Universidad de las Ciencias Informáticas
Facultad 3

Desarrollo de una aplicación informática para el proceso de gestión de las comisiones disciplinarias en la Facultad 3

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

Autor:

Etna Lina LavinScott

Tutores:

Ing. Liset González Polanco

Ing. Lazaro Heredia Maso

Ciudad de la Habana

Curso 2013-2014

La posibilidad de realizar un sueño es lo que hace que la vida sea interesante

Paulo Coelho

DECLARACIÓN DE AUTORÍA

Se declara que Etna Lina Lavin Scott es el único autor del trabajo de diploma Desarrollo de una aplicación informática para el proceso de gestión de las comisiones disciplinarias en la Facultad 3 y se le reconocen a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Firma del Autor

Etna Lina Lavin Scott

Firma del tutor

Ing. Liset González Polanco

Firma del tutor

Ing. Lazaro Heredia Manso

Agradecimientos

La vida más que un regalo de dios es una bendición que nos da nuestros padres en especial nuestras madres; por eso hoy le agradezco a mi mamá por haber permitido que hoy yo respire, ame, llore y sueñe; sueño que hoy estoy cumpliendo, el de ser ingeniera en ciencias informática.

No me quiero ir sin agradecer a todos aquello que de una forma u otra alegraron mi estancia en la universidad, aquellos que compartieron conmigo malos y buenos momentos.

También quiero agradecerle a mi amada abuela que más que eso fue mi segunda madre, a mi adorable hermano, a mis tíos a mis primos como también a mi pareja Yoyi que lo amo muchísimo.

Y no me puedo despedir, ni mucho menos terminar estos agradecimientos sin antes decirles "GRACIAS" con toda sinceridad a mis tutores que de no ser por ellos parte de mi sueño aún no se estaría cumpliendo; en especial a mi tutor Lazaro que al menos para mí fue mi compañero de tesis.

A todos los que formaron parte de mi vida en esta universidad

GRACIAS

Dedicatoria

Necesitaría varias hojas para dedicarle este trabajo a todos mis seres queridos como mi abuela, mi hermano, mi tía y mis primitos. En la línea anterior faltó mencionar una persona, sí, faltó mencionar a mi mamá y a mi papá que para mí es la misma persona: Sandra Scott Gutiérrez así se nombra y se apellida la persona que se quitó todo lo que tenía para dárselo a sus hijos, la persona que sacrificó su felicidad por dársela a sus hijos, la persona que trabaja día a día para darle a sus hijos todo lo que necesitan y lo que no también, por esa persona hoy soy ingeniera en ciencias informáticas y por tal razón con todo el amor de mundo, si tengo que quitarme el corazón y dárselo en una bandeja; le dedico todo este trabajo a mi madre.

“Ama profunda y apasionadamente será la única forma de vivir la vida por completo”

Resumen

El presente trabajo de diploma tiene como propósito informatizar el proceso de comisiones disciplinarias en la Facultad 3. Actualmente no se cuenta con un sistema que informatice dicho proceso y se detectan afectaciones en la gestión de la información atendiendo a cuatro factores de evaluación que son: Calidad de la información, Oportunidad de la información, Cantidad de información y Relevancia de la información. Para su cumplimiento 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, de 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 Facultad 3 de un sistema que gestiona de forma automatizada muchos de los procesos de las comisiones disciplinarias contribuyendo a la gestión de la información.

Palabras claves

Comisión disciplinaria, Facultad 3, gestión de información, software.

Índice

INTRODUCCIÓN

La introducción de las Tecnologías de la Información y las Comunicaciones (TIC) en la vida del hombre ha llevado a que este redefina la forma en que ejecuta los procesos de su vida cotidiana y por tanto como interactúa en la sociedad. Desde los comienzos de la humanidad fue necesario el reconocimiento de un conjunto de leyes, normas de conducta que hicieran posible la vida pacífica, el desenvolvimiento normal de las actividades humanas, con el convencimiento que la violación de tales normas trae consigo una sanción, un castigo. La historia demuestra cómo fueron dándose pasos desde la ley de Talión “Ojo por ojo, diente por diente” y los distintos derechos ya escritos como el sistema romano, el francés, el anglosajón. Estos escritos determinaban las reglas y los castigos definidos para su incumplimiento.

En Cuba actualmente se posee como máxima ley, la Constitución Socialista de 1987, que establece las normas fundamentales a seguir por todos los cubanos. Las instituciones, en el país, también poseen un conjunto de reglas, por lo que si alguien incumple con alguna de las políticas determinadas, deberá ser sancionado por una comisión destinada para esta tarea. Específicamente el Ministerios de Educación Superior (MES) se rige por el reglamento disciplinario para los estudiantes de la educación superior, actualmente el decreto ley 240 del año 2007. La Universidad de las Ciencias Informáticas (UCI), al pertenecer al MES, está sujeta a este reglamento, por lo que si un estudiantes incumple con algún o algunos de los artículos descritos en él, debe de ser procesado en Comisión disciplinaria, que es el órgano designado por una facultad para procesar una denuncia.

En la Facultad 3 actualmente el proceso de Comisiones Disciplinarias (CD) se realiza de forma manual, provocando afectaciones en la gestión de la información que se obtiene del trabajo de las comisiones disciplinarias, ya que la documentación generada es archivada al concluir un proceso. La obtención de reportes para los análisis que se requieren por los niveles de dirección de la facultad es manual, lo que provoca atrasos y da margen a que el resultado no sea confiable. Las consultas a los documentos son complejas y se dificulta su acceso futuro, lo cual afecta la disponibilidad de la información. Aunque existen intentos por informatizar dichos procesos no se cuenta con un sistema en la facultad.

A partir de la situación problemática identificada se plantea el siguiente **problema a resolver:** ¿cómo contribuir al proceso de Comisiones Disciplinarias para mejorar la gestión de la información en la Facultad 3? Teniendo en cuenta el problema identificado el **objeto de estudio** es: el proceso de las CD en la Enseñanza Superior teniendo como **objetivo general**: desarrollar un sistema informático para informatizar el procesos de CD que contribuya a mejorar la gestión de la

información en la Facultad 3, enmarcándose en el **campo de acción**: el proceso de las CD en la Facultad 3. Como **idea a defender**: si se informatiza el proceso de CD entonces mejora la gestión de la información en la Facultad 3.

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **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 un análisis del sistema para el proceso de comisiones disciplinarias, mediante la metodología de desarrollo de software establecida.
- Elaborar el 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 darle cumplimiento a los **objetivos específicos** de la investigación se trazan las siguientes **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 metodología de software, herramientas y tecnologías a utilizar.
- Implementación de las funcionalidades asociadas a cada historia de usuario.
- Realización del diseño de casos de prueba asociados a las funcionalidades implementadas.
- Ejecución de pruebas de software al código fuente obtenido en la implementación de las diferentes funcionalidades.
- Validación de los resultados obtenidos a través de pruebas de caja negra asociados a las diferentes funcionalidades.
- Validación de la propuesta de solución de la investigación frente al problema planteado.

Para realizar la investigación se utilizaron los siguientes métodos de investigación:

Métodos teóricos:

- Histórico-lógico: favoreció el estudio de los antecedentes, el desarrollo, las regularidades y tendencias actuales de gestión, planificación de las comisiones disciplinarias así como los sistemas existentes en el mundo y en Cuba.
- Analítico-sintético: permitió el procesamiento de la información y arribar a las conclusiones prácticas y teóricas de la investigación, así como precisar las herramientas utilizadas para el diseño y la implementación del sistema.
- Modelación: se utilizó para la creación de abstracciones que explican la realidad, este método es un instrumento de la investigación que permite la creación de modelos, descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

Método empírico:

- Entrevista: se aplica a la asesora de trabajo educativo con el fin de saber las principales deficiencias que afectan este proceso (comisión disciplinaria) en la UCI así como las necesidades y los requerimientos para elaborar la propuesta.

Estructura del trabajo de diploma

Para el desarrollo del presente trabajo se proponen tres capítulos, los cuales están estructurados de la siguiente forma:

Capítulo 1: Fundamentación Teórica: en este capítulo se realiza un estudio acerca de los sistemas existentes que de una forma u otra realizan el proceso de sanción. En el capítulo también se estudiaron las diferentes metodologías y herramientas que pueden ser empleadas para implementar el sistema de gestión de la información de los procesos de las comisiones disciplinarias.

Capítulo 2: Descripción y análisis de la propuesta de solución: en este capítulo se realiza la planificación, diseño e implementación donde se manejan elementos relacionados con la descripción de las funcionalidades del sistema a implementar así como el diseño del mismo. Muestra las relaciones de los diferentes artefactos generados durante el proceso de implementación teniendo en cuenta la metodología que será definida.

Capítulo 3: Validación de los resultados: en este capítulo se muestra de los resultados de las pruebas realizadas al sistema informático obtenido como resultado de la investigación. El análisis de estos resultados valida la solución propuesta.

CAPÍTULO 1:FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El presente capítulo aborda los elementos que permitieron realizar una correcta caracterización del sistema, sustentados en un estudio de los conceptos fundamentales a los que está relacionado. Se realiza un análisis de sistemas existentes en el mundo que de una manera u otra informatizan procesos relacionados con el sistema a desarrollar. Se ofrece el resultado de las principales herramientas, plataforma de desarrollo y metodología de desarrollo de software a utilizar para el desarrollo de la investigación. Se estudian los diferentes patrones empleados en el desarrollo de sistemas informáticos y finalmente se hace alusión a las pruebas para evaluar la calidad del sistema desarrollado.

1.2. Conceptos fundamentales

Un concepto es la idea que forma el entendimiento. Se trata de un pensamiento que es expresado mediante palabras. Un concepto es, por lo tanto, una unidad cognitiva de significado. Nace como una idea abstracta (es una construcción mental) que permite comprender las experiencias surgidas a partir de la interacción con el entorno y que, finalmente, se verbaliza (se pone en palabras). (1)

En este epígrafe se dejan plasmado los conceptos fundamentales tratados en todo el contexto de la investigación, como vía de establecer el mismo entendimiento se enuncian a continuación aquellos que son de vital importancia para el entendimiento de los procesos descriptos, los mismos fueron tomados del Diccionario de la real academia de la lengua española.

Reglamento: Conjunto de normas, reglas o leyes creadas por una autoridad para regir una actividad o un organismo. Es un conjunto de disposiciones orgánicas emanadas del poder público competente para hacer efectivo el cumplimiento de las leyes administrativas. (1)

Denuncia: Es la acción y efecto de denunciar (avisar, noticiar, declarar la irregularidad o ilegalidad de algo, delatar). La denuncia puede realizarse ante las autoridades correspondientes (lo que implica la puesta en marcha de un mecanismo judicial) o de forma pública (sólo con valor testimonial). (1)

Comisión disciplinaria: Conjunto de personas encargadas por una autoridad de velar por la buena conducta y disciplina. (1)

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. (1)

Opinión: Idea, juicio o concepto que se tiene sobre alguien o algo. (1)

Declaración: Acción y efecto de declarar o explicar 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. (1)

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 y para el bien de los gobernados. (1)

1.3. Descripción del proceso

Una comisión disciplinaria comienza con una denuncia de indisciplina que llega a la facultad, la analiza el decano, el mismo decide si procede o no procede, 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, Resolución 240 del 2007. Una vez que está asignada la denuncia la comisión disciplinaria tiene un plazo de 30 días hábiles para todo el proceso de la comisión; fecha que es asignada por la asesora de trabajo educativo, y que se tienen en cuenta la cantidad de estudiantes que han sido denunciados. (Solo en casos excepcionales se le otorga 10 días hábiles más, aprobados por el decano).

Cuando la comisión disciplinaria tiene la denuncia comienza a solicitar las declaraciones, de la persona que denuncia, la persona denunciada, la FEU, UJC, el tutor o profesor guía y las evidencias que dan el hecho por probado, las mismas se pueden obtener mediante entrevistas individuales, declaraciones de testigos, registro de conversaciones.

Una vez que ya se cuente con todas esas opiniones la comisión, integrada por el presidente, el secretario y el vocal realizan una primera reunión donde discuten todas las declaraciones y llegan a un primer consenso de la tipificación de la falta (faltas graves, menos graves y muy graves), se elabora el dictamen el cual contiene la medida propuesta por la comisión y que debe ser analizada por el decano. Con todo esto se conforma un expediente y es enviado 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, el decano revisa si está de acuerdo o no con la sanción propuesta, la tipificación y es enviado a la secretaria docente donde se realiza la resolución, esta es notificada a los infractores y se les comunica sobre el proceso de apelación recogido en el Capitulo XI: De las apelaciones, del citado reglamento, esclareciendo que cuenta con 10 días hábiles para entregar la solicitud. Una vez pasado ese período si no hay solicitud de apelación emitida por parte de los sancionados se archivan las resoluciones en el expediente docente y se comunica al denunciante el resultado de la CD, el asesor de trabajo educativo toma las medidas necesarias para que el o los infractores cumplan la sanción impuesta.

En caso de emitirse una solicitud de apelación el expediente es entregado al departamento de Asesoría jurídica, el cual cuenta con 30 días hábiles para proceder al finalizar esa fecha comunica al decano si la apelación tuvo lugar.

Puede ocurrir que una CD al analizar el caso determine que es un hecho que no se puede dar por probado y emita como dictamen que no procede, el cual es presentado al decano y si este lo aprueba es archivada la denuncia por el asesor de trabajo educativo, en caso de no estar de acuerdo el decano, se asigna a otra comisión.

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

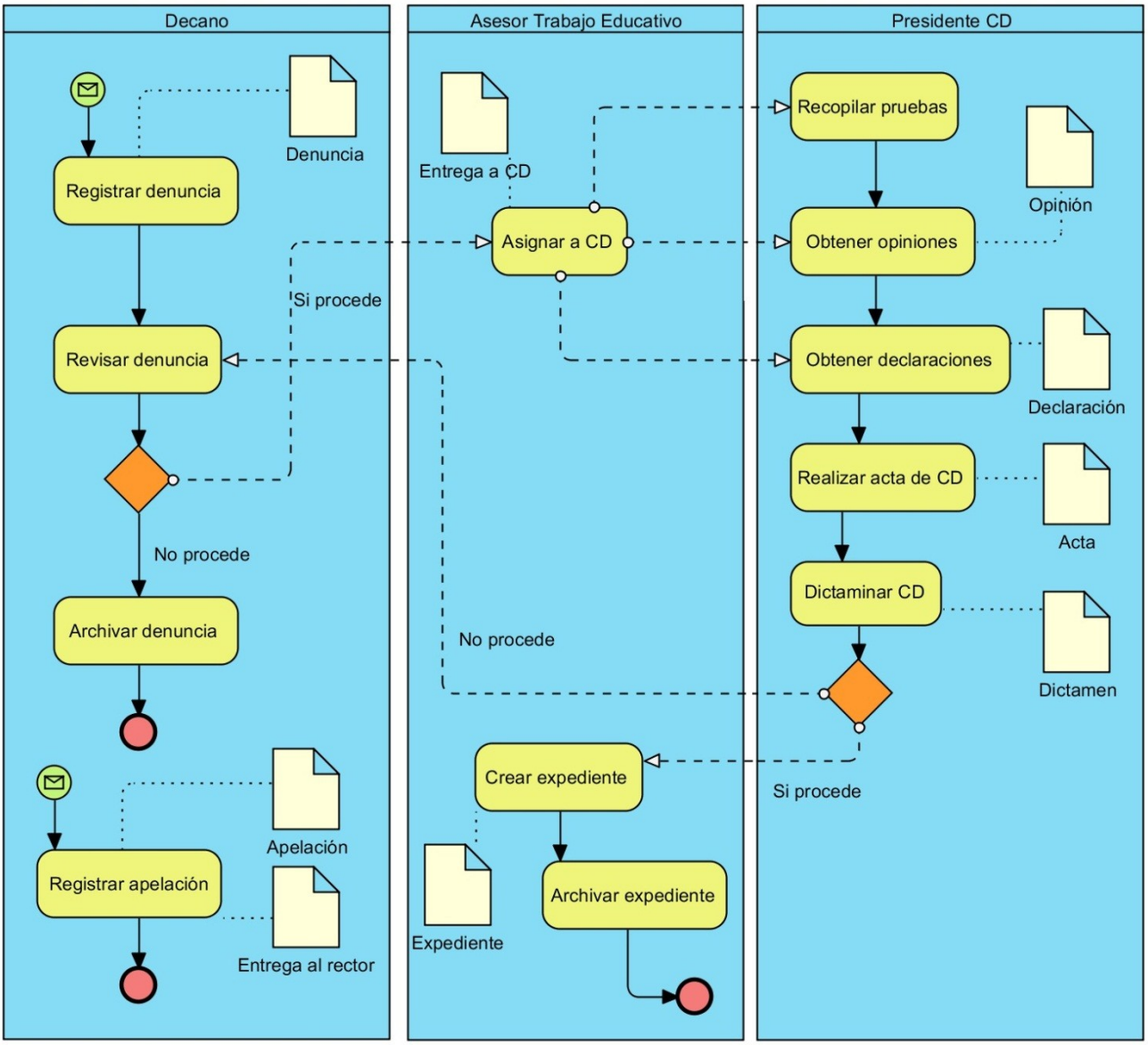

Imagen 2 Diagrama de proceso de negocio

1.4. Sistemas informáticos existentes en el mundo

Actualmente existe una tendencia hacia el uso de las TIC para lograr que los procesos que se realizan en todas las esferas transcurran de forma eficiente y acertada. El sector educativo no escapa a esta tendencia, por lo que se han creado varios sistemas informáticos que facilitan la gestión de la información. A continuación se muestran algunos ejemplos:

- Dumbo**:** *Portal Integrado para la Gestión de Peticiones de Cambio e Incidencias de la Universidad de Murcia.*

Herramienta que integra de una manera ágil la gestión de incidencias y atención a usuarios en las dos vertientes que éstas conllevan: los usuarios que las originan y los expertos que se encargan de resolverlas. Dumbo es considerado una Solución tecnológicamente avanzada, atendiendo a que es un sistema web apoyado sobre una arquitectura de tres capas que garantiza fácil acceso, instalación cero e independencia de la plataforma cliente utilizada. Ofrece valiosa información a responsables de Sistemas de Información y a equipos directivos. (2)

Este sistema es privativo y ajustado a las necesidades propias del cliente. El mismo no es multiplataforma, solo opera en el sistema 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

- Mecanismo de seguridad del sistema GREHU.

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 nuevo sistema a desarrollar.

Este sistema, con sus innumerables salidas, tanto estadísticas como gráficas hace posible que en mayor o menor medida, los directivos relacionados con la gestión del personal, puedan conocer y prever las posibilidades 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. (3)

Esta aplicación informática presenta varias funciones como el control de sanciones y amonestaciones, funcionalidad que puede ser parte del nuevo sistema. Pero a partir de lo estudiado se llega a la conclusión que no puede ser utilizado en la facultad debido a que trabaja sobre el sistema operativo Windows y es un software a la medida.

- Software de Control de Asistencia.

El Sistema de Control de Asistencia de la empresa IBIX[1] es un software que permite llevar registros automáticos del tiempo laborado e incidencias del personal en base a los turnos y políticas definidas por la empresa. Se obtienen variados reportes como: control de asistencia de los trabajadores, faltas, retardos y tiempo extra.

Este sistema obliga el cumplimiento de la jornada de trabajo del personal y establece un control sobre el tiempo extra, el cual representa un alto costo en las empresas. Mejora el desempeño en el registro de los trabajadores y productividad del personal .Es fácil de usar en ambiente Windows XP, 7 ,8 y en sistemas multiusuario y multiempresa. (4)

Esta aplicación informática procesa información específica de la empresa para la que fue creada. No se conoce con que herramientas y tecnologías fueron desarrolladas además que no puede operar en el sistema operativo GNU/Linux. Sin embargo posee características que debe tomarse en cuenta como la obtención de variados reportes que dan paso a una buena toma de decisiones.

El estudio de estos sistemas ha permitido descubrir que ninguno puede ser utilizado en la institución debido a que son software a la medida que procesan información específica de los lugares para los cuales fueron desarrollados. Son software privativos y de alto costo de adquisición, por lo que no son de código abierto, imposibilitando modificarlo en función de las necesidades propias. No se conoce cómo fue su desarrollo, ni la tecnología que se utilizó. Por estos motivos surge la necesidad de un sistema que cumpla con todas las regulaciones y normas para llevar a cabo el control y registro de los procesos definidos en los análisis disciplinarios de las comisiones disciplinarias de la Facultad 3.

1.5. Formulación de la propuesta de solución

1.5.1. Metodología para el desarrollo

“Un proceso de desarrollo de software encierra una serie de actividades y resultados encaminados a producir un software. La metodología de desarrollo se encarga de elaborar estrategias, centradas

[1] IBIX es una empresa reconocida por fabricar sus propios equipos (OEM - Original Equipment Manufacturer) e integrar tecnologías de los principales fabricantes como Metrologic, HID, Dallas Semiconductors y Recognition Systems.

en las personas o los equipos, orientadas hacia la funcionalidad y la entrega. Su objetivo es elevar la calidad del software a través de un mayor control sobre el proceso" (5).Proporciona, además, un conjunto de procedimientos, técnicas y ayudas con un soporte documental, indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado y definiendo además qué personas deben participar en el desarrollo de las actividades y qué papel deben desempeñar.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología, lo que se obtiene son clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. En la ingeniería de software existen dos clasificaciones de metodologías, las tradicionales y las ágiles. A continuación se presenta un análisis de las mismas: (6)

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

Es una metodología de desarrollo tradicional que puede adaptarse a cualquier proyecto y se divide en cuatro fases de desarrollo o flujos de trabajo las cuales son denominadas: Inicio, Elaboración, Construcción y Transición. Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones y en cada ciclo de iteración se hace necesario el uso de artefactos. Los elementos que componen al RUP son las actividades que se determinan en cada iteración, los trabajadores (que son las personas involucradas en cada proceso) y los artefactos que pueden ser un documento, un modelo o un elemento de modelo.

RUP se basa en casos de uso para describir lo que se espera del software. Basándose en el Lenguaje Unificado de Modelado (UML) como lenguaje principal. Contiene abundante documentación y con él se trabaja muy organizado. (5)

Es una metodología tradicional o robusta que exige un equipo de trabajo de más de nueve personas. Centra su atención en llevar una documentación absoluta de todo el proyecto, está focalizada en documentación, planificación y procesos. Se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante. Características por la que se descarta esta metodología y se escogerá una metodología ágil.

Las metodologías ágiles están especialmente orientadas para entornos variables, proyectos pequeños donde los individuos y las interacciones entre ellos, son más importantes que las herramientas y los procesos empleados y en donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. La prioridad de este enfoque es satisfacer al cliente mediante tempranas y continuas entregas que le aporte un valor. Por lo que es más importante crear un producto software que funcione sin tener que escribir documentación exhaustiva,

tributando con una elevada simplificación pero sin renunciar a las prácticas esenciales para asegurar la calidad del producto.

- **Scrum**

Método iterativo e incremental que enfatiza prácticas y valores de la administración de proyectos por sobre las demás disciplinas del desarrollo. Metodología que maneja los riesgos a niveles aceptables. 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 sprintes, un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de quince minutos del equipo de desarrollo para coordinación e integración. Sin embargo Scrum no propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un framework (marco de trabajo en idioma español) ágil de administración de proyectos que puede ser combinado con cualquiera de las metodologías ágiles existentes. Además de que está especialmente indicada para proyectos con un rápido cambio de requisitos. (7)

- **Extreme Programming (XP)**

Metodología más destacada dentro de los procesos ágiles de desarrollo de software, utilizada para el desarrollo de proyectos cortos. Solo se requiere de un equipo de desarrollo pequeño y en pareja, trabajando en el mismo local físico. Se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (6)

Es una metodología que se basa en pruebas unitarias las cuales se realizan a los principales procesos y a posibles fallas que pudieran ocurrir, también en la reutilización de código para lo cual se crean patrones o modelos estándares. Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. (7)

Después de analizada estas dos metodologías se escogió la metodología XP debido que es una metodología ágil que requiere de un pequeño grupo de desarrollo. Otras de las razones por la que se escogió la metodología XP es que esta empieza en pequeño y añade funcionalidades con

retroalimentación continua. El manejo del cambio se convierte en parte sustantivo del proceso. El costo del cambio no depende de la fase o etapa. No introduce funcionalidades antes que sean necesarias. El cliente se convierte en miembro del equipo y decide que se implementa, además de saber el estado real y el progreso del proyecto. Puede añadir, cambiar o quitar requerimientos en cualquier momento, así como obtener lo máximo de cada semana de trabajo o un sistema funcionando cada tres o cuatro meses. El programador decide cómo se implementan los procesos y puede cambiar los requerimientos en base a nuevos descubrimientos. Puede pedir al cliente en cualquier momento aclaraciones de los requerimientos y estima el esfuerzo para implementar el sistema.

1.5.2. Marco de trabajo

Un framework es una estructura de software integrado por componentes personalizables e intercambiables que incluyen soportes de programas, bibliotecas y lenguajes de guiones para desarrollar o unir los componentes de un proyecto. Es el esqueleto de una aplicación que debe de ser adaptado por el programador permitiéndoles escribir un código mejor, más entendible y mantenible.

Un marco de trabajo permite separar en tres capas a la aplicación:

- La lógica de presentación, que administra las interacciones entre el usuario y el software.
- La lógica de datos, que permite el acceso a un agente de almacenamiento persistente u otros.
- La lógica de dominio o de negocio, que manipula los modelos de datos de acuerdo a los comandos recibidos desde la presentación.

Existen varios framework de desarrollo entre los que se analizaron:

- **CodeIgniter**

Posee un diseño compacto para crear aplicaciones web completas. Proporciona un amplio conjunto de bibliotecas para tareas comunes, así como una interfaz simple y estructura lógica para acceder a estas bibliotecas. Permite enfocarse creativamente en el proyecto, reduciendo al mínimo la cantidad de código necesario para una tarea determinada. Pero uno de sus puntos débiles es no tener soporte para AJAX[2], además de que es un framework de bajo nivel. (8)

- **Zend**

Se basa en la simplicidad y en las mejores prácticas orientadas a objetos. Se centra en la creación de aplicaciones de web 2.0 seguras, confiables, y consumir aplicaciones disponibles de

[2] **AJAX**(acrónimo de ***A**synchronous **J**avaScript **A**nd **X**ML*), es una técnica de desarrolloweb para crear aplicaciones interactivas. Es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página

proveedores líderes como Google, Amazon, Yahoo!, Flickr, entre otros. Zend es un excelente framework, pero no es buena opción en este caso ya que necesita de un servidor dedicado para poder sacar el máximo provecho a las funciones con las que este cuenta. Se configura mediante la línea de comandos, entonces cualquier cambio que se necesitara hacer se realizaría a través de la línea de comandos del servidor dedicado. En el caso de un servidor compartido por ejemplo, se tendría que bajar, configurar en localhost, modificar, subir, configurar en servidor. Requiere de una larga curva de aprendizaje. Además de que su estructura de archivos no está definida por sí misma, cualquier desarrollador puede modificarla, esto indica que no sería compatible con otra aplicación desarrollada con Zend.(9)

- **Symfony 2**

Posee un reducido número de requisitos previos, lo cual hace que sea muy fácil de instalar en cualquier configuración (Linux o Windows). Es compatible con casi cualquier sistema de base de datos. Permite construir aplicaciones robustas en un contexto empresarial. Incluye herramientas adicionales que ayudan a probar, depurar y documentar el proyecto. Adicionalmente ofrece los beneficios de una activa comunidad de código abierto. Es totalmente gratuito. (10)

Después de hacer un análisis detallado de estos framework se decidió que Symfony 2 es la mejor opción ya que es un framework de código abierto para desarrollo en PHP basado en la arquitectura Modelo-Vista-Controlador (la estructura de cada sitio o aplicación se compone de tres capas: una base de datos, una interfaz de usuario y una capa de comunicación entre las otras dos).También contiene una serie de facilidades y ventajas que ayudan a que el trabajo sea menos complicado debido a que es fácil de instalar y configurar en cualquier plataforma, libera a los desarrolladores de la tarea de crear funcionalidades menores. Las aplicaciones desarrolladas con Symfony 2 son compatibles con la mayoría de las plataformas, bibliotecas e infraestructuras que existen. Promueve el uso de buenas prácticas de programación y genera código fácilmente comprensible por el desarrollador. Si alguna vez encuentran dificultades, se contará con la colaboración de una comunidad de cientos de miles de programadores y con la seguridad de que cualquier posible defecto será corregido en versiones posteriores.

- **ORM Doctrine 2**

El marco de trabajo Symfony 2 utiliza Doctrine como mapeador de objetos relacionales escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un SGBD (Sistema de gestión de base de datos).Necesita un bajo nivel de configuración para empezar un proyecto. Puede generar clases a partir de una base de datos existente y después se puede especificar relaciones y añadir funcionalidades extra a las clases autogeneradas. Una característica importante de Doctrine es la posibilidad de escribir consultas de

base de datos utilizando un dialecto de SQL (por sus siglas en inglés Structured Query Language) denominado DQL (Doctrine Query Language). Además de:

- Soporte para datos jerárquicos
- Soporte para hooks [3]y eventos para manejar la lógica de negocio relacionada
- Diversos comportamientos del modelo (conjuntos anidados, internacionalización, log[4], índice de búsqueda)
- Una función "compilar" que combina varios archivos PHP del framework en uno solo para evitar el descenso de rendimiento que provoca incluir varios archivos PHP

1.5.3. Gestor de base de datos

Un Sistema Gestor de Base de Datos (SGBD) es un software que permite la definición de bases de datos, así como elegir la estructura de los datos necesarios para el almacenamiento y búsqueda de los mismos. Los SGBD sirven de interfaz entre los usuarios, la base de datos y los sistemas informáticos que interactúan con él y permite definir los datos a distintos niveles de abstracción y manipularlos, garantizando la seguridad e integridad de los mismos. (11)

- **PostgreSQL**

PostgreSQL es un sistema de gestión de bases de datos objeto relacionales que ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento. Es una herramienta gráfica de diseño y administración de base de datos (BD) de licencia gratuita que no ha presentado caídas. Puede operar sobre distintas plataformas, incluyendo Linux, Windows, Unix, Solaris y MacOS X (12). Representa ahorros considerables en costos de operación. Presenta gran capacidad de almacenamiento y buena escalabilidad ya que es capaz de ajustarse al número de CPU (Unidad Central de Procesamiento) y a la cantidad de memoria disponible de forma óptima, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.(13)

- **Sistema Gestor de Base de Datos MySQL**

Es uno de los más importantes en cuanto al diseño y la programación de bases de datos de tipo relacional. Se usa como servidor a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo. Una de las características más interesantes de MySQL es que permite recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a las necesidades existentes. Por otro lado, es conocido por desarrollar alta velocidad en la búsqueda de datos e información, a diferencia de otros sistemas. Las plataformas que utiliza son de diversos tipos y entre ellas se pueden mencionar; LAMP, MAMP, SAMP, BAMP y WAMP

[3] Hooks: métodos que pueden validar o modificar las escrituras y lecturas de la base de datos
[4]Log es un registro oficial de eventos durante un rango de tiempo determinado que generalmente se guarda en un fichero de texto

(aplicables a Mac, Windows, Linux, BSD, Open Solaris, Perl y Phyton entre otras). El código fuente se puede descargar y está accesible a cualquiera. Este gestor no presenta gran volumen de documentación por lo que sus utilidades no son bien conocidas debido a que no están archivadas. Además de que no es intuitivo, como otros programas

Después de estudiados los gestores de base de datos anteriores se decide utilizar PostgreSQL debido a que posee gran capacidad de almacenamiento. Es altamente escalable tanto en la gran cantidad de datos que puede manejar como en el número de usuarios simultáneos que puede soportar. Es un SGBD multiplataforma con licencia de código abierto. Posee una interfaz gráfica y facilita enormemente la administración e incluso pueden realizarse cambios en el modelo de datos desde la herramienta de manera muy sencilla.

1.5.4. Servidores web Apache 2

- **Apache 2**

El servidor Apache se desarrolla dentro del proyecto HTTP Server de la Apache Software Foundation (Fundacción de Software Apache). 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.

Se escoge este servidor debido a que es multiplataforma, lo que lo hace prácticamente universal. Es una tecnología gratuida 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.

1.5.5. Lenguajes utilizados

1.5.5.1. Lenguaje de Modelado

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

Marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. Es una notación (esquemática en su mayor parte) con la que se construyen sistemas por medio de conceptos orientados a objetos. Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir, detallar los artefactos, documentar y construir un sistema. En otras palabras, es el lenguaje en el que está descrito el modelo, pero no define un proceso oficial de desarrollo. (14)

El lenguaje unificado de diagrama o notación (UML) sirve para especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer en primer lugar o cómo diseñar el sistema, sino que simplemente ayuda a visualizar el diseño y a hacerlo más accesible para otros. Se compone de muchos elementos de esquematización que representan las diferentes partes de un sistema de software. Los elementos UML se utilizan para crear diagramas, que representan alguna parte o punto de vista del sistema (15).En este caso el lenguaje UML se utilizó en la realización del diagrama de clases, en el modelo de datos y en la realización de los diagramas de despliegue y de componentes.

- **Notación para la Modelación de Procesos de Negocio (BPMN por sus siglas en inglés)**

BPMN es una notación para el modelado de procesos de negocios a través de diagramas. Tiene como objetivo principal servir como soporte para la gestión por procesos, como una notación que pueda ser entendida fácilmente desde los analistas que crean los bocetos iniciales del proceso, los desarrolladores, técnicos responsables de implementar la tecnología que ejecutará estos procesos, hasta las personas que los ejecutan y aquellas que llevarán a cabo el monitoreo y supervisión de los procesos. En otras palabras esta notación crea un enlace entre las etapas de diseño e implementación. Logra ajustarse a las necesidades que presenta el proyecto para proveer un modelado de procesos que resulte fácilmente entendible para todos los usuarios del negocio y los clientes que estarán manejando el proceso de negocio. Las cuatro clases que componen la lista de elementos centrales son los Objetos de Flujo, Objetos de Conexión, Carriles y Artefactos. (16) En la presente solución BPMN fue utilizado para obtener el modelo de negocio.

1.5.5.2. Lenguajes de Programación

Lenguajes de programación del lado del servidor:

- **PHP 5.3**

Para la programación del sistema del lado del servidor se decidió usar el lenguaje PHP, que significa Hypertext Preprocessor y es un lenguaje interpretado de propósito general, ampliamente usado y que está diseñado esencialmente para el desarrollo web y para la programación de servidores web de código abierto. Es un lenguaje multiplataforma que puede usarse para desarrollar aplicaciones en cualquiera de los sistemas operativos existente. PHP brinda muchas funciones para la explotación de bases de datos de manera sencilla. Brinda interfaces para el acceso a la mayoría de las bases de datos existentes como MySQL, PostgreSQL, Oracle, DB2 y Microsoft SQL Server (17). Es el lenguaje utilizado por el framework seleccionado, maximizando las posibilidades de ser seleccionado para utilizarlo en la propuesta de solución.

- **Twig 1.2**

Twig es un motor y lenguaje de plantillas, integrado con Symfony2 para crear las plantillas de la aplicación, las cuales tienen una sintaxis concisa, fáciles de leer y de escribir. Twig se caracteriza por ser rápido, seguro y flexible. La característica más importante de Twig es que su sistema está implementado con herencia entre plantillas, lo que permite crear un "esqueleto" de plantilla base que contenga todos los elementos comunes del sitio y define los bloques que las plantillas descendientes pueden sustituir, ahorrando así código y tiempo en el trabajo(18).En la propuesta de de solución se utilizó la característica más importante de Twig, posibilitando definir una plantilla base de la que heredan los layout que definen las características de cada menú y las plantillas de las funcionalidades que heredan de estos layout, logrando disminuir la cantidad de código repetido.

Lenguajes de programación del lado del cliente:

- **JavaScript 1.2**

JavaScript es un lenguaje interpretado que se utiliza principalmente para crear páginas web dinámicas. Reduce la carga del servidor al hacerse cargo de gran parte de las funciones del cliente. Fue utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de la página web. Se conoce como un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Con él se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. Por su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica (19).

Como biblioteca de JavaScript se utilizó JQuery, debido a que es rápida y concisa, facilita el manejo de eventos, animaciones, y las interacciones Ajax para el desarrollo web. JQuery está diseñado para modificar la forma en que se escribe JavaScript, posibilitó además visualizar toda la información en tablas dinámicas.

- **HTML 5**

HTML, siglas de HyperText Markup Language (lenguaje de marcas de hipertexto), en su versión 5 es un lenguaje de marcas hipertextuales. Es un lenguaje usado para describir la estructura y el contenido en forma de texto, puede describir, hasta un cierto punto, la apariencia de un documento. HTML 5 se ha convertido en el formato más fácil para la creación de páginas web debido a su sencillez debido a que no hay que compilar el código para ver si funciona, se puede ver en forma inmediata el resultado del trabajo y también es usado para complementar el texto con objetos tales como imágenes (20). Este lenguaje se ha elegido por sus innumerables características, de las cuales se destacan que es un lenguaje estático para el desarrollo de sitios web que permite

describir hipertexto presentando el texto de forma estructurada y agradable. Se utilizan para definir texto, tablas, y otros elementos que forman parte del diseño de la página web.

- **CSS 3**

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTMLo XML por sus siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible') y por extensión en XHTML (por sus siglas del inglés eXtensible HyperText Markup Language).Permite separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. (21)La novedad más importante que aporta CSS, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos o hacks, que a menudo complicaban el código de las web.(20)

Las ventajas de utilizar CSS es que tienen un control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo. Por otro lado los Navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad ya que una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Facilitando que el documento HTML en sí mismo sea más claro de entender y se consigue reducir considerablemente su tamaño.

1.5.6. Herramientas informáticas

- **Visual Paradigm for UML 8.0**

Es una herramienta CASE (Ingeniería de Software Asistida por Computadora) profesional que soporta el ciclo de vida completo del desarrollo de Software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Fue la herramienta de modelado de software elegida para generar y representar los artefactos del sistema debido a que es multiplataforma; es decir, tiene la capacidad de ejecutarse sobre diferentes sistemas operativos; además de permitir crear elementos del modelo UML para un amplio alcance de objetivos, ubicar esos elementos en diagramas y paquetes, crear conectores entre ellos, documentarlos, generar código para el software que se está construyendo y realizar ingeniería inversa del código existente en varios lenguajes. Sustenta todos los modelos UML y todos sus diagramas, además para modelado de procesos de negocio soporta BPMN y los diagramas que contiene esta notación, diagramas de procesos de negocio (BPD) y de flujos de datos (DFD). Puede modelar interfaces de usuario, capturar y trazar requisitos, tiene las características que precisa para diseñar y administrar su desarrollo e implementación. (22)

Se seleccionó Visual Paradigm por ser una herramienta de modelado visual UML muy fácil de utilizar, que proporciona un ambiente colaborativo y de integración con el entorno de desarrollo NetBeans.

- **Subversion 1.5**

Subversion 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 HTTP Apache como un módulo de extensión, esto le da una gran ventaja e interoperabilidad y acceso instantáneo a las características existentes que ofrece este servidor.(23)

Esta herramienta permitió el acceso al repositorio a través de redes, garantizando disponibilidad de la información porque es la vía para conectarse a 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 7.3**

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es una aplicación informática destinada a los programadores que incluye un número de herramientas que facilitan el trabajo, tales como un editor de texto, un compilador, depuradores de código, administración de proyectos, integración con sistemas controladores de versiones o repositorios. NetBeans es un IDE de código abierto, libre y gratuito pensado para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación (24).Esta herramienta permitió que en un solo entorno se editara el código de la aplicación, así como depurarlo. Además, permitió la integración con el subversión para el control de versiones.

- **PgAdmin**

Herramienta para administrar el gestor de base de datos PostgreSQL, pues es una aplicación gráfica, con licencia de código abierto. Está diseñado para responder a las necesidades de los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. También incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados. (12)Permitió el manejo y visualización de tablas, consultas, usuarios, funciones, secuencias, entre otros.

1.5.7. Patrón de Arquitectura

Lo patrones de arquitectura se ocupan de definir la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

- **Patrón Modelo-Vista-Controlador (MVC).**

Permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control. Este patrón se ve usualmente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el sistema de gestión de base de datos y el controlador representa la lógica del negocio, que está formado por tres niveles:

- **Modelo**: representa la información con la que trabaja la aplicación, o sea, su lógica de negocio.
- **Vista**: convierte el modelo en una página web que facilita al usuario interactuar con ella.
- **Controlador**: es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista), lo que permite un mantenimiento más sencillo de las aplicaciones. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica referida a los datos, lo que permite que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos que la aplicación utiliza. (25)(Ver Imagen 2: Arquitectura Modelo Vista Controlador)

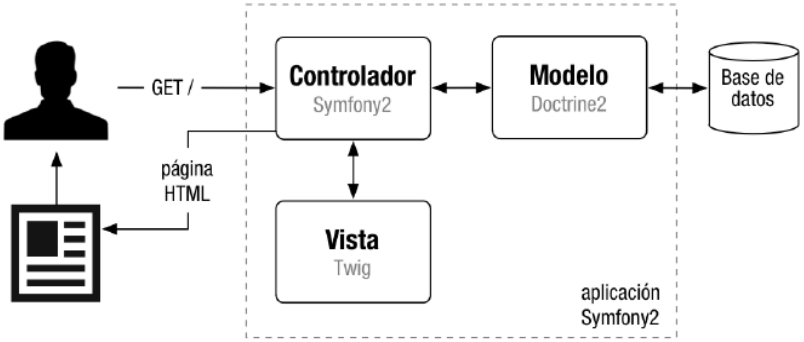

Imagen 3 Arquitectura MVC

1.5.8. Patrones de Diseño.

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.

- **Patrones GRASP** (Patrones Generales de Software para Asignación de Responsabilidades): representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos expresados en forma de patrones. Dentro de sus patrones se encuentran Experto, Creador, Alta cohesión, Bajo acoplamiento y Controlador (26).
- **Patrones GoF (Gand of Four): l**os patrones Gof proponen soluciones a problemas concretos, no son teorías genéricas. Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje. Se utilizan en situaciones frecuentes. Favorecen la reutilización de código. Ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar. Estos patrones se clasifican según su propósito en creacionales, estructurales y de comportamiento(26).

En el capítulo 2 se muestra una explicación de los patrones de diseño que fueron empleados en la implementación de la solución.

1.5.9. Métricas

Las métricas son una medida cuantitativa que permiten lograr una visión profunda de la eficacia del proceso del software. Ayudan en la planificación, seguimiento y control de un proyecto de software y evalúan la calidad del producto

1.5.9.1. 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. (27)

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.

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

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. (27)

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.

Q1: especificidad de requisitos.

Cuanto más cerca de 1 esté el valor de Q**1**, menor será la ambigüedad de la especificación. (27)

1.5.9.2. Métricas para el diseño

Ayudan a evaluar los modelos de análisis y diseño, ofrecen una indicación de la complejidad y del código fuente, así como facilitan el diseño de pruebas más efectivas. (26)

Diseñadas para evaluar los siguientes atributos de calidad:

- **Responsabilidad*:*** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.
- **Complejidad de implementación**: Consiste en el grado de dificultad que tiene implementado un diseño de clases determinado.
- **Reutilización**: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento*:*** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, y está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento*:*** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

- **Cantidad de pruebas**: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (sistema, módulo, clase, conjunto de clases, etc.) diseñado.

➢ **Tamaño Operacional de Clase (TOC):**

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

- **Responsabilidad:** Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación:** Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización:** Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Relaciones entre clases (RC):

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.
- **Reutilización:** Un aumento de las RC implica una disminución en el grado de reutilización 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.

En el capítulo 3 podrá una explicación más detallada de estas métricas así como un ejemplo de cómo se utilizó para evaluar el diseño del sistema.

1.5.10. Pruebas

El objetivo es diseñar una serie de casos de pruebas que tengan una alta probabilidad de encontrar errores. Para ello se aplican las técnicas de pruebas del software, las cuales facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes de software y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento. (26)

➢ **Pruebas de caja negra**

Las pruebas de caja negra también denominadas pruebas de comportamiento se concentran en los requisitos funcionales del software. Permite derivar conjuntos de condiciones de entrada que ejercitaran por completo todos los requisitos del sistema. Pruebas que se le aplican al software con el objetivo de encontrar errores, tales como funciones incorrectas o ausentes, errores en la interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento, de inicialización y de terminación. (26)

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.

- **Pruebas de caja blanca**

La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al aplicar este tipo de prueba se podrán derivar casos de pruebas que garanticen que todas las rutas independientes dentro del módulo se han ejecutado por lo menos una vez, garantizando los lados verdaderos y falsos de todas las decisiones lógicas. Así como ejecutar todos los bucles y dentro de sus límites operacionales las estructuras de datos internos para asegurar su validez. (26)

Técnica camino básico: permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

Para aplicar esta prueba se han definido una serie de pasos a seguir que a continuación se describen:

1. **Notación del grafo de flujo:** usando el código como base, se realiza la representación del grafo de flujo (grafo del programa) mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
    - Nodo: cada círculo, denominado nodo, representa una o más sentencias procedimentales.
    - Arista: las flechas del grafo de flujo, denominadas aristas, representan flujo de control y son análogas a las flechas del diagrama de flujo.
    - Región: las áreas delimitadas por aristas y nodos se denominan regiones.
2. **Complejidad ciclomática:** es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos

independientes[5] del conjunto básico de un programa y propone que se realice al menos un caso de prueba por cada camino independiente, de manera que se asegure la ejecución de cada sentencia como mínimo una vez. La complejidad ciclomática V (G), se calcula de tres formas:

- Número de regiones del grafo de flujo.
- V (G) = A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
- V (G) = P+ 1, donde P es el número de nodos predicado[6] contenidos en el grafo de flujo G.

3. **Determinar un conjunto básico de caminos linealmente independientes**: el valor de V (G) coincide con el número de caminos linealmente independientes de la estructura de control del programa.
4. **Obtención de casos de prueba:** se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

[5] Camino del programa que introduce un nuevo conjunto de sentencias de proceso o una nueva condición. Está constituido por al menos una arista que no haya sido recorrida anteriormente a la definición del camino.

[6] Cada nodo que contiene una condición, y está caracterizado porque dos o más aristas emergen de él.

1.6. Conclusiones parciales

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

- El análisis de los sistemas informáticos desarrollados en otros países y en Cuba dio como resultado que no pueden ser utilizados en la Facultad 3 debido a las características propias del proceso de comisiones disciplinarias y que las soluciones informáticas no son adaptables a las necesidades por lo que se hace necesario desarrollar una aplicación para ello.
- La selección de la metodología de desarrollo XP permitió lograr mayor organización, planificación y ejecución de las actividades e hitos a cumplir.
- El empleo de las herramientas seleccionadas favorece el logro de los objetivos trazados. Su flexibilidad e integración influyen positivamente en el incremento de la productividad.
- El uso del modelo arquitectónico y la correcta aplicación de los patrones de diseño definido posibilitó visualizar el sistema desde diferentes puntos de vistas.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN

2.1. Introducción

Durante este capítulo se obtendrán un conjunto de artefactos pertenecientes a la metodología XP que serán generados y ayudarán en el desarrollo del sistema informático. En la fase de Planificación se describen elementos como son las Historias de Usuarios. En la fase de diseño se definen las tarjetas CRC y en la fase de desarrollo encontrarán las tareas de programación definidas para cada historia de usuario.

2.2. Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

2.2.1. Historias de usuarios

Las historias de usuario son la técnica utilizada por 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 de poseer. Estas fichas quedan estructuradas de la siguiente manera:

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 define la HU

Nombre: Nombre descriptivo de la HU

Programador: Nombre del programador encargado de desarrollar la HU

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

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

El cliente realizó una descripción de las historias de usuarios (HU) con el mínimo nivel de detalle, ayudado por el equipo de desarrollo teniendo en cuenta las funciones que debía realizar para lograr identificarlas. También es necesario destacar que las HU han jugado un papel importante en la estimación de los tiempos requeridos para el desarrollo del proyecto.

Se confeccionaron un total de 10 HU que se encuentran desde el Anexo1 hasta el Anexo 7. A continuación se muestra la Tabla 1 Historia de Usuario número 2. Gestionar denuncias, la cual es un ejemplo de estas tarjetas.

Historia de usuario	
Número:2	**Nombre Historia de Usuario:** Gestionar denuncias
Modificación de Historia de Usuario Número: 1	
Usuario: Etna Lina Lavin Scott	**Iteración asignada:1**
Prioridad en Negocio: Alta	**Puntos Estimados:** 1,14 semanas (8días)
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1,14 semanas (8días)

Descripción:

- Permite adicionar denuncia.

El sistema debe mostrar la opción de adicionar denuncia, el usuario debe llenar diferentes campos como fecha de creación, procedencia y escribir una descripción de la misma, la cual será almacenada en el sistema al dar clic en el botón Guardar. Después se elige el botón Siguiente donde se buscarán las personas que intervienen en la denuncia dígase denunciado o/y denunciante y se almacenan los datos.

A continuación el sistema es capaz de ofrecer la opción de subir algún documento necesario para el proceso.

- Permite listar denuncia.

El sistema debe de ser capaz de mostrar un listado con todas las denuncias existentes. En dicha lista debe mostrar diferentes datos de la denuncia como son: Fecha de la denuncia, denunciante y estado.

- Permite actualizar denuncia.

El sistema debe permitir la opción de actualizar cualquiera de los campos que conforman la denuncia con el objetivo de rectificar posibles errores introducidos.

- Permite revisar denuncia.

El sistema debe ser capaz de mostrarle un listado con todas las denuncias listas para ser revisadas, el usuario selecciona una de las denuncias mostradas en la lista y el sistema

muestra toda la información que tenga de esa denuncia. • Permite asociar personas a denuncias. El sistema debe ser capaz de asociar cada denuncia con las personas implicadas correctamente. • Permite adicionar datos de una persona. El sistema debe permitir introducir datos de la (s) persona (s), por ejemplo: especificar el rol de denunciante a denunciado. • Permite actualizar datos de la persona. El sistema posibilita actualizar datos de la (s) persona (s), por ejemplo: cambiar el rol de denunciante a denunciado, en correspondencia con la situación, y en función de corregir errores. • Permite asignar denuncia a comisiones disciplinarias. El sistema debe permitir la asignación de la denuncia a una comisión ya creada en el sistema • Permite asociar evidencias a la denuncia. El sistema debe ser capaz de asociar las evidencias a la denuncia.
Observaciones:

Tabla 1Historia de Usuario número 2. Gestionar denuncias

La historia de usuario mostrada pertenece a la iteración número 1 y fue seleccionada esta debido a la importancia que tiene tanto para el cliente como para el equipo de desarrollo, pues engloba las funcionalidades críticas del proceso y que el sistema informático deberá suplir como parte de garantizar la calidad en el mismo.

2.2.2. Requisitos

Son declaraciones de los servicios que deben proporcionar el sistema, de la manera que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. Describen con detalle lo que el sistema debe de hacer. (5)

2.2.3. Captura y análisis de los requisitos

“La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir el mismo. Este proceso puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas y depende mucho de las personas que participen en él. Por la complejidad que esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa”. (26)

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

2.2.3.1. Requisitos Funcionales

Una vez realizadas las distintas historias de usuarios se procede a documentar las necesidades del cliente, identificando los requisitos a partir de las historias de usuario. 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 por la investigación.

RF_Autenticar usuario.	RF_Adicionar plantillas	RF_Adicionar denuncias.
RF_Actualizar usuario.	RF_Actualizar plantillas	RF_Actualizar denuncias.
RF_ Listar usuario.	RF_Listar plantillas	RF_Asociar personas denuncia
RF_Adicionar comisión	RF_Mostrar vista previa plantilla	RF_Adicionar datos persona
RF_Actualizar comisión	RF_Generar pdf	RF_Actualizar datos persona
RF_Asociar integrantes comisión	RF_Actualizar roles	RF_Asociar adjuntos denuncia
RF_ Listar comisiones	RF_Actualizar origen denuncia	RF_Revisar denuncia
RF_Adicionar apelación	RF_Actualizar origen opinión	RF_Asignar denuncia a cd
RF_Actualizar apelación	RF_Estudiantes denunciados	RF_Actualizar fechas
RF_Listar apelaciones	RF_Cantidad de CD	RF_Actualizar términos
RF_Adicionar notificación	RF_Estudiantes reincidentes	RF_Listar términos
RF_Listar notificación	RF_Estudiantes con condicional	RF_Adicionar dictamen
RF_Actualizar acta	RF_Sancionados por	RF_Adicionar opinión

separación

RF_Actualizar dictamen	RF_Listar cd	RF_Listar opiniones
RF_Listar denuncias	RF_Asociar adjuntos cd	RF_Listar declaraciones
RF_Actualizar opinión	RF_Asociar persona opinión	RF_Adicionar acta
RF_Adicionar declaración	RF_Actualizar declaración	RF_Actualizar decisión
RF_Adicionar decisión		

2.2.3.2. Requisitos no funcionales.

Son restricciones de los servicios o funciones ofrecidos por el sistema. Son propiedades emergentes de dicho sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento y rendimiento. De forma alternativa, se utilizan para definir las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de los datos que utilizan en las interfaces del sistema. (5)

Requerimientos de apariencia o interfaz externa

- La interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma.
- La comunicación entre el servidor Web y las máquinas clientes será mediante el protocolo HTTP y entre el servidor y el directorio activo mediante LDAP.
- 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 la utilización del sistema.

Requerimientos de usabilidad

- La aplicación propuesta será usada por personas con al menos 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.

Requerimientos de rendimiento

- Para un buen funcionamiento de la aplicación se seguirán las diferentes técnicas de elaboración de sitios web, que faciliten el acceso rápido a sus páginas.

Requerimientos de soporte

- Se necesita un servidor de bases de datos que soporte volúmenes de datos (PostgreSQL 9.1).
- Se necesita un servidor web (Apache).
- El sistema será de software libre (Linux), atendiendo a las políticas de migración de la Universidad.

Requerimientos 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 que deseen 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.

Requerimientos 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 PostgresSql 9.1.
- Computadoras clientes con un navegador web (Firefox 20.0 o superior).

Requerimientos de hardware

- En el cliente se requiere una máquina con 256 MB de RAM como mínimo, con microprocesador a 1 GHz de velocidad de procesamiento o superior, tarjeta de red que posibilite que todas las máquinas implicadas en la funcionalidad de la aplicación estén conectadas a la red.
- En el servidor requiere como mínimo 1024 MB de RAM, microprocesador a 3 GHz de velocidad y tarjeta de red que posibilite la conexión.

2.3. Diseño

Durante todo el tiempo de vida del proyecto el diseño es revisado, actualizado debido a cambios que se presentan durante el desarrollo de forma constante. A continuación encontrarán los artefactos correspondientes a la fase de Diseño.

2.3.1. Tarjetas CRC (Clase - Responsabilidad – Colaborador)

Las tarjetas CRC fueron las bases para la obtención del modelo entidad relación. Cada tarjeta se convirtió en objeto, sus responsabilidades en métodos públicos y sus colaboradores en llamados a otras clases. En la metodología XP el proceso de diseño es iterativo, por lo que la creación de las

tarjetas es según las iteraciones y se le añaden responsabilidades y colaboradores según se haga necesario. 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 Tarjeta CRC, se muestra cómo se distribuye esta información.

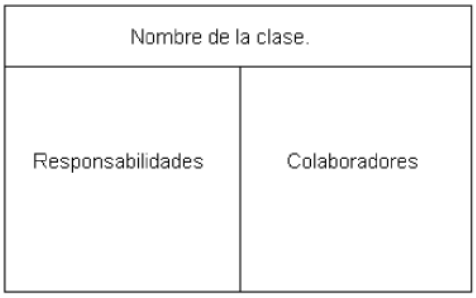

Imagen 4 Tarjeta CRC

A continuación las tablas 2; 3 y 4, que muestran ejemplos de tarjetas CRC confeccionadas durante la fase de Diseño, el resto se podrá consultarse en los Anexos del 8 al 23.

Proceso	
Responsabilidades Gestionar los procesos de Denuncia, Comisión Disciplinaria y Apelación Atributos 1. Fecha de creación 2. Fecha de culminación	Colaboradores 1. Estado 2. PersonaProceso[7] 3. Adjunto 4. Notificación 5. EstadoPaso[8] 6. Tipo de proceso

Tabla 2 Tarjeta CRC número 5

Comisión	
Responsabilidades Gestionar las comisiones Atributos 1. Nombre 2. Descripción **3.** Activo	Colaboradores 1. Persona

Tabla 3 Tarjeta CRC número 18

[7] En el caso de PersonaProceso es el nombre de una tabla intermedia de la base datos, que almacena los datos de las personas y los procesos.

[8] En el caso de EstadoPaso es el nombre de una tabla intermedia de la base datos, que almacena los estados y los pasos en los que se encuentran los procesos.

Denuncia	
Responsabilidades Gestionar las denuncias Atributos 1. Descripción de la falta 2. Observaciones 3. Fecha de revisión	Colaboradores 1. Proceso 2. DecisionDenuncia[9] 3. ProcedenciaDenuncia[10]

Tabla 4 Tarjeta CRC número 15

2.3.2. Diagrama de clases

La imagen 4 Diagrama de clase entidades del sistema SIPCD muestra un total de veintiocho clases y las relaciones entre ellas. Además se obtuvieron cinco 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 de que el diagrama presenta un bajo acoplamiento y una alta cohesión.

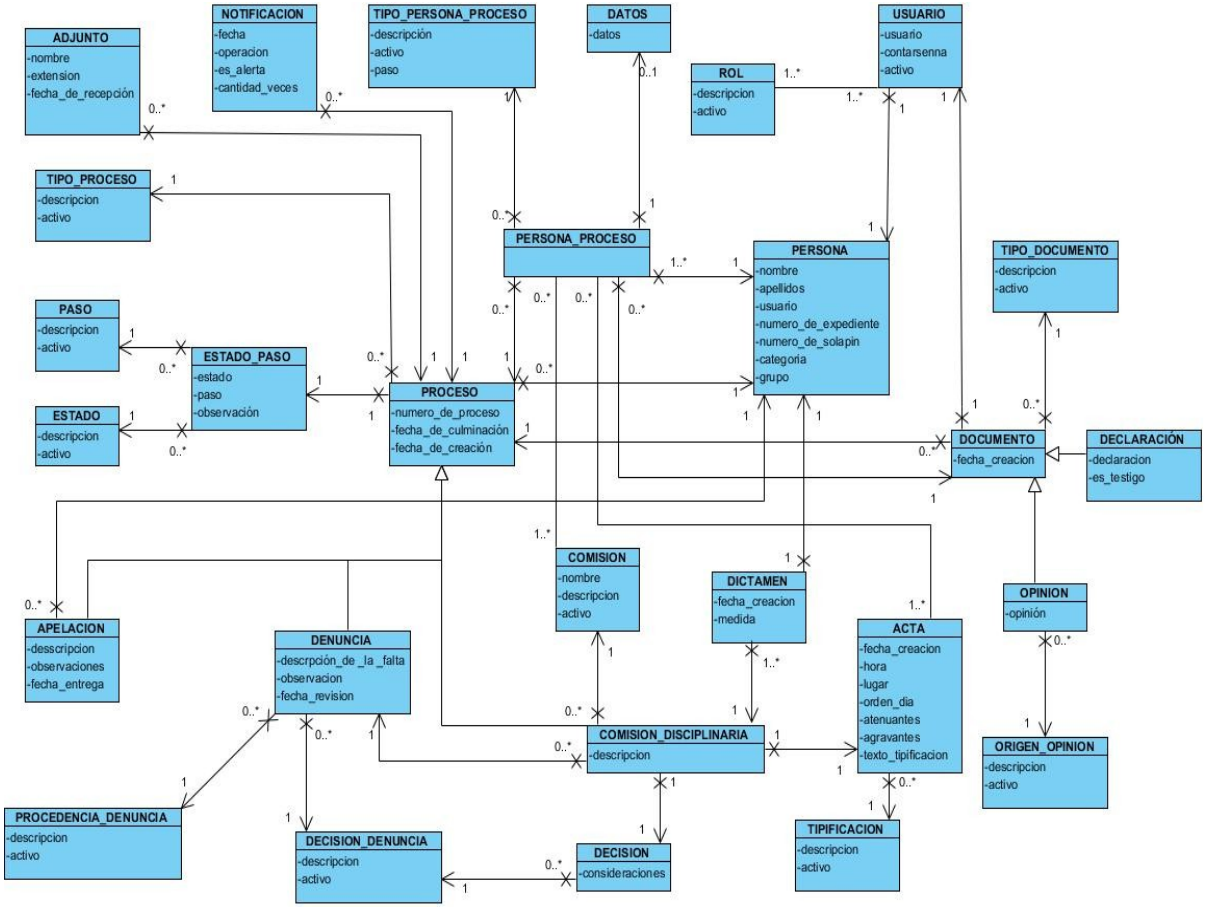

Imagen 5 Diagrama de clase entidades del sistema SIPCD

[9] En el caso de DecisiónDenuncia es el nombre de una tabla de la base datos, que almacena las decisiones de procede o no procede una denuncia.

[10] En el caso de ProcedenicaDenuncia es el nombre de una tabla de la base datos, que almacena el área que envía la denuncia.

2.3.3. Modelo de datos

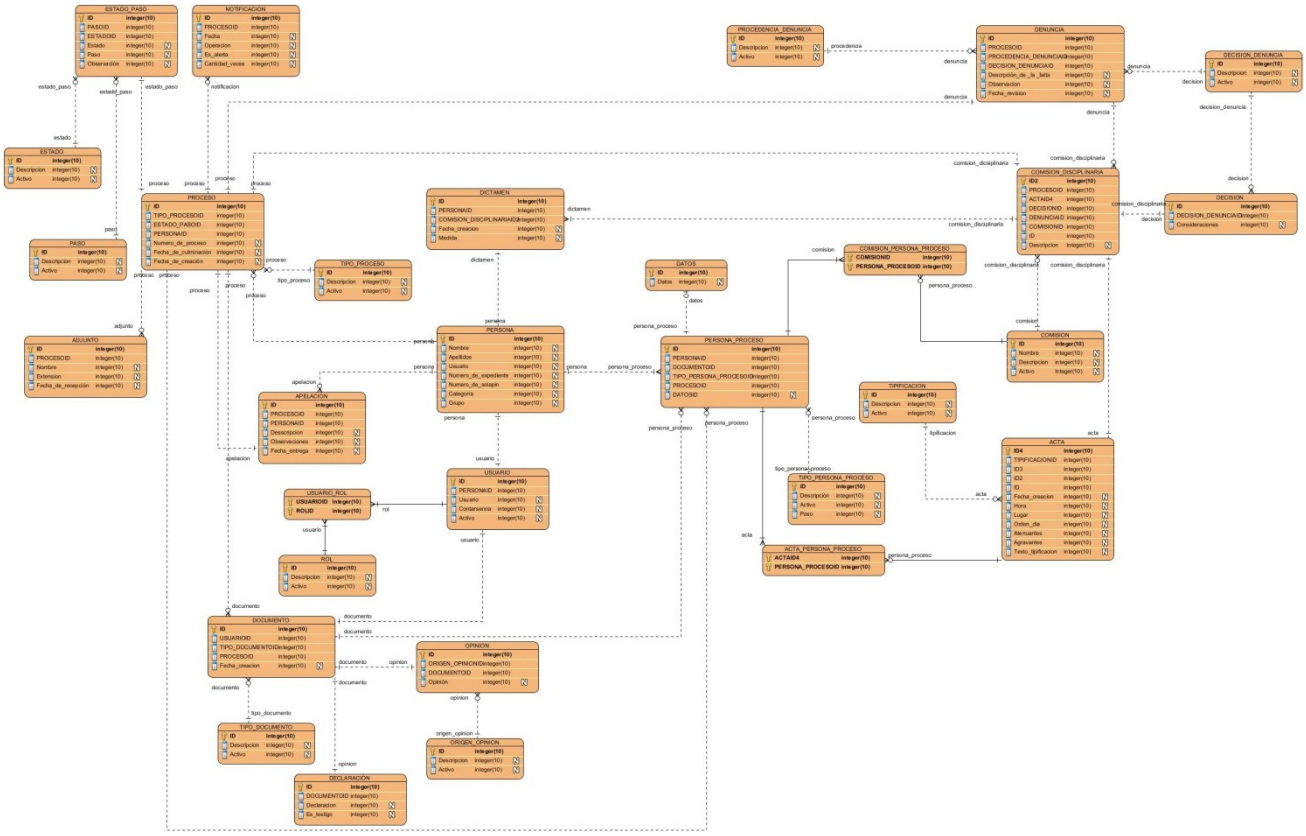

Imagen 6 Modelo de datos del sistema SIPCD

Este modelo de datos representa las tablas (entidades) importantes para el funcionamiento del negocio y muestra los datos que serán contenidos en el sistema. El modelo de datos obtenido cuenta con un total de treinta y una tablas, donde once de ellas son nomencladores, encargados de gestionar conceptos específicos del negocio ya predefinidos, como por ejemplo: la tabla Procedencia_Denuncia, que se relaciona con la tabla Denuncia. Las restantes tablas facilitan el registro de datos que son necesarios para el funcionamiento de la aplicación, por ejemplo: la tabla Usuario registra los datos referentes a las personas que se han autenticado en el sistema como usuario y contraseña. (Ver Imagen 5 Modelo de datos del sistema SIPCD)

2.4. Patrones de diseño

El marco de trabajo Symfony 2 trae estos patrones implícitamente, dándole solución a una serie de problemas recurrentes que se presentan. Symfony 2 implementa numerosas funcionalidades que el cliente debe conocer, para así hacer uso de estos, configurándolo según sus necesidades. A continuación se describen los patrones empleados:

- **Inyección de dependencias:** permite desacoplar unos componentes de otros y reducir la interdependencia entre objetos y bibliotecas. Consiste en pasar a cada componente todo lo que necesita a través de sus constructores, métodos o campos. Se trata de un

objeto “inteligente” que conoce las dependencias de los servicios y es capaz de construir cualquier servicio que se le solicite y de entregarlo bien configurado, listo para su uso y sin preocupación de instanciar e “inyectar” sus dependencias. Permite estandarizar y centralizar la forma en que se construyen los objetos en la aplicación.

- **Unidad de trabajo:** ya que Doctrine conoce todas las entidades gestionadas, cuando se llama al método flush() para persistir los objetos, se calcula el conjunto de cambios y ejecuta la(s) consulta(s) más eficiente(s) posible(s). Por ejemplo, si se persiste un total de 100 objetos y posteriormente se llama a flush(), Doctrine creará una sola declaración preparada y la volverá a utilizar en cada inserción. Este patrón se conoce como Unidad de trabajo, y se utiliza porque es rápido y eficiente. La unidad de trabajo se manifiesta en las clases controladora cuando se va a persistir un determinado objeto que contiene otros.

Patrones de diseño GRASP

- Experto: Consiste en asignar una responsabilidad al experto en información, o sea, a la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos; tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se deba conectar con el objeto producido en cualquier evento. Al escogerlo como creador, brinda soporte al bajo acoplamiento.
- Bajo acoplamiento: Se basa en asignar una responsabilidad para mantener el bajo acoplamiento. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño.
- Alta cohesión: Asigna una responsabilidad de modo que la cohesión siga siendo alta. Como el patrón Bajo Acoplamiento, también Alta Cohesión es un principio que se debe tener presente en todas las decisiones de diseño; es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño.
- Controlador: Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se acepten. La clase controlador debería utilizarse con los eventos sistemáticos de un caso, almacenando de esta forma la información referente al estado del caso.

Estos patrones fueron utilizados en la propuesta de solución de la manera descrita a continuación:

El **patrón Experto** se aplicó en cada clase definida para la solución, las responsabilidades de las clases dependen de la información que poseen, permitiendo que cada clase sea experta en realizar las tareas para las cuales se definieron sus responsabilidades; permitiéndole al **patrón Controlador** asignar la responsabilidad de controlar el flujo de eventos del sistema. La aplicación del **patrón Alta cohesión** permitió que cada clase fuera responsable de lograr tareas bien definidas en áreas funcionales específicas, las clases contienen el número de responsabilidades que necesitan ni más ni menos, de esta manera se logra un comportamiento muy bien definido para cada clase utilizada en el desarrollo de la solución. El **patrón Bajo acoplamiento** disminuyó las dependencias, aumentado así la reutilización de las clases. Por último el **patrón Creador** reflejado 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.

Patrones de diseño GOF

Patrones de Creación: 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.

- 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.

Patrones 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.

- Fachada: es una interfaz que actúa como mediadora entre dos capas en la aplicación. Este patrón se empleó en las clases controladora 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 layoutDenuncia.html.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.

Patrones de 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.

- 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.
- 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.

2.5. Implementación

En esta fase se genera todo el código fuente necesario para satisfacer las HU definidas para la solución y se describen todas las tareas realizadas en cada iteración. Al inicio de cada iteración, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario. Todas las HU son traducidas en tareas de programación.

2.5.1. Iteraciones

Previa a las iteraciones se realizó una reunión con los miembros del equipo de desarrollo donde se transformó el contenido de las HU en tarjetas CRC y luego en tareas de programación. A continuación se presenta la Tabla 5 Tareas de Programación de la iteración 1, que muestra las tareas de programación definidas en la primera iteración, debido a la importancia que tiene tanto para el cliente, como para el equipo de desarrollo, por ser las primeras tareas a realizar y por tener una alta prioridad en negocio; las demás tablas se podrán encontrar en los Anexos 24 y 25.

Iteración: Número de la iteración.

HU: Nombre de la Historia que va a ser implementada.

Tareas de programación: Nombre de las tareas de programación definida para una HU.

Iteración	HU	Tareas de programación
	Gestionar Usuario	Crear Interfaz para autenticar usuario
		Listar Usuarios
		Crear código fuente para actualizar usuario
		Crear código fuente para autenticar usuario
	Gestionar Comisión	Crear interfaz para cada configuración de la comisión
		Crear menú para la gestión de la comisión
		Listar comisiones
		Crear código fuente para adicionar o actualizar comisiones
	Gestionar denuncia	Crear interfaz para gestionar denuncia
		Crear interfaz para cada configuración de la denuncia
		Crear menú para la gestión de la denuncia

Iteración 1		Crear código fuente para adicionar o actualizar denuncia
		Listar denuncia
		Crear código fuente para adicionar o actualizar datos de la persona
		Crear código fuente para asociar adjuntos a la denuncia y asignar denuncia a CD.
	Gestionar CD (Comisión Disciplinaria)	Crear interfaz para gestionar CD
		Crear interfaz para cada configuración de la CD
		Crear menú para la gestión de CD
		Listar CD, opiniones , declaraciones, decisiones
		Crear código fuente para adicionar o actualizar opiniones.
		Crear código fuente para adicionar o actualizar declaraciones
		Crear código fuente para adicionar o actualizar decisiones
		Crear código fuente para adicionar o actualizar dictamen
		Crear código fuente para adicionar o actualizar acta
		Crear código fuente para asociar adjuntos a CD y asociar persona a la opinión

Tabla 5 Tareas de Programación de la iteración 1

2.5.2. Estándares de codificación

"Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armónico, como si un único programador hubiera escrito todo el código de una sola vez, la legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. Posibilita que un equipo de programadores mantenga un código de calidad sobre el que se efectuarán luego revisiones del código" (26).

Cabecera del archivo:

- Siempre es importante que todos los archivos .php inicien con una cabecera específica que indique información relevante del mismo. Es de cada equipo decidir si se quiere o no agregar más datos.(Ver Imagen 6 Ejemplo de cabecera con datos relevantes).

```
/*
 * @access public
 * @package SIPCD.CDBundle.Controller
 * @author Etna Lina Lavin
 */
```

Imagen 7 Ejemplo de cabecera con datos relevantes

- Cada archivo especificará el paquete que lo contiene y el uso de otras clases.(Ver Imagen 7 Ejemplo de cabecera donde cada archivo específica el paquete que lo contiene).

```php
<?php

namespace ArqBase\ComisionBundle\Controller;

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

Imagen 8 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.

- Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben bastar para ello.(Ver Imagen 8 Ejemplo 1 de comentario de una función y 9 Ejemplo 2 de comentario de una función)

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

Imagen 9 Ejemplo 1 de comentario de una función

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

Imagen 10 Ejemplo 2 de comentario de una función

Clases:

Las clases serán colocadas en un archivo .php, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo de la clase y siempre empezará en mayúscula. En lo posible, procurar que los nombres de clase tengan una sola palabra. (Ver Imagen 10 Ejemplo de nombre de una clase).

```php
class DenunciaController extends BaseController {
```

Imagen 11 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[11] 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, ésta será separada por un signo de underscore "_". (Ver Imagen 11 Ejemplo de nombre de la variable).

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

Imagen 12 Ejemplo de nombre de la variable

Llaves:

Siempre utilizar llaves para ganar en legibilidad. (Ver Imagen 12 Ejemplo de la utilización de llaves)

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

Imagen 13 Ejemplo de la utilización de llaves

Poner espacios entre signos:

Si tienes un signo debes de poner espacios a ambos lados. (Ver Imagen 13 Ejemplo de espacio entre signos)

```
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 14 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 ésta razón siempre que se necesite se va a utilizar comillas simples, aunque en casos de la interpolación de variables, se

[11] La notación húngara es aquella donde se coloca el tipo de datos antes del nombre de variable: *strNombre* para un string (cadena).

permiten las dobles. (Ver Imagen 14 Ejemplo de comillas simples e Imagen 15 Ejemplo de comillas dobles).

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

Imagen 15 Ejemplo de comillas simples

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

Imagen 16 Ejemplo de comillas dobles

Números dentro del código:

Convertir en constante un número cuando se necesite. (Ver Imagen 16 Ejemplo de constante)

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

Imagen 17 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 está 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 17 Ejemplo de variables inicializadas)

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

Imagen 18 Ejemplo de variables inicializadas

2.5.3. Diagrama de componente

"Un componente es una parte física de un sistema (módulo, base de datos, código fuente, binario o ejecutable), es decir, la materialización de una o más clases" (26). El diagrama de componentes muestra las organizaciones y dependencias lógicas entre los componentes del software. El diagrama de componente del sistema SIPCD (Ver Imagen 18 Diagrama de componentes) evidencia las interacciones entre los componentes del sistema y los del bundle[12] CD. SIPCD cuenta con los componentes Security (para la seguridad), Routing (para las rutas de acceso), Config (para las configuraciones) y el Ctr_Frontal es el controlador frontal que recibe cada petición realizada. Estos componentes permiten un adecuado manejo de la seguridad y las configuraciones.

[12] Bundle: es un directorio que contiene todo tipo de archivos dentro una estructura jerarquizada de directorios.

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 del módulo), Vista (encargado de la construcción y manejo de las interfaces de usuario). El paquete Controlador con el componente Controladoras, es utilizado para dar respuesta a las peticiones realizadas por los usuarios, 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 SIPCD. 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).

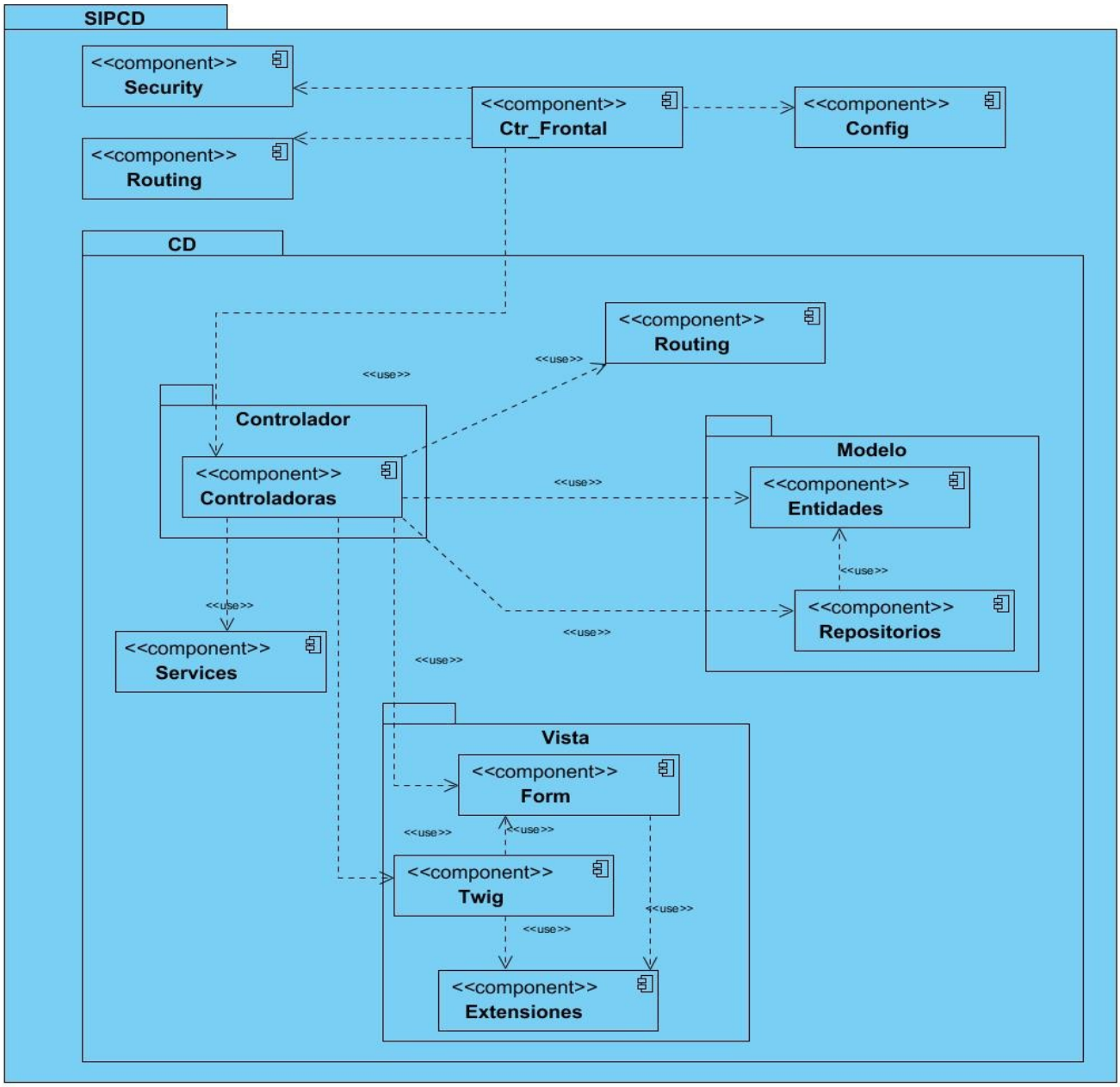

Imagen 19 Diagrama de componentes

2.5.4. Descripción del método

Este método es el encargado de salvar y actualizar una denuncia la cual tiene como atributos, fecha de creación, procedencia y la descripción. (Ver Imagen 19 Código fuente del método salvarActualizarDenuncia)

```
public function salvarActualizarDenunciaAction($id_proceso) {
    $peticion = $this->getRequest();
    $em = $this->getDoctrine()->getManager();
    if (is_null($id_proceso)) {
        $denuncia = new Denuncia();
    } else {
        $denuncia = $em->getRepository('ComisionBundle:Denuncia')->find($id_proceso);
        $denuncia->setFechaCreacion($denuncia->getFechaCreacionToString());
    }
    $procedencia = $em->getRepository('ComisionBundle:Denuncia')->qbProcedencia();
    $frm = $this->createForm(new AdicionarDenunciaType(), $denuncia, array('qb_procedencia' => $procedencia));
    if ($peticion->getMethod() == 'POST') {
        $frm->bind($peticion);
        $denuncia->setFechaCreacion($this->getObjetoFecha($denuncia->getFechaCreacion()));
        $this->salvarActualizarProceso($denuncia, static::denuncia, static::iniciar_denuncia, static::pendiente,
                $this->get('security.context')->getToken()->getUser());
        if ($peticion->get('_siguiente_')) {
            return $this->redirect($this->generateUrl('asociar_personas_denuncia', array(
                'id_proceso' => $denuncia->getIdProceso())));
        } else {
            return $this->redirect($this->generateUrl('salvar_actualizar_denuncia', array(
                'id_proceso' => $denuncia->getIdProceso())));
        }
    }
    return $this->render('ComisionBundle:Denuncia:salvar_actualizar_denuncia.html.twig', array(
                'frm' => $frm->createView(),
                'id_proceso' => $id_proceso,
                'paso' => static::iniciar_denuncia
    ));
}
```

Imagen 20 Código fuente del método salvarActualizarDenuncia

El siguiente método es el encargado de crear un formulario para introducir los datos referentes a la denuncia. (Ver Imagen 20 Ejemplo de un formulario en plantilla Twig).

```
{% block contenido %}
    <div class="hero-unit">
        <legend>Adicionar denuncia</legend>
        {% include 'ComponentesBundle:Errores:errores_form.html.twig' %}
        <form id="frmDenuncia" class="formulario" action="{{ path('salvar_actualizar_denuncia', {'id_proceso': id_proceso })
              method="POST" {{ form_enctype(frm) }}>
            <table>
                <tr>
                    <th>
                        <div class="control-group">
                            <label class="control-label">Fecha creación:</label>
                            <div class="controls">
                                <div id="dp3" class="input-append date" data-date-format="dd-mm-yyyy" >
                                {{ form_widget(frm.fecha_creacion) }}<span class="add-on"><i class="icon-th"></i></span>
                                </div>
                            </div>
                        </div>
```

Imagen 21 Ejemplo de un formulario en plantilla Twig

La seguridad del módulo se gestionó mediante el archivo security.yml, donde se especificaron las reglas para el control de acceso, los roles y las rutas permitidas para cada usuario. Este archivo es utilizado por todos los módulos de la aplicación. La seguridad se maneja mediante las entradas de las rutas y los roles que tienen acceso a cada ruta. (Ver Imagen 21 Especificación de la seguridad en el archivo security.yml)

```
security:
    role_hierarchy:
        ROLE_ADMIN:       ROLE_EDITOR
        ROLE_SUPER_ADMIN: [ROLE_EDITOR, ROLE_ADMIN]
        ROLE_DECANO: [ROLE_ADMIN]
        ROLE_ASESOR: [ROLE_SUPER_ADMIN]
    firewalls:
        area_segura:
            remember_me:
                key:      "%secret%"
                lifetime: 31536000 # 365 days in seconds
            pattern:    ^/
            anonymous: ~
            form_login:
                login_path: /seg/login
                check_path: /seg/login_check
                success_handler: login_success_handler
                failure_handler: login_failure_handler
                default_target_path: /base/principal
            logout:
                path:       /seg/logout
                target:     /seg/login
```

```
access_control:
    - { path: ^/seg/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/seg/login_check, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/base/principal, roles: [ROLE_EDITOR, ROLE_SUPER_ADMIN, ROLE_ADMIN, ROLE_DECANO, ROLE_ASESOR] }
    - { path: ^/base, roles: [ROLE_SUPER_ADMIN, ROLE_ADMIN, ROLE_DECANO, ROLE_ASESOR] }
    - { path: ^/seg, roles: [ROLE_SUPER_ADMIN, ROLE_DECANO, ROLE_ASESOR] }
    - { path: ^/com, roles: [ROLE_SUPER_ADMIN,ROLE_DECANO, ROLE_DECANO, ROLE_ASESOR] }
    - { path: ^/comision, roles: [ROLE_SUPER_ADMIN, ROLE_ADMIN, ROLE_DECANO, ROLE_ASESOR] }
providers:
    usuarios:
        entity: { class: SeguridadBundle:Usuario, property: username }
encoders:
    ArqBase\SeguridadBundle\Entity\Usuario:
      algorithm:   sha512
      iterations: 10
      encode_as_base64: true
```

Imagen 22 Especificación de la seguridad en el archivo *security.yml*

2.5.5. Diagrama de despliegue

El modelo de despliegue representa un modelo de objetos que muestra la forma en que se distribuye físicamente el sistema, teniendo en cuenta la funcionalidad entre cada nodo, que representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar. Dichos nodos poseen relaciones que representan medios de comunicación entre ellos. El modelo de despliegue del sistema SIPCD describe las relaciones físicas de los distintos nodos que componen al mismo. (Ver Imagen 22 Diagrama de despliegue del sistema SIPCD).

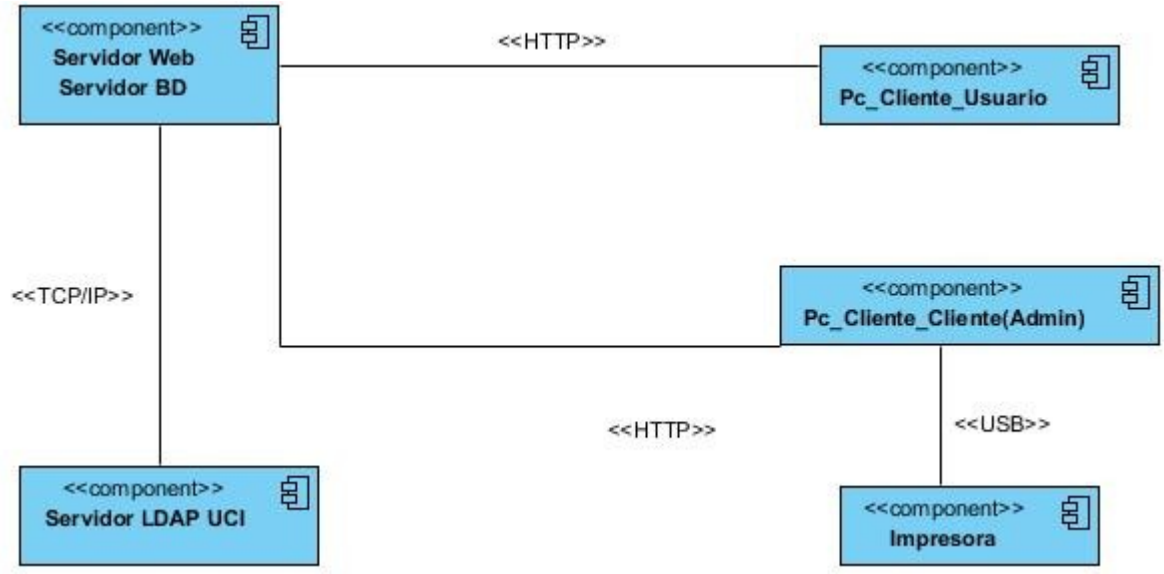

Imagen 23 Diagrame de despliegue del sistema SIPCD

Mediante la presente imagen se describe la utilización de servidor que contendrán los servicios web y de base de datos, el cual se conectará mediante el protocolo TCP/IP, al Servidor LDAP de la UCI, mediante el protocolo HTTP se conectarán las computadoras clientes, estas pueden estar conectadas mediante USB a una impresora.

2.6. Conclusiones parciales

En el presente capítulo se arribaron a las siguientes conclusiones:

- La realización de los artefactos que se describen no poseen una alta complejidad estructural y facilitan la realización de la implementación así como que las clases que los componen son reutilizables, garantizando un Bajo acoplamiento y una Alta cohesión.
- La utilización de los estilos de códigos a tener en cuenta en la implementación de la solución propuesta facilitó un buen entendimiento del código y una mejora considerable del nivel de complejidad de las clases.

CAPÍTULO 3: VALIDACIÓN DE LOS RESULTADOS.

3.1. Introducción

Durante el desarrollo de este capítulo se presentarán los resultados obtenidos durante el desarrollo de la investigación y la fase de prueba. Se muestran los resultados de las pruebas de liberación de calidad que fueron hechas con el objetivo de verificar la funcionalidad del sistema en función de los casos de pruebas descritos, y las pruebas unitarias encargadas de verificar el código.

3.2. Validación de requisitos

La validación de requisitos tiene como objetivo demostrar que su especificación define realmente el sistema que el usuario necesita o el cliente desea. Examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedades, sin inconsistencias, sin omisiones, que los errores detectados han sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto. Para llevar a cabo la validación de requisitos se realizaron revisiones técnicas por parte del equipo de desarrollo. Se revisó un conjunto de factores para asegurar la calidad de los mismos, entre los que se destacan la estabilidad y la ausencia de ambigüedades.

Para medir la estabilidad de los requisitos se aplicó la métrica Estabilidad como se muestra a continuación:

Teniendo en cuenta que se identificaron un total de 55 requisitos funcionales, de los cuales cinco resultaron modificados (dos por inserción y tres por eliminación), se calcula:

$$ETR = [(55 – 5) /55] * 100 = 90.90$$

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 95, 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:

$$ETR = [(55 – 2) /55] * 100 = 96.36$$

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

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

$$Q1 = 55/ 55 = 1$$

Como resultado de la aplicación de la métrica se obtuvo un valor de 1. Dicha cifra demuestra que los requisitos son entendibles, específicos y que no poseen ambigüedades.

3.3. Validación del diseño

Las métricas referentes al tamaño para las clases orientadas a objetos (OO) se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema OO 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 8, (véase Tabla 8 Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica) para determinar el TOC de cada clase. A continuación se muestran tres tablas encaminadas a un mejor entendimiento de la utilización de esta métrica:

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

Tabla 6 Clasificación de las clases según el tamaño

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.

Tabla 7 Modo en que afecta el TOC a los atributos de calidad

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 8 en la columna criterio.

Atributo	Categoría	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

Tabla 8 Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica TOC

Relaciones entre clases (RC)

"La métrica RC está dada por el número de relaciones de uso de una clase con otras" (27). A continuación se muestran las tablas encaminadas a un mejor entendimiento de la utilización de esta métrica.

Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 9 Modo en que afectan las RC a los atributos de calidad

Para determinar el valor de los atributos de calidad, se calcula el promedio de asociaciones de uso y se emplean en la Tabla 10 Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica en la columna criterio.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	Cantidad de relaciones de uso > 2
Complejidad de mantenimiento	Baja	Cantidad de relaciones de uso <= Promedio
	Media	Promedio <= Cantidad de relaciones de uso <= 2* Promedio
	Alta	Cantidad de relaciones de uso > 2* Promedio
Cantidad de pruebas	Baja	Cantidad de relaciones de uso <= Promedio
	Media	Promedio <= Cantidad de relaciones de uso <= 2* Promedio
	Alta	Cantidad de relaciones de uso > 2* Promedio

Tabla 10 Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica RC

Con el fin de evaluar la calidad del diseño y analizar el tamaño de las clases involucradas en la funcionalidad gestionar denuncia, se aplicaron las métricas TOC y RC ya explicadas. A continuación se muestra la aplicación de las métricas a las clases correspondientes a la funcionalidad gestionar denuncia.

Métrica TOC

A continuación se muestran los resultados obtenidos tras la aplicación de la métrica tamaño operacional de clase, donde:

CA: cantidad de atributos de la clase.

CO: cantidad de operaciones de la clase.

CP: cantidad de procedimientos

R: responsabilidad de la clase

CI: complejidad de implementación de la clase.

Reu: reutilización de la clase.

No	Clases	CA	CO	CP	TC	R	CI	Reu
1	Denuncia	2	16	18	Pequeño	Media	Media	Media
2	ProcedenciaDenuncia	2	7	9	Pequeño	Baja	Baja	Alta
3	DecisiónDenuncia	2	7	9	Pequeño	Baja	Baja	Alta
4	Proceso	3	31	34	Grande	Alta	Alta	Baja
5	TipoProceso	1	7	8	Pequeño	Baja	Baja	Alta
6	Adjunto	3	10	13	Pequeño	Baja	Baja	Alta
7	Persona	7	15	22	Medio	Media	Media	Media
8	EstadoPaso	1	12	13	Pequeño	Baja	Baja	Alta
9	Datos	0	5	5	Pequeño	Baja	Baja	Alta
10	TipoPersonaProceso	3	12	15	Pequeño	Media	Media	Media
11	PersonaProceso	0	20	20	Pequeño	Media	Media	Media

Tabla 11 Resultados de la aplicación de la métrica TOC a las clases de la funcionalidad gestionar denuncia

Se trabajó con un total de 11 clases para un promedio de 16.18 de cantidad de procedimientos, obteniéndose como resultados los datos que a continuación se muestran:

Cantidad de clases: 11	Baja	Media	Alta
Responsabilidad	54%	36%	10%
Complejidad de implementación	54%	36 %	10%
Reutilización	10%	36%	54%

Tabla 12 Resultados generales de la aplicación de la métrica TOC a las clases de la funcionalidad gestionar denuncia

El resultado de aplicar la métrica TOC demuestra que de un total de 11 clases pertenecientes a la funcionalidad Gestionar Denuncia, 6poseen baja responsabilidad, lo que representa un 54% del total. Además, se obtuvosolo una clase con alta complejidad de implementación y el mismo número con baja reutilización. Los resultados demuestran que dicha funcionalidad posee un nivel alto de reutilización y un nivel bajo de complejidad y responsabilidad en el diseño propuesto.

Métrica RC

A continuación se muestran los resultados obtenidos para cada atributo de calidad evaluado, tras la aplicación de la métrica relaciones entre clases:

No	Clases	CR	A	CM	CP
1	Denuncia	1	Bajo	Baja	Baja
2	ProcedenciaDenuncia	1	Bajo	Baja	Baja
3	DecisiónDenuncia	2	Medio	Media	Media
4	Proceso	6	Alto	Alta	Media
5	TipoProceso	2	Medio	Media	Media
6	Adjunto	1	Bajo	Baja	Baja
7	Persona	2	Medio	Media	Media
8	EstadoPaso	1	Bajo	Baja	Baja
9	Datos	2	Medio	Media	Media
10	TipoPersonaProceso	1	Bajo	Baja	Baja
11	PersonaProceso	1	Bajo	Baja	Baja

Tabla 13 Resultados de la aplicación de la métrica RC a las clases de la funcionalidad gestionar denuncia

Dónde:

CR: cantidad de relaciones de uso.

CM: complejidad de mantenimiento.

A: acoplamiento entre clases.

CP: cantidad de pruebas.

Se obtuvo un promedio de asociaciones entre clases. A continuación se muestran los datos que resultaron tras la aplicación de la métrica:

Cantidad de clases: 11	Baja	Media	Alta
Acoplamiento	54%	36%	10%
Complejidad de mantenimiento	54%	36%	10%
Cantidad de pruebas	54%	36%	10%

Tabla 14 Resultados generales de la aplicación de la métrica RC a las clases de la funcionalidad gestionar denuncia

Los resultados obtenidos luego de aplicar la métrica RC, demuestran que la mayoría de las clases que componen la funcionalidad Gestionar Denuncia, representadas por un 55% poseen pocas relaciones de uso, por lo que tienen un bajo acoplamiento, además poseen un bajo grado de complejidad de mantenimiento y bajo grado de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas al software.

Luego de realizar la validación del diseño apoyado de las métricas TOC y RC se comprobó que el sistema presenta un bajo acoplamiento así como una alta reutilización. Por lo que está presente dos de los patrones de diseños empleados en la solución. La aplicación también resultó ser fácil de mantener por su propio desarrollador o cualquier otro que no haya participado en la creación del misma.

3.4. Pruebas de Caja Negra

A continuación se muestra un ejemplo (Ver Imagen 23 Caso de prueba funcionalidad Adicionar denuncia) de un caso de prueba de uno de los requisitos que componen la funcionalidad Adicionar Denuncia, el resto de estos se incluyen en el documento Casos de Pruebas de SIPCD.

Escenario 1 (Datos generales)	Descripción	Fecha de creación	Procedencia	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Adicionar denuncia con los datos correctos.	Permite adicionar una denuncia con sus datos correspondientes .	v 14/05/2014	v Residencia	v Se le realiza una denuncia al estudiantes Maceo gonzalez Peres por haberle falatado el respeto al profesor guia Eduado Ramos	El sistema debe de adicionar la denuncia	1- Seleciona la opcion deuncias en el menu superior;aparece la interfaz correspondiente;se selecciona la opción denunciar;eliges el boton adicionar;se llenan los campos fecha de creación,procedencia y descripción;se salvan los datos.
EC 1.2 Adicionar denuncia con campos vacios	No permite adicionar una denuncia porque existen campos obligatorios vacios .	N/A	v Residencia	v Se le realiza una denuncia al estudiantes Maceo gonzalez Peres por haberle falatado el respeto al profesor guia Eduado Ramos	El sistema no debe de adicionar la denuncia	
		v 14-may	N/A	v Se le realiza una denuncia al estudiantes Maceo gonzalez Peres por haberle falatado el respeto al profesor guia Eduado Ramos Torres		
		v 14/05/2014	v Residencia	N/A		

Imagen 24 Caso de prueba funcionalidad Adicionar denuncia

Los casos de pruebas fueron realizados para que a partir de ellas se realizaran las pruebas de funcionalidad. Se realizaron un total de 13 casos de pruebas los cuales fueron realizados en dos escenarios, para cuando los datos son correctos y para cuando existan campos vacíos. En la tabla también se recogen datos como: descripción, respuesta del sistema y el flujo central.

3.4.1. Pruebas aplicadas por el grupo de calidad del Centro de gobierno electrónico

A partir de los casos de pruebas entregados los especialistas del grupo de calidad del centro, aplicaron las pruebas 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 perfectamente y presenta una interfaz agradable a la vista de los usuarios. (Ver tabla 15 Resumen de los resultados de las pruebas aplicadas).

Iteración	Cantidad de no conformidades	Cantidad de no conformidades de casos de prueba	Cantidad de no conformidades de la aplicación	Cantidad de no conformidades que no proceden	Cantidad de no conformidades resueltas
1	65	2	63	12	53
2	19	0	19	3	16
3	6	0	6	1	5

Tabla 15 Resumen de los resultados de las pruebas aplicadas

La aplicación fue sometida a tres iteraciones encontrándose un total de 90 no conformidades. En la primera iteración se encontraron 65 no conformidades, 12 no procedieron y fueron resueltas 53. Para la segunda iteración se encontraron 19 no conformidades de las cuales se resolvieron 16 y 3 no procedieron. En la tercera y última iteración solo se encontraron 6 no conformidades, 1 no procedió y fueron resueltas 5. Después de concluidas las pruebas el grupo de calidad liberó la aplicación entregándosele al desarrollador una acta en la que consta que la aplicación está apta para ser utilizada. (Ver Anexo 26)

3.5. Pruebas de Caja blanca

Para aplicar las pruebas de caja blanca se empleó la técnica camino básico. Esta permitió obtener una medida de la complejidad lógica para el diseño de los casos de prueba y usar dicha medida como guía para la definición de un conjunto básico de caminos de ejecución. Se tomó como ejemplo el código correspondiente al método salvarActualizarDenuncia porque el mismo constituye una funcionalidad principal del sistema y posee una alta prioridad para el usuario, especificada en la Historia de usuario Gestionar denuncia (mostrada en la Tabla 1 Historia de Usuario número 2). Del mismo se obtuvo el grafo de flujo que se muestra en la imagen 24 (Grafo de flujo del método salvarActualizarDenuncia).

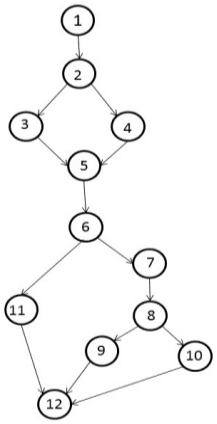

Imagen 25 Grafo de flujo del método salvarActualizarDenuncia

Luego se calculó la complejidad ciclomática:

1. V (G) = 4
2. V (G) =A-N+2 = 14-12+2 = 4
3. V (G) = P+1 =3+1 = 4

El valor V (G) fue igual al número de caminos linealmente independiente de la estructura de control del programa por lo que se definieron los siguientes cuatro caminos:

Camino 1: 1-2-3-5-6-11-12

Camino 2:1-2-3-5-6-7-8-9-12

Camino 3: 1-2-3-5-6-7-8-10-12

Camino 4: 1-2-4-5-6-7-8-10-12

Luego de elaborado el grafo de flujo y los caminos a recorrer, se realizaron los casos de prueba que forzaron la ejecución de cada uno de esos caminos. Se escogieron los datos de forma tal que las condiciones de los nodos predicados estuvieran adecuadamente establecidas, con el fin de comprobar cada camino. Se realizaron las pruebas correspondientes obteniéndose que el código se ejecuta de manera correcta para el método seleccionado. También se probaron el resto de los métodos existentes en la solución obteniendo resultados satisfactorios que validan la correcta ejecución del código que conforma la solución informática propuesta.

3.6. Valoración de las variables de la investigación

En el marco teórico de la investigación se definieron en el problema a resolver como variables:

- Variable Independiente: Proceso de las CD.
- Variable dependiente: Gestión de la información.

Enmarcado al proceso de CD que se realiza en la Facultad 3 una buena gestión 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 definieron algunos indicadores de la gestión de la información.

- Disponibilidad

  La información esté disponible en el momento que se desea consultar.

- Integridad

  El acceso a la información solo es posible por el personal autorizado.

  La información nunca se pierda.

- Tiempo

Indicadores	Antes	Después
Disponibilidad	Para una persona disponer de la información tiene que dirigirse al departamento de secretaría docente que es 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.	Para una persona disponer de toda la información necesaria solo tendrá que acceder al sistema SIPCD, incluso si la misma se encuentra archivada en secretaria.
Integridad	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 solo 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.
Tiempo	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.

Tabla 16 Tabla comparativa de los indicadores seleccionado de la gestión de la información.

3.7. Conclusiones parciales

Se concluye que:

- El análisis y la validación de los requisitos demostraron que el sistema no posee una alta 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 XP.

- Los artefactos del diseño no poseen una alta complejidad estructural por lo que facilitan la realización de la implementación. La mayoría de las clases que los componen son reutilizables, garantizando un Bajo acoplamiento y una Alta cohesión.
- La aplicación de los diferentes tipos de pruebas al final de cada iteración, y las revisiones durante todo el desarrollo, posibilitaron la corrección de errores.
- Los resultados de cada prueba, incluyendo las unitarias, generaron una retroalimentación permanente, mostrando a los desarrolladores la calidad de su trabajo.

Conclusiones generales

Luego de realizado el presente trabajo, se concluye que:

- El análisis de sistemas de comisiones disciplinarias, demostró que es más factible crear una solución propia que responda a las particularidades de la Facultad 3 y cumpla con el paradigma de independencia tecnológica por la que aboga el país.
- La realización del análisis mediante la metodología de desarrollo de software seleccionada XP, garantizó el entendimiento de las funcionalidades del sistema por parte de los miembros del equipo de desarrollo.
- La obtención de los artefactos del diseño demostraron que no poseen una alta complejidad estructural y se demuestra que facilitan la realización de la implementación. La mayoría de las clases que los componen son reutilizables y se encuentran lo menos acopladas posible.
- La implementación de la solución garantizó una aplicación funcional flexible a cambios y sencilla de operar, que responde a los requisitos pactados con el cliente.
- La validación de la solución demostró que las funcionalidades del sistema son correctas, que las entradas se aceptan, se producen salidas satisfactorias y los componentes funcionan de forma adecuada.
- Con la propuesta de solución se contribuye a la gestión de la información del proceso de Comisiones Disciplinarias de la Facultad 3.

Recomendaciones

- Realizar el despliegue del sistema SIPCD en la Facultad 3.
- Ampliar el campo de acción para la universidad.

Bibliografía

1. Real Academia Española. [En línea] 2014. http://lema.rae.es/drae/.

2. Sistema Dumbo. [En línea] [Citado el: 9 de febrero de 2014.] www.um.es/atica2/contenidos/calidad/dumbo.html.

3. Reyes, Gonzalez. Sistema GREHU. [En línea] [Citado el: 15 de enero de 2014.] http://renia.cujae.edu.cu/index.php/revistacientifica/article/view/50.

4. Ortega, Alejandro y Olivas, Guillermo. IBIX. [En línea] [Citado el: 4 de febrero de 2014.] http://www.ibix.com/control-de-asistencia.

5. Sommerville, Ian.Ingeniería del Software. México DF : Editora Pearson, 2005.

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

7. Penadés Letelier, Patricio y Mendez, Carmen.Metodologias Agiles para el desarrollo de software. Valencia : s.n., 2006.

8. etnassoft. [En línea] [Citado el: 30 de enero de 2014.] http://www..com/biblioteca/codeigniter-guia-del-usuario-en-espanol-v-2-1-3/.

9. Zend Framework. [En línea] 2014. [Citado el: 22 de enero de 2014.] http://framework.zend.com/.

10. ¿Porque elegior symfony 2? [En línea] [Citado el: 30 de enero de 2014.] http://www.4rsoluciones.com/por-que-elegir-symfony2/.

11. PostgreSQL. PostgreSQL. PostgreSQL. [En línea] 2012. [Citado el: 3 de Diciembre de 2012.] http://www.postgresql.org/about/.

12. Pgadmin, PostgreSQL tools. [En línea] [Citado el: 22 de enero de 2014.] http://www.pgadmin.org/.

13. Postgresql-es. [En línea] [Citado el: 21 de enero de 2014.] http://www.postgresql.org.es/.

14. Larman, Craig.UML y Patrones. s.l. : Pearson.

15. Leon, Eduardo.Tutorial Visual Paradigm for UML.

16. Fingar, Peter ySmitch, Howard.Business Process Managment:The Third Wave. 2009.

17. Gutierrez, Andres Felipe.Kumbia PHP Framework,¿PorqueProgramar debería ser más fácil? 2007.

18. Twig. [En línea] [Citado el: 4 de febrero de 2014.] http://twig.sensiolabs.org/.

19. Eguiluz, Javier.Introducción a JavaScript. 2008.

20. Castro, Elizabeth.HTML con XHTML y CSS. s.l. : Amaya, 2008.

21. Eguiluz, Javier.Introducción a CSS. 2007.

22. Boost Productivity with Innovative and Intuitive Technologies. [En línea] [Citado el: 15 de enero de 2014.] http://www.visual-paradigm.com/.

23. Control de versiones Open Source de siguiente generación. [En línea] [Citado el: 22 de enero de 2014.] http://svnbook.red-bean.com/index.es.html.

24. Netbeans. [En línea] [Citado el: 21 de enero de 2014.] https://netbeans.org/.

25. Bahit, Eugenia.“POO y MVC en PHP”. 2009.

26. Pressman, Roger S. Ingenieria de requisitos.Un enfoque practico ,6ta edición. Métricas del producto para el software.

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

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

29. Bradenbaugh, Jerry.Programacion de aplicaciones web con JavaScript . Madrid : EDICIONES ANAYA MULTIMEDIA (GRUPO ANAYA, S.A.), 2000.

30. Pressman, Roger S.Ingenieria de requisitos. Un enfoque practico. España : s.n. 5ta edición .

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

32. Symfony. [En línea] [Citado el: 12 de noviembre de 2012.] http://www.symfony.com/.

33. Ben Collins - Sussman, Brian W. Fitzpatrick, C. Michael Pilato.Control de versiones con Subversion: Revisión 1967.

34. Lenguaje PHP. [En línea] [Citado el: 12 de enero de 2013.] http://lenguajephpjc.blogspot.com/2011/05/caracteristicas-del-lenguaje-php.html..

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

36. Symfony en español. [En línea] 12 de marzo de 2013. http://gitnacho.github.com/symfony-docs-es/book/doctrine.html.

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

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

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

40. Doctrine Documentation. [En línea] http://www.doctrine-project.org/documentation/manual/1_0/en/introduction-to-models.

41. Patrones del "Gand of Four". Madrid, España : s.n.

42. Fabien Potencier, François Zaninotto.Symfony, la guía definitiva. 2008.

43. Introducción a HTML. [En línea] http://www.aulaclic.es/html/t_1_1.htm.

44. Pacheco, Nacho.Manual de Twig. 2011.

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

46. 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.

47. INTECO.Guía Prática de Gestión de Requisitos. Diciembre 2008.

48. Bonanata, Maximiliano.Programación y Algoritmos. s.l. : MP Ediciones S.A.

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

50. de la Torre LLorente, César, y otros, y otros.Guía de Arquitectura N-Capas Orientada al Dominio con .NET 4.0. España : Krasis Press, 2010.

51. Méndez, Gonzalo.Ingeniería de requisitos. Madrid : s.n., 2008.

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

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

54. Boost Productivity with Innovative and Intuitive Technologies. [En línea] [Citado el: 15 de enero de 2014.] http://www.visual-paradigm.com/solution/freeumltool/.

55. Genbeta:dev. [En línea] [Citado el: 21 de enero de 2014.] http://www.genbetadev.com/herramientas/netbeans-1.

56. Open Source Software Engineering Tools. [En línea] [Citado el: 22 de enero de 2014.] http://subversion.tigris.org/.

57. Castro Sáenz, Alfonso.Compendio historico de derecho romano. 2002.

58. Denzer, Patricio.PostgresSQL. 2006.

59. Scannapieco, Antonio.EL DERECHO ROMANO. 2000.

60. Jacobson, Ivar y Booch, Grady y Rumbaugh, James.El proceso unificado de desarrollo de software. s.l. : Addison Wesley, 2005.

Anexos

Anexo 1: Tabla 17 Historia de Usuario Gestionar Usuario

Historia de usuario	
Número:1	**Nombre Historia de Usuario:** Gestionar usuario
Modificación de Historia de Usuario Número:	
Usuario: Etna Lina Lavin Scott	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2,14 semanas (15 días)
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2,14 semanas (15 días)
Descripción: • Permite autenticar usuario. El sistema debe de ser capaz de permitir la autenticación de los usuarios ya registrados. • Permite actualizar usuario. El sistema debe de ser capaz de permitir la actualización de algunos de los datos del usuario (rol, usuario y si el mismo se encuentra activo). • Permite listar usuarios. El sistema debe de mostrar un listado con todos los usuarios del sistema.	
Observaciones: • La autenticación en el sistema se realiza mediante la conexión al servidor LDAP de la UCI y localmente para los usuarios registrados. Al ingresar un usuario por primera vez en el sistema vía LDAP, el mismo pasa a ser parte del sistema con un rol que no posee privilegios, estos serán asignados por el administrador del sistema.	

Anexo 2: Tabla 18 Historia de usuario Gestionar comisiones disciplinarias

Historia de usuario	
Número:3	**Nombre Historia de Usuario:** Gestionar comisión disciplinarias
Modificación de Historia de Usuario Número:	
Usuario: Etna Lina Lavin Scott	**Iteración asignada:1**
Prioridad en Negocio: Alto	**Puntos Estimados:** 0.86 semanas (6días)
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.86 semanas (6días)
Descripción: • Permite asociar adjuntos CD. El sistema debe de ser capaz de asociar adjuntos a la CD correspondiente. • Permite adicionar opinión. El sistema debe permitir la opción de adicionar opinión sobre un estudiante específico que forma parte de una Comisión Disciplinaria. (Fecha recepción, Procedencia, Opinión). • Permite asociar persona opinión. El sistema debe de ser capaz de asociar a la opinión la persona que la emite. • Permite listar opiniones. El sistema debe de ser capaz de mostrar un listado con las opiniones ya existentes mostrando información de: (Fecha, Procedencia, Persona que la emite) • Permite adicionar/actualizar declaración. El sistema debe permitir la opción de adicionar/actualizar declaración de los implicados (nombre y apellidos, cargo, declaración y quien la realiza testigo o infractor). • Permite listar declaraciones. El sistema debe de ser capaz de mostrar un listado con las declaraciones ya existentes mostrando información de: (Fecha, si es de un implicado o un testigo) • Permite adicionar/actualizar decisión. El sistema debe permitir la opción de adicionar/actualizar una decisión de la CD, si al analizar las pruebas, opiniones y declaraciones se detecta que la misma procede, se pasa a continuar con el proceso, sino se archiva y se notifica al decano. Los datos recogidos son: (Decisión y consideraciones) • Permite adicionar/actualizar acta. El sistema debe permitir la opción de adicionar/actualizar un acta por cada estudiante analizado,	

siendo la misma el documento concluyente de la comisión. Los datos recogidos son: (Fecha, hora y lugar donde se realiza la reunión, las personas invitadas, el orden del día, atenuantes y agravantes, la tipificación de la falta y una descripción de la misma) • Permite adicionar/actualizar dictamen. El sistema debe permitir la opción de adicionar/actualizar un dictamen que contiene la fecha de creación del mismo y la medida impuesta. • Permite listar CD. El sistema debe de ser capaz de mostrar un listado con las comisiones mostrando la información de: (Fecha, Número de Comisión) • Permite listar comisiones archivadas. El sistema debe ser capaz de mostrar un listado con las comisiones que han sido archivadas debido a que las mismas no procedieron.
Observaciones:

Anexo 3: Tabla 19 Historia de usuario Gestionar notificaciones

Historia de usuario	
Número:4	**Nombre Historia de Usuario:** Gestionar notificaciones
Modificación de Historia de Usuario Número:	
Usuario: Etna Lina Lavin Scott	**Iteración asignada:2**
Prioridad en Negocio: Bajo	**Puntos Estimados:** 0.86 semanas (6días)
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 0.86 semanas (6días)
Descripción: • Permite adicionar notificaciones. • Permite listar notificaciones El sistema debe ser capaz de mostrar un listado con las notificaciones del usuario autenticado.	
Observaciones:	

Anexo 4:Tabla 20 Historia de usuario Gestionar comisiones

Historia de usuario	
Número:7	**Nombre Historia de Usuario:** Gestionar comisiones
Modificación de Historia de Usuario Número:1	
Usuario: Etna Lina Lavin Scott	**Iteración asignada: 1**
Prioridad en Negocio: Alto	**Puntos Estimados:** 0.86 semanas (6días)
Riesgo en Desarrollo: Medio	**Puntos Reales:** 0.86 semanas (6días)
Descripción: • Permite adicionar/actualizar comisión. El sistema debe permitir la opción de adicionar/actualizar una comisión: (nombre, si esta activa, descripción). Además las personas que formaran parte de dicha comisión asignándole el rol correspondiente. • Permite listar comisiones. El sistema debe de ser capaz de mostrar un listado con todas las comisiones ya creadas mostrando varia información como: nombre, descripción, si esta activa y el jefe de la comisión.	
Observaciones:	

Anexo 5:Tabla 21 Historia de usuario Gestionar plantillas

Historia de usuario	
Número:8	**Nombre Historia de Usuario:** Gestionar plantillas
Modificación de Historia de Usuario Número:	
Usuario: Etna Lina Lavin Scott	**Iteración asignada:3**
Prioridad en Negocio: bajo	**Puntos Estimados:** 0.86 semanas (6días)
Riesgo en Desarrollo: bajo	**Puntos Reales:** 0.86 semanas (6días)
Descripción: • Permite adicionar/actualizar plantilla. El sistema debe permitir la opción de adicionar/actualizar una plantilla para un determinado documento o reporte. • Permite listar plantilla. El sistema debe de ser capaz de mostrar un listado con todas las plantillas ya creadas mostrando varia información como: nombre de la plantilla y si esta activa. • Permite mostrar vista previa de la plantilla. • Permite generarla en formato pdf.	
Observaciones:	

Anexo 6: Tabla 22 Historia de usuario Gestionar apelación

Historia de usuario	
Número:9	**Nombre Historia de Usuario:** Gestionar apelación
Modificación de Historia de Usuario Número:	
Usuario: Etna Lina Lavin Scott	**Iteración asignada: 2**
Prioridad en Negocio: bajo	**Puntos Estimados:** 0.86 semanas (6días)
Riesgo en Desarrollo: bajo	**Puntos Reales:** 0.86 semanas (6días)
Descripción: • Permite adicionar/actualizar apelación. El sistema debe permitir la opción de adicionar/actualizar una apelación donde debe seleccionar el implicado en la comisión disciplinaria que la solicita, la fecha de solicitud, fecha de entrega, descripción. • Permite listar apelación. El sistema debe de ser capaz de mostrar un listado con todas las apelaciones mostrando la información de: (Fecha de solicitud, involucrado que solicita)	
Observaciones:	

Anexo 7: Tabla 23 Historia de usuario Gestionar nomencladores

Historia de usuario	
Número:10	**Nombre Historia de Usuario:** Gestionar nomencladores
Modificación de Historia de Usuario Número:	
Usuario: Etna Lina Lavin Scott	**Iteración asignada:2**
Prioridad en Negocio: Medio	**Puntos Estimados:** 0.86 semanas (6días)
Riesgo en Desarrollo: bajo	**Puntos Reales:** 0.86 semanas (6días)
Descripción: • Permite actualizar origen de la denuncia. • Permite actualizar origen de la opinión. • Permite actualizar la tipificación de la falta.	
Observaciones: Un nomenclador es una clase que no cambia sus valores en determinado tiempo.	

Anexo 8: Tabla 24 tarjeta CRC Usuario

Usuario	
Responsabilidades 1. Autenticar usuario **Atributos** 1. Usuario 2. Contraseña 3. Activo 4. Roles 5. Persona	**Colaboradores** 1. Persona 2. Rol

Anexo 9: Tabla 25 tarjeta CRC Persona

Persona	
Responsabilidades 1. Guardar datos de persona **Atributos** 1. Nombre 2. Apellidos 3. usuario 4. Número del expediente 5. Solapín 6. Grupo 7. Categoría	**Colaboradores**

Anexo 10: Tabla 26 tarjeta CRC Rol

Rol	
Responsabilidades 1. Almacena los roles de usuarios **Atributos** 1. Descripción 2. Activo	**Colaboradores**

Anexo 11: Tabla 27 tarjeta CRC Documento

Documento	
Responsabilidades 1. Almacena los documentos **Atributos** 1. Fecha de creación 2. Autor 3. Estado 4. Tipo de documento	**Colaboradores** 1. Proceso

Anexo 12: Tabla 28 tarjeta CRC Adjunto.

Adjunto	
Responsabilidades 1. Permitir guardar archivos **Atributos** 1. Nombre 2. Extensión 3. Fecha de recepción	**Colaboradores** 1. Proceso

Anexo 13: Tabla 29 tarjeta CRC Estado

Estado	
Responsabilidades 1. Guarda los estado del proceso **Atributos** 1. Descripción 2. Activo	**Colaboradores**

Anexo 14: Tabla 30 tarjeta CRC EstadoPaso

EstadoPaso	
Responsabilidades 1. Almacena los estados y los pasos del proceso **Atributos** 1. Descripción 2. Activo	**Colaboradores** 1. Estado 2. Paso 3. Usuario 4. Observación 5. Proceso

Anexo 15: Tabla 31 tarjeta CRC Paso

Paso	
Responsabilidades 1. Almacena lo pasos del proceso **Atributos** 1. Descripción 2. Activo	**Colaboradores**

Anexo 16: Tabla 32 tarjeta CRC Tipo de documento

Tipo de documento	
Responsabilidades 1. Almacena los tipos de documentos **Atributos** 1. Descripción 2. Activo	**Colaboradores**

Anexo 17: Tabla 33 tarjeta CRC Tipo PersonaProceso

Tipo PersonaProceso	
Responsabilidades 1. Almacena los tipos de personas **Atributos** 1. Descripción 2. Activo	**Colaboradores** 1. Paso

Anexo 18: Tabla 34 tarjeta CRC Notificación

Notificación	
Responsabilidades 1. Enviar alertas 2. Notificar eventos **Atributos** 1. Fecha 2. Operación 3. Es alerta 4. Cantidad de veces notificado	**Colaboradores** 1. Proceso

Anexo 19: Tabla 35 tarjeta CRC PersonaProceso

PersonaProceso	
Responsabilidades Almacena las personas que están asociadas a un proceso con su rol **Atributos** 1. Tipo de persona	**Colaboradores** 1. Persona 2. Datos 3. Proceso 4. Tipo Persona Proceso

Anexo 20: Tabla 36 tarjeta CRC Comisión disciplinaria

Comisión disciplinaria	
Responsabilidades 1. Almacena los datos de las comisiones disciplinarias **Atributos** 1. Falta cometida 2. Tipo de falta	**Colaboradores** 1. Denuncia 2. Decisión 3. Comisión 4. Conclusión 5. Resolución 6. Prueba

Anexo 21: Tabla 37 tarjeta CRC DecisiónDenuncia

DecisiónDenuncia	
Responsabilidades 1. Almacena decisiones **Atributos** 1. Procede 2. No procede	**Colaboradores**

Anexo 22: Tabla 38 tarjeta CRC ProcedenciaDenuncia

ProcedenciaDenuncia	
Responsabilidades 1. Almacena las procedencias de las denuncias **Atributos** 1. Nombre 2. Descripción	**Colaboradores**

Anexo 23: Tabla 39 tarjeta CRC Datos

Datos	
Responsabilidades 1. Almacena los datos de las personas proceso **Atributos** 1. Datos	**Colaboradores** 1. PersonaProceso

Anexo 24: Tabla 40 Tareas de Programación de la iteración 2

Iteración	HU	Tareas de programación
Iteración 2	Gestionar Reporte	Crear interfaz para gestionar reporte.
		Crear menú para gestionar reporte.
		Crear código fuente para reportar estudiantes denunciados, reincidentes, con condicional y sancionados por separación.
		Crear interfaz para cada configuración del reporte.
		Crear código fuente para reportar la cantidad de CD.
	Gestionar Apelación	Crear interfaz para gestionar apelación.
		Crear menú para gestionar apelación.
		Listar apelaciones.
		Crear interfaz para cada configuración de la apelación.
		Crear código fuente para adicionar o actualizar apelación.
	Gestionar nomencladores	Crear interfaz para gestionar nomencladores.
		Crear menú para gestionar nomencladores.
		Crear código fuente para actualizar roles.
		Crear código fuente para actualizar origen de denuncia.
		Crear código fuente para actualizar origen de la opinión.
		Crear interfaz para cada configuración de los nomencladores.
		Crear código fuente para actualizar tipificación de la falta.
	Gestionar notificaciones	Crear interfaz para gestionar notificaciones.
		Crear menú para gestionar notificaciones.
		Listar notificaciones.
		Crear código fuente para adicionar notificaciones.

Anexo 25: Tabla 41 Tareas de Programación de la iteración 3

Iteración	HU	Tareas de programación
Iteración 3	Gestiona plantillas	Crear interfaz para gestionar plantillas.
		Crear menú para la gestión de las plantillas.
		Listar plantillas.
		Crear código fuente para adicionar o actualizar plantillas.
		Crear interfaz para cada configuración de las plantillas.
		Crear código fuente para mostrar una vista previa de la plantilla.
		Crear código fuente para generar la plantilla a pdf.
	Gestionar términos de tiempo	Crear interfaz para gestionar términos de tiempo.
		Crear interfaz para cada configuración de los términos de tiempo.
		Crear menú para la gestión de términos de tiempo.
		Listar términos.
		Listar prórroga.
		Crear código fuente para actualizar fechas y términos.

Anexo 26: Acta de liberación entregado por el centro de calidad de CEGEL

UCI Universidad de las Ciencias Informáticas | FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

CEGEL

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: 11/06/2014

Emitida a favor de: Tesis Sistema para Informatizar los Procesos de Comisiones Disciplinarias (SIPCD)

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
Sistema para Informatizar los Procesos de Comisiones Disciplinarias (SIPCD)	*1.0*	*0*	*3*	*Evaluación dinámica* *Pruebas de Funcionalidad*	*11/06/14*

Ing. Raúl Velázquez Alvarez
Asesor de Calidad del Centro

Etna Lina Lavin Scott
Autor

UCI Universidad de las Ciencias Informáticas
Centro de Gobierno Electrónico

1

Anexo 27: Reconocimiento obtenido en el evento de la Jornada Científica

UCI Universidad de las Ciencias Informáticas

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FEDERACIÓN ESTUDIANTIL UNIVERSITARIA

otorga el presente

RECONOCIMIENTO

a: Etna Luna Lanin Scott

por obtener: Mención

con el trabajo: Desarrollo de una Aplicación Inf para el proceso de gestión de las Comisiones disciplinarias en la Fac. 3.

En la XII edición de la JORNADA CIENTÍFICA ESTUDIANTIL en la FACULTAD 3.

"La tecnología es el reflejo del fanatismo del hombre por sobrevivir"

DADO A LOS 9 DÍAS DEL MES 5 DEL 2014

"AÑO 56 DE LA REVOLUCIÓN"

ERNESTO ORTIZ MUÑOZ
PRESIDENTE DE LA FEU

UCI Universidad de las Ciencias Informáticas
FACULTAD 3

DrC. RAFAEL RODRÍGUEZ PUENTE
DECANO