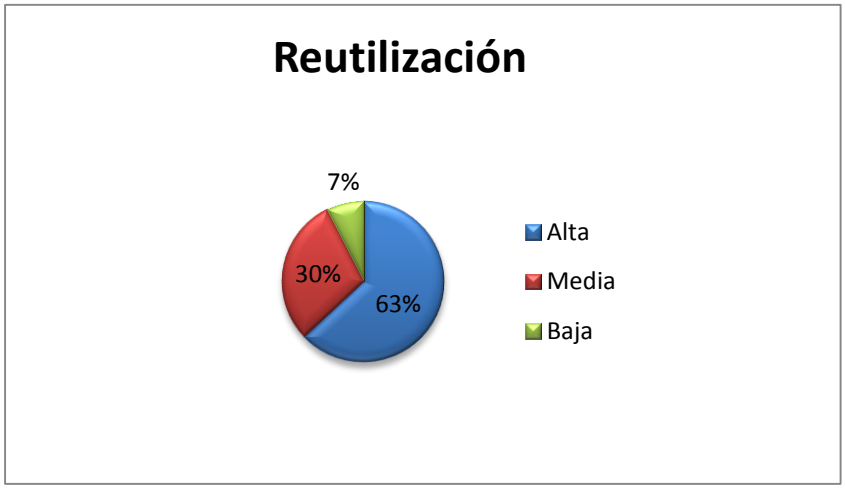

Imagen 7.Representación de la evaluación del atributo reutilización

Después de realizar la aplicación de la métrica TOC se llega a la conclusión de que el diseño propuesto cumple con lo requerido inicialmente, ya que el mismo dio como resultado una baja responsabilidad y complejidad de implementación y una alta reutilización, permitiendo la eliminación de clases repetidas, teniendo en cuenta que más de la mitad de las clases poseen baja dependencia respecto a otras, evidenciando que el sistema no tendrá una alta complejidad de implementación.

Relaciones entre clases (RC)

A continuación se muestra un ejemplo del resultado de la aplicación de esta métrica al diseño.

Tabla 18.Acoplamiento

Acoplamiento	Cantidad de clases	Promedio
Ninguno	2	5,714
Bajo	15	42,857
Medio	10	28,571
Alto	9	25,714

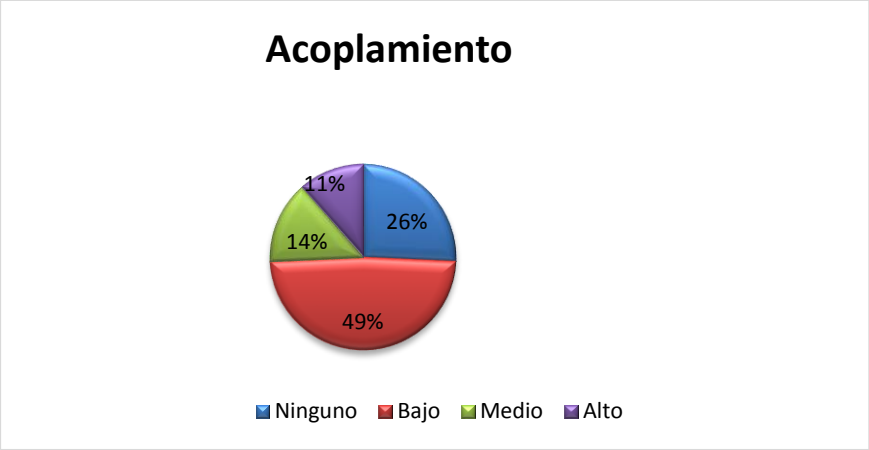

Imagen 8.Representación de la evaluación del atributo acoplamiento

Tabla 19.Complejidad de mantenimiento

Complejidad de Mantenimiento	Cantidad de clases	Promedio
Baja	23	63,88888889
Media	3	8,333333333
Alta	9	25

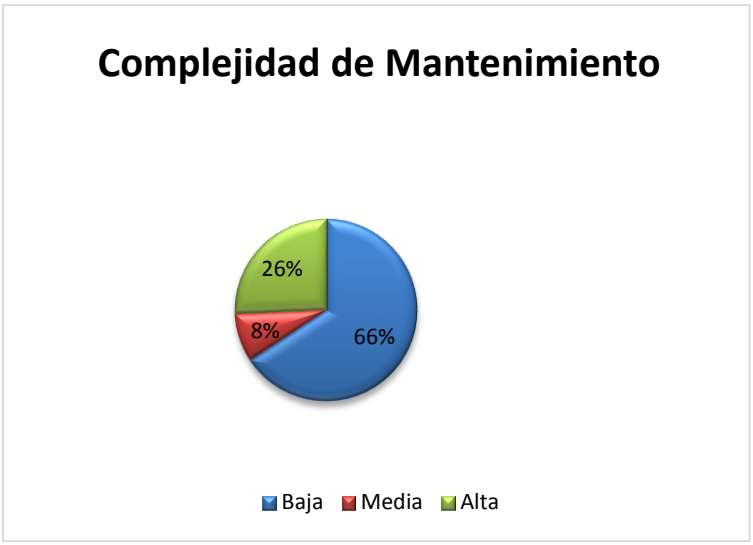

Imagen 9.Representación de la evaluación del atributo complejidad de mantenimiento

Tabla 20.Cantidad de pruebas

Cantidad de Pruebas	Cantidad de clases	Promedio
Baja	26	72,22222222

Media	0	0
Alta	9	25

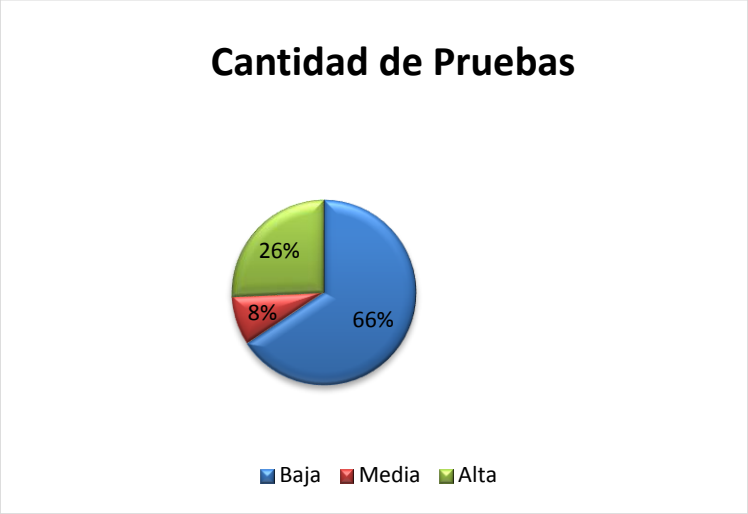

Imagen 10.Representación de la evaluación del atributo cantidad de pruebas

Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que el diseño propuesto cumple con los atributos de calidad que fueron evaluados, confirmando que las clases están diseñadas correctamente ya que presentan un bajo acoplamiento, poca complejidad de mantenimiento y poca cantidad de pruebas que implica que se necesitará menos esfuerzo a la hora de realizar pruebas unitarias a estas clases.

2.9 Conclusiones del capítulo

En el presente capítulo se abordaron aspectos fundamentales del análisis y diseño de la propuesta de solución concluyéndose que:

- El análisis y la validación de los requisitos demostraron que el sistema posee una baja complejidad de implementación, que los requisitos son estables y no poseen ambigüedades, posibilitando el tránsito confiable a la próxima fase propuesta por la metodología de desarrollo.
- La realización de las historias de usuarios facilitaron representar las funcionalidades y establecer la estimación de los tiempos requeridos para el desarrollo del proyecto.

- La estimación del esfuerzo y el plan de iteraciones permitieron crear un cronograma que facilita la implementación teniendo en cuenta los compromisos con el cliente, dando como resultado que tomaría un tiempo de 14 semanas.
- Se diseñaron un total de 30 tarjetas CRC que permitió definir la relación clase, responsabilidad, colaboración según las clases básicas identificadas en el sistema.
- La validación del diseño realizado, a través de métricas, permitió la proyección futura de resultados, basándose en la premisa de que un buen diseño es la base para generar una buena implementación.

Capítulo 3: Implementación y prueba.

3.1 Introducción

En el desarrollo de este capítulo se abordarán los aspectos fundamentales del proceso de implementación y pruebas. Se expondrán los detalles de la ejecución de la fase de desarrollo propuesta por la metodología. En el transcurso de esta fase se definirán los estándares de codificación, la seguridad del sistema y el modelo de datos. Se mostrarán los resultados de las pruebas de aceptación que fueron hechas con el objetivo de aumentar la calidad del sistema, reduciendo el número de errores detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su corrección.

3.2 Fase de desarrollo

En esta fase de la metodología se genera todo el código fuente de las especificaciones recogidas en las HU definidas para la solución, de manera tal que se cumplan los requisitos de todo el sistema. XP plantea que el mejor artefacto que se puede generar es el código, su buena calidad influirá en gran medida en el producto final. La misma es influenciada por los estándares de codificación los cuales serán tenidos en cuenta en esta fase. (13)

3.3 Estándares de codificación

Los estándares de codificación son una serie de convenciones que deben seguir los desarrolladores manteniendo buenas prácticas definidas por la ingeniería de software, para obtener un código fácil de comprender y de alta calidad. Posibilita que un equipo de programadores mantenga un código legible sobre el que se efectuarán luego revisiones; con el objetivo de regular la calidad de la implementación estableciendo un estándar de desarrollo común por el cual se rige la programación del sistema.

Cabecera del archivo

Los archivos *.php* inician con una cabecera específica que indique información relevante del mismo, tales como: nombre del autor, paquete al que pertenece, acceso al archivo, etc. Cada programador decide si debe agregar más datos. (Ver Imagen 11).

```
/*
 * @access public
 * @pakage SIPCD.CDBundle.Controller
 * @author Yisselly y Diana
 */
```

Imagen 11.Ejemplo de cabecera con datos relevantes

Cada archivo especificará el paquete que lo contiene y el uso de otras clases. (Ver Imagen 12**)**

```
<?php

namespace ArqBase\ComisionBundle\Controller;

use ArqBase\BaseBundle\Controller\BaseController;
use ArqBase\ComisionBundle\Entity\Denuncia;
use ArqBase\ComisionBundle\Form\Denuncia\AdicionarDenunciaType;
```

Imagen 12.Ejemplo de cabecera donde cada archivo específica el paquete que lo contiene

Comentarios en las funciones

Todas las funciones deben tener un comentario antes de su declaración, explicando qué hacen y los parámetros que reciben, en caso que sea necesario, así ningún programador tendrá que analizar el código de una función para conocer su utilidad, tanto el nombre como el comentario que acompañe a la función deben bastar para ello. (Ver Imagen 13 y 14).

```
/*
 * Detalles de la denuncia
 */
public function verDenunciaArchivadaAction($paso, $id_proceso) {
```

Imagen 13.Ejemplo 1 de comentario de una función

```
//busca la comision disciplinaria dado el id de proceso
$comision = $em->getRepository('ComisionBundle:ComisionDisciplinaria')->find($id_proceso);
```

Imagen 14.Ejemplo 2 de comentario de una función

Clases

Las clases serán colocadas en un archivo *.php* aparte, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo de la clase y siempre empezará en mayúscula. Mientras sea posible, se debe procurar que los nombres de clase tengan una sola palabra, en caso de más de una deberá ponerse en mayúscula a continuación de la otra. Las clases siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de la declaración de la clase explicando su utilidad. (Ver Imagen 15)

```
class DenunciaController extends BaseController {
```

Imagen 15.Ejemplo de nombre de una clase

Nombres de variables:

Los nombres deben ser descriptivos y concisos. No usar ni grandes frases ni pequeñas abreviaturas para las variables, ni adoptar la notación húngara 7 en el código. Siempre es mejor saber qué hace una variable con sólo conocer su nombre. Esto aplica para los nombres de variables, funciones, argumentos de funciones y clases. Todos los nombres de variables deben estar en minúscula. En caso de usar más de una palabra, esta será separada por un signo de underscore "_". (Ver Imagen 16)

```
if (is_null($id_proceso)) {
    $denuncia = new Denuncia();
```

Imagen 16.Ejemplo de nombre de la variable

Llaves

Siempre utilizar llaves para ganar en legibilidad. (Ver Imagen 17)

```
if (is_null($id_proceso)) {
    $denuncia = new Denuncia();
} else {
    $denuncia = $em->getRepository('ComisionBundle:Denuncia')->find($id_proceso);
    $denuncia->setFechaCreacion($denuncia->getFechaCreacionToString());
}
```

Imagen 17.Ejemplo de la utilización de llaves

Poner espacios entre signos

Si se tiene un signo se debe de poner espacios a ambos lados. (Ver Imagen 18)

```
if ($peticion->get('boton_select') == '_finalizar_') {
    $comision = $this->cambiarEstadoPaso($comision, static::iniciar_comision,
    $this->salvarObjeto($comision);
    return $this->redirect($this->generateUrl('index_asignar'));
} else {
    $denuncia = $this->cambiarEstadoPaso($denuncia, static::asignar_denuncia,
    $this->salvarObjeto($denuncia);
    return $this->redirect($this->generateUrl('asignar_denuncia_acd', array('
}
```

Imagen 18.Ejemplo de espacio entre signos

Cadenas de texto entre comillas

PHP tiene dos formas de poner strings o cadenas de texto. Con comillas simples y con comillas dobles. La diferencia es que si se usa comillas dobles y se coloca dentro del texto un nombre de

variable, el compilador lo interpretará y reemplazará por su valor. Por esta razón siempre se va a usar comillas simples a menos que se necesite hacer la interpolación de variables que permiten las dobles. Cuando se usan caracteres de escape "\" intensivamente, usar dobles comillas, por ser un caso especial. (Ver Imagen 19 y 20)

```
$denuncia = $em->getRepository('ComisionBundle:Denuncia')->find($id_proceso);
```

Imagen 19.Ejemplo de espacio entre signos

```
$texto = "La denuncia debe tener al menos un infractor.";
```

Imagen 20.Ejemplo de comillas dobles

Números dentro del código

Convertir en constante un número cuando se necesite. (Ver Imagen 21)

```
//tipo_proceso
    const denuncia = 1;
    const comision_disciplinaria = 2;
```

Imagen 21.Ejemplo de constante

No usar variables sin inicializar

Si no se tiene control sobre el valor de una variable, se verifica que esté inicializada. Pero sólo se usa esta opción cuando no se tenga el control o no se esté seguro que valor pueda tener la variable (Como en variables que llegan por parámetro o por GET, etc.). (Ver Imagen 21)

```
if (is_null($id_proceso)) {
    $denuncia = new Denuncia();
} else {
    $denuncia = $em->getRepository('ComisionBundle:Denuncia')->find($id_proceso);
    $denuncia->setFechaCreacion($denuncia->getFechaCreacionToString());
}
```

Imagen 22.Ejemplo de variables inicializadas

3.4 Diagrama de componente

Representa cómo un sistema de software es dividido en componentes, y las dependencias entre estos. Este tipo de diagrama contiene además interfaces y sus relaciones, pudiendo contener también paquetes que se utilizan para agrupar elementos del modelo.

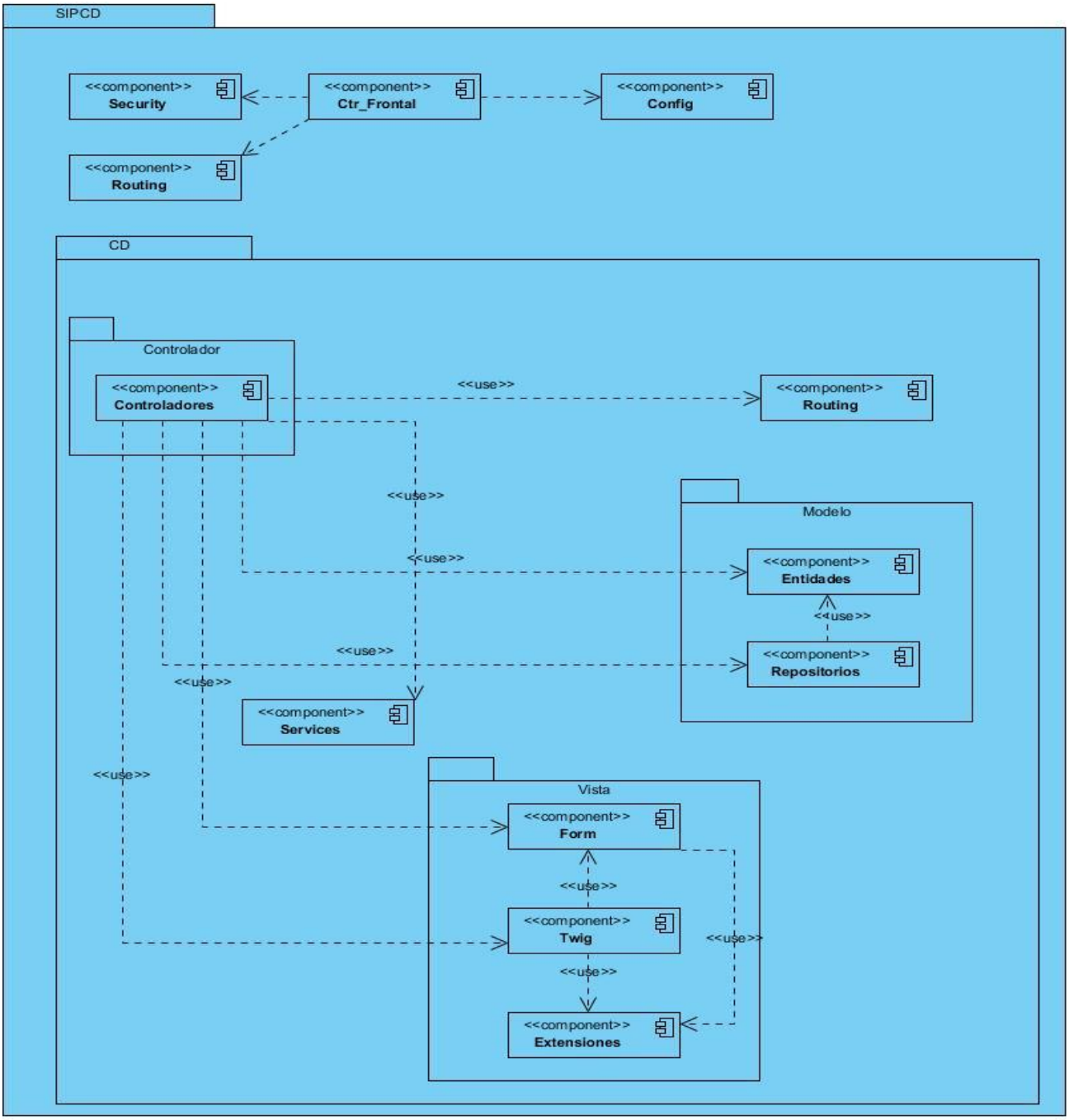

Imagen 23.Diagrama de componentes del sistema SIPCDII

El diagrama de componentes del sistema SIPCDII evidencia las interacciones entre los componentes del sistema y los del bundle[4] CD. SIPCDII cuenta con los componentes *Security* (para la seguridad),

[4] Un bundle es un conjunto estructurado de archivos que se encuentran en un directorio y que implementan una sola característica dividida en funcionalidades.

Routing (para las rutas de acceso), *Config* (para las configuraciones) y el *Ctr_Frontal* (controlador frontal) que recibe cada petición realizada. Estos componentes permiten un adecuado manejo de la seguridad y las configuraciones.

El bundle CD está compuesto por tres paquetes fundamentales, *Controlador* (encargado de dar respuesta a las peticiones de los usuarios), *Modelo* (contiene las clases entidades), *Vista* (encargado de la construcción y manejo de las interfaces de usuario). El paquete *Controlador* con el componente *Controladores*, para dar respuesta a las peticiones realizadas por los usuarios, el mismo posee una relación con el componente *Routing* para el manejo de las rutas definidas para el bundle y el componente *Services* que ofrece los servicios públicos a los que se puede acceder desde cualquier parte del sistema SIPCDII. El componente *Controladoras* se relaciona con el componente *Entidades* para el manejo de las entidades, el componente *Entidades* es usado por el componente *Repositorio* que contiene las clases repositorios encargadas de realizar las consultas a la base de datos. El paquete *Vista* incluye los componentes *Form* (encargado del trabajo con los formularios), *Twig* (para las interfaces de usuarios y plantillas del módulo) y *Extensiones* (posee los javascript y css).

3.5 Diagrama de despliegue

Para la utilización de la aplicación el usuario debe conectarse a esta mediante la Pc_Usuario o la Pc_Cliente utilizando un navegador web. Dicha PC interactuará mediante una conexión segura con el servidor, el cual brinda servicios web y de base de datos. Además el sistema debe contar con una impresora conectada a la Pc_Cliente para imprimir los documentos generados.

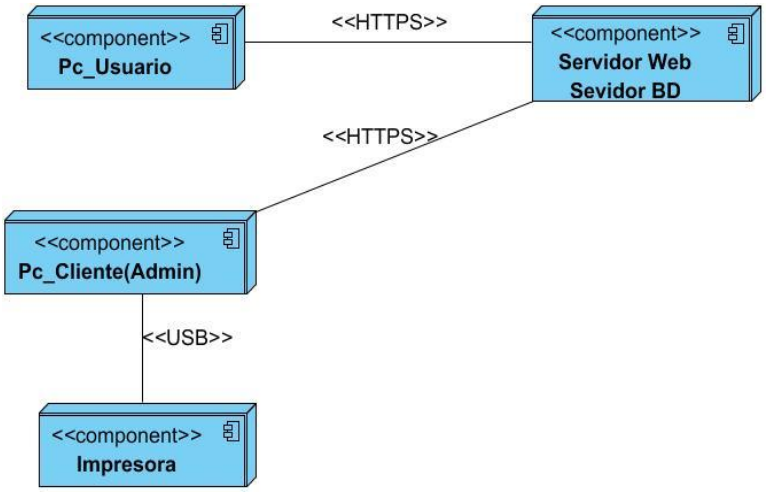

Imagen 24.Diagrama de despliegue del sistema SIPCDII

Para el despliegue de la aplicación, se debe contar con:

- Un servidor web con los módulos: php5-xsl, php5-pgsql, php5-mencache, php5-cli, php-apc, php-soap, el php5-intl y php5-ldap.
- Un servidor de base de datos y copias de seguridad.
- Pc clientes con un navegador web (Mozilla Firefox 20.0 o superior).
- Opcionalmente periféricos como impresoras y escáneres.

3.6 Modelo de datos

El modelo de datos representa las tablas (entidades) importantes para el funcionamiento del negocio y muestra los datos que serán contenidos en el sistema. Este cuenta con un total de 31 tablas, donde 6 de ellas son nomencladores, encargados de gestionar conceptos específicos del negocio ya predefinidos. Las restantes tablas facilitan el registro de datos que son necesarios para el funcionamiento de la aplicación.

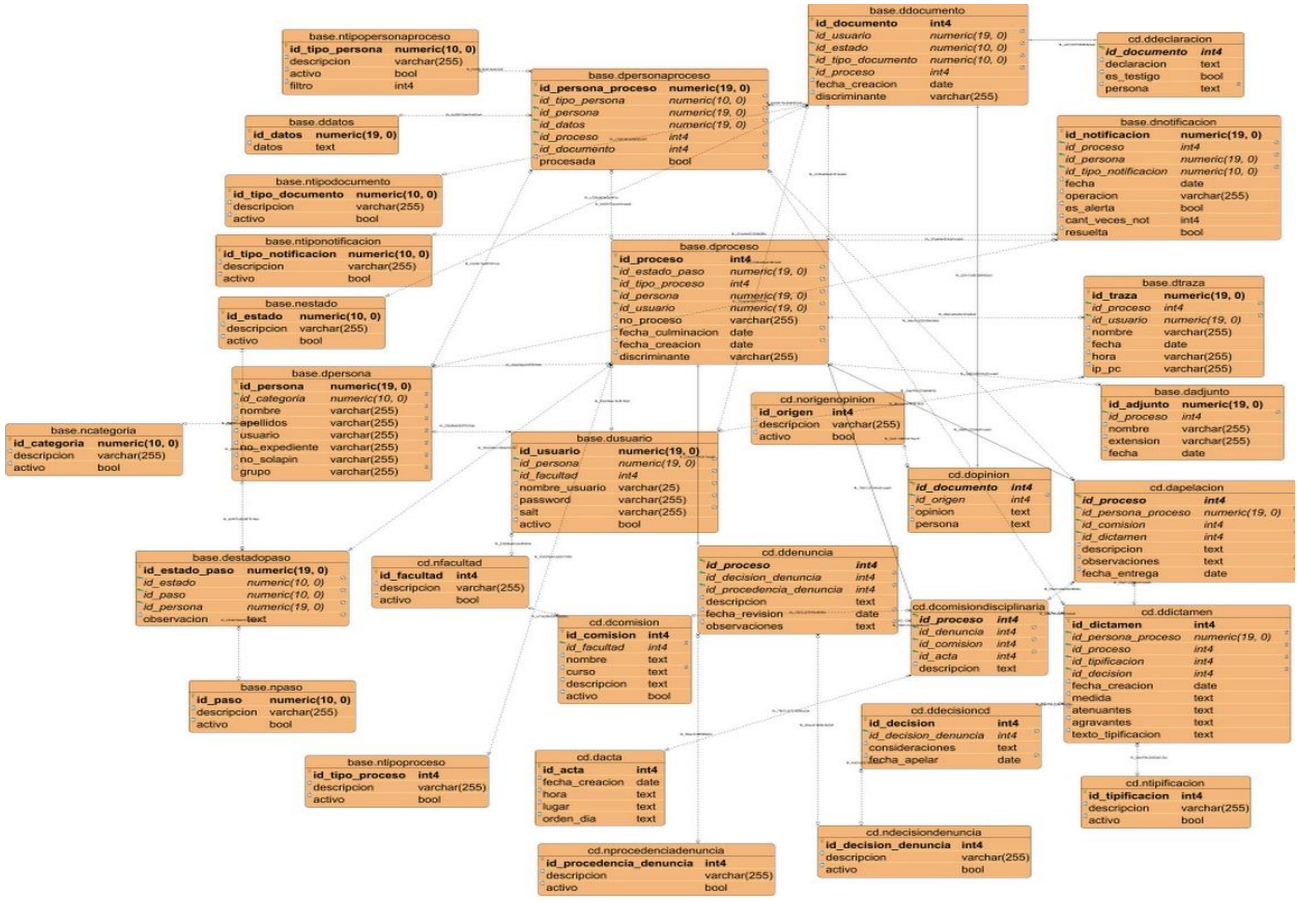

Imagen 25.Modelo de datos del sistema SIPCDII

3.7 Fase de pruebas

Se escogen algunas de las pruebas planteadas por (Pressman, 2007) para aplicarlas a la aplicación implementada. Este autor propone como pauta a seguir que el proceso debe realizarse desde la parte más pequeña del software hasta la más grande, formando una espiral entre los niveles de prueba. El objetivo fundamental de las pruebas es determinar si el código generado funciona correctamente y realiza las operaciones deseadas. Además permiten comprobar que la solución cumple con los requisitos acordados con el cliente. Por último, verifican que el sistema sea capaz de funcionar normalmente en un ambiente parecido o en el ambiente de despliegue.

3.7.1 Prueba de caja negra

A continuación se muestra un ejemplo (Ver tabla 21) con los escenarios de un caso de prueba de uno de los requisitos que componen la funcionalidad “Adicionar Apelación”. La tabla muestra datos como: descripción, respuesta del sistema y el flujo central. El resto de estos se incluyen en el documento Casos de Pruebas de SIPCDII.

Tabla 21.Caso de Prueba de la funcionalidad Adicionar Apelación

Escenario	Descripción	Fecha de creación	Fecha de entrega	Descripción	Observaciones	Respuesta del sistema	Flujo central
EC 1.1 Gestionar apelación con los valores correctos	Gestionar una apelación con valores correctos.	V	V	V	V	El sistema crea una apelación con los valores adicionados.	1. El usuario selecciona la opción Apelaciones
		01/05/2015	14/05/2015	El estudiante cometió fraude.	El estudiante a cometido indisciplinas en varias ocasiones.		
EC 1.2 Adicionar Apelación con los valores incorrectos	Adicionar una Apelación con valores incorrectos.	N/A	V	V	V	El sistema no permite insertar ese valor	
		fecha	14/05/2015	El estudiante cometió fraude.	El estudiante a cometido indisciplinas en varias ocasiones.		

		V	N/A	V	V		
		01/05/2015	Fecha	El estudiante cometió fraude.	El estudiante a cometido indisciplinas en varias ocasiones.		
		V	V	N/A	V		
		01/05/2015	14/05/2015	texto	El estudiante a cometido indisciplinas en varias ocasiones.		
		V	V	V	N/A		
		01/05/2015	14/05/2015	El estudiante cometió fraude.	texto		

Se obtuvieron un total de 36 casos de pruebas los cuales fueron creados para realizar las pruebas de funcionalidad al sistema. Estas pruebas fueron confeccionadas en tres escenarios, para cuando los datos sean: correctos, incorrectos o existan campos incompletos.

Pruebas aplicadas por el centro de calidad de Centro de gobierno electrónico(CEGEL)

A partir de los casos de pruebas entregados los especialistas del centro aplicaron la prueba de caja negra o de funcionalidad las cuales constituyen pruebas de liberación. Estas pruebas se realizan con el fin de comprobar que el sistema funciona correctamente y presenta una interfaz agradable a la vista de los usuarios. (Ver tabla 22)

Tabla 22.Resumen de las pruebas aplicadas por el centro CEGEL

	Primera Iteración	Segunda Iteración	Total
Cantidad de no conformidades	12	4	16
Cantidad de no conformidades de casos de pruebas	0	0	0
Cantidad de no conformidades de la aplicación	12	4	16

Cantidad de no conformidades que no proceden	3	0	3
Cantidad de no conformidades resueltas	9	4	13

La aplicación fue sometida a dos iteraciones encontrándose un total de 16 no conformidades .En la primera iteración se detectaron 12 no conformidades donde 3 no procedieron y fueron resueltas 9.

Para la segunda iteración se encontraron 4 no conformidades y fueron resueltas las 4. Después de concluida las pruebas de funcionalidad el área de calidad el centro CEGEL liberó la aplicación entregándosele a los desarrolladores un acta en la que consta que la aplicación está apta para ser utilizada. (Ver Anexo 46)

3.8 Validación de las variables

Para darle cumplimiento al objetivo planteado se validaron las variables identificadas. A continuación se muestran los resultados obtenidos:

Variable independiente: informatización del proceso de comisiones disciplinarias.

Variables dependientes: oportunidad de su información.

Enmarcado al proceso de CD que se realiza en la UCI una buena oportunidad de la información es cuando esta está disponible siempre y cuando haga falta, que la información no se pierda, que solo pueda acceder a esta la persona autorizada y en el tiempo requerido. A partir de este concepto se realizó una comparación entre la forma en que se realiza actualmente el proceso de CD y como sería el mismo con la utilización del sistema. (Ver tabla 23).

Los indicadores por los que se evaluó la oportunidad de la información son:

- Disponibilidad de la información
- Tiempo de obtención de la información
- Integridad de la información

Tabla 23.Tabla comparativa de los indicadores seleccionados

Indicador	Antes	Después
Disponibilidad de la información	Para una persona disponer de la información tiene que dirigirse al departamento de secretaría docente de cada facultad que es	Para una persona disponer de toda la información necesaria solo tendrá que acceder al sistema SIPCD, incluso si

	donde está archivado el expediente de cada comisión realizada, corriendo el riesgo de que el departamento se encuentre cerrado o no encontrarse el encargado de los expedientes, provocando no poder disponer de la información deseada.	la misma se encuentra archivada en secretaría.
Tiempo de obtención de la información	La generación de documentos y la obtención de reportes estadísticos se realizan de forma manual o mediante la utilización de software de ofimática, lo cual da margen a errores humanos. La información se movía de un lado a otro provocando pérdida de tiempo debido a que la entrega de cada información generada es manual.	Se accede a la información a través del sistema disminuyendo considerablemente el tiempo en que se realiza cada operación. La generación de documentación se realiza mediante plantillas ya definidas. Mediante la utilización de las funcionalidades de búsqueda se obtiene la información de los procesos. Los reportes estadísticos son generados por la aplicación, reduciendo el tiempo de obtención y la posibilidad de error.
Integridad de la información	A la información puede acceder cualquier personal que no esté autorizado a obtener la misma, así como hay facilidad de pérdida de información, existe riesgo de deterioro de los documentos.	A la información sólo tiene acceso las personas autorizadas debido a que en el sistema la seguridad se maneja a través de roles que son asignados por el administrador. Existe muy poca probabilidad de pérdida de información debido a que esta se almacena correctamente en el servidor y se disponen de copias de seguridad.

3.9 Conclusiones del capítulo

El presente capítulo abordó las fases finales del proceso de desarrollo establecido por la metodología concluyendo que:

- En la fase de desarrollo se estableció el estándar de codificación a utilizar lo que permitió un mejor entendimiento del código.
- La modelación de los artefactos diagrama de componentes, diagrama de despliegue y modelo de datos, permitieron al equipo de desarrollo un mejor entendimiento del sistema.
- En la fase de prueba se examinaron las funcionalidades del sistema a través de las pruebas de caja negra, obteniéndose durante las 3 iteraciones 16 no conformidades, de las cuales 9 procedieron y se resolvieron y 4 no procedieron.
- Los resultados de las pruebas, generaron una retroalimentación permanente entre el equipo de desarrollo y el personal del centro CEGEL, mostrándoles a los desarrolladores la calidad de su trabajo.

Conclusiones generales

Se concluye que:

- El análisis de sistemas informáticos relacionados con temas disciplinarios, demostró que es más factible crear una solución propia que responda a las particularidades de la UCI.
- La realización del análisis mediante la metodología de desarrollo seleccionada permitió lograr mayor organización, planificación y ejecución de las actividades e hitos a cumplir.
- El diseño permitió obtener una vista de la estructura estática del sistema teniendo en cuenta distintos patrones y se contribuyó a la escalabilidad de la solución mediante el uso de estilos arquitectónicos como el MVC, lo cual propició la alta reutilización de las clases diseñadas.
- El proceso de implementación permitió desarrollar la solución a partir del diseño elaborado, cumpliendo con todos los requisitos identificados. El uso de estándares de codificación contribuyó a la legibilidad del código y a futuros mantenimientos de la solución.
- La validación de la solución demostró que las funcionalidades del sistema son correctas y los componentes funcionan de forma adecuada.

Se obtuvo el sistema SIPCD 2.0 para el proceso de las comisiones disciplinarias en la UCI, contribuyendo a la oportunidad de la información que gestiona.

Recomendaciones

- Realizar el despliegue del sistema SIPCD en la UCI.
- Generalizar el sistema a las demás universidades del país.
- Continuar perfeccionando las funcionalidades y reportes del sistema SIPCD.

Bibliografía

1. **Figueredo, Pedro Espronceda.** La Batalla de Ideas una nueva etapa de lucha de los cubanos. *Radio Granma.* [En línea] 5 de diciembre de 2013. [Citado el: 10 de enero de 2015.] http://www.radiogranma.icrt.cu/index.php/component/k2/item/1898-la-batalla-de-ideas-una-nueva-etapa-de-lucha-de-los-cubanos.

2. **Informáticas, Universidad de la Ciencias.** UCI. *UCI.* [En línea] 2012. [Citado el: 7 de mayo de 2015.]

3. **Martinto, MSc. Pedro Carlos Pérez.** *Diseño metodológico de la investigación científica.* s.l. : UCI.

4. **Martinto, MSc. Pedro Carlos Pérez.** *Diseño metodológico de la investigación científica.* s.l. : UCI, 2013.

5. **Real Academia Española.** [En línea] 2015. [Citado el: 8 de febrero de 2015.] http://lema.rae.es/drae/.

6. **Universidad de Extremadura.** BATUSI. [En línea] [Citado el: 4 de febrero de 2015.] http://www.unex.es/organizacion/servicios-universitarios/servicios/siue/funciones/servicio_usuario/BATUSI.

7. **UNIVERSIDAD DE MURCIA.** Dumbo. [En línea] 2015. [Citado el: 1 de febrero de 2015.] http://www.um.es/.

8. **Carvajal Riola, José Carlos.** *Metodologías Ágiles: Herramientas y Modelo de desarrollo para aplicaciones.* España : s.n., 2008.

9. **Pressman, Roger S.** Un enfoque prácitco,6ta edición.Métricas del producto para el software. *Ingeniería de Requisitos.* 2015.

10. **Pressman, Roger.** *Ingeniería del Software, Un enfoque práctico.* s.l. : Mc Graw Hill, 2007.

11. SCRUM.org. *Improving the Profession of Software Development.* [En línea] [Citado el: 15 de enero de 2015.] https://www.scrum.org/.

12. **Letier, Patricio y Padanés, María Carmen.** *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : s.n., 2007.

13. **ing. Informático Sanchez Mendoza, María A.** *Metodologías De Desarrollo De Software.* Perú S.A.C. : s.n., Junio 7 del 2004.

14. **Escalona, M. J y Koch, N.** Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.

15. CodeIgniter. *CodeIgniter.* [En línea] [Citado el: 10 de febrero de 2015.] http://www.codeigniter.com/.

16. ¿Porque elegir symfony 2? [En línea] 2014. [Citado el: 1 de febrero de 2015.] http://www.4rsoluciones.com/por-que-elegir-symfony2/..

17. Doctrine . *Doctrine Documentation.* [En línea] [Citado el: 10 de febrero de 2015.] http://www.doctrine-project.org/documentation/manual/1_0/en/introduction-to-models.

18. **PostgreSQL.** PostgreSQL. [En línea] 3 de diciembre de 2012. [Citado el: 17 de enero de 2015.] http://www.postgresql.org/about/..

19. Bases de Datos en MySQL. [En línea] [Citado el: 5 de febrero de 2015.] http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf.

20. Microsoft. *SQL Server.* [En línea] [Citado el: 10 de febrero de 2015.] http://www.microsoft.com/es-es/server-cloud/products/sql-server/.

21. Windows Server. [En línea] [Citado el: 10 de febrero de 2015.] https://technet.microsoft.com/es-es/library/cc753433%28v=ws.10%29.aspx.

22. **Apache .** The Apache Software Foundation. *The Apache Software Foundation.* [En línea] [Citado el: 20 de Noviembre de 2012.] http://www.apache.org/.

23. **Salinas Caro, Patricio y Histchfeld K, Nancy.** Tutorial de UML. [En línea] [Citado el: 7 de febrero de 2015.] http://www.dcc.uchile.cl/~psalinas/uml/.

24. BPMN Business Process Modeling Notation. [En línea] 2014. [Citado el: 9 de febrero de 2015.]

25. ApredéraProgramar. [En línea] 2014. [Citado el: 9 de febrero de 2015.] http://www.aprenderaprogramar.com/index.php?option=com_content&id=492:ique-es-php-y-ipara-que-sirve-un-potente-lenguaje-de-programacion-para-crear-paginas-web-cu00803b&Itemid=193.

26. PHP. *PHP.* [En línea] [Citado el: 10 de febrero de 2015.] http://php.net/.

27. El motor de plantillas para PHP que separa el código HTML. [En línea] 2014. [Citado el: 8 de febrero de 2015.]

28. **Bradenbaugh, Jerry.** *Aplicaciones JavaScript.* s.l. : Anaya Multimedia, 2000.

29. jQuery. [En línea] [Citado el: 10 de febrero de 2015.] https://jquery.com/.

30. **Musciano, Chuck y Kennedy, Bill.** *HTML, La Guía Completa.* s.l. : O'Reilly, 1999.

31. Control de versiones Open Source. [En línea] [Citado el: 5 de febrero de 2015.] http://svnbook.red-bean.com/index.es.

32. **NetBeans.** NetBeans. *NetBeans.* [En línea] 2012. [Citado el: 16 de noviembre de 2012.] http://netbeans.org/.

33. Pgadmin, PostgreSQL tools. [En línea] [Citado el: 7 de febrero de 2015.] http://www.pgadmin.org/..

34. **Leon, Eduardo.** *Tutorial Visual Paradigm for UML.* 2011.

35. Desarrollo Web. [En línea] [Citado el: 7 de febrero de 2015.] http://www.desarrolloweb.com/articulos/que-es-mvc.html.

36. **de la Torre LLorente, César.** *Guía de Arquitectura N-Capas Orientada al Dominio con .NET 4.0.* España : s.n., 2010.

37. **Larman.** *UML y Patrones,.* s.l. : Tomo I,.

38. **liliam Celia Beyris Soulary, Ludisley La Torre Hernández, Mariela Cepero Núñez.** *Proceso de medición y análisis para el polo de hardware y automática.* s.l. : Grupo Editorial Ediciones Futuro, 2010.

39. La prueba del software. [En línea] [Citado el: 8 de febrero de 2015.]

40. **Sánchez Fornaris, Maite y Alcantara Rabí, Dayanis.** *Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas. .* 2009.

41. **Huebe, Lourdes Pérez.** *Ingeniería de Requerimientos.* Pachuca : s.n., 2005.

42. **Sommerville, Ian.** *Ingeniería del software.* Madrid : Pearson Educación S.A., 2005. 84-7829-074-5.

43. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* s.l. : PEARSON, 1999.

44. **Bahit, Eugenia.** *POO y MVC en PHP. El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC.*

45. **Ubuntu.** Guía Ubuntu. [En línea] [Citado el: 12 de enero de 2013.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

46. VisualParadigm. [En línea] http://www.visual-paradigm.com/product/vpuml/.

47. **Jacobson, Ivar, Booch, Greddy y Rumbaugh, James.** *El proceso unificado de desarrollo de software. Traducción al español.* Madrid : Pearson Educacion S.A, 2000.

48. **Kotonya, G y Sommerville.** *I. Requirements Engineering: Processes and Techniques.* 2000.

49. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico. 5ta edición.* s.l. : McGraw-Hill, 2002. ISBN: 8448132149.

50. *Ayuda de RUP. Rational Unified Process.* s.l. : Copyright (C) IBM Corporation 1987. Versión 7.0.12005.

51. Sitio oficial de php. [En línea] [Citado el: 12 de noviembre de 2012.] http://www.php.net/manual/es/intro-whatcando.php.

52. Manual de twig en español. [En línea] http://gitnacho.github.io/Twig/.

53. Aprendiendo UML en 24 horas. [En línea] http://www.scribd.com/doc/3708746/Aprendiendo-UML-en-24-horas.

54. **Fabien Potencier, François Zaninotto.** *Symfony, la guía definitiva.* 2008.

55. **Pacheco, Nacho.** *Manual de Twig.* 2011.

56. **Erika Camacho, Fabio Cardezo, Gabriel Núñez.** *Arquitecturas de software. Guía de estudio.* 2004.

57. **Stefan Kung, Lubbe Onken, Simon Large.** TortoiseSVN. *TortoiseSVN.* [En línea] [Citado el: 4 de Febrero de 2013.] http://tortoisesvn.net/docs/release/TortoiseSVN_es/index.html.

58. **Craig Larman, Prentice Hall.** El modelo de diseño. *UML y Patrones 2ª Edición.* 2003.

59. **Pantaleón, Marta E. Zorrilla.** Universidad de Cantabria. *Universidad de Cantabria.* [En línea] [Citado el: 21 de Marzo de 2013.] http://personales.unican.es/zorrillm/PDFs/Docencia/Master/02%20-%20Modelos%20de%20datos%20ER-UML-relacional.pdf.

60. **Netciel.** NETCIEL. *NETCIEL.* [En línea] 2010. [Citado el: 2 de Febrero de 2013.] http://www.netciel.com/es/stack-de-desarrollo-web/42-lapp-web-stack.html.

61. **Potenccer, Fabian.** [En línea] 2013. [Citado el: 4 de Febrero de 2013.] http://fabien.potencier.org/article/49/what-is-symfony2.

62. **PRUEBASDESOFTWARE.** pruebasdesoftware. *pruebasdesoftware.* [En línea] 2005. [Citado el: 4 de Febrero de 2013.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

63. **Bonanata, Maximiliano.** *Programación y algoritmos.* s.l. : MP ediciones S.A. Página 17.

64. **Source, JavaScript.** The JavaScript Source. *The JavaScript Source.* [En línea] [Citado el: 5 de Diciembre de 2012.] http://www.javascriptsource.com/faq/index.html.

65. **Pressman, Roger S.** *Ingeniería de software un enfoque práctico.* 1998. 84-481-3214-9.

66. Curso práctico de Modelado de Negocios con BPMN y UMl. [En línea] 2014. [Citado el: 17 de enero de 2015.] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.

67. Testing Software. [En línea] [Citado el: 7 de febrero de 2015.] http://blog.elevenpaths.com/2014/09/qa-pruebas-para-asegurar-la-calidad-del.html.

68. Universidad de San Martín de Porres (USMP). [En línea] 8 de febrero de 2015. http://www.usmp.edu.pe/.