SISTEMA DE INFORMACIÓN OFICINA DE CONTROL INTERNO DISCIPLINARIO

SILVIA JULIANA BERMUDEZ GALLO

LEIDY TATIANA TARAZONA LIZCANO

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE CIENCIAS FÍSICO MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2011

SISTEMA DE INFORMACIÓN OFICINA DE CONTROL INTERNO DISCIPLINARIO

SILVIA JULIANA BERMUDEZ GALLO

LEIDY TATIANA TARAZONA LIZCANO

Proyecto para optar el título de:

Ingenieras de Sistemas

Director:

Ing. ENRIQUE TORRES LOPEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE CIENCIAS FÍSICO MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2011

DEDICATORIA

A Dios quien siempre está a mi lado dándome la fuerza, valentía y sabiduría para que todo salga bien,

A mis padres quienes siempre me brindaron su apoyo y confianza para lograr esta meta,

A mi hermana que me ha acompañado y apoyado en todo momento,

A Jose Antonio por su amor y apoyo incondicional

Silvia Juliana Bermúdez Gallo.

A mis padres por todo su esfuerzo y apoyo para poder iniciar y culminar con éxito mi carrera,

A mis hermanas que me han acompañado y apoyado en todo momento,

Leidy Tatiana Tarazona Lizcano.

AGRADECIMIENTOS

Nuestros más sinceros agradecimientos a:

Dios, quien ha estado con nosotros en todos los momentos de nuestra vida, guiándonos, ayudándonos, mostrándonos el camino a seguir y dándonos fortaleza en los momentos difíciles.

A nuestros padres, que nos han tenido paciencia para mostrarnos la forma correcta y apropiada de hacer las cosas. Por el apoyo moral y anímico que nos ofrecieron, ya que sin éste hubiese sido muy difícil alcanzar este triunfo.

Al Ingeniero Enrique Torres López, nuestro director de proyecto, por brindarnos su apoyo incondicional y conocimientos necesarios y ayudarnos a sortear los diversos obstáculos que se nos presentaron durante el desarrollo del proyecto.

Al Doctor Javier Octavio Trillos, quien al momento de iniciar el proyecto se encontraba a cargo de la dirección de la Oficina de Control Interno Disciplinario por su valiosa colaboración, dedicación y aporte de conocimientos a favor del proyecto.

A todas y a cada una de las personas que de una u otra manera aportaron para que pudiéramos alcanzar ésta tan anhelada meta.

TABLA DE CONTENIDO

INTRODUCCION.....22

PARTE I. FUNDAMENTOS.....23

1. PRESENTACION DEL PROYECTO.....23

1.1 DESCRIPCION DEL PROYECTO. 23

1.1.1 OBJETIVO GENERAL. 23

1.1.2 OBJETIVOS ESPECÍFICOS. 23

1.2 JUSTIFICACION. 25

1.2.1 ANTECEDENTES Y DESCRIPCIÓN DEL PROBLEMA.....25

1.2.2 IMPACTO 25

1.2.3 VIABILIDAD. 26

2. MARCO TEÓRICO.....27

2.1 OFICINA DE CONTROL INTERNO DISCIPLINARIO 27

2.1.1 MISIÓN.....27

2.1.2 VISION. 27

2.1.3 ESTRUCTURA ORGANIZATIVA..... 28

2.1.4 FUNCIONES DEL DIRECTOR.....28

2.2 PROCEDIMIENTO. 29

2.3 GENERALIDADES DEL ENTORNO DE DESARROLLO 29

2.3.1 APLICACIONES ORIENTADAS A LA WEB.....29

2.3.1.1 LA WEB COMO SISTEMA DE INFORMACIÓN. 30

2.3.1.2. TECNOLOGÍAS WEB. 31

2.3.1.3 PLATAFORMAS DE DESARROLLO WEB. 33

2.3.1.4 WEB 2.0. 34
2.3.2 DISEÑOS CON EL ESTÁNDAR UML. 35
2.3.2.1 INTRODUCCIÓN A UML 35
2.3.2.2 ELEMENTOS COMUNES A TODOS LOS DIAGRAMAS. 36
2.3.2.3 DIAGRAMAS DE CASOS DE USO. 38
2.3.2.4 DIAGRAMAS DE SECUENCIA. 40
2.3.2.5 DIAGRAMA DE CLASES. 41
2.3.2.6 DIAGRAMA ENTIDAD – RELACIÓN 43
2.3.3 JAVA ENTERPRISE EDITION 5.0 (JEE 5)....44
2.3.4 . ENTORNO DE DESARROLLO....45
2.3.4.1 JAVA SERVER FACES (JSF)....45
2.3.4.2 OBJECT RELATIONAL MAPPING / JPA 46
2.3.4.3 EJB 3.0 46
2.3.5 SERVIDOR DE APLICACIONES – JBOSS....46
2.3.6 ENTERPRISE ARCHITECT 47
2.3.7 CONSTRUCCIÓN DE PROTOTIPOS DE SOFTWARE. 47

3. METODOLOGÍA DE DESARROLLO....50

3.1 CICLO DE VIDA DEL PROYECTO. 50
3.1.1 ANÁLISIS DE REQUERIMIENTOS: 50
3.1.2 DISEÑO....51
3.1.3 IMPLEMENTACIÓN DE LA APLICACIÓN. 51
3.1.4 PRUEBAS DEL SOFTWARE. 52
3.1.5 AJUSTES. 52
3.2 METODOLOGIA DE DESARROLLO DEL PROYECTO. 53
3.2.1 MODELO DE CONSTRUCCIÓN DE PROTOTIPOS....53
3.2.2 ESTRUCTURA DEL MODELO DE CONSTRUCCIÓN DE PROTOTIPOS. 54
3.2.3 PROCEDIMIENTO PARA LA METODOLOGÍA PLANTEADA. 55

PARTE II. DESARROLLO DEL SISTEMA.....57

4. APLICACIÓN DE LA METODOLOGÍA.....57

4.1 LEVANTAMIENTO DE REQUERIMIENTOS 57

4.1.1 DESCRIPCIÓN GENERAL.....57

4.1.2 FUNCIONES QUE DEBE CUMPLIR EL SISTEMA.....58

4.1.2.1 DEL REGISTRO DE LAS QUEJAS, INFORMES O ANONIMOS 58

4.1.2.2 ADMINISTRACIÓN O MANTENIMIENTO DEL SISTEMA 60

4.1.2.3 DOCUMENTOS SECRETARIALES.....60

4.1.2.4 GESTION DE LAS QUEJAS,INFORMES O ANONIMOS O PROCESO INVESTIGATIVO.....61

4.1.2.5 CONSULTAS.....62

4.1.2.6 ESTADISTICAS.....62

4.2 ESTÁNDARES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN.....62

4.2.1 ASPECTOS GENERALES.....62

4.2.2 DOCUMENTACIÓN DE LOS DIAGRAMAS DE DISEÑO.....66

4.2.3 SINTAXIS DE NOMBRES EN JAVA.....67

4.2.4 DOCUMENTACIÓN 71

4.2.5 CAPA DE PRESENTACIÓN.....71

4.3 DIAGRAMAS UML 76

4.3.1 DIAGRAMA DE CASOS DE USO.....77

4.3.1.1 IDENTIFICACIÓN DE LOS ACTORES 77

4.3.1.2 CASOS DE USO POR ACTOR.....78

4.3.2 DIAGRAMAS DE SECUENCIAS 83

4.3.3 DIAGRAMA DE CLASES 85

4.4 PROTOTIPO INICIAL.....86

4.4.1 GENERALIDADES DEL PROTOTIPO INICIAL 86

4.4.2 INTERFAZ RESULTADO DEL PROTOTIPO INICIAL.....87

4.5 PROTOTIPO FINAL.....90

4.5.1 GENERALIDADES DEL PROTOTIPO FINAL 90

4.5.2 PROYECTO ENMARCADO EN EL ESQUEMA DE SEGURIDAD DE LA UIS. 100
4.5.2.1 ESTRUCTURA DE LA BASE DE DATOS SOPORTE 100
4.5.2.2 ENTORNO DE NAVEGACIÓN.....103
4.5.2.3 ENTORNO DE CONTROL DE DATOS.....103
4.5.2.4 AUDITORÍA 104
4.5.3 ORGANIZACIÓN DE DIRECTORIOS. 104
4.5.4 DOCUMENTACIÓN DE PROGRAMAS FUENTE.....108

5. CONCLUSIONES.....118

6. RECOMENDACIONES 120

7. BIBLIOGRAFÍA.....121

LISTADO DE FIGURAS

Figura 1.Elementos de un Sistema de Información 31
Figura 2. Historia de UML 36
Figura 3. Ejemplo de una Nota UML. 37
Figura 4. Ejemplo de una Dependencia UML. 37
Figura 5. Ejemplo de Diagrama de Caso de Uso 39
Figura 6. Ejemplo de Diagrama de Secuencia 40
Figura 7. Ejemplo de Diagrama de Clases 42
Figura 8. Ejemplo Diagrama Entidad-Relacion 43
Figura 9. Modelo de Construccion de Prototipos. 54
Figura 10. Estructura del Modelo de Construcción de Prototipos 54
Figura 11 - Plantilla Principal División de Servicios de Información 72
Figura 12. Plantilla de Contenido. Division de Servicios de Informacion 72
Figura 13. Plantilla Contenido con Formulario. Division Servicios de Informacion 73
Figura 14. Casos de Uso Secretaria CID 78
Figura 15. Caso de Uso: Recibir informes, quejas o anonimos 79
Figura 16. Casos de Uso asociados con director CID(parcial) 81
Figura 17. Caso de Uso. Analizar informes, quejas o anónimos. 82
Figura 18.Diagrama Secuencia caso de uso. Recibir Quejas, Informes o anonimos 84
Figura 19. Diagrama de Clases parcial SIOCID 85
Figura 20. Vista de Pantallazo inicial. Prototipo Inicial 87
Figura 21. Vista registro de una queja, informe o anónimo. Prototipo Inicial 88
Figura 22. Vista de la queja, informe o anónimo ya registrado. Prototipo Inicial. 89
Figura 23. Listado de Quejas, informes o anónimos. Prototipo Inicial. 90
Figura 24. Vista del pantallazo inicial. Prototipo Final. 91
Figura 25. Vista Registro Queja. Prototipo Final. 92

Figura 26. Vista para el registro de investigados y declarantes. Prototipo Final. ...93
Figura 27. Vista de las quejas informes o anónimos. Prototipo Final.....94
Figura 28Vista Detalle de la queja, informe o anónimo. Prototipo Final.....95
Figura 29. Vista de las quejas informes o anónimos. Prototipo Final.....96
Figura 30. Vista del detalle de la queja 97
Figura 31. Vista cuadro desplegable del detalle de la queja 97
Figura 32. Vista del pantallazo para iniciar el modulo de consultas. 98
Figura 33. Vista inicial de la opción consultar por queja. 99

LISTADO DE TABLAS

Tabla 1. Descripcion Caso de Uso: Recibir informes, quejas o aninomos.....80
Tabla 2. Descripcion Caso de uso. Analizar informes ,quejas o anonimos.....83

RESUMEN

1. **TITULO:** DESARROLLO DE LA VERSION DEL SISTEMA DE INFORMACIÓN DE LA OFICINA DE CONTROL INTERNO DISCIPLINARIO PARA LA UNIVERSIDAD INDUSTRIAL DE SANTANDER. *

2. **AUTORES:** BERMUDEZ GALLO SILVIA JULIANA, TARAZONA LIZCANO LEIDY TATIANA**

3. **PALABRAS CLAVE:** Web, Quejas, Cliente Servidor, Java, Prototipos, Control Interno, Investigación.

4. DESCRIPCION:

Ante las exigencias de ahorro en tiempo y en agilidad en los procesos que se llevan a cabo en la oficina de Control Interno Disciplinario de la Universidad Industrial de Santander los cuales se vienen desarrollando de manera manual generando una gran cantidad de documentos, y ante la necesidad de ser más eficientes en las respuestas generadas de los procesos, se requiere la creación de un sistema de información que los minimice, que realice online los procesos, que brinde soporte y eficacia probatoria de los documentos que allí se procesen, que respete y cumpla las leyes colombianas existentes para este tipo de procesos, que tenga una interfaz agradable, dinámica, de fácil manejo, pero que brinde la seguridad necesaria que permita proteger y mantener la calidad de reserva sumarial o de información confidencial.

Es por esto que se hace necesario la realización de un proyecto cuyo objetivo general sea: "Diseñar, implementar y poner en funcionamiento el sistema de información orientado a web que permita el registro, seguimiento y control de los procesos que adelanta la oficina de Control Interno Disciplinario, manejo de alarmas de vencimiento de trámites, consultas e informes solicitados."

También generar mediante un ambiente manejado por auditorias la confianza necesaria para que a la información que allí este consignada se le pueda hacer el seguimiento respectivo y se pueda determinar qué tipo de acciones se le realizan, desde que equipo y por cual persona.

Este nuevo producto se realizo con los estándares de calidad y eficiencia de la División de Servicios de Información de la Universidad Industrial de Santander establecidos para los desarrollos de los nuevos sistemas de información.

———————————————————

* Trabajo de Investigación

** Facultad de Ingenierías Físico – Mecánicas, Escuela de Ingeniería de Sistemas e Informática. Director: Ingeniero Enrique Torres López.

SUMMARY

1. **TITLE:** DEVELOPMENT OF INFORMATION SYSTEM VERSION OF THE OFFICE OF INTERNAL DISCIPLINARY CONTROL AT THE INDUSTRIAL UNIVERSITY OF SANTANDER. *

2. **AUTHORES:** BERMUDEZ GALLO SILVIA JULIANA, TARAZONA LIZCANO LEIDY TATIANA **

3. **KEYWORDS:** Web Information System, Complaints, Reports, Anonymous, Client Server Applications, Java, Model Prototypes Construction.

4. DESCRIPTION:

Before the requirements of saving in time and in agility in the processes that there carry out in the office of Internal Disciplinary Control of the Industrial University of Santander which they come developing in a manual way generating a great quantity of documents, and before the need to be more efficient in the answers generated of the processes, there asks from itself the creation of an information system that minimizes them, that realizes online the processes, which there offers support and evidential efficiency of the documents that there are processed, That respects and fulfills the Colombian existing laws for this type of processes, which there has an agreeable, dynamic interface, of easy managing, but that offers the necessary safety that allows to protect and to support the quality of summing reservation or of confidential information.

It is for this that makes to itself necessary the accomplishment of a project which general aim is: " To design, to help and to put in functioning the information system orientated to web that allows the record, follow-up and control of the processes that advances the office of Internal Disciplinary Control, managing alarm of maturity of steps, consultations and requested reports. "

Also to generate by means of an environment handled by audits the necessary confidence in order that to the information that there this one recorded could do the respective follow-up to him and it could determine what type of actions are realized, since I equip and for which it presents.

This new product I realize with the standards of quality and efficiency of the Division of Services of Information of the Industrial University of Santander established for the developments of the new information systems.

———————————————————

* Project

** Physique-Mechanics Sciences Department, Computer Science Engineering, Engineer Enrique Torres Lopez.

TERMINOS Y DEFINICIONES

Queja: Es una de las formas en que se acciona o pone en movimiento el aparato disciplinario, y constituye un supuesto de reclamación, denuncia o crítica de la actuación administrativa. También es el medio por el cual el beneficiario pone de manifiesto su inconformidad con la actuación de determinado funcionario o con la forma y condiciones en que se preste o no un servicio.

Inhibitorio: Decisión que no hace tránsito a cosa decidida, a través del cual el operador disciplinario se abstiene de adelantar una actuación disciplinaria por encontrar, entre otros aspectos, que la queja es manifiestamente temeraria, se refiera a hechos disciplinariamente irrelevantes, de imposible ocurrencia o presentados de manera absolutamente inconcreta o difuso.

Indagación Preliminar: Etapa opcional que se adelanta dentro del procedimiento ordinario, cuya finalidad es establecer la procedencia de la investigación disciplinaria, verificando la ocurrencia de la conducta, determinando si es constitutiva de falta disciplinaria; o identificando e individualizando al autor de la misma.

Pliego de Cargos: Es la evaluación de la investigación y constituye la presunción de incursión en la falta disciplinaria, que se hace en contra del disciplinado, con la cual ha de ser juzgado y sometido a un fallo, ya sea absolutorio o sancionatorio.

Descargos: Pieza procesal en la cual el disciplinado se pronuncia y ejerce su defensa frente a los cargos objeto de acusación, solicitando práctica de pruebas.

Fallo: Decisión de fondo en la cual una vez cumplidas todas las etapas procesales, se resuelve la responsabilidad del investigado, a través de una absolución o una imposición de sanción.

Sanción: Decisión de carácter administrativo que se impone a un servidor público considerando responsable de cometer una falta disciplinaria, previo el agotamiento de un proceso ordinario, en la cual se cumple una función preventiva, correctiva y garantizadora de los principios Constitucionales y Legales que se deben observar en el ejercicio de la función pública.

Citación: Mecanismo mediante el cual se le solicita al presencia de una o ambas partes del proceso.

Notificación: Actuación procesal a través de la cual se hace efectivo el principio de publicidad de las actuaciones administrativas, dándose a conocer las decisiones disciplinarias a los sujetos procesales, personalmente, por estado, por edicto, por estrados o por conducta concluyente.

Particular: Persona de la comunidad que no es funcionario ni proveedor ni estudiante de la universidad.

Estudiante UIS: Persona matriculada en algún programa académico de la UIS.

Empelado UIS: Personal vinculado a la Universidad mediante contrato de trabajo.

UAA: Unidad Académica y Administrativa de la Universidad Industrial de Santander.

Quejoso: Es la persona que presenta la queja, el informe o el anónimo en la oficina de control interno.

Investigado: Es la persona sobre la cual recae la queja, el informe o el anónimo presentado por el quejoso.

Declarante: Es la persona que es citada dentro de un proceso investigativo para que dé su versión de los hechos que son parte de la investigación.

OCID: Oficina Control Interno Disciplinario.

Interfaz: La idea fundamental en el concepto de interfaz, es el de mediación. La interfaz es lo que “media”, lo que facilita la comunicación, la interacción, entre dos sistemas de diferente naturaleza, típicamente el ser humano y una máquina como el computador. Esto implica, además, que se trata de un sistema de traducción, ya que los dos se comunican con lenguajes diferentes: verbo-icónico en el cado del hombre y binario en el caso de la máquina.

Java: Lenguaje de Programación que se caracteriza por tener una arquitectura que permite que el código escrito funcione en multitud de sistemas operativos sin ser modificado.

Método: Es una operación que define como se comporta un objeto.

Módulo: Se utiliza como sinónimo de Subsistema.

Sistema de Información: Aplicación comercial para el computador. Está constituida por la base de datos, los programas de aplicación, los procedimientos manuales y automatizados, e incluye los sistemas computacionales que realizan procesamiento.

Clase: Una clase de objetos describe un grupo de objetos con propiedades similares, con relaciones comunes entre otros y con una semántica común.

INTRODUCCION

El presente documento corresponde al informe de desarrollo del proyecto de grado "SISTEMA DE INFORMACIÓN OFICINA DE CONTROL INTERNO DISCIPLINARIO"

En la actualidad, el mundo moderno y digitalizado, obliga a las organizaciones, ya sean de carácter público o privado a ofrecer facilidades de comunicación entre las entidades y los diferentes usuarios que interactúan con ellas.

Por todo lo anterior, la universidad dando cumplimiento a estas directrices normativas y tecnológicas, implantará el sistema de Control Interno Disciplinario.

Este proyecto pretende diseñar, implementar e implantar una versión del sistema de control interno disciplinario, tomando como esquema las diferentes solicitudes que puede hacer un usuario: Quejas, Informes o Anónimos, y darles trámite de manera efectiva teniendo como premisa que cada solicitud es importante y debe resolverse de manera efectiva y en el menor tiempo posible.

Para la realización de éste proyecto se contó con el apoyo de la División de Servicios de Información (DSI) de la Universidad Industrial de Santander, entidad encargada de la administración y desarrollo de la tecnología de información, recibiendo todo el soporte y asesoría necesarios para ello, utilizando los estándares definidos por la DSI para el desarrollo de los nuevos sistemas de información y herramientas tales como Informix, Java EE 5, J-boss seam y diferentes tipos de software ya licenciados por la UIS.

En el transcurso del desarrollo del proyecto se realizaron diversas reuniones con la unidad directamente beneficiada, la Dirección de Control Interno Disciplinario para realizar el levantamiento de requerimientos, refinar prototipos y producto final, proceso que se describirá en el presente documento.

PARTE I. FUNDAMENTOS

1. PRESENTACION DEL PROYECTO

1.1 DESCRIPCION DEL PROYECTO.

1.1.1 Objetivo General. Diseñar, implementar y poner en funcionamiento el sistema de información orientado a web que permita el registro, seguimiento y control de los procesos que adelanta la oficina de Control Interno Disciplinario, manejo de alarmas de vencimiento de trámites, consultas e informes solicitados.

1.1.2 Objetivos Específicos. El sistema de información de la oficina de Control Interno Disciplinario tiene como objetivos específicos:

- Ofrecer a la comunidad en general un medio para que den a conocer a la oficina de Control Interno Disciplinario las inquietudes que tiene sobre el comportamiento inadecuado de miembros de la comunidad universitaria.

- Permitir el registro de quejas e informes de acciones que afectan el nombre y buen funcionamiento de la universidad.

- Facilitar el análisis de las quejas e informes recibidos, hacer las citaciones pertinentes a través de correo certificado, medios electrónicos o edictos en caso de ser necesario.

- Crear un entorno para la gestión de pruebas consistente en la recolección, registro y análisis de las pruebas realizadas.

- Facilitar formatos para la elaboración del borrador del fallo de acuerdo al análisis de las pruebas realizadas.

- Facilitar a la Secretaría General de la universidad la revisión del fallo y su interacción con la oficina de Control Interno Disciplinario, para llegar a un consenso sobre el fallo definitivo.

- Facilitar a la dirección de la universidad el registro del fallo definitivo y su comunicación a las personas involucradas.

- Facilitar la consulta de las quejas e informes recibidos, el trámite seguido y el fallo dado para uno de ellos. Esta consulta se puede hacer por persona involucrada, por unidad académico administrativa, por rango de fechas.

- Generar estadísticas de número de quejas por período, por unidad académico administrativas y por personas involucradas.

- Crear las pistas de auditoría necesarias para hacer seguimiento de las actividades realizadas por los usuarios autorizados para hacer uso del sistema, su consulta e impresión.

1.2 JUSTIFICACION.

1.2.1 Antecedentes y descripción del problema.

La División de Servicios de Información de la Universidad Industrial de Santander en su afán de generar soluciones informáticas de la más alta calidad técnica que faciliten el proceso de modernización institucional y que faciliten a la comunidad universitaria el acceso a la información ve de vital importancia para toda la institución, contar con un sistema de información orientado a la web que permita el registro de los procesos que adelanta la Oficina de Control Interno Disciplinario, su seguimiento y control, generación de alarmas de los trámites que se deben seguir, generación de reportes, consultas y estadísticas, debidamente integrado con los sistemas de información de la institución y realizado con tecnologías avanzadas cumpliendo los estándares de desarrollo establecidos por la universidad para tal fin.

1.2.2 Impacto.

Este sistema es un medio que le permite la dirección de Control Interno Disciplinario de la universidad modernizar su manera de hacer los trámites legales de las quejas, informes o anónimos que llegan hasta este despacho, permitiendo que sean más eficientes las respuestas a los mismos, que los procesos sean más agiles, controlados y que se eviten olvidos de pruebas y citaciones que se deben realizar cada día, además de organizar una agenda para que tanto el director de la oficina como su asistente y secretaria estén enterados de todo lo que se debe realizar día a día.

La oficina de Control Interno Disciplinario y sus funcionarios podrán de una manera práctica llevar a cabo funciones que antes podrían llegar a ser dispendiosas, de una manera segura y fácil para todos, logrando así cumplir con

sus funciones que han sido estipuladas por la universidad como son atender las solicitudes y quejas presentadas por la comunidad universitaria y el público en general dándoles el debido proceso legal y transparente.

1.2.3 Viabilidad.

La viabilidad del proyecto es total, debido a que este es realizado con herramientas de desarrollo de software de libre distribución, o adquiridos actualmente por la división de Servicios de Información, y se desarrolla en los servidores pertenecientes a la universidad.

En cuanto a hardware se cuenta con instalaciones adecuadas, con los equipos requeridos y el soporte tecnológico necesario para el desarrollo del mismo, Además se tiene la supervisión por parte del director del proyecto y la colaboración del equipo de trabajo de la División de Servicios de información.

2. MARCO TEÓRICO

2.1 OFICINA DE CONTROL INTERNO DISCIPLINARIO.

2.1.1 Misión.

Atendiendo lo dispuesto en la ley 734 de 2002, corresponde a las oficinas de control disciplinario conocer de los asuntos disciplinarios contra los servidores públicos de sus dependencias, para el caso de la Universidad industrial de Santander los destinatarios de la ley disciplinaria son el personal docente y administrativo vinculado a través de una relación legal y reglamentaria y los trabajadores oficiales vinculados mediante contrato laboral.

Si bien la finalidad descrita en la ley es fundamentalmente sancionadora, la garantía de la función pública puede lograrse también desde una esfera preventiva, advirtiendo e informando al servidor de sus derechos, deberes y obligaciones para con la Universidad, asunto que se logra con la continua capacitación y asesoría a los servidores de la institución.

2.1.2 Visión.

La oficina de Control Interno Disciplinario pretende concientizar al servidor de su rol preponderante en la sociedad, el compromiso por el respeto y cumplimiento de sus deberes funcionales y la necesidad de salvaguardar el patrimonio público. En ese orden de ideas, se busca prevenir las faltas disciplinarias, combatir la corrupción y retornar a la figura del servidor público como persona digna, ética y responsable al servicio de la comunidad.

2.1.3 Estructura Organizativa.

Mediante Acuerdo Superior Nº 070 de 1998, se creó la Oficina de Control Interno Disciplinario como una dependencia adscrita a la Rectoría con las funciones determinadas en la ley disciplinaria y en los reglamentos universitarios en cuanto a la titularidad de la acción disciplinaria.

En la Resolución de Rectoría Nº 361 de 2000, se asignaron funciones al Director de la Oficina de Control Interno Disciplinario.

2.1.4 Funciones del director

- Asumir el conocimiento de todos los procesos disciplinarios que se adelanten contra los servidores de la Universidad.
- Efectuar la evaluación de la investigación mediante auto de formulación de cargos o archivo definitivo.
- Cumplida la instrucción, remitir el expediente al funcionario que de conformidad con los reglamentos internos sea el competente para proferir la decisión respectiva.
- Responder por la oportuna realización de las diligencias requeridas en cada una de las etapas de los procesos disciplinarios de su competencia.
- Reportar a la División de Registro y Control de la Procuraduría General de la Nación los resultados de las Investigaciones adelantadas.
- Realizar las demás funciones que le asigne el Rector, relacionadas con el cargo que desempeña.

2.2 PROCEDIMIENTO

En la oficina de Control Interno Disciplinario se sigue el siguiente procedimiento para la recepción y posterior solución de las quejas, informes o anónimos que sean presentados ante esta oficina.

En la secretaria de esta oficina se recepcionan las quejas, los informes o los anónimos que luego serán analizados para decidir si se inicia un proceso o se archiva la queja, en caso de archivarse se dará por terminado el proceso. En caso contrario se inicia el proceso realizando las respectivas citaciones, se procederá a realizar las pruebas que se consideren pertinentes y luego de su análisis se procederá a conferir un fallo que puede ser absolutorio o condenatorio. Se pueden presentar durante el proceso apelaciones a las decisiones que se vayan tomando y cuando se emita el comunicado final se dará por terminado el proceso.

2.3 GENERALIDADES DEL ENTORNO DE DESARROLLO

2.3.1 Aplicaciones orientadas a la Web

En los primeros días de la Web, los sitios Web consistían de páginas estáticas, permitiendo una interacción limitada con el usuario. Al comienzo de los años 90, estas limitaciones fueron superadas cuando los servidores Web fueron reemplazados para permitir comunicaciones a través del desarrollo de fragmentos de código que eran ejecutados del lado del servidor. A partir de entonces las aplicaciones dejaron de ser estáticas y solamente editadas por aquellos “gurúes” del HTML y se permitieron a usuarios normales interactuar con las aplicaciones por primera vez.

Este fue un paso fundamental para llegar a la Web que hoy en día conocemos. Sin la interacción no existiría el comercio electrónico, el Web-mail , Internet-banking, blogs, comunidades online.

2.3.1.1 La Web como Sistema de Información.[1]

La evolución de Internet como red de comunicación global y el surgimiento y desarrollo de la Web como servicio imprescindible para compartir información, creó un excelente espacio para la interacción del hombre con la información hipertextual, a la vez que sentó las bases para el desarrollo de una herramienta integradora de los servicios existentes en Internet. Los sitios Web, como expresión de sistemas de información, deben poseer los siguientes componentes:

•Usuarios.

•Mecanismos de entrada y salida de la información.

•Almacenes de datos, información y conocimiento.

•Mecanismos de recuperación de información.

Se pudiese definir sistema de información como el conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su posterior uso, generados para cubrir una necesidad (objetivo). Dichos elementos formarán parte de alguna de estas categorías:

- Personas
- Datos
- Actividades o técnicas de trabajo.
- Recursos materiales en general (típicamente recursos informáticos y de comunicación, aunque no tienen por qué ser de este tipo obligatoriamente).

Todos estos elementos interactúan entre sí para procesar los datos (incluyendo procesos manuales y automáticos) dando lugar a información más elaborada y distribuyéndola de la manera más adecuada posible en una determinada organización en función de sus objetivos.

[1] Rodríguez Perojo Keilyn, Ronda León Rodrigo. El Web como Sistema de Información. http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm

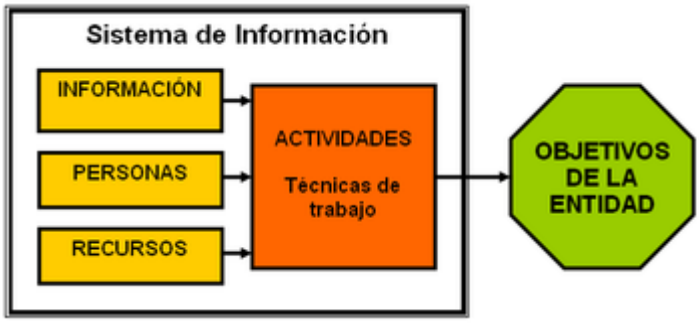

Figura 1.Elementos de un Sistema de Información[2]

Normalmente el término es usado de manera errónea como sinónimo de sistema de información informático, en parte porque en la mayoría de los casos los recursos materiales de un sistema de información están constituidos casi en su totalidad por sistemas informáticos, pero siendo estrictos, un sistema de información no tiene por qué disponer de dichos recursos (aunque en la práctica esto no suela ocurrir). Se podría decir entonces que los sistemas de información informáticos son una subclase o un subconjunto de los sistemas de información en general[2].

Actualmente, los sistemas de información se encuentran al alcance de las grandes masas de usuarios por medio de Internet; así se crean las bases de un nuevo modelo, en el que los usuarios interactúan directamente con los sistemas de información para satisfacer sus necesidades de información.

2.3.1.2. Tecnologías Web.[3]

Inicialmente, era difícil la construcción de aplicaciones sofisticadas. La primera generación de aplicaciones Web era primitiva, en general basada en formularios

[2] Tomado de: http://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n
[3] López Gustavo, Blanco Ignacio. Tesis en Ingeniería Informática. http://materias.fi.uba.ar/7500/blanco-tesisingenieriainformatica.pdf

con información y aplicaciones de búsqueda. Incluso estas aplicaciones básicas requerían de un alto seniority para su construcción.

A través del tiempo, el conocimiento necesario para construir aplicaciones ha sido reducido. Hoy en día, es relativamente sencillo construir aplicaciones sofisticadas utilizando las modernas plataformas y lenguajes, como pueden ser PHP, .NET o Java.

Primera generación – CGI

Common Gateway Interface (CGI) fue la tecnología reinante desde aproximadamente 1993 hasta fines de los '90 cuando los lenguajes de scripting comenzaron a ganar importancia.

CGI trabaja encapsulando la información provista por el usuario en variables de ambiente. Estas luego son accedidas por scripts o programas desarrollados comúnmente en Perl o C. Estos programas procesan la información provista por los usuarios, y luego envían código HTML con la información procesada a la salida estándar, que a su vez es capturada por el servidor Web y pasada al usuario.

Scripting.

La falta de manejos de sesiones y control de autorización por parte de CGI impidió el desarrollo de aplicaciones Web comerciales con esa tecnología.

Los desarrolladores Web comenzaron entonces a utilizar lenguajes de script, como JavaScript o PHP para resolver esos problemas. Básicamente los lenguajes de script son ejecutados en el servidor Web y como son no compilados son desarrollados e implementados más fácilmente.

Los lenguajes de script tienen algunas desventajas:

- La mayoría de los lenguajes no son tipados y no promueven buenas prácticas de programación.
- Son más lentos en comparación con los lenguajes compilados (a veces hasta 100 veces más lentos).
- Es difícil (no imposible) escribir aplicaciones de múltiples capas porque en general las capas de presentación, aplicación y datos residen en la misma máquina, limitando de esta forma la escalabilidad y seguridad.
- La mayoría no soporta nativamente métodos remotos o llamadas a Web services, lo que hace difícil la comunicación entre servidores de aplicación y con Web services externos.

De cualquier manera a pesar de las desventajas aplicaciones grandes y frecuentemente accedidas han sido desarrolladas utilizando lenguajes de script, como eGroupWare (egroupware.org), que está escrita en PHP. Además muchas aplicaciones de Internet banking han sido desarrolladas en ASP.

Los lenguajes de script incluyen, ASP, Perl, Cold Fusion y PHP. De cualquier manera, muchos de esos podrían ser considerados como lenguajes interpretados híbridos, en particular las últimas versiones de PHP y Cold Fusion.

2.3.1.3 Plataformas de desarrollo web.

Una vez que los lenguajes de script alcanzaron los límites de performance y escalabilidad, los proveedores más grandes evolucionaron hacia la plataforma de Sun J2EE y cuya evolución se encuentra en JEE6 y la de Microsoft .NET.

J2EE

- Utiliza el lenguaje Java para producir aplicaciones Web.
- Permite la creación de grandes aplicaciones distribuidas.

- Provee un buen control de sesión y manejo de autorización.
- Permite la creación de aplicaciones de múltiples capas.

Una de las desventajas de J2EE es que posee una curva de aprendizaje importante, lo que provoca una difícil inserción de diseñadores Web y programadores en sus primeros pasos.

.NET

- Simplifica la creación de aplicaciones pequeñas a programadores que se están iniciando y a diseñadores gráficos.
- Permite la creación de grandes aplicaciones distribuidas.
- Provee un buen control de sesión y manejo de autorización.
- Permite a los programadores la utilización de su lenguaje de programación favorito, el que es compilado a código nativo.

2.3.1.4 Web 2.0.

El concepto original de la Web (en este contexto, llamada Web 1.0) eran páginas estáticas

HTML que no eran actualizadas frecuentemente. El éxito de las punto com dependía de webs más dinámicas (a veces llamadas Web 1.5) donde los CMS servían páginas HTML dinámicas creadas al vuelo desde una actualizada base de datos. En ambos sentidos, el conseguir hits (visitas) y la estética visual eran considerados como unos factores muy importantes.

Los propulsores de la aproximación a la Web 2.0 creen que el uso de la Web está orientado a la interacción y a redes sociales, que pueden servir contenido que explota los efectos de las redes con o sin crear webs interactivas y visuales. Es

decir, los sitios Web 2.0 actúan más como puntos de encuentro, o webs dependientes de usuarios, que como webs tradicionales.

2.3.2 Diseños con el estándar UML.[4]

2.3.2.1 Introducción a UML

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa: Booch, OMT y OOSE (Object-oriented software engineering). UML ha puesto fin a las llamadas “guerras de métodos” que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

[4] Ferré Grau Xavier, Sánchez S. María Isabel. Desarrollo Orientado a Objetos con UML. Facultad de Informática - UPM

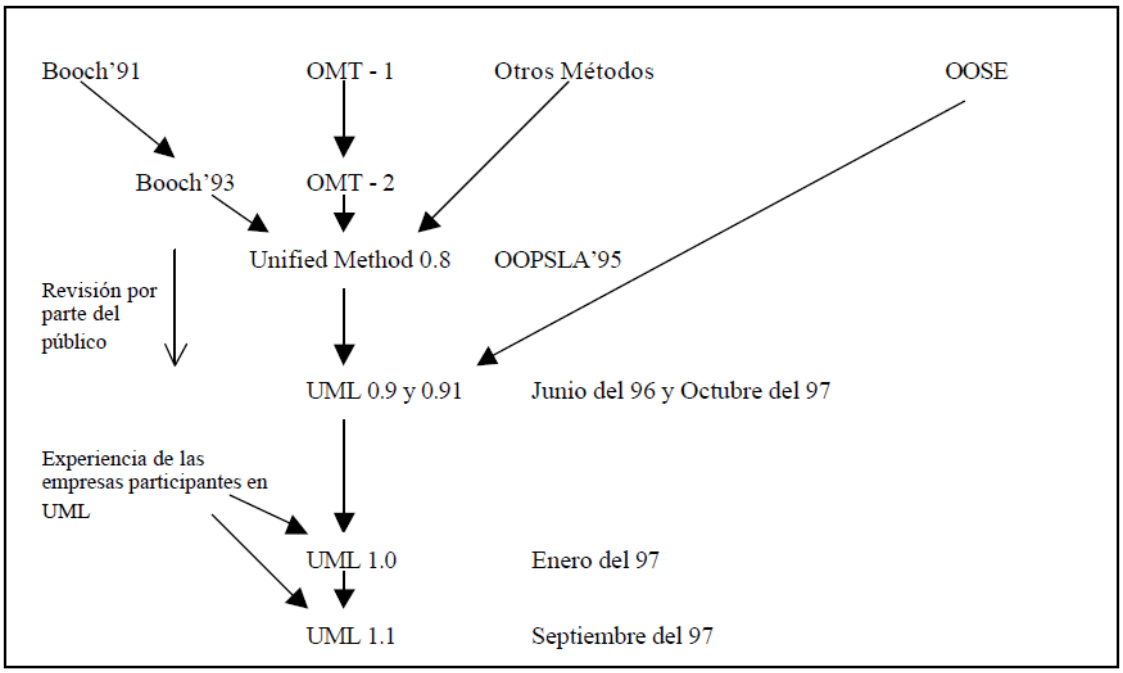

Figura 2. Historia de UML

2.3.2.2 Elementos comunes a todos los diagramas.

Nota

Una nota sirve para añadir cualquier tipo de comentario a un diagrama o a un elemento de un diagrama. Es un modo de indicar información en un formato libre, cuando la notación del diagrama en cuestión no nos permite expresar dicha información de manera adecuada.

Una nota se representa como un rectángulo con una esquina doblada con texto en su interior.

Puede aparecer en un diagrama unida a un elemento por medio de una línea discontinua. Puede contener restricciones, comentarios, el cuerpo de un procedimiento, etc.

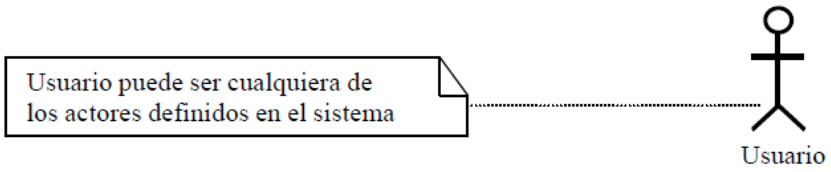

Figura 3. Ejemplo de una Nota UML.

Dependencias

La relación de dependencia entre dos elementos de un diagrama significa que un cambio en el elemento destino puede implicar un cambio en el elemento origen (por tanto, si cambia el elemento destino habría que revisar el elemento origen).

Una dependencia se representa por medio de una línea de trazo discontinuo entre los dos elementos con una flecha en su extremo. El elemento dependiente es el origen de la flecha y el elemento del que depende es el destino (junto a él está la flecha).

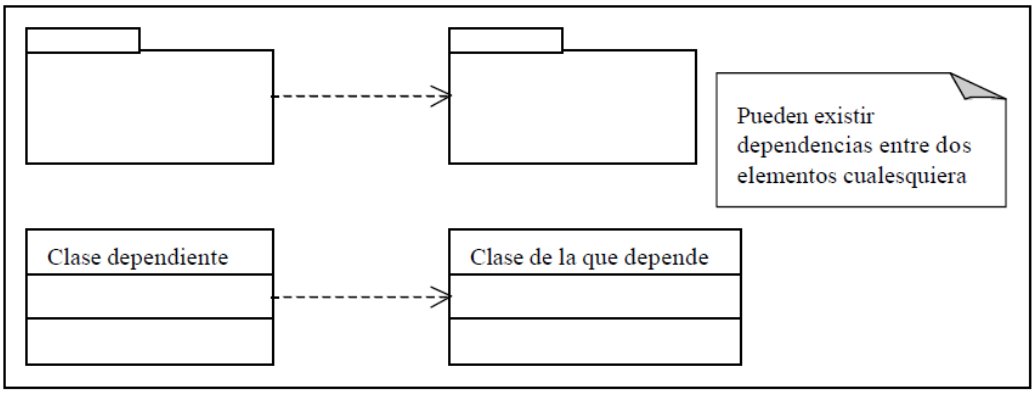

Figura 4. Ejemplo de una Dependencia UML.

2.3.2.3 Diagramas de Casos de Uso.

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Elementos del diagrama de casos de uso.

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: actores, casos de uso y relaciones entre casos de uso.

Actores

Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo.

Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.).

Casos de Uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Relaciones entre Casos de Uso

Entre dos casos de uso puede haber las siguientes relaciones:

· Extiende: Cuando un caso de uso especializa a otro extendiendo su funcionalidad. (extend)

· Uso: Cuando un caso de uso utiliza a otro. (include)

Se representan como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta <<extiende>> o <<uso>> según sea el tipo de relación.

En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

En la Figura se muestra un ejemplo de Diagrama de Casos de Uso para un cajero automático.

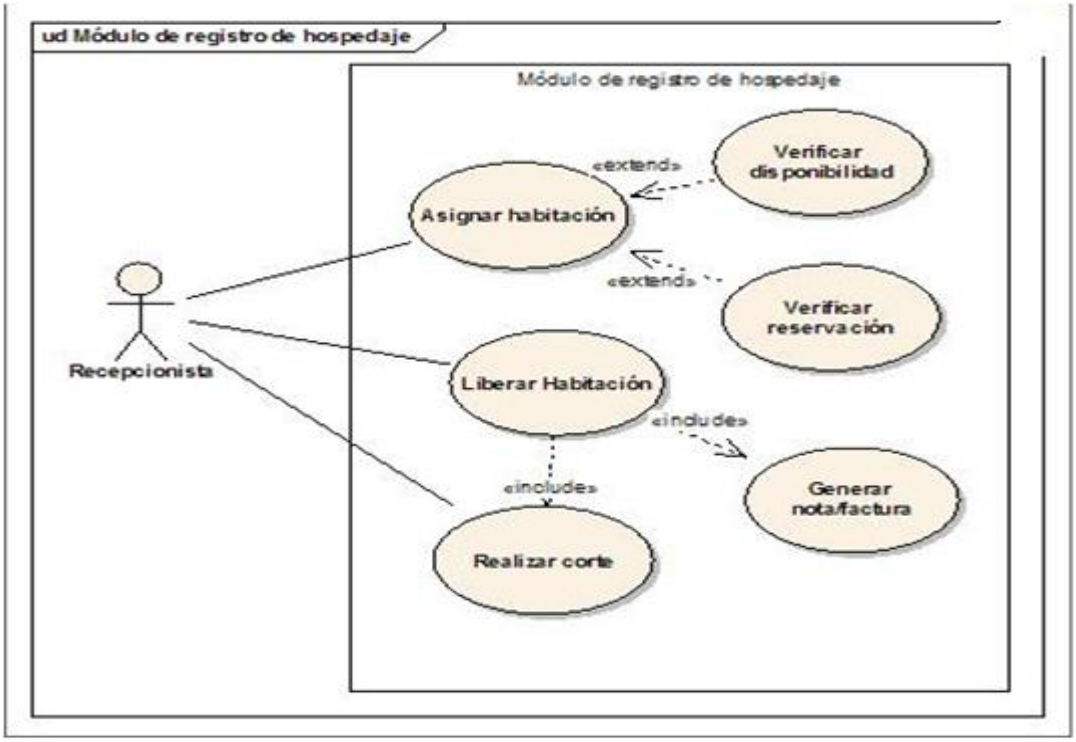

Figura 5. Ejemplo de Diagrama de Caso de Uso

2.3.2.4 Diagramas de Secuencia.

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo.

Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

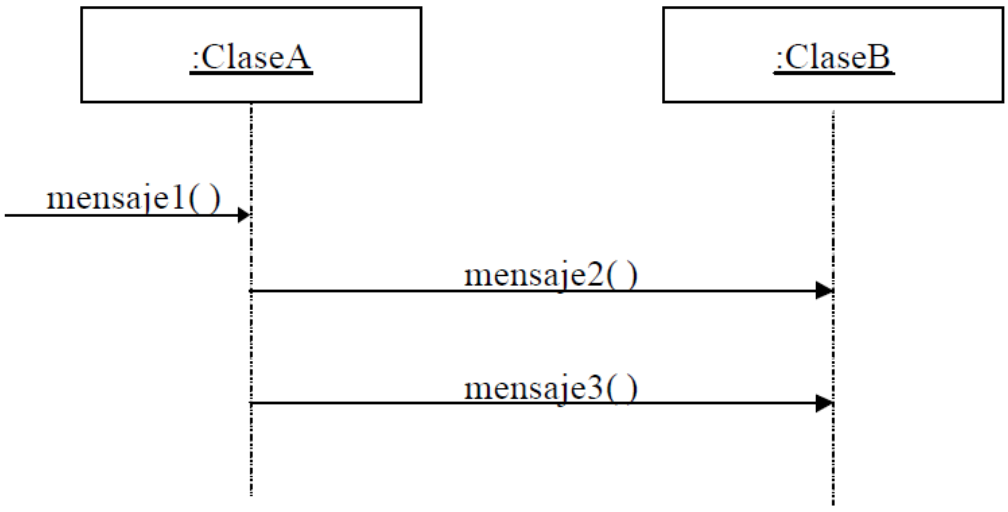

Figura 6. Ejemplo de Diagrama de Secuencia

2.3.2.5 Diagrama de Clases[5].

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Representación de: - Requerimientos en entidades y actuaciones. - La arquitectura conceptual de un dominio - Soluciones de diseño en una arquitectura - Componentes de software orientados a objetos, y dentro de sus características principales tenemos:

- Propiedades también llamados atributos o características, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso.
- Operaciones comúnmente llamados métodos, son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar.
- Interfaz es un conjunto de operaciones que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto. Hace referencia a polimorfismo.
- Herencia se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una persona puede especializarse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona,

[5] Fuente: http://es.wikipedia.org/wiki/Diagrama_de_clases

pero además cada uno tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, etc.

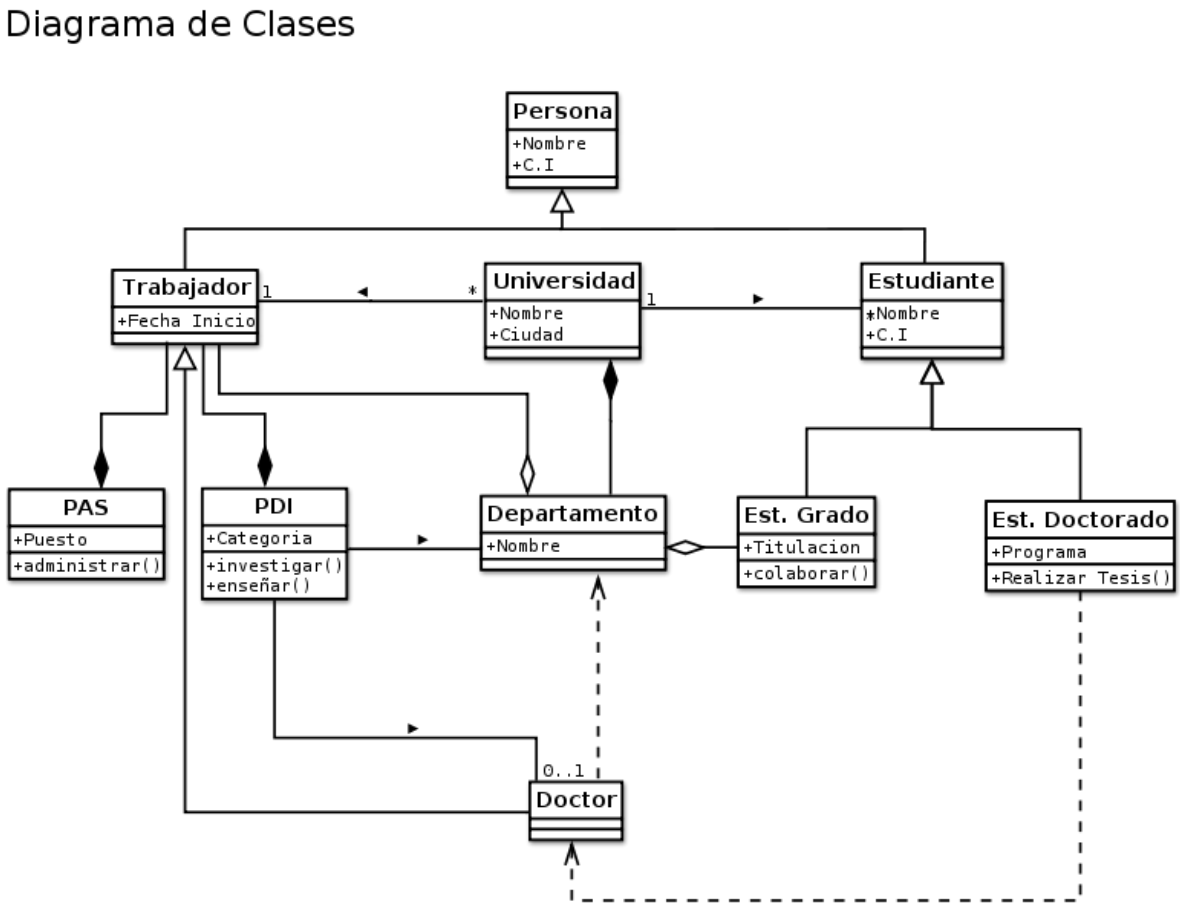

Figura 7. Ejemplo de Diagrama de Clases

Al diseñar una clase se debe pensar en cómo se puede identificar un objeto real, como una persona, un transporte, un documento o un paquete. Estos ejemplos de clases de objetos reales, es sobre lo que un sistema se diseña. Durante el proceso del diseño de las clases se toman las propiedades que identifican como único al objeto y otras propiedades adicionales como datos que corresponden al objeto.

2.3.2.6 Diagrama Entidad – Relación[6]

Un diagrama o modelo entidad-relación (a veces denominado por su siglas, E-R "Entity relationship", o, "DER" Diagrama de Entidad Relación) es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades.

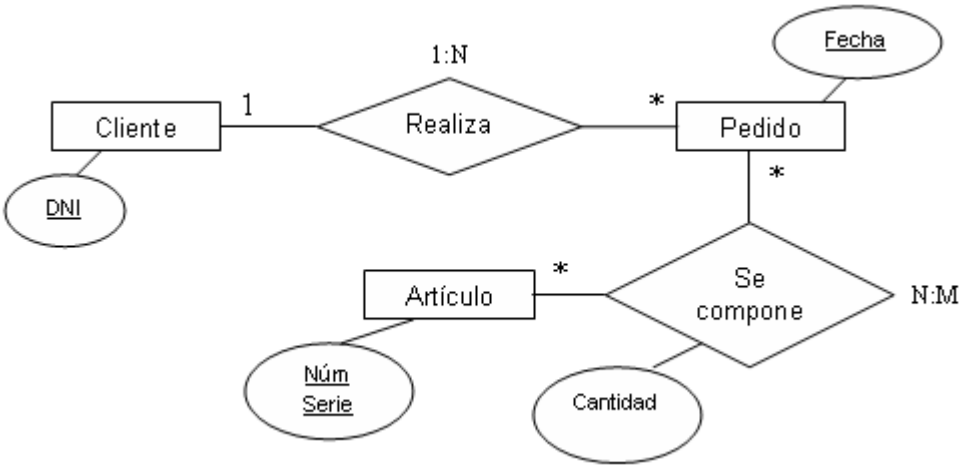

Figura 8. Ejemplo Diagrama Entidad-Relacion

El Modelo Entidad-Relación.

- Se elabora el diagrama (o diagramas) entidad-relación.
- Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

Dado lo complejo de esta técnica se necesita cierto entrenamiento y experiencia para lograr buenos modelos de datos.

[6] Fuente: http://es.wikipedia.org/wiki/Diagrama_entidad-relaci%C3%B3n

El modelado de datos no acaba con el uso de esta técnica. Son necesarias otras técnicas para lograr un modelo directamente implementable en una base de datos.

Brevemente:

- Transformación de relaciones múltiples en binarias.
- Normalización de una base de datos de relaciones (algunas relaciones pueden transformarse en atributos y viceversa).
- Conversión en tablas (en caso de utilizar una base de datos relacional).

2.3.3 Java Enterprise Edition 5.0 (JEE 5)

Introducción a Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

Java EE5

Java Enterprise Edition 5 (Java EE 5) se centra en hacer más fácil el desarrollo, sin embargo, conserva la riqueza de la plataforma J2EE 1.4. Ofrece funciones como Java Server Faces (JSF) y la tecnología de servicios web API, Java EE 5 hace que la codificación sea más simple y directa, pero mantiene el poder que ha establecido a Java EE como la primera plataforma para servicios web y desarrollo de aplicaciones empresariales.

El SDK de Java EE 5 y Java Application Platform SDK prestan apoyo a las especificaciones Java EE 5, y el SDK características adicionales, tales como tiempo de ejecución de Open ESB, Portlet Container, y Sun Java System Access Manager.

2.3.4 Entorno de Desarrollo

2.3.4.1 Java Server Faces (JSF)

Java Server Faces es una solución integral al problema de proveer una experiencia de usuario rica y a la vez sencilla en aplicaciones web. Para los desarrolladores de software, JSF provee una API estandarizada, fácil de usar y orientada a objetos, permitiendo crear interfaces de usuario con componentes reutilizables. Esto también repercute en la consistencia de tales interfaces, brindándole al usuario una experiencia de máxima calidad.

Su principal ventaja consiste en que el desarrollador sólo debe aprender el modelo de interfaces de usuario de JSF una sola vez, lo que le permitirá usar cualquier componente que cumpla los estándares de este modelo, aún si tales componentes provienen de terceros.

2.3.4.2 Object Relational Mapping / JPA

Más conocida por su sigla JPA, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares.

2.3.4.3 EJB 3.0

Enterprise JavaBeans (EJB) es una arquitectura de componentes para la construcción de aplicaciones empresariales ejecutadas en servidores. Tiene por propósito proveer una forma estándar de implementar este tipo de aplicaciones, haciéndose cargo de aspectos comunes y repetitivos como la persistencia, la integridad transaccional y la seguridad, permitiendo que el desarrollador pase a preocuparse exclusivamente por la lógica del negocio en sí.

JBoss AS fue de los primeros servidores de aplicaciones en adoptar las especificaciones de EJB 3.0. Este modelo de EJB simplifica el desarrollo eliminando la necesidad de una interfaz “Home” y descriptores de despliegue, reemplazándolos por anotaciones. Facilita la implementación de la persistencia por medio del api JPA.

2.3.5 Servidor de Aplicaciones – Jboss

El servidor de aplicaciones JBoss es una herramienta certificada para el desarrollo de aplicaciones empresariales Java. Su madurez y el esfuerzo de muchos desarrolladores, e incluso las sugerencias que han realizado muchos usuarios, han permitido que JBoss AS (Application Server) se popularice

ampliamente y sea común en los currículos de muchos desarrolladores. Encuestas recientes muestran que es el servidor de aplicaciones más popular actualmente.

Es reconocido por soportar los estándares más recientes. De hecho, es el primer servidor de aplicaciones en alcanzar la certificación J2EE 1.4 cuando salió su versión 4.0 (actualmente va en la versión 5, liberada a finales de 2008). Pero JBoss no sólo marca la pauta en la adopción de estándares con su servidor de aplicaciones, sino en la imposición de los mismos. Recientemente se le eligió para hacer parte del Java Community Process (JCP). Además en los últimos años ha estado a la cabeza del desarrollo de Java Enterprise llegando a establecerse en todas las especificaciones de requerimientos de Java (Java Specification Requests, JSRs).

2.3.6 Enterprise Architect

Enterprise Architect es una herramienta desarrollada por Sparx Systems que ofrece la capacidad de realizar el modelado de un proyecto y apoyar el desarrollo del mismo durante todo su ciclo de vida. Enterprise Architect logra esto usando UML (Unified Modeling Language).

Su valor como herramienta radica en la capacidad de permitir a los ingenieros desarrolladores comunicar sus ideas y su visión sobre los proyectos facilitando la administración y la redistribución de esta información.

2.3.7 Construcción de Prototipos de Software.

Un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y, enterarse más acerca

del problema y sus posibles soluciones. Un prototipo de software apoya a dos actividades del proceso de ingeniería de requerimientos:

- Obtención de requerimientos: les permite adquirir nuevas ideas para los requerimientos y encontrar áreas fuertes y débiles del software.
- Validación de requerimientos. El prototipo puede revelar errores y omisiones en los requerimientos. La construcción de prototipos puede utilizarse como un análisis de riesgo y una técnica de reducción. Un modelo iterativo del proceso, como el desarrollo incremental, se utiliza junto con un lenguaje diseñado para el desarrollo rápido de aplicaciones. Por lo tanto, las técnicas utilizadas para desarrollar un prototipo para validar los requerimientos también se utilizan para desarrollar el sistema de software mismo.

Ventajas:

- Al demostrar las funciones del sistema se identifican las discrepancias entre los desarrolladores del software y los usuarios.
- Durante el desarrollo del prototipo el personal del desarrollo de software puede darse cuenta de que los requerimientos son inconsistentes y/o están incompletos.
- Se dispone rápidamente de un sistema que funciona y demuestra la factibilidad y usabilidad de la aplicación a administrar.

En sistemas grandes y complejos una forma de resolver la dificultad de evaluación es utilizar un enfoque evolutivo para el desarrollo de sistemas. Esto significa proporcionar al usuario un sistema incompleto y después modificarlo y aumentarlo en el momento en que los requerimientos del usuario sean claros.

El enfoque de construcción de prototipos desechables es para ayudar a refinar y clasificar la especificación del sistema. El prototipo se escribe, evalúa y modifica. La evaluación del prototipo informa del desarrollo de la especificación detallada del

sistema que se incluye en el documento de requerimientos de este. Una vez que se ha redactado la especificación, el prototipo ya no es útil y se desecha.

Prototipo No funcional [7]

Es un modelo no funcional a escala configurado para probar ciertos aspectos de diseño. Un ejemplo de este enfoque es un modelo a escala completa de un automóvil que se usa para pruebas en un túnel de viento. El tamaño y forma del automóvil son precisos, pero el automóvil no es funcional. En este caso solo se incluyen las características del automóvil que son fundamentales para la prueba en el túnel de viento.

Un modelo no funcional a escala de un sistema de información podría producirse cuando la codificación requerida por las aplicaciones es demasiado extensa para incluirse en el prototipo, pero se puede conseguir una idea útil del sistema a través de la elaboración de un prototipo de la entrada y la salida. En este caso, el procesamiento, debido al excesivo costo y el tiempo requerido, no podría incluirse en el prototipo. Sin embargo, aún se podrían tomar algunas decisiones sobre la utilidad del sistema con base en la entrada y la salida incluidas en el prototipo.

[7] Fuente: Kendall, Julie E. Analisis y Diseño de sistemas. Capitulo 6. Página 153

3. METODOLOGÍA DE DESARROLLO

3.1 CICLO DE VIDA DEL PROYECTO.

A continuación se hace una descripción de las diferentes actividades que se llevaron a cabo durante el transcurso del proyecto, buscando conceptualizar la metodología que se aplicó en el desarrollo del nuevo sistema de información de la oficina de Control Interno Disciplinario de la Universidad Industrial de Santander.

3.1.1 Análisis de Requerimientos:

En el análisis de requerimientos se especificó, junto con los funcionarios de Control Interno Disciplinario, la función y comportamiento que debería tener el sistema de información que se desarrolló en el proyecto, se indicó la interfaz con otros elementos del sistema, se estableció la integración con el sistema de información institucional y se fijaron los estándares de diseño que cumple el sistema.

El análisis de requerimientos permitió la representación de la información y las funciones que fueron traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministró los medios para valorar la calidad de los programas.

En esta etapa se hicieron reuniones periódicas con los funcionarios de control interno disciplinario, y fueron ellos los que definieron las características y alcances del software que se desarrolló y que se adapta plenamente a las exigencias de los procesos de control interno disciplinario.

3.1.2 Diseño

El diseño del software fue realmente un proceso de muchos pasos que se centró en cuatro atributos distintos: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo).

En esta etapa de diseño se hizo una traducción de los requisitos a una representación del software donde se pudo evaluar su calidad con la codificación.

El diseño se efectuó, mediante modelos UML (Lenguaje de Modelado Unificado) que incluyó los diagramas que han sido seleccionados dentro de los estándares de desarrollo de software utilizados en la División de Servicios de Información, que son: de casos de uso, de clases y de secuencia, utilizando la herramienta de modelaje Enterprise Architect.

3.1.3 Implementación de la Aplicación.

El diseño se tradujo en una forma legible por la máquina. El paso de generación de código se llevó a cabo en esta tarea.

En esta etapa se procedió a generar el software que se había diseñado. Se realizó teniendo en cuenta los parámetros establecidos por la División de Servicios de Información en cuanto a los estándares técnicos y de calidad que caracterizan las aplicaciones que son generadas para el servicio de la Universidad, teniendo como base el Lenguaje de programación JAVA 5, frameworks como: seam, java server faces (JSF), Enterprise Java Beans (EJB 3.0) e Informix como motor de base de datos.

3.1.4 Pruebas del software.

Corresponden a los procesos que permiten verificar la calidad de un producto software y el cumplimiento de los requerimientos establecidos en la fase de análisis de requerimientos.

Las pruebas de software se integran dentro de las diferentes fases del ciclo de desarrollo del software establecidas en la ingeniería de software.

Una vez generado el código, comienzan las pruebas, proceso utilizado para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Las pruebas se centran en los procesos lógicos internos del software, asegurando que todas las sentencias sean probadas y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

Estas pruebas se aplicaron de forma permanente a lo largo del desarrollo del proyecto por parte del equipo de trabajo, y se abrió un espacio dónde los usuarios finales interactuaron con la aplicación con el objetivo que detectaran posibles fallos que hayan sido omitidos en el momento del desarrollo.

3.1.5 Ajustes.

Después de las Pruebas, cada uno de los errores detectados o las observaciones hechas por los usuarios, debidamente analizadas, deben ser tenidos en cuenta para ajustar el sistema, de tal manera que se adapte plenamente a los requerimientos establecidos.

También estos se hicieron permanentemente a lo largo del desarrollo del proyecto, a la par con la realización de las pruebas, en la medida en que se detectaron errores o inconsistencias en el software que se estaba desarrollado.

Este proceso está incluido dentro del proceso iterativo de refinamiento que se le dio al sistema, para adaptarse plenamente a las necesidades del cliente que en el presente caso es la Oficina de control interno disciplinario.

3.2 METODOLOGIA DE DESARROLLO DEL PROYECTO.

3.2.1 Modelo de Construcción de Prototipos.

Teniendo como base las actividades anteriormente descritas, se dispuso como metodología de desarrollo el MODELO DE CONSTRUCCION DE PROTOTIPOS.

Se eligió esta metodología debido a que es muy frecuente que los clientes/usuarios que están solicitando el sistema, en este caso la oficina de Control Interno Disciplinario, definan un conjunto de objetivos generales para el software, pero no identifican los requisitos detallados de entrada, proceso o salida.

En otros casos, el responsable del desarrollo del software puede no estar seguro de la eficacia de un algoritmo, o no haber comprendido plenamente el requerimiento del usuario.

Para éstas y otras muchas situaciones, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque, ya que la entrega de prototipos, que hacen parte integral del proyecto en su conjunto, permitirán la corrección temprana de errores o la redefinición del sistema en caso de ser necesario, y los prototipos tanto funcionales como no funcionales permitirán la familiarización del usuario con el sistema que se está desarrollando.

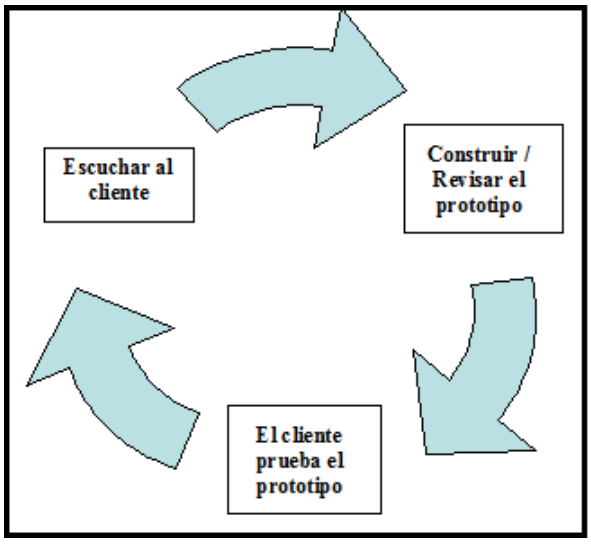

Figura 9. Modelo de Construccion de Prototipos.

3.2.2 Estructura del modelo de construcción de prototipos.

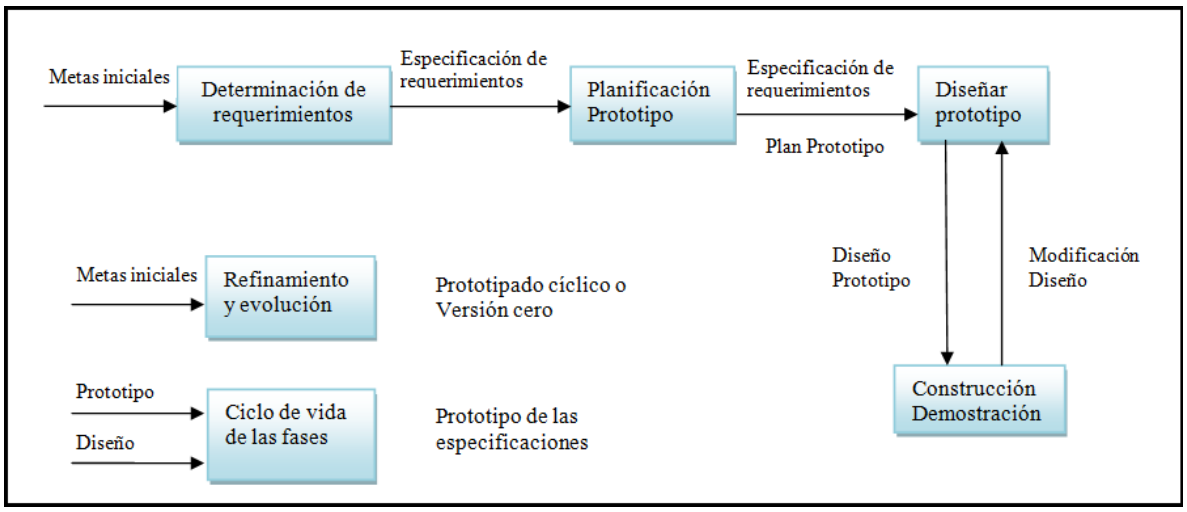

Figura 10. Estructura del Modelo de Construcción de Prototipos

Ésta metodología planteada, permitió que el proyecto se llevara a cabo de manera eficiente, debido a:

- En la creación de los prototipos iniciales se trabajó con unas ideas aproximadas de lo que pretendía alcanzar la oficina de control interno disciplinario, estas ideas dieron resultado a un producto o prototipo inicial, el cual evolucionó dando como resultado un prototipo más maduro, que cumple con todos los requerimientos del cliente.
- Con el uso del modelo de prototipos se dio la facilidad de mejorar, de manera temprana los prototipos, teniendo en cuenta las sugerencias del usuario solicitante del proyecto, de manera que se pudieran cubrir a cabalidad las necesidades por las cuales se desarrolló el proyecto.
- Es importante conocer de forma temprana si los diferentes prototipos de los servicios, cumplen a satisfacción con los requerimientos expuestos por los funcionarios de la oficina de control interno disciplinario.
- En este sistema de desarrollo no se libera un prototipo sin haber realizado todas las pruebas necesarias del mismo, pero los inconvenientes resultan inevitables y se da la facilidad de detectarlos y corregirlos de manera oportuna.

3.2.3 Procedimiento para la metodología planteada.

- La construcción de prototipos comienza con la recolección de los requisitos. La oficina de control interno disciplinario de la Universidad Industrial de Santander (quienes son los usuarios administradores del sistema) cuentan de manera verbal lo que desean alcanzar con esta nueva versión y los problemas que inevitablemente deben ser subsanados.
- El desarrollador y usuario se reúnen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesaria una mayor definición.
- Teniendo la abstracción del sistema por parte del desarrollador, se realiza todo el diseño de los objetos mediante los diagramas UML: Casos de uso, de clases, de secuencia y entidad – relación.

- Luego se produce el Diseño del Prototipo que se enfoca sobre la representación de los aspectos del software visibles al usuario (por ejemplo, métodos de entrada esquemas de navegación y formatos de salida), en ésta etapa se presenta al usuario el diseño tanto de los diagramas UML como el prototipo NO funcional, para que éste lo apruebe y se adecúe plenamente a su solicitud.
- Se procede con la construcción del prototipo que se ha propuesto.
- El prototipo es evaluado por el usuario y se utiliza para refinar los requisitos de tal forma que el usuario identifique aspectos que deban ser replanteados o mejorados.
- Se produce un proceso iterativo en el que el prototipo es “afinado” (Refinamiento del prototipo) para que satisfaga las necesidades del usuario, al mismo tiempo que facilita al que lo desarrolla una mejor comprensión de lo que hay que hacer y poder entregar el producto final requerido.

PARTE II. DESARROLLO DEL SISTEMA.

4. APLICACIÓN DE LA METODOLOGÍA

4.1 LEVANTAMIENTO DE REQUERIMIENTOS

Se presentan los requerimientos, generados después de las primeras reuniones con el cliente, en donde se pudo abstraer la idea principal de lo que se alcanzó durante el desarrollo del proyecto, que sirvió como base para las primeras etapas del análisis y diseño del sistema de información de la oficina de Control Interno Disciplinario.

4.1.1 Descripción General.

Este sistema permite la recepción de las Quejas, Informes o Anónimos por parte de ciudadanos particulares, estudiantes o funcionarios y permite su gestión y tratamiento por parte de la Dirección de Control Interno Disciplinario. Es importante aclarar, que la recepción de las Quejas, Informes o Anónimos se centrará en la Secretaria de la Dirección de Control Interno Disciplinario, quienes se encargarán de gestionarla, procesarla y dar respuesta a los mismos.

Los requisitos de este sistema se describen a continuación.

4.1.2 Funciones que debe cumplir el sistema.

El módulo brinda soporte a la gestión de la OCID, para dar respuesta oportuna a los requerimientos de los Quejosos:

4.1.2.1 Del registro de la queja, informe o anónimo.

Diligenciamiento del registro de la Queja.

Cuando la secretaria ingresa a esta sección debe preguntar el tipo de vinculación de la persona con la universidad y luego proceder a buscarla mediante el nombre o número de documento en la base de datos correspondiente.

En caso de no encontrarla deberá llenar el siguiente formulario:

Tipo Vinculacion* : menú desplegable con opciones para elegir: empleado UIS, estudiante UIS, particular.

Tipo Documento* : menú desplegable con opciones para elegir: cedula de ciudadanía, cedula extranjería, entre otros.

Identificación* : campo numérico, donde se especificará el número de documento de identidad del quejoso se debe aclarar que este campo es necesario para que pueda realizar seguimiento a la queja.

Primer Nombre* : campo para datos alfabéticos, donde se especificará el primer nombre del quejoso.

Segundo Nombre : campo para datos alfabéticos, donde se especificará el segundo nombre del quejoso.

Primer Apellido* : campo para datos alfabéticos, donde se especificará el primer apellido del quejoso.

Segundo Apellido : campo para datos alfabéticos, donde se especificará el segundo apellido del quejoso.

Correo electrónico : campo que solo admitirá formatos de correo electrónico, donde se especificará el correo electrónico del solicitante.

Dirección* : campo alfanumérico, donde se especificará la dirección de contacto con el solicitante.

Motivo de la Queja*: menú desplegable con opciones para elegir los motivos de la queja éstos varían de acuerdo a los procesos que han sido presentados con anterioridad, pueden ser agregados o modificados por el Administrador, cuando lo estimen conveniente.

Descripción* : campo para redacción de la queja.

Ayuda en línea : El sistema debe permitir la ayuda en línea para explicar al Solicitante el funcionamiento del módulo. La redacción y contenido está a cargo de OCID.

2. Luego de diligenciar el Formulario de la Queja y registrarla, se debe proceder a hacer el registro del investigado en el que se muestre el número de radicado de la solicitud (consecutivo), que durante todo el proceso no puede cambiar.

(*) Campos obligatorios

4.1.2.2 Administración o Mantenimiento al Sistema.

Al módulo de administración tiene acceso solo el director el cual tendrá la posibilidad de hacerle mantenimiento a las tablas soporte, en dónde se puede redefinir parámetros básicos para el funcionamiento del sistema, tales como:

- Tipos de Quejas.
- Tipos de Vinculación.
- Tipos de Notificaciones.
- Motivos de la Queja.
- Estado de las Quejas.
- Estado de las Pruebas.
- Tabla parámetros.
- Entidades Externas.
- Actividades.
- Formatos.

4.1.2.3 Documentos Secretariales

Al modulo de documentos secretariales tienen acceso solo los usuarios registrados y autorizados por la OCID en general serán la secretaria, la profesional adscrita a la oficina y el director CID, los cuales podrán:

- Realizar Citaciones a declaraciones, descargos y pruebas.
- Realizar Notificaciones sobre las Citaciones que han realizado dependiendo de los tiempos estipulados por la legislación colombiana.
- Realizar Constancias que permitan tener soporte de los documentos entregados y recibidos.
- Realizar Comunicaciones a los diferentes actores involucrados en el proceso investigativo como lo son quejosos investigados, o a entidades como la

procuraduría u otras a nivel externo que lo requieran o unidades UIS cuando hubiere lugar.

4.1.2.4 Gestión de las Quejas, Informes o Anónimos o Proceso Investigativo.

1. Listado de Quejas, Informes o Anónimos: Aparecen 3 para seleccionar si es una queja, un informe o un anónimo, organizados en una tabla que consta de las siguientes columnas:

Numero de Radicado, Fecha de Radicado, Estado de la Queja, Informe o Anónimo respectivamente.

Los estados de la Queja serán definidos así:

- Recibida: cuando la queja, el informe o el anónimo no han sido analizados por OCID pero están registrados en el sistema.
- Análisis: cuando la queja empieza su curso normal de un proceso investigativo normal.
- En Pruebas: cuando se dio por iniciado un proceso formal y se empiezan a realizar las pruebas pertinentes del caso.
- Archivada: cuando no se considero pertinente iniciar un proceso debido a falta de pruebas contundentes o no se considero que el motivo fuera procedente como tal.
- Finalizada: cuando se ha dictado un fallo sancionatorio condenatorio o absolutorio de la queja, informe o anónimo recibido.

Si esta en el estado Recibida solo tiene como opción permitida el Análisis.

Si esta en el estado Análisis se puede realizar los autos de apertura de indagaciones o investigaciones según sea el caso, se puede archivar el proceso o de ser muy grave dictar auto de cargos.

4.1.2.5 Consultas

Al modulo de consultas tienen acceso las vicerrectorias académica y administrativa, la rectoría, secretaria general, el personal de la OCID en el cual podrán ver todos los detalles de las quejas, de los investigados, de los quejosos en qué estado del proceso se encuentran, que documentos como citaciones, comunicaciones, solicitudes, notificaciones o constancias se han realizado pero no podrá modificar ningún detalle de los mismos.

4.1.2.6 Estadísticas

Debe permitir al administrador seleccionar entre los tipos de solicitud, y poder visualizar, entre otros, los siguientes datos estadísticos:

- Por UAA.
- Por rangos de fechas.

4.2 ESTÁNDARES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN.[8]

4.2.1 Aspectos Generales

Interfaz de desarrollo

El IDE de desarrollo a utilizar es el JBoss Developer Studio, el cual debe ser instalado en la carpeta por defecto del instalador.

[8] Fuente: Estandares de la División de Servicios de Información de la Universidad Industrial de Santander.

Este y todos los programas necesarios se pueden descargar, por el personal autorizado, del equipo establecido para tal fin.

Servidor de aplicaciones

En cuanto al servidor de aplicaciones de desarrollo se debe utilizar el mismo que se encuentra en los servidores de producción y desarrollo, el cual debe ser instalado en la carpeta C:\jboss-5.0.0.GA.

JAVA

La versión del compilador de JAVA debe ser la 1.5, la cual debe ser instalada en el directorio C:\java1.5.

SEAM

La versión del seam a utilizar es la 2.1.2.GA. (jboss-seam-2.1.2.GA.zip).

Espacio de trabajo

El espacio de trabajo se debe crear en C:\workspace.

El nombre del proyecto para los JPA debe estar conformado de la siguiente manera:

[Sistema]Entidades

Por ejemplo: AcademicoEntidades (La primera letra de cada palabra en mayúscula).

El repositorio para los JPA debe estar conformado de la siguiente manera:

[Sistema]JPA

Por ejemplo: AcademicoJPA (La primera letra de cada palabra en mayúscula).

El Server name en el IDE de desarrollo se debe llamar JBoss 5 y el nombre del JBoss Runtime Enviroment se debe llamar JBoss 5.

Servidor de versiones

Para su configuración se debe instalar en el JBoss Developer Studio los siguientes paquetes:

- subclipse-site-1.4.7
- ajdt_1.6.1a_for_eclipse_3.4
- org.tmatesoft.svn_1.2.1.eclipse

Una vez instalados se debe solicitar el nombre de usuario y la contraseña al Ingeniero encargado.

Cualquier inquietud acerca de la instalación y configuración del servidor de versiones, dirigirse con el Ingeniero encargado.

También se debe instalar el siguiente programa: TortoiseSVN-1.5.8.15348-win32-svn-1.5.5.msi para conectarse con el servidor de versiones de la documentación.

Plantillas, estilos, imágenes y formateador

Descargar del servidor de versiones de documentación las carpetas Estilos, Plantillas e Imágenes y copiarlas en la careta view de la aplicación.

Patrones a utilizar

Los patrones a utilizar corresponden a cada una de las capas implementadas:

- Capa de presentación: Modelo Vista Controlador, el cual es implementado por Java Server Faces (JSF).
- Capa de lógica de negocio: Session Façade
- Capa de persistencia: Entity Access Object (EAO)

Rich Faces

La versión a utilizar es la incluida en el jboss-seam-2.1.2.

Código HTML

Está prohibido el uso de etiquetas HTML en las páginas.

Anotaciones en la entidad

Las anotaciones en la entidad se deben colocar antes del método get correspondiente y no en la declaración del atributo.

Llaves compuestas

Se debe utilizar la anotación EmbeddedId, la cual obliga a declara un objeto del tipo de la llave dentro del EJB de entidad.

HashCode e Equals en EJB de entidad

Se debe utilizar únicamente los campos que conforman la llave primaria. En el caso de las llaves compuestas, el atributo que hace referencia a ésta.

JPA externos

Para utilizar los JPA externos (academico.jar, recursos-humanos.jar, etc), éstos deben copiarse en la carpeta raíz. En el caso de Windows C:\, para Linux \).

En cada uno de los archivos persistence.xml de su aplicación, se debe utilizar <jar-file>file:/jpa.jar</jar-file>.

Por ejemplo:

<jar-file>file:/academico.jar</jar-file>

<jar-file>file:/recursos-humanos.jar</jar-file>

<jar-file>file:/general-UIS.jar</jar-file>

Servicios

Para crear un servicio se debe tener en cuenta las siguientes reglas:

- La interfaz debe contener la anotación Remote
- La implementación de la interfaz debe contener la anotación RemoteBinding con su respectivo nombre jndi (igual al utilizado por la anotación Name). Por ejemplo:

@RemoteBinding(jndiBinding = "ConsultarGenerales")

Para utilizar el servicio se debe utilizar la anotación EJB indicando el nombre jndi. De acuerdo al ejemplo anterior sería:

@EJB(mappedName = "ConsultarGenerales")

4.2.2 Documentación de los Diagramas de Diseño

Casos de Uso.

Para el desarrollo del modelo de casos de uso, se debe realizar un diagrama de casos de usos por módulo del sistema a implementar siguiendo el estándar propuesto por el Lenguaje Unificado de Modelado 2.1 (UML). Se deben tener en cuenta los siguientes los siguientes puntos:

Identificación de Actores

Se identifican con el rol que desempeñan en el sistema.

Diagrama de Casos de Uso

El diagrama de casos de uso se tiene que realizar con la herramienta Enterprise Architect. Cada caso de uso constituye un flujo completo de eventos especificando la interacción que toma lugar entre el actor y el sistema.

Casos de Uso

Se deben identificar con una acción. Los verbos que se pueden utilizar se encuentran al final del documento.

La información mínima requerida por cada caso de uso es la siguiente:

- Descripción completa
- Precondiciones y postcondiciones
- Descripción del escenario básico y alternos
- Diagrama de clases
- Si el caso de uso es complejo se debe incluir:
- Diagrama de secuencia y/o diagrama de actividades.

4.2.3 Sintaxis de Nombres en Java

Reglas de sintaxis generales

- En esta sección se especifican las reglas de sintaxis generales para todos los identificadores (nombres de variables, clases, métodos, etc.)
- Todos los nombres de los identificadores deben estar en español.
- Siempre se deben utilizar nombres que sean claros, concretos y libres de ambigüedades. Usando palabras completas evitando acrónimos y abreviaturas.
- Los nombres deben estar definidos sin espacios en blanco, sin guiones (_ , -), ni comillas (“,’), sin operadores (+, - , / , *), sin tildes, utilizar la n en vez de la ñ y sin caracteres especiales.
- No se debe utilizar la mayúscula para diferenciar entre identificadores distintos. Ejemplo: contador y Contador.

- No se deben diferenciar dos identificadores solo con numerales en cualquier posición. Ejemplo: contador1, contador2, 1contador, 2contador.
- Las siguientes partículas están prohibidas en la declaración de los nombres de identificadores: artículos (el, la, los, unos, unas, un), determinantes demostrativos (este, ese, aquel, aquellos), cardinales (uno, dos, etc.), pronombres de cualquier tipo (yo, tu, el, me, te, se, este, ese, mi, tu, su, etc.).
- Se deben utilizar máximo 5 palabras por nombre, las 3 primeras palabras van completas, a partir de la cuarta palabra se quitan las vocales a la palabra exceptuando la última vocal y la primera si la palabra empieza por vocal. No se utiliza ningún separador entre las palabras, se separa cada palabra utilizando su primera letra en mayúscula. Ejemplo: hacerMantenimientoConsultaUsrs, hacerMantenimientoAsignaturasCntxto.

Clases

- Los nombres de las clases deben iniciar siempre en mayúscula, deben ser simples y descriptivos.
- Los nombres de las clases de Entidad deben ser sustantivos en singular.
- Para las clases de Entidad siempre se debe definir la anotación @Table que indica la tabla de la base de datos relacionada para su persistencia. Además se debe utilizar el parámetro name de la anotación @Entity para identificar el EJB (@Entity(name = “xxx”)). El nombre de la clase puede ser distinto al nombre de la tabla y debe seguir el estándar de identificadores mencionado en el numeral anterior.
- Los nombres de los EJB utilizados por el patrón Session Façade está conformado por un verbo autorizado (Ver anexo de verbos). Además debe incluir al final las letras “EJB”. Ejemplo: RegistrarMatriculaEstudianteEJB.
- La interfaz asociada al EJB debe llevar el mismo nombre del EJB sin el sufijo “EJB”. Ejemplo: RegistrarMatriculaEstudiante.

- Para los EJB de entidad, cuando la clase representa una relación entre dos entidades, el nombre se forma uniendo los nombres de las entidades involucradas.
- Los nombres de las clases que representan excepciones terminaran siempre con el sufijo “EXCEPCION”.
- El nombre de la clase no contendrá detalles sobre la implementación interna de la misma. Por ejemplo ArrayEstudiantes no es nombre válido.

Métodos

- Se deben utilizar los verbos autorizados para su codificación.
- El nombre de los métodos debe iniciar con minúscula la primera palabra, las siguientes palabras inician en mayúscula, sin separadores.
- La primera palabra del nombre de los métodos debe ser un verbo en infinitivo y debe representar una acción o comportamiento de la clase.
- El nombre del método debe describir claramente el comportamiento del mismo.
- En lo posible no se deben usar verbos genéricos aplicables a todo como: procesar, gestionar, manejar. Ejemplo: procesarEstudiante(), gestionarCliente(), en este caso el verbo no aclara el cometido real del método.

Paquetes

- El nombre de los paquetes debe iniciar con minúscula la primera palabra, las siguientes palabras inician en mayúscula, sin separadores.
- La estructura de los paquetes es la siguiente:

  co.edu.uis.[sistema].[aplicación].[módulo].[caso de uso]

  Ejemplo:
  co.edu.uis.financiero.egresos.contratacion.ordenarPrestacionServicio.

- La estructura para el paquete donde van a estar las entidades comunes para todos los sistemas es el siguiente:

  co.edu.uis.sistema.entidades

- La estructura para el paquete donde van a estar los servicios comunes para todos los sistemas es el siguiente:

  co.edu.uis.sistema.servicios

Variables

- Para el nombre de las variables utilizar palabras completas en singular (máximo tres palabras). El nombre deber ir en plural cuando la variable representa una lista o un conjunto de elementos.
- Si las palabras no son suficientes para la descripción, se debe hacer un comentario, al frente de la variable.
- Las constantes (final) van en mayúscula sostenida separando las palabras por guión de piso (_).
- Para los argumentos, deben iniciar siempre con la letra “a” y posteriormente el nombre según lo establecido anteriormente.
- Para las instancias de las clases, las entidades llevan el mismo nombre de la clase, solo que la palabra inicial va en minúscula. Si se necesita más de una instancia, para diferenciarlas se debe adicionar una palabra que identifique el rol que desempeña. Ejemplo: Estudiante estudiantePregrado, Estudiante estudiantePostgrado.
- La variable del EntityManager se debe llamar em.
- La variable de tipo FacesMessages se debe llamar facesMessages.

Librerías (JAR)

El nombre de las librerías no sigue el mismo estándar de las clases. Éstas se deben escribir en minúscula, separando cada palabra con un guión (-). Ejemplo: recursos-humanos.jar.

Nombres de archivos

Se sigue el mismo estándar descrito en las reglas de sintaxis generales.

4.2.4 Documentación

La documentación relacionada con el diseño del sistema reside en la base de datos de Enterpirse Architect.

La documentación del código fuente se debe hacer en cada una de las clases, siguiendo el estándar de JAVADOC y documentando la definición de la clase, descripción de los métodos get y set y descripción de los parámetros de entrada y salida de cada uno de los métodos que componen la clase.

4.2.5 Capa de Presentación

Plantilla principal

La plantilla principal de una página es la siguiente:

Figura 11 - Plantilla Principal División de Servicios de Información

Contenido

Esta sección se refiere al caso de uso implementado. La estructura de esta sección es:

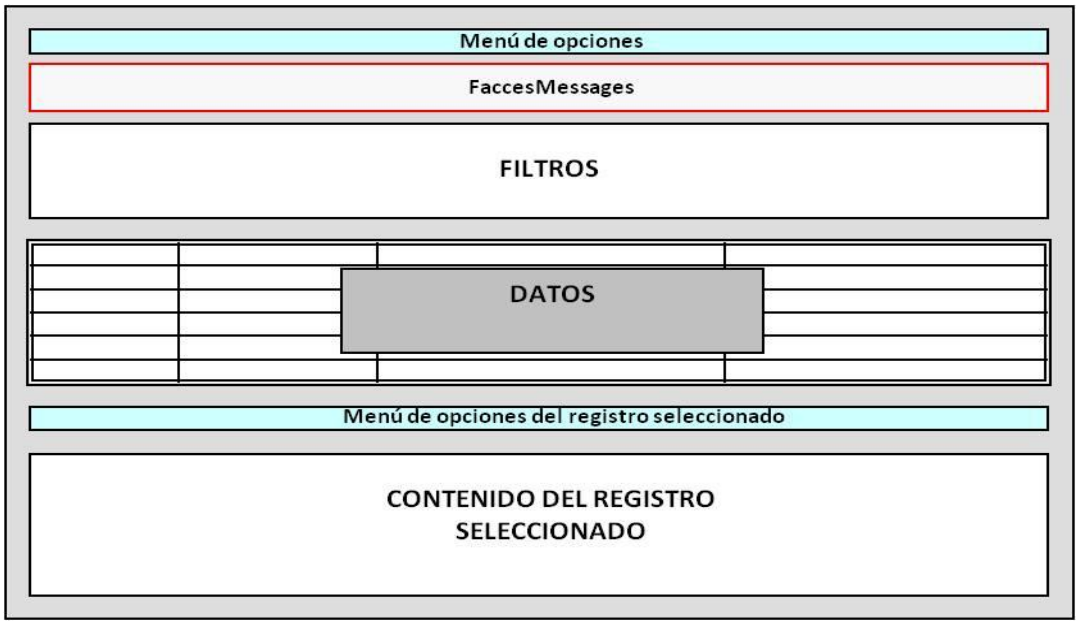

Figura 12. Plantilla de Contenido. Division de Servicios de Informacion

Para las páginas que muestran formularios:

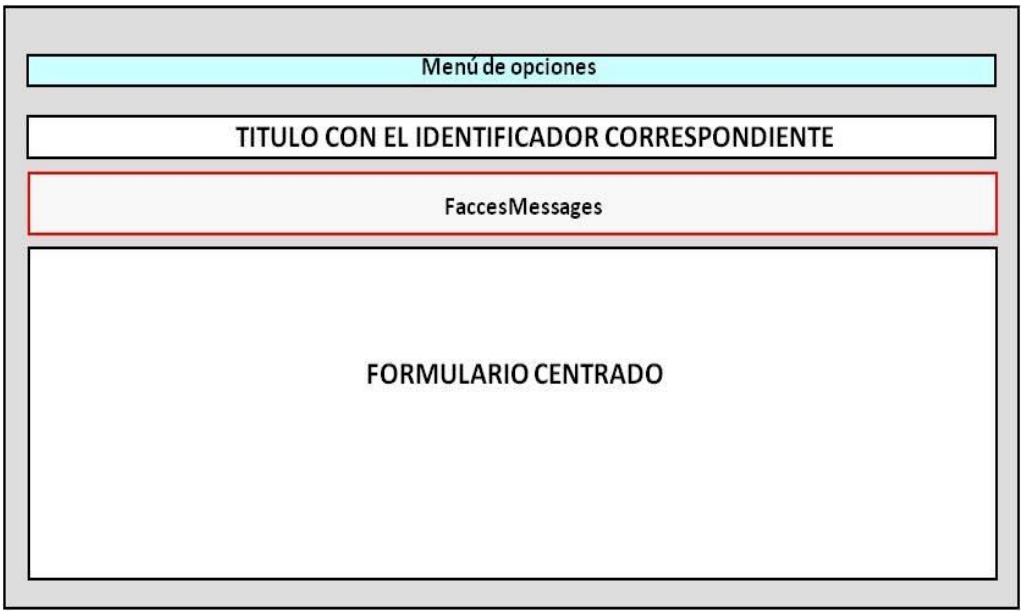

Figura 13. Plantilla Contenido con Formulario. Division de Servicios de Informacion

Paginación

Para la paginación se debe tener en cuenta el tipo de consulta que se va a realizar y la memoria consumida por ésta en el servidor de base de datos. De acuerdo a esto la aplicación decide el número de registros máximos que debe retornar la consulta.

Por ejemplo, se quiere consultar los estudiantes de la sede Bucaramanga. De acuerdo al análisis realizado, no hay un consumo alto de memoria, por lo que se decide retornar 100 estudiantes máximos. Esto indica que cuando se implemente el método de consulta en JPQL, el parámetro setMaxResults debe tener el valor de 100.

Texto

Existen cinco tipos de plantillas para mostrar texto en la página. Éstas se encuentran dentro de la carpeta Plantillas.

- etiqueta.xhtml: Texto o datos.
- etiquetaColumna.xhtml: El título de la columna en una tabla
- titulo.xhtml: Titulos
- etiquetaNavegacion.xhtml: Utilizado en la barra de navegación
- mostrar.xhtml: Permite visualizar dos etiquetas (etiqueta, dato) al mismo tiempo. Su objetivo es la de visualizar datos como si fuera un formulario.

La sintaxis para utilizar las plantillas son:

```
<s:decorate template="../Plantillas/[nombrePlantilla]">
     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>
</s:decorate>
```

Donde nombrePlantilla puede ser etiqueta.xhtml, etiquetaColumna.xhtml, etiquetaTitulo.xhtml o etiquetaNavegacion.xhtml.

Para la plantilla mostrar.xhtml:

```
<s:decorate template="../Plantillas/mostrar.xhtml">
     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>
     <h:outputText
     value="#{formatoWeb.usuario.primerNombre}"/>
</s:decorate>
```

Edición

Para capturar cualquier tipo de dato en una caja de texto se debe utilizar la plantilla edicion.xhtml de la siguiente manera:

```
<s:decorate id="dcr[identificador]"
     template="/Plantillas/edicion.xhtml">

     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>

     <h:inputText id="txt[nombreCajaDeTexto]" value="[valor]"
          required="true">

          <a:support event="onblur"
     reRender="dcr[identificador]"/>
     </h:inputText>

</s:decorate>
```

dcr[identificador] es el nombre del elemento decorate. Por ejemplo: dcrPrimerNombre.

txt[nombreCajaDeTexto] es el nombre de la caja de texto. Por ejemplo: txtPrimerNombre.

Tablas estáticas

Se debe usar el control panelGrid de Java Server Faces.

Tablas dinámicas

Se debe usar el control dataTable de Rich Faces, agregando las siguientes líneas de programación en su definición:

onRowMouseOver="this.style.backgroundColor='#E1E1E1'"

onRowMouseOut="this.style.backgroundColor='#{a4jSkin.tableBackgroundColor}'

Las cuales indican el color de la fila cuando el puntero del mouse esta sobre ésta.

Listas desplegables

Para cualquier tipo de lista se debe utilizar el control de Java Server Faces.

Mensajes del sistema

Los mensajes del sistema son textos enviados por los EJB de sesión, ya sea de confirmación de una acción o errores en el procedimiento. Dichos errores se deben mostrar en la etiqueta FacesMessages la cual debe estar definida al comienzo de la página después del menú si existiera éste. El código es el siguiente:

```
<h:messages globalOnly="true" styleClass="message"/>
```

Para mostrar errores de validación en los formularios, éstos deben aparecer al frente de cada control y no globalmente.

4.3 DIAGRAMAS UML

Dentro del proceso de análisis y diseño del sistema de información se generaron los diagramas UML, que se presentarán a continuación, es de destacar que durante el transcurso del proyecto el diseño se fue adaptando a medida que se fue perfeccionando el prototipo inicial.

Aquí se presentarán los esquemas básicos de cada uno de los diagramas que se generaron, producto del diseño, con el fin de ilustrar el fondo y la forma como fueron producidos.

4.3.1 Diagrama de Casos de Uso.

4.3.1.1 Identificación de los Actores

Dentro del sistema de la oficina de Control Interno Disciplinario se pueden identificar los siguientes actores que aunque no hacen parte activa del sistema, ni tienen ningún manejo sobre él son importantes porque de ellos depende el uso del sistema y el inicio del proceso:

- **Persona Particular**: Son todas aquellas personas ajenas a la universidad ciudadanos del común sin ningún tipo de vinculación con la misma que presentan ante la oficina de control interno las quejas o anónimos.
- **Estudiante UIS**: Son todas aquellas personas que se encuentran matriculadas en alguno de los programas académicos de la universidad.
- **Empleado UIS**: Son todas aquellas personas que están vinculadas a la universidad.

También dentro del sistema se pueden identificar los siguientes actores, cada uno con roles, acciones y permisos distintos:

- **Secretaria CID**: Es la encargada de la recepción y registro de las quejas, los informes y los anónimos, el registro de los quejosos, investigados declarantes, también de la elaboración de algunos documentos como las citaciones, comunicaciones, constancias, notificaciones y solicitudes.
- **Director CID**: Es el funcionario que tendrá a su cargo la administración del sistema y gozara de todos los permisos de acceso al sistema.
- **Vicerrectores (vices)**: Son el Vicerrector académico o Vicerrector administrativo de la UIS.

- **Secretaria General**: Es la encargada de la recepción del borrador del fallo, generar los actos administrativos y enviarlos a los vicerrectores o al rector dependiendo si el fallo es de primera o segunda instancia.
- **Rector**: Es la persona que decide en segunda y última instancia cual será la decisión final del proceso que ha llegado hasta su despacho.
- **Administrador**: Es la persona que será la encargada de definir los roles, asignar los permisos necesarios y requeridos para el buen funcionamiento del sistema.

4.3.1.2 Casos de uso por Actor

Actor: Secretaria CID

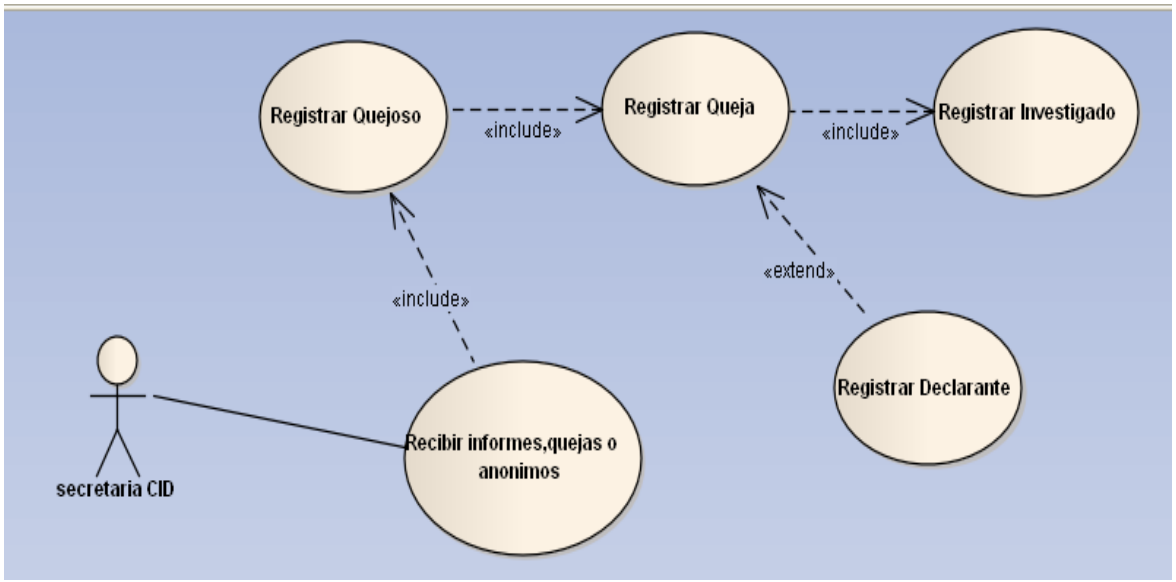

Figura 14. Casos de Uso Secretaria CID

Se identifican cinco casos de uso asociados a Secretaria CID:

- Recibir informes, Quejas o anónimos
- Registrar Quejoso
- Registrar Queja

- Registrar Investigado
- Registrar Declarante

Aquí se documenta la información del caso de uso: Recibir informes, Quejas o anónimos.

Caso de uso: Recibir informes, Quejas o anónimos

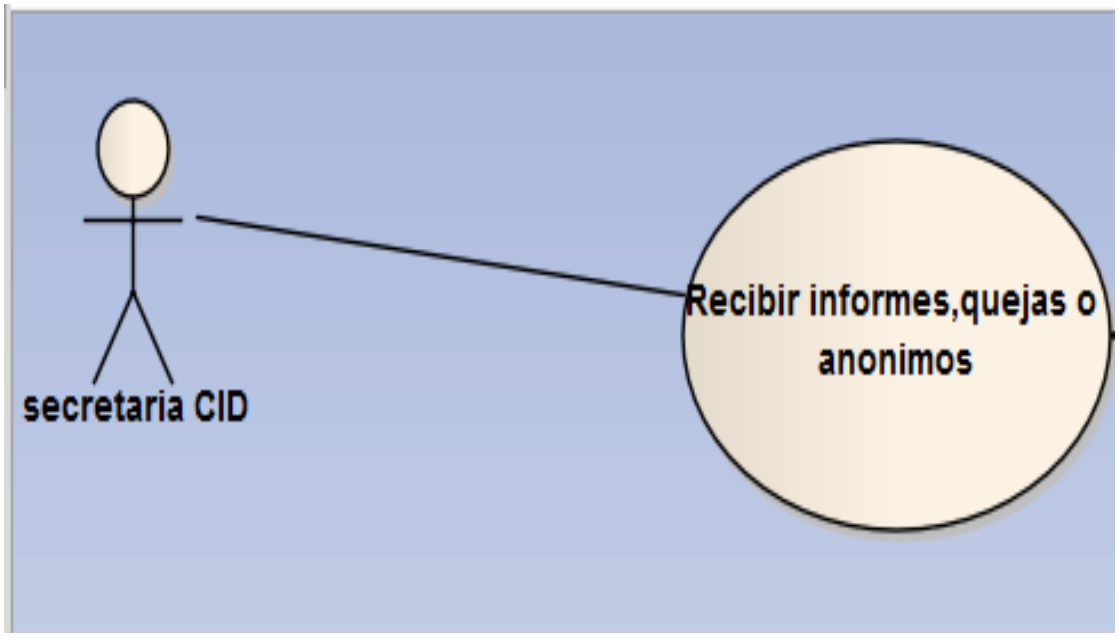

Figura 15. Caso de Uso: Recibir informes, quejas o anonimos

CASO DE USO:	Recibir informes, quejas o anónimos.
ACTORES:	Secretaria CID.
RESUMEN:	Este caso de uso refleja la situación de recibir todos los documentos que generen alguno de los tipos descritos

	como son el informe la queja o el anónimo para ser estudiado por el director de CID. Ella será la encargada de hacer el traspaso de estos documentos para su posterior estudio.
PRECONDICIONES:	El usuario secretaria CID debe validarse e identificarse.
FLUJO PRINCIPAL:	Se reciben los documentos y se clasifican para su posterior envío.
SUBFLUJO:	Se registra el quejoso, la queja, la(s) persona(s) que posiblemente serán investigadas y por último los posibles declarantes si los hay.

Tabla 1. Descripcion Caso de Uso: Recibir informes, quejas o aninomos

Actor: Director CID

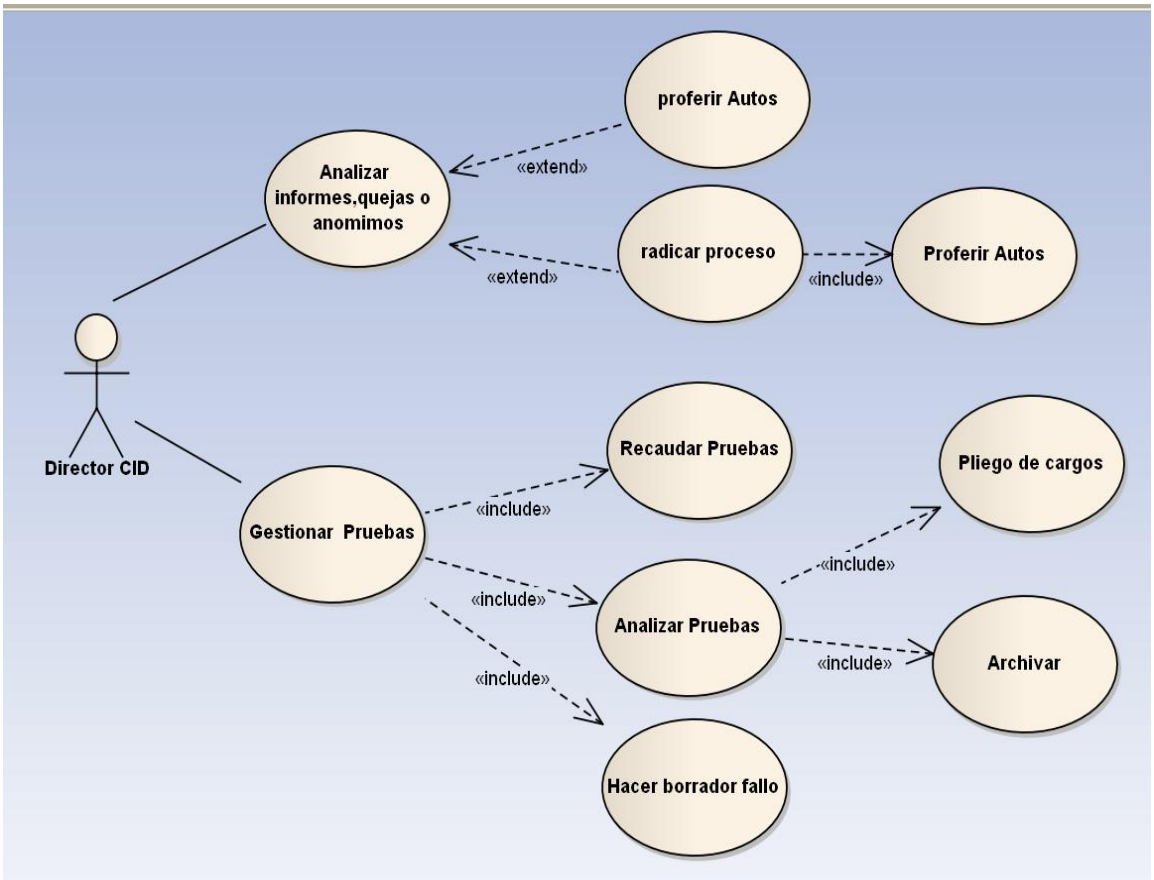

Figura 16. Casos de Uso asociados con director CID(parcial)

Sin duda alguna éste es el actor que mayor interacción tiene con el sistema y el cual maneja los parámetros más importantes dentro del funcionamiento del mismo. Se identifican algunos casos de uso principales asociados a Director CID:

- Analizar informes, Quejas o anónimos
- Hacer Citaciones
- Hacer Solicitudes y Comunicaciones
- Gestionar Pruebas
- Realizar Consultas

Aquí se documenta la información del caso de uso: Analizar informes, quejas o anónimos.

Caso de Uso: Analizar Informes, quejas o anónimos.

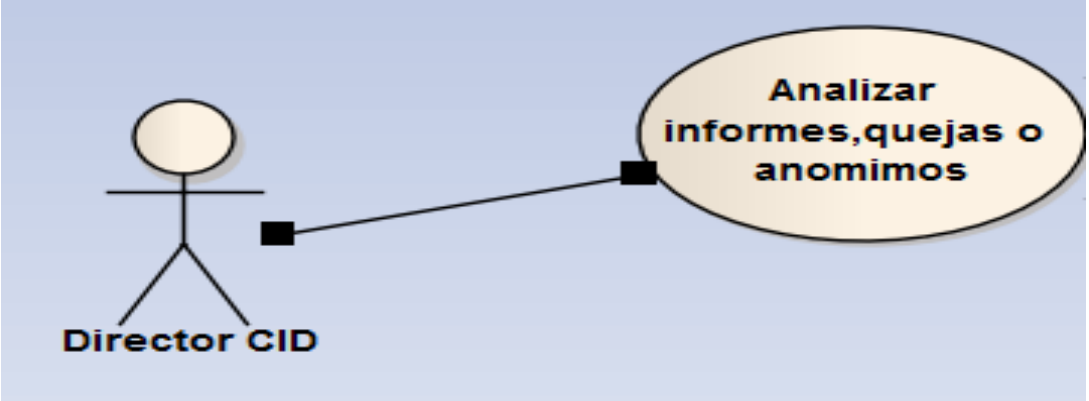

Figura 17. Caso de Uso. Analizar informes, quejas o anónimos.

CASO DE USO:	Analizar informes, quejas o anónimos.
ACTORES:	Director CID.
RESUMEN:	Se analizará si la solicitud en cualquiera de sus formas es procedente como para abrir una investigación o no procede.
PRECONDICIONES:	1. Validación previa de usuario. 2. Haberse recibido una queja por parte de la secretaria CID.
POSTCONDICIONES	1. Proferir Autos. 2. Radicar proceso.

FLUJO PRINCIPAL:	Se Profiere Auto inhibitorio si la queja no es procedente.
SUBFLUJOS:	En caso de la queja ser procedente se radica el proceso y se procede a hacer un Auto de apertura de investigación.

Tabla 2. Descripcion Caso de uso. Analizar informes ,quejas o anonimos

4.3.2 Diagramas de Secuencias

A continuación se presentaran los diagramas de secuencia, que amplían de manera considerable la visión general sobre el funcionamiento e interacción del sistema con el usuario ya sea éste ciudadano, personal de control interno o personal de las unidades académico administrativas.

Se elaboró el diagrama de secuencia del caso de uso, considerado mas importante, con el fin de ampliar la información que éste brinda y lograr abstraer mejor la idea de cada elemento del sistema.

Para el Caso de Uso. Recibir Quejas, informes o anonimos.

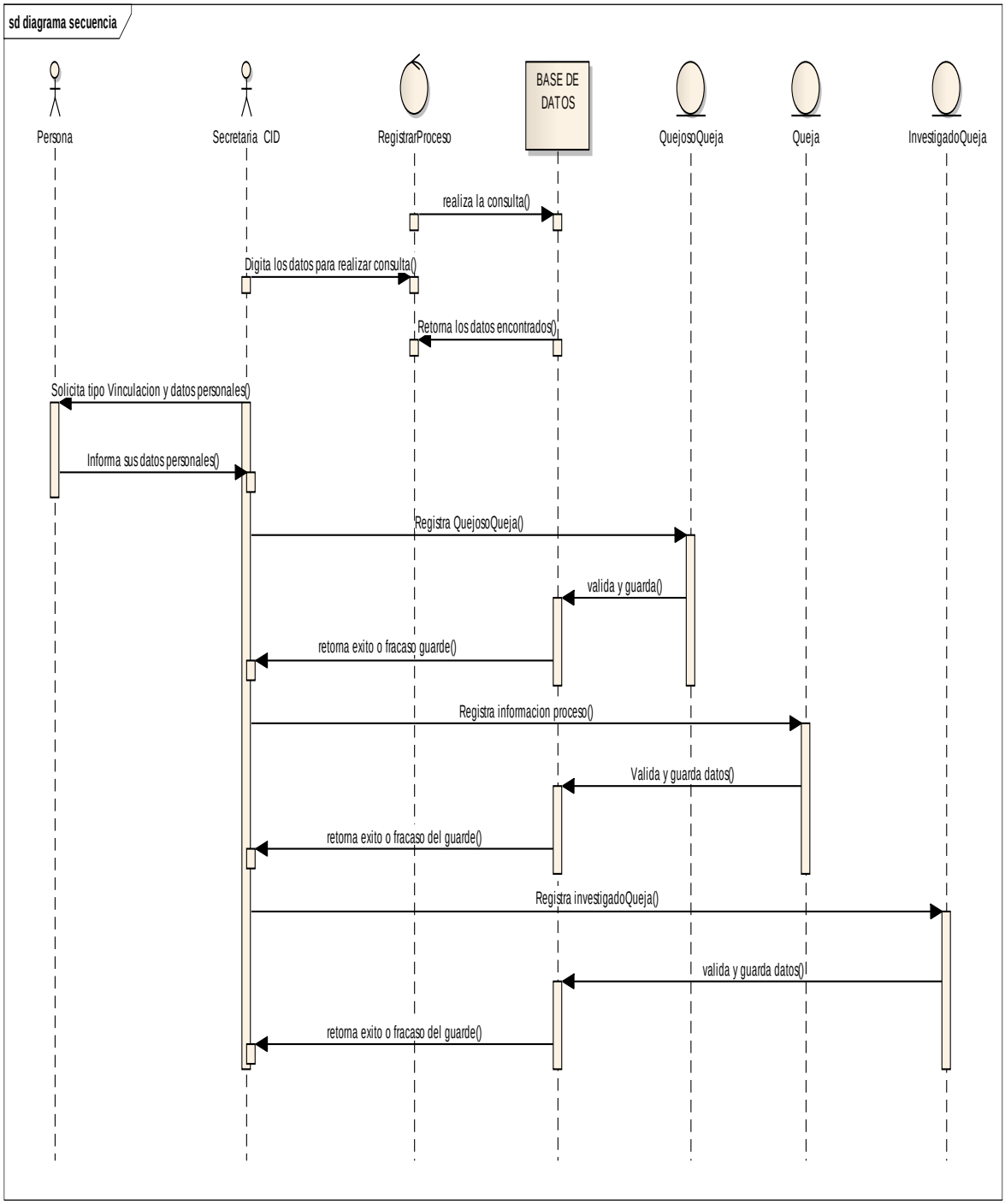

Figura 18. Diagrama de Secuencia para el caso de uso. Recibir Quejas, Informes o anonimos

4.3.3 Diagrama de Clases

Aquí se presenta el esquema asociado al caso de uso Registrar quejas, informes o anónimos, que hace parte del complejo diagrama del proyecto.

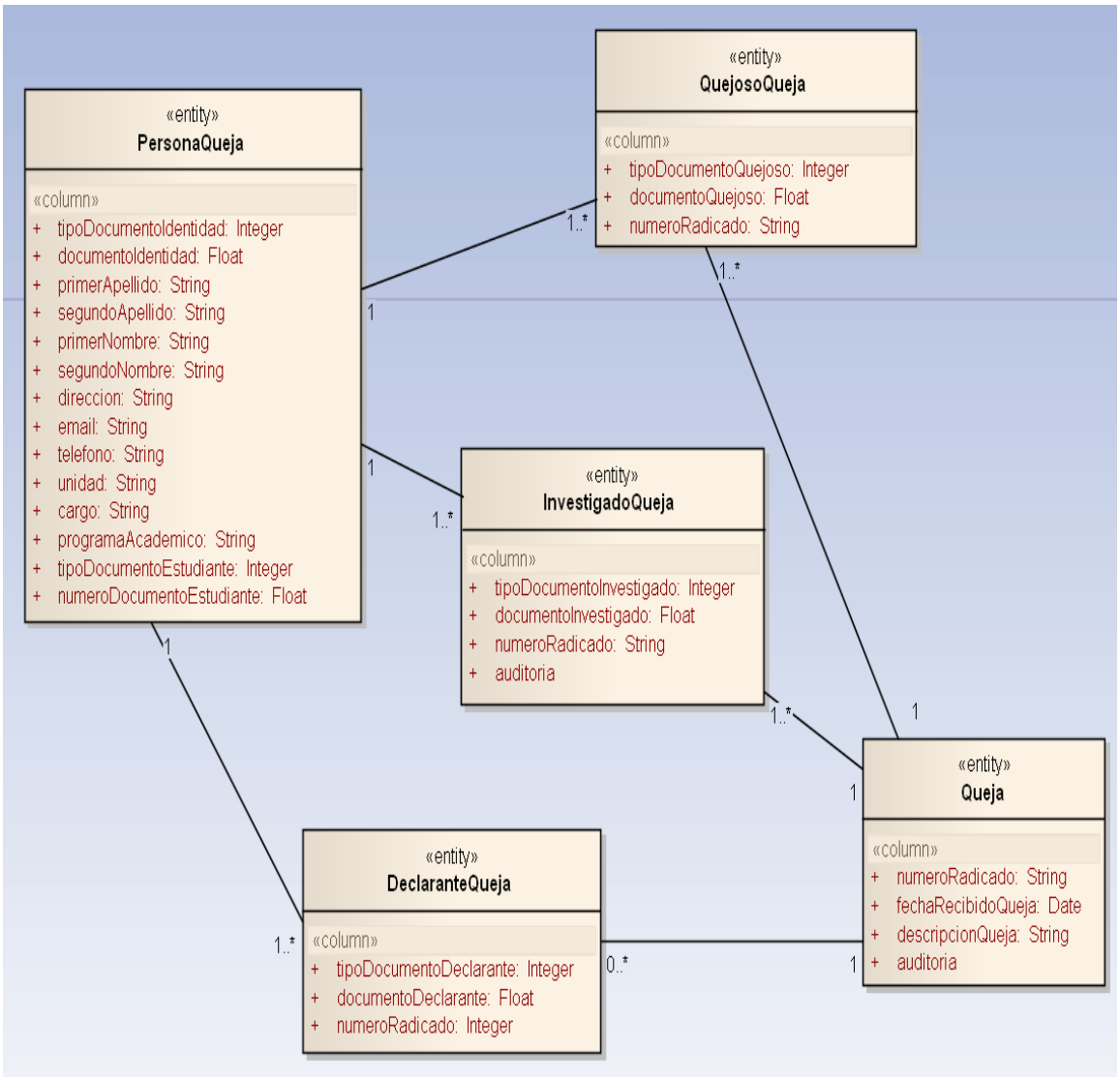

Figura 19. Diagrama de Clases parcial SIOCID.

4.4 PROTOTIPO INICIAL

4.4.1 Generalidades del prototipo inicial

El proceso de construcción del prototipo inicial comenzó desde el momento en manifestar el interés de desarrollar el proyecto y continuó con una serie de encuentros periódicos, con el Director de Control Interno Disciplinario de la Universidad Industrial de Santander, quien será el directo responsable de la gestión del nuevo sistema de información.

En éstos encuentros se escuchó y analizó cada uno de los requerimientos deseados para el sistema y se fue estructurando un prototipo no funcional que serviría como carta de navegación en el transcurso del desarrollo del sistema.

El prototipo no funcional se creó como una página web básica sin ningún tipo de programación de fondo, que mostraría gráficamente el esquema de navegación del sistema, de ésta manera se definieron detalles de alta importancia tales como la información que el usuario quiere ver, la que desea ocultar, el contenido de cada una de las vistas y los datos que se le van a solicitar a cada uno de los usuarios que interactuarían con el sistema.

Se manejo un esquema cíclico de perfeccionamiento y refinamiento de tal manera que se llegara a un nivel de acercamiento detallado de lo que quiere alcanzar el usuario con el sistema, de esta manera se logra que el proceso de refinamiento del prototipo funcional sea mínimo, agilizando de esta manera el proceso de desarrollo del proyecto.

El tiempo destinado a esta etapa fue bastante amplio, ya que del resultado obtenido dependería la eficiencia del desarrollo y programación del sistema.

Durante el transcurso de la construcción del prototipo se definió todo el diseño integrado por los diagramas de casos de uso, de clases, de datos y de secuencia.

4.4.2 Interfaz resultado del prototipo inicial

Se presenta a continuación apartes de la interfaz resultante de la construcción del prototipo inicial (no funcional).

Figura 20. Vista de Pantallazo inicial. Prototipo Inicial

Vista del registro de una queja un informe o un anónimo.

Figura 21. Vista registro de una queja, informe o anónimo. Prototipo Inicial

Vista del detalle o resumen de la queja, informe o anónimo, después de haber sido registrada con éxito en el sistema para su posterior análisis.

Figura 22. Vista de la queja, informe o anónimo ya registrado. Prototipo Inicial.

Listado de las quejas, informes o anónimos para ser procesados bien sea en proceso investigativo o en documentos secretariales.

Figura 23. Listado de Quejas, informes o anónimos. Prototipo Inicial.

4.5 PROTOTIPO FINAL

4.5.1 Generalidades del Prototipo Final

Al término del desarrollo del sistema de información de la Oficina de Control Interno Disciplinario, se obtuvo un prototipo que se adapta plenamente a las características que se han definido desde el comienzo del desarrollo del proyecto.

Tomando como punto de partida el prototipo no funcional, se logró consolidar el producto definitivo, junto con un proceso de refinamiento por parte del cliente, vendrá a ser parte del sistema de información que mejorara considerablemente la

manera como se manejan las quejas, informes y anónimos en la oficina de Control Interno Disciplinario.

El producto cumple todos y cada uno de los parámetros que se definieron en el documento de requerimientos que se menciona en el apartado 4.1 de este documento, y después de un proceso de evaluación de parte del cliente, ha sido aceptado por éste, manifestando su plena conformidad con el resultado.

A continuación se describirá de forma general las funcionalidades más importantes, clasificadas por módulos, propias del sistema de información de la Oficina de Control Interno Disciplinario de la Universidad Industrial de Santander.

Pantallazo inicial

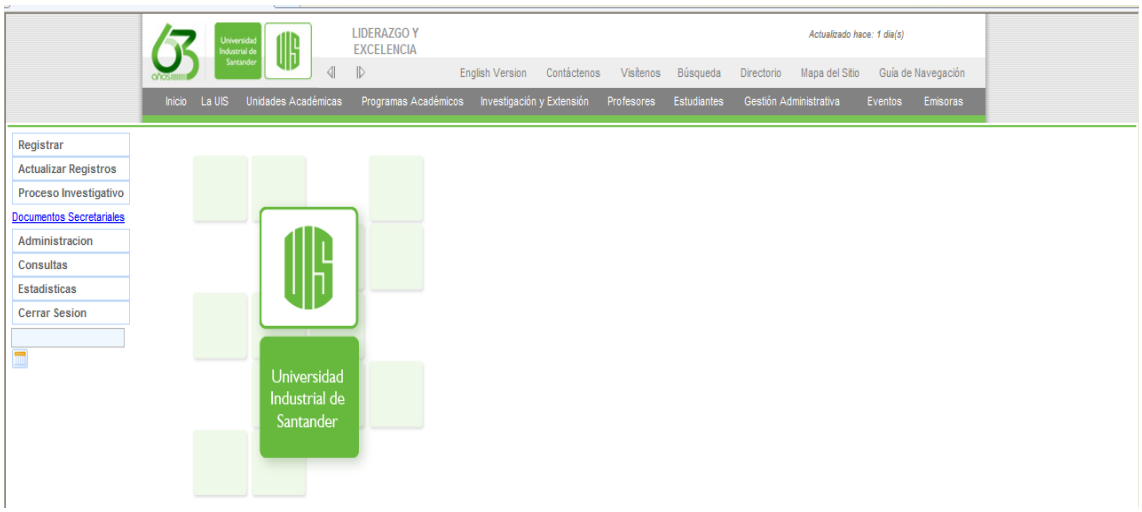

Figura 24. Vista del pantallazo inicial. Prototipo Final.

Modulo de registro de la queja.

- El sistema permite al personal de Control Interno Disciplinario el registro de la queja interpuesta por parte de: Estudiantes, Empleados UIS y Ciudadanía en general, en caso de estar vinculado de manera directa a la universidad (Estudiantes, Empleados UIS) se confronta ésta información con las bases de datos internas de la UIS.

- Se registra la descripción de la queja, se debe especificar también un motivo y automáticamente el sistema la pondrá en estado de recibida.

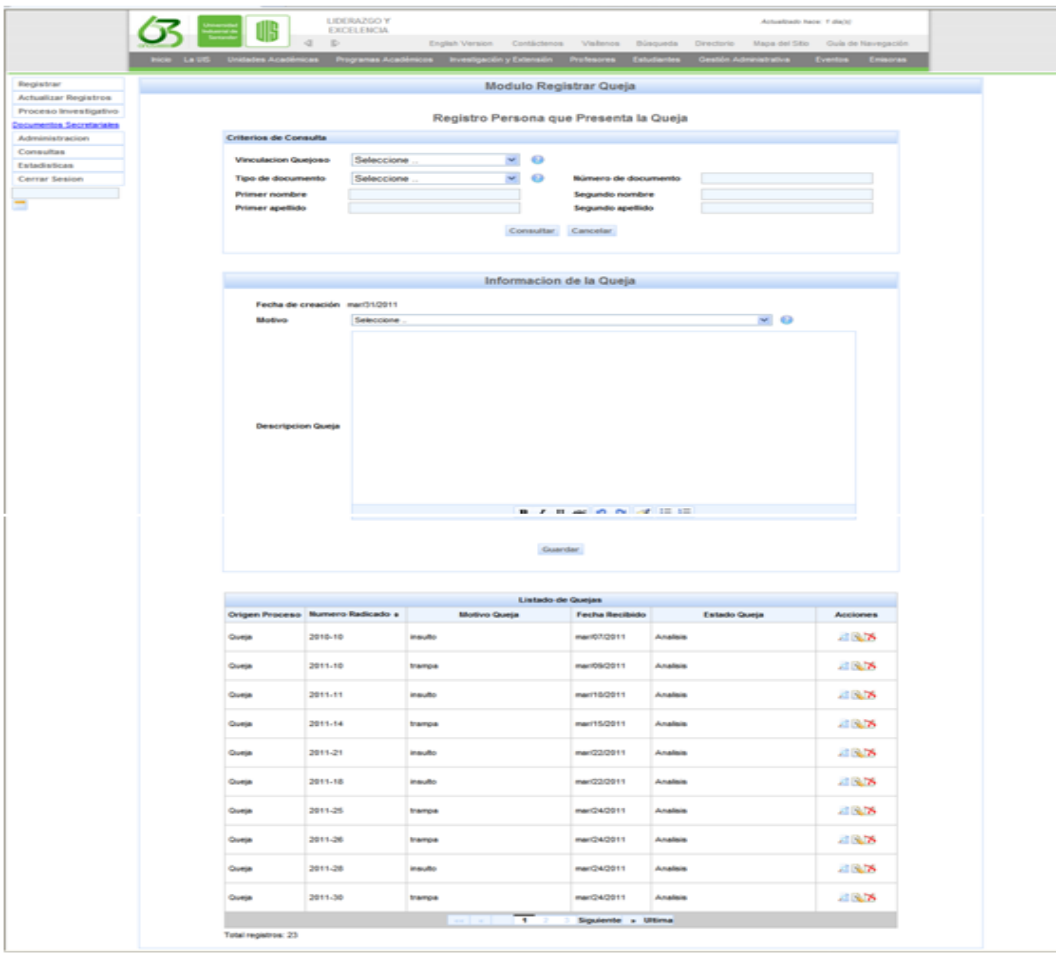

Figura 25. Vista Registro Queja. Prototipo Final.

- Al finalizar el registro de la queja, del quejoso(s) en la tabla de la parte de abajo se muestra el numero de radicado de la queja el motivo la fecha de radicado el estado y se da la opción de ver, eliminar o modificar la queja informe o anónimo, de agregar investigado(s) y en caso de tener algún declarante también se puede hacer.

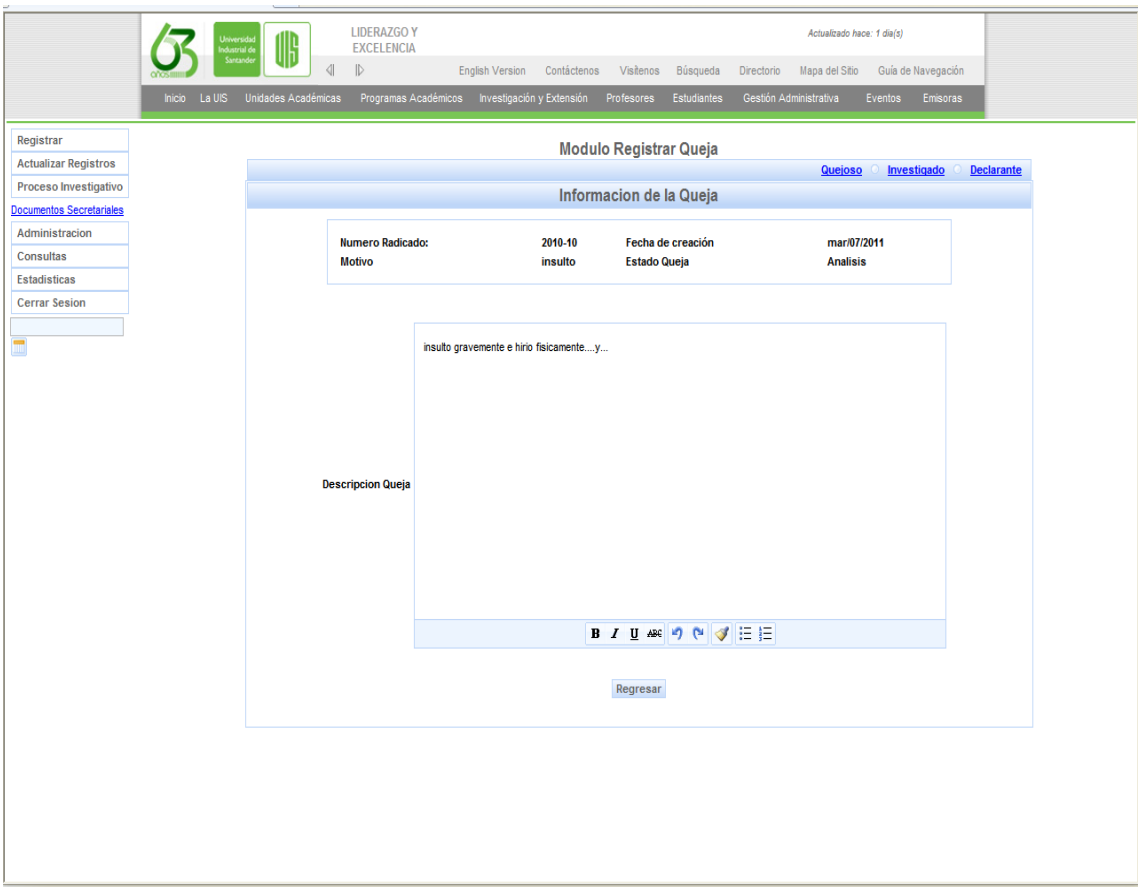

Figura 26. Vista para el registro de investigados y declarantes. Prototipo Final.

El modulo de Registro de Informes es similar al de la queja solo que únicamente puede ser interpuesto por personal adscrito a la universidad, y en el modulo anónimo no se solicita ningún tipo de identificación de la persona que lo interpone.

Modulo de Proceso Investigativo

- El director de Control Interno Disciplinario, podrá visualizar las quejas, los informes y los anónimos, cuyo estado de proceso sea “Recibido”, “Analizado” entre otros, para llevar a cabo el proceso investigativo.

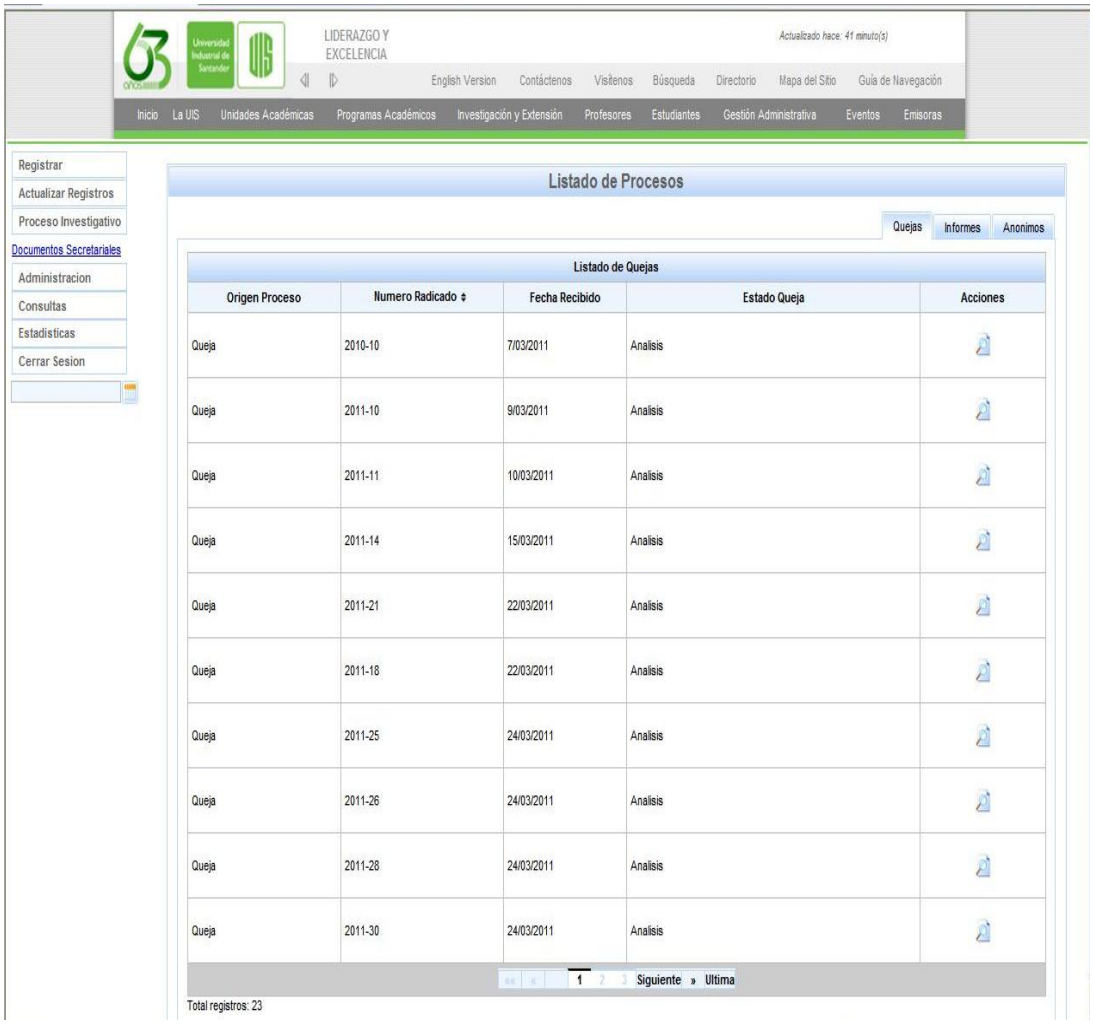

Figura 27. Vista de las quejas informes o anónimos. Prototipo Final

- El director podrá escoger cualquier queja, informe o anónimo, ver su descripción completa, las personas que están involucradas en el proceso y decidir si la queja, informe o anónimo procede para abrir una indagación preliminar o da por terminado el proceso sin haberlo iniciado porque no amerita abrir una investigación. Estas opciones están dentro de un combo que lo llevara a la correspondiente página dependiendo de la elección que haga.

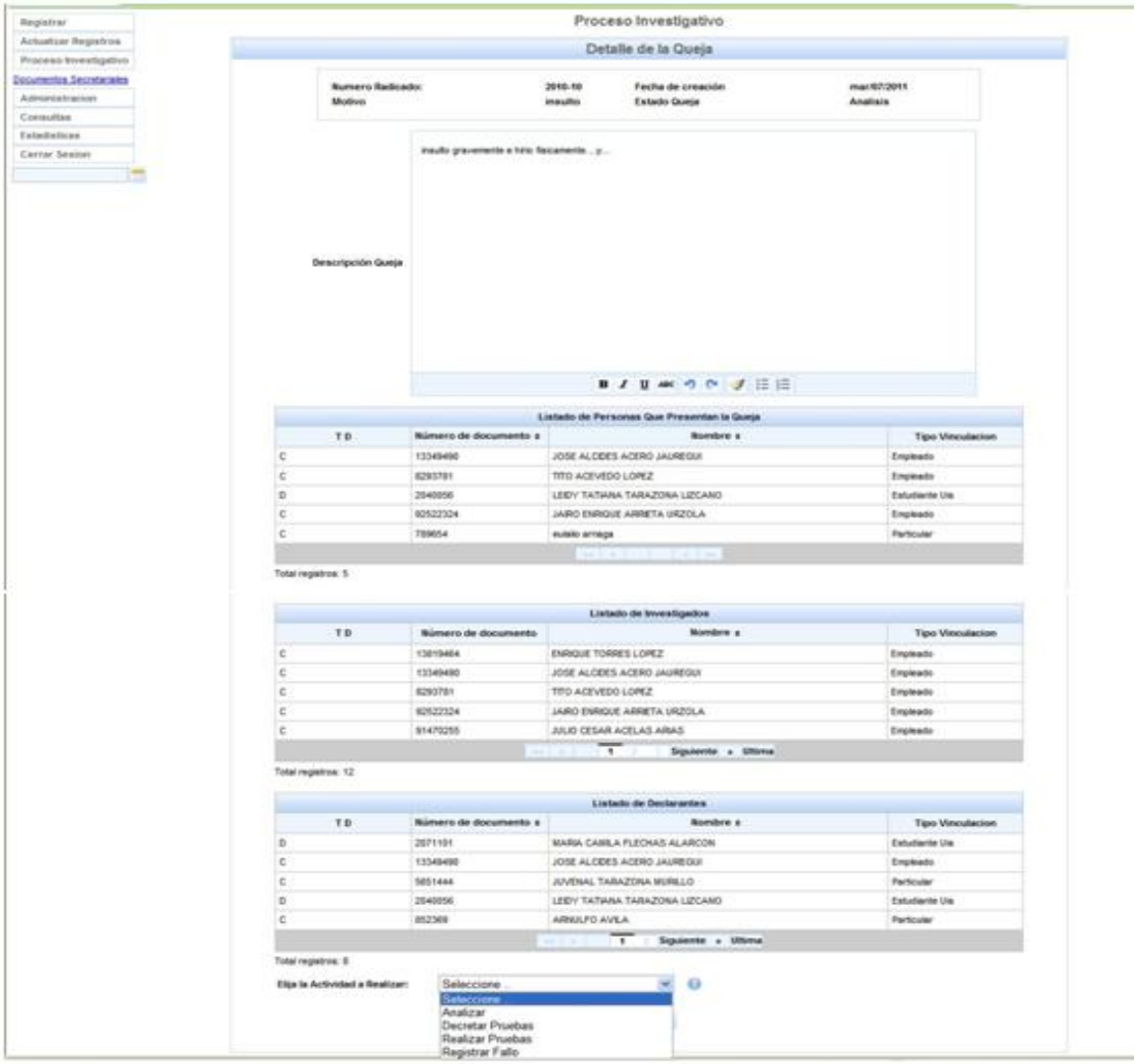

Figura 28Vista Detalle de la queja, informe o anónimo. Prototipo Final.

- El director de Control Interno también podrá decretar pruebas, realizar pruebas o registrar el borrador del fallo del proceso que ha seleccionado.

Modulo de Documentos Secretariales.

Al igual que en el modulo de proceso investigativo todo el personal de Control Interno Disciplinario previamente logueado podrá acceder a las quejas, informes o anónimos cuyo estado de proceso sea “Recibido”, “Analizado” entre otros.

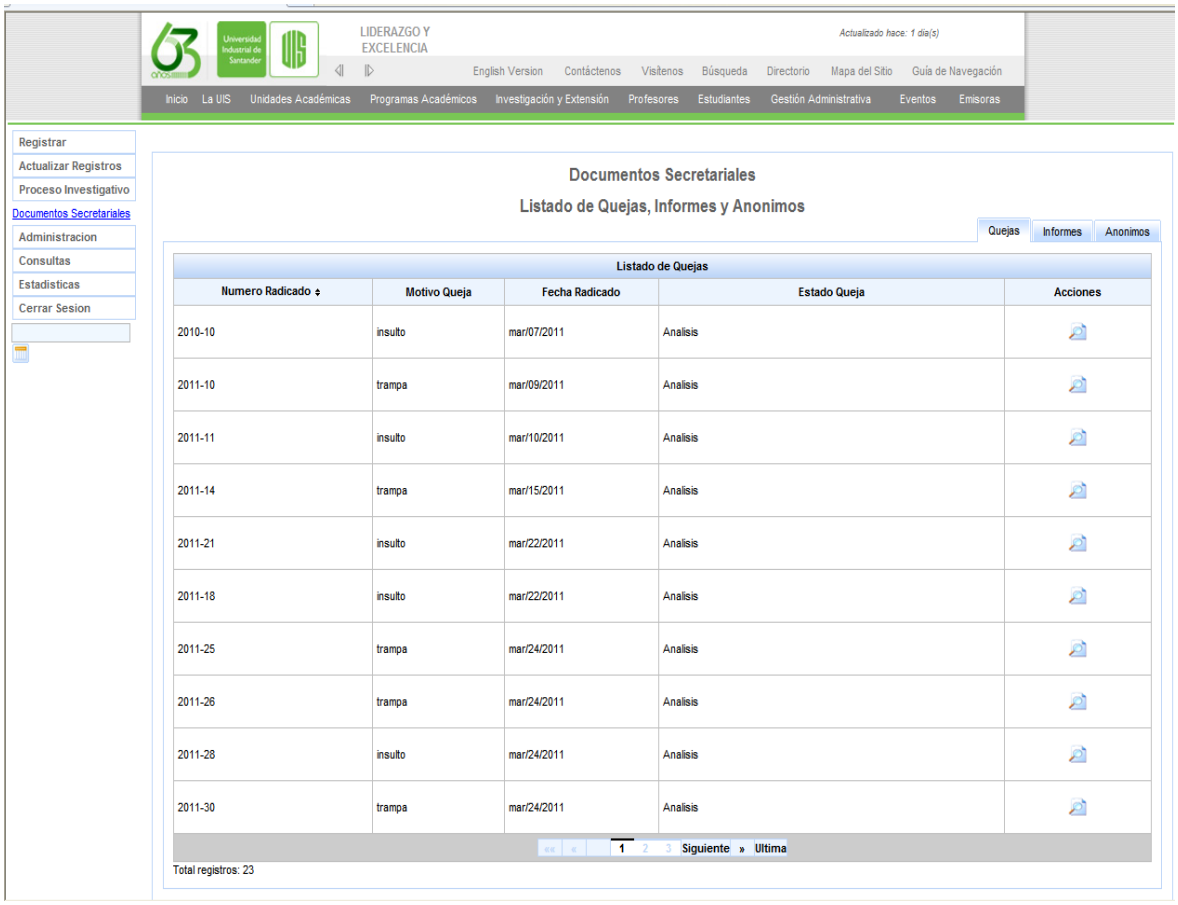

Figura 29. Vista de las quejas informes o anónimos. Prototipo Final

Acto seguido podrá ver el detalle de la queja y podrá escoger entre los diferentes tipos de documentos que están previamente estipulados para el manejo de un proceso investigativo de ley como son las citaciones, comunicaciones, constancias, notificaciones o solicitudes.

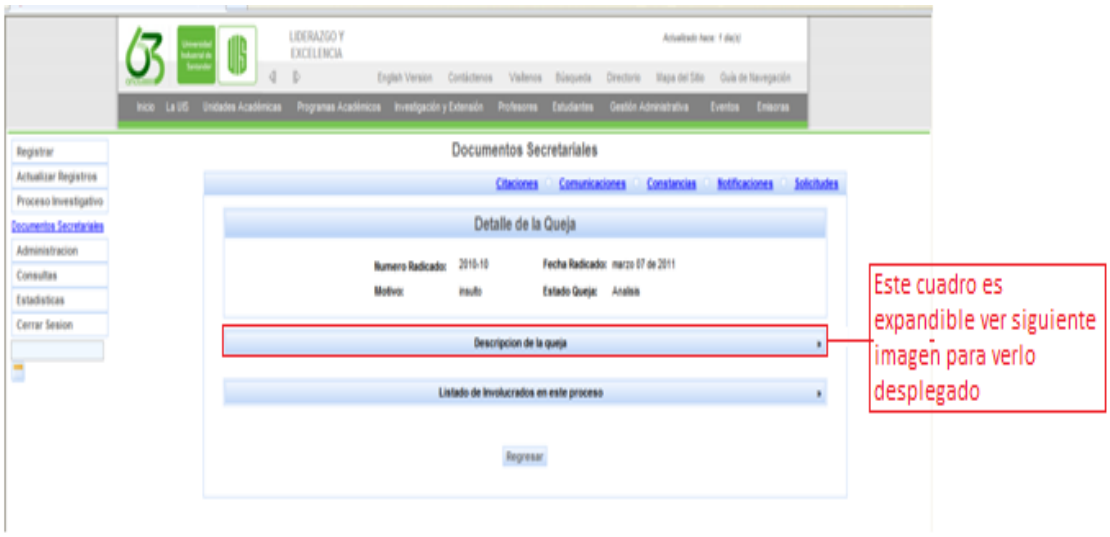

Figura 30. Vista del detalle de la queja

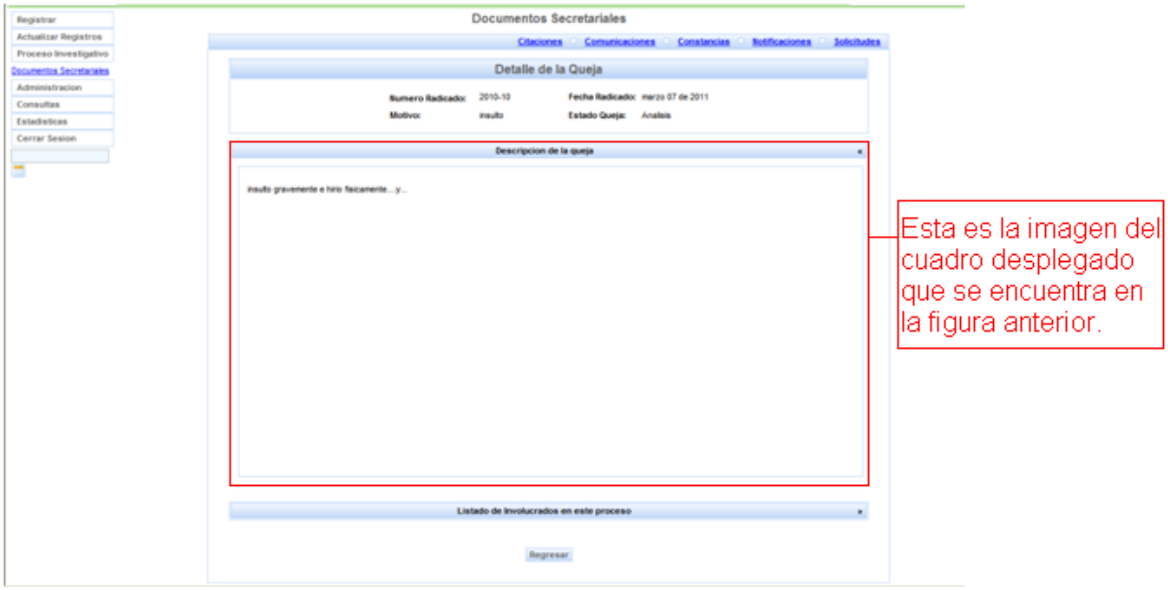

Figura 31. Vista cuadro desplegable del detalle de la queja

El uso de los paneles que se pueden expandir/ocultar se realizó con el fin de aprovechar el espacio en pantalla y para una mejor presentación.

Módulo de Consultas

El módulo de consultas podrá ser accedido por todo el personal de la universidad es un módulo en el cual está disponible toda la información correspondiente del proceso en modo solo lectura lo que implica que no podrá modificarse ni borrarse nada**.**

Las consultas podrán hacerse por investigado, por motivo, por queja o por quejoso como se muestra a continuación.

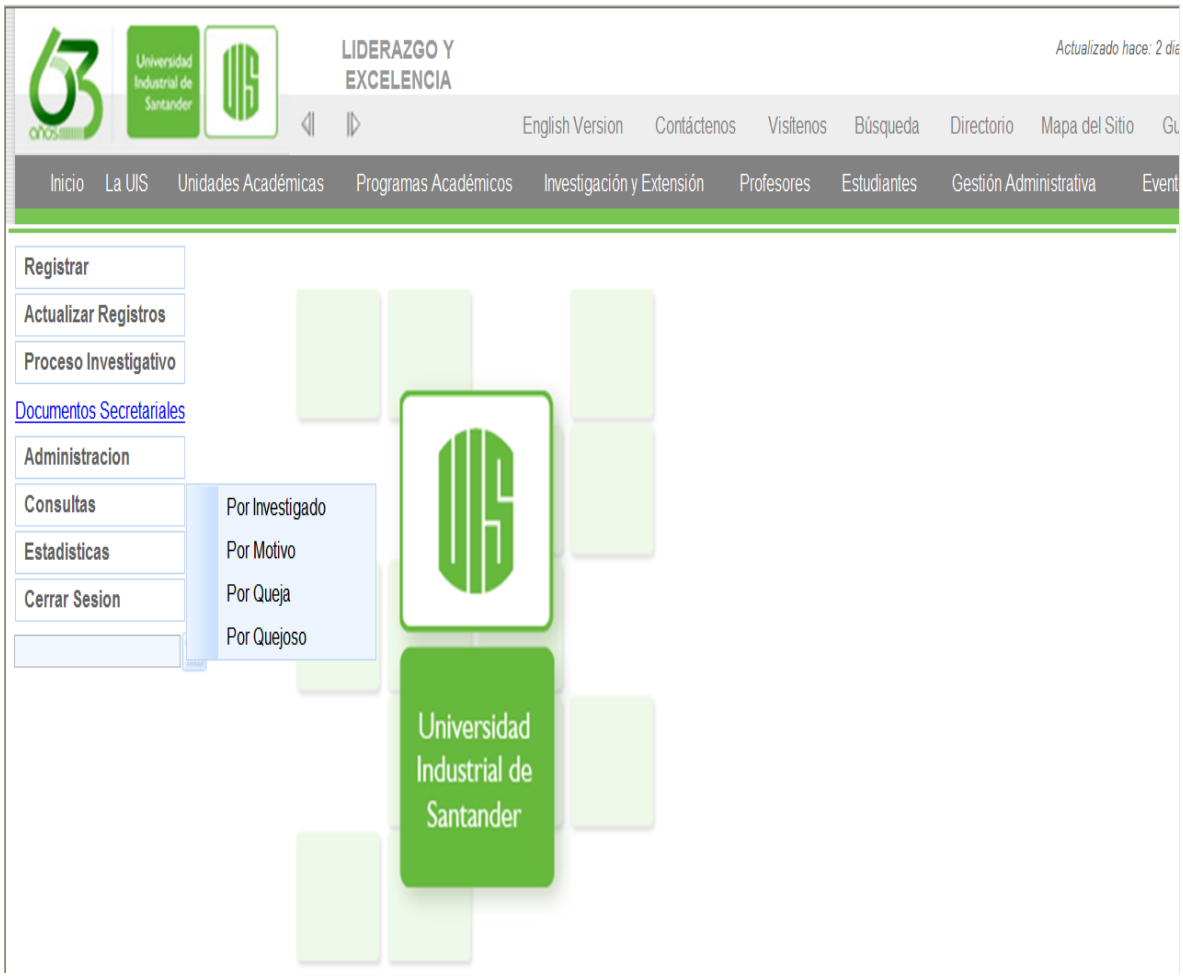

Figura 32. Vista del pantallazo para iniciar el modulo de consultas.

Aquí una muestra el pantallazo del modulo consultas opción consultar por Queja en el cual se deberá conocer el número de radicado de la misma en caso de no existir saldrá un mensaje donde se le informe al usuario que no existe ningún proceso con ese número de radicado.

Así mismo sucederá con las opciones consultar por quejoso, por investigado o por motivo.

Figura 33. Vista inicial de la opción consultar por queja.

4.5.2 Proyecto enmarcado en el esquema de Seguridad de la UIS.

Para este proyecto se utiliza el esquema de seguridad definido por la División de Servicios de Información para los diferentes sistemas de información que apoyan la gestión de la Universidad Industrial de Santander, el cual está basado en la estructura de roles – usuarios.

Los roles se establecen en cada una de las unidades académico administrativas, UAA, responsables de cada sistema, de acuerdo a las actividades que realizan. A cada uno de los roles definidos se le asocian los usuarios de acuerdo a las funciones que desempeñen.

4.5.2.1 Estructura de la Base de Datos soporte

La base de datos que soporta el esquema de seguridad contempla básicamente las siguientes tablas:

Sistema: Contiene información de los sistemas de información de la universidad. Para cada sistema se específica: Nombre, descripción del sistema, fecha y hora de creación en la base de datos, fecha y hora de inicio de vigencia del sistema, fecha y hora de cierre de vigencia del sistema.

Rol: contiene información de los diferentes roles definidos para cada sistema de información, como: Nombre asignado al rol, descripción del rol, fecha y hora de creación, fecha y hora de inicio de vigencia del rol, fecha y hora de cierre de vigencia del rol.

Usuario: Contiene información de los posibles usuarios de los sistemas de información. Entre esta información está: tipo y número de documento de

identidad del usuario, fecha y hora de creación del usuario, fecha y hora de inicio de vigencia del usuario, fecha y hora de cierre de vigencia del usuario.

Sistema–rol: Contiene los roles definidos para cada uno de los sistemas de información, indicando: rol, sistema, fecha y hora de creación del rol – sistema, fecha y hora de inicio de vigencia del rol en el sistema, fecha y hora de cierre de vigencia del rol en el sistema.

Rol–usuario: Contempla los usuarios asociados a cada uno de los roles definidos, considerando: Rol, usuario, fecha y hora de creación del rol – usuario, fecha y hora de inicio de vigencia del usuario en el rol, fecha y hora de cierre de vigencia del usuario en el rol.

Menú–rol–sistema: Contiene los menús asociados a los roles en los distintos sistema de información, contemplando: Sistema de información, nombre del menú, descripción del menú, fecha y hora de creación del menú, fecha y hora de inicio de vigencia del menú asociado al rol, fecha y hora de cierre de vigencia del menú asociado al rol.

Opción–menú–rol: Contempla las opciones definidas para cada una de los posibles menús establecidos para cada sistema de información. Contiene: Nombre de la opción, descripción de la opción, nombre del menú superior, nombre del menú que contiene la opción, nombre del programa a ejecutar cuando la opción es la de más bajo nivel, fecha y hora de creación de la opción del menú, fecha y hora de inicio de vigencia de la opción, fecha y hora de cierre de la opción.

Tabla–sistema: Contiene información de las tablas que conforman la base de datos que soporta cada uno de los sistemas de información. Considera: Sistema de información, nombre de la tabla, descripción de la tabla.

Tipo–permiso: Establece para cada tabla de un sistema de información, los roles que tienen permisos para incluir registros, para modificar registros o para eliminar registros en ella. Contiene: Sistema de información, nombre de la tabla, clase de

permiso (inclusión, modificación, eliminación de registros), fecha y hora de creación del permiso, fecha y hora de inicio de vigencia del permiso, fecha y hora de fin de vigencia del permiso.

Acceso–tabla: Define para las tablas de un sistema de información si un rol tiene permiso sobre toda la información de la tabla o sobre una parte de esta. Considera: Sistema, nombre de la tabla, clase de acceso (total, parcial), fecha y hora de creación del permiso, fecha y hora de inicio de vigencia del permiso, fecha y hora de fin de vigencia del permiso.

Atributo–tabla: Establece los atributos sobre los cuales se debe controlar el acceso a una tabla, cuando a un rol se le concede permiso para hacer uso parcial de la información existente en una tabla. Contiene: Sistema de información, nombre de la tabla, nombre del atributo sobre el cual se controla el acceso a la tabla, descripción del atributo, fecha y hora de creación del atributo, fecha y hora de inicio de vigencia del atributo, fecha y hora de fin de vigencia del atributo.

Valor–atributo-proceso: Contiene los valores que deben tener los atributos definidos en cada tabla en la tabla atributo – tabla que permiten el acceso a la información asociada a estos valores. Específica: Sistema de información, nombre de la tabla, nombre del atributo, valor del atributo, descripción, fecha y hora de creación del valor del atributo, fecha y hora de inicio de vigencia del valor del atributo, fecha y hora de fin de vigencia del valor del atributo.

Acceso-sistema: Contempla el histórico de acceso que un usuario ha realizado a un sistema, identificando las opciones que ha seleccionado. Contiene: Login de usuario, rol, identificación de la sesión, sistema, opción seleccionada, fecha y hora de ingreso, fecha y hora de salida.

4.5.2.2 Entorno de Navegación

Para cada sistema de información, la UAA responsable define los roles necesarios para el adecuado uso del sistema de información de acuerdo a las funciones que realice y establece los usuarios asociados a cada uno de ellos.

Para cada rol se define el menú de inicio, el cual le permite a cada usuario que hace parte de este rol, empezar la navegación por las distintas opciones que le ofrece el sistema, hasta llegar al nivel más bajo en el cual se ejecuta el proceso que soporta la actividad que desea realizar.

Este entorno está soportado por las siguientes tablas de las base de datos del esquema de seguridad: Rol, usuario, sistema, sistema-rol, usuario-rol, menú-rol, opción-menú rol, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

4.5.2.3 Entorno de Control de Datos

Para los roles definidos en cada uno de los sistemas de información se especifican las tablas a las cuales puede acceder, el tipo de transacción que puede realizar sobre estas tablas (inclusión, modificación o eliminación de registros), si tiene acceso total o parcial a la información que contiene la tabla.

Para el acceso a la información de la tabla de manera parcial, se debe establecer el atributo o atributos seleccionados, los valores que estos atributos deben tener para autorizar el acceso solicitado.

Este entorno está soportado por las siguientes tablas de las base de datos del esquema de seguridad: Rol, usuario, sistema, sistema-rol, usuario-rol, tabla–

sistema, tipo-permiso, acceso-tabla, atributo-tabla, valor atributo proceso, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

4.5.2.4 Auditoría

Todas las tablas que conforman la base de datos soporte del esquema de seguridad tienen el historial de las transacciones realizadas sobre cada una de ellas.

El historial de las transacciones de cada tabla contiene información de los registros incluidos en la tabla, de los registros modificados y de los registros eliminados. Adicionalmente, en cada transacción se especifica: Fecha de la transacción, hora de la transacción, tipo de transacción (I/U/D), tipo y número de documento de identidad del usuario que realizó la transacción, login, rol asociado, dirección IP y MAC del equipo desde el cual llevó a cabo la transacción.

4.5.3 Organización de Directorios.

Consideraciones iniciales

Para cada uno de los sistemas principales se debe crear un proyecto que siga los estándares para la estructura de directorios y nombres. En estos proyectos van las entidades propias de cada sistema, así como los componentes comunes a varias aplicaciones.

Las entidades van, en cada sistema, empaquetadas en un .jar, de tal manera que el desarrollo de una nueva aplicación no implica la implementación de éstas.

En el paquete general van los elementos que son utilizados por todos los módulos.

Las entidades van a estar todas en co.edu.uis.[Sistema].entidades.

Los servicios van a estar todos en co.edu.uis.[Sistema].servicios.

Para la creación de las entidades se toma la estructura de la base de datos. Por lo tanto se hace indispensable comenzar por este paso, seguido de la creación de los componentes para posteriormente desarrollar las demás aplicaciones.

Carpetas

Deben comenzar con minúscula y están basados en la estructura de directorios que se genera mediante “Seam-Gen” cuando se crea un nuevo proyecto. Las carpetas que se crean son las siguientes:

- bootstrap
- classes
- dist
- exploded-archives
- lib
- nbproject
- resources
- src
    - hot
    - main
    - test
- test-build
- view

A continuación se hace una breve descripción de las carpetas con las que el desarrollador tiene contacto directo.

Carpeta src

En esta carpeta se guardan los archivos fuentes de las clases del sistema distribuidos en dos subcarpetas de la siguiente manera:

Carpeta main

Contiene las clases de apoyo al proceso. No incluye a las entidades, pues estas estarán disponibles en otro lugar que se describe más adelante en este documento.

Carpeta hot

Contiene los casos de uso reflejados en los EJBs y sus interfaces.

Carpeta resources

Contiene los archivos de configuración de la aplicación.

Carpeta view

Contiene las páginas, imágenes, estilos y todo lo referente a aspecto de la aplicación.

Carpeta dist

Contiene los archivos jar, war y ear de la aplicación. Estos se generan automáticamente.

Carpeta lib

Contiene las librerías necesarias para la ejecución de la aplicación.

Nombres de archivos

Se sigue el mismo estándar dado para el nombre de variables descrito en el apartado 4.2.3 de éste documento.

Organización del código fuente dentro de la estructura de directorios.

Paquetes

La composición del nombre de los paquetes se sigue teniendo en cuenta la siguiente sintaxis:

co.edu.uis.[sistema].[aplicación].[módulo].[caso de uso]

Ejemplo:

co.uis.edu.co.siocid.administracion.mantenimiento.Actividad

En el caso de los proyectos principales (los que empaquetan entidades y componentes para cada sistema) los paquetes serán los siguientes:

Administración (src/action): co.edu.uis.[Sistema].[Aplicación].administracion

Clases generales (src/action): co.edu.uis.[Sistema].[Aplicación].general

Paginas (carpeta view)

Dentro de esta carpeta se tendrán la siguiente estructura:

- Plantillas : Plantillas del sitio
- imágenes
- estilos
- scripts (Si son necesarios. No es obligatorio y se deben evitar en la medida de lo posible).
- Administración : Todo lo referente a la administración
- generales : Páginas de uso global en la aplicación
- ayudas
- módulos

4.5.4 Documentación de programas fuente[9].

Nombre de archivos, variables, constantes, atributos, métodos y parámetros.

Los nombres dentro del código fuente siguen los parámetros establecidos en el estándar general de nombres, con las siguientes particularidades.

- Los nombre de los parámetros de los métodos empiezan con guión de piso, el resto del nombre sigue el estándar para nombres de variables. Una vez dentro del código fuente, se deben asignar a una nueva variable con el mismo nombre pero sin el guión de piso.
- Los nombres de constantes o variables finally, se escriben en mayúscula sostenida separando las palabras por guiones de piso.

Comentarios Java:

Los programas en Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los comentarios de implementación están delimitados por /*.. */ cuando se trata de varias líneas y // cuando se trata de una línea. Los comentarios de documentación (conocidos como "comentarios JavaDoc") son específicos de Java y están delimitados por :

/** ... */. Los comentarios de documentación se pueden extraer a ficheros HTML usando la herramienta javadoc.

Los comentarios de implementación están destinados a comentar el código o para comentarios sobre la implementación en particular, buscan dar una orientación y hacer aclaraciones sobre la implementación a quien observa el código fuente. Los comentarios de documentación están destinados a describir la especificación del código, desde una perspectiva independiente de la implementación, están hechos

[9] Fuente: Estándares de la División de Servicios de Información de la Universidad Industrial de Santander

para ser leídos por desarrolladores que pueden no tener necesariamente el código fuente a mano.

Los comentarios se deben usar para dar una visión general del código y para proporcionar información adicional que no esté disponible fácilmente en el propio código.

Orden dentro de los archivos de código fuente:

Cada archivo de código fuente Java debe contener una única clase o interfaz pública y deben seguir el siguiente orden:

- Sentencias package.
- Sentencias import.
- Comentario de documentación de la clase/interfaz.
- Declaraciones de clase e interfaz.
- Comentario de la implementación de la clase/interfaz si fuera necesario.
- Declaración de constantes (solo se declaran dentro de interfaces).
- Declaración de atributos. (la declaración se debe hacer uno por línea y al frente debe ir un comentario de implementación de ser necesario.)
- Declaración de variables. (Según los estándares no se deben declarar variables públicas, para ello se definen los métodos gets() y sets()). Las variables aquí declaradas serán privadas y se utilizarán como indicadores de estado de la clase. Las variables se deben declarar en el siguiente orden:
  - Variables estáticas: organizadas por ámbito o accesibilidad de la siguiente manera: públicas, protegidas, de paquete (sin modificador) privadas.
  - Variables de instancia: organizadas de la misma manera que las estáticas.
  - Al igual que los atributos se debe declarar una por línea para facilitar los comentarios de implementación frente a ellas en caso de necesitarse.
  - Comentario de documentación sobre cada uno de los constructores (no aplica a entidades).

- Declaración de constructores. (Siempre se declarará el constructor sin parámetros, se requiera o no una acción dentro de este. No aplica a entidades).
- Comentario de documentación sobre cada uno de los métodos que lo requieran.
- Declaración de métodos del la lógica del negocio: estos deben ir agrupados por funcionalidad en lugar de por ámbito, siendo el objetivo de esta organización el hacer el código de mas fácil lectura y compresión.
- Declaración de métodos gets() , sets() e is().
- Sobre escritura del método equals().
- Sobre escritura del método hashCode().
- Sobre escritura del método compare().
- Sobre escritura del método compareTo().

Especificaciones sobre el formato del código fuente:

Tabulación:

La unidad de tabulación será de 2 espacios.

Longitud de línea

Se deben evitar líneas de 80 o más caracteres ya que algunas herramientas no las manejan bien, además que líneas demasiado extensas hacen difícil la lectura del código.

Ruptura de líneas

- Cuando una expresión no cabe en una única línea, se debe romper de acuerdo a estos principios generales:
  - Romper después de una coma.
  - Romper antes de un operador.

- Preferir las rupturas de alto nivel a las de bajo nivel. (por ejemplo no romper por operador dentro de paréntesis.
- Alinear la nueva línea con el principio de la expresión al mismo nivel que la línea anterior y dar cuatro tabulaciones.

Declaraciones y sentencias

Sentencias package e imports:

Deben seguir las siguientes reglas de formato.

- Estas sentencias no van tabuladas.
- Se debe declarar una por línea.
- No utilizar comodines.

Variables locales:

Todas las variables locales deben inicializarse en el sitio en donde se declaran.

Las variables deben declararse al inicio de cada método y no esperar a declararlas hasta su primer uso. La única excepción son las variables de bucles que en Java se pueden declarar dentro de la sentencia for. Apropósito de estas variables, se utilizarán las letras de la i a la z en la medida que se vayan necesitando.

Clases, interfaces y métodos:

En las declaraciones se deben seguir las siguientes reglas de formato:

- Ningún espacio entre el nombre del método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea que la sentencia de declaración.
- La llave de cierre "}" comienza una línea nueva tabulada para coincidir con su sentencia de apertura correspondiente, excepto cuando es un bloque vacío

que la llave de cierre "}" debe aparecer inmediatamente después de la de apertura "{".

- Los métodos están separados por una línea en blanco.
- Las clases e interfaces principales no van tabuladas
- Los bloques de código que pertenecen a una clase o método van tabulados.

Sentencias simples:

Se debe escribir solo una sentencia por línea, y no escribir más de una separadas por punto y coma en la misma línea sin importar lo cortas que puedan ser.

Sentencias compuestas:

Las sentencias compuestas son sentencias que contienen una lista de sentencias encerradas entre llaves "{" y "}" y deben seguir las siguientes reglas de formato.

- Las sentencias internas deben estar tabuladas un nivel más que la sentencia compuesta.
- La llave de apertura debe estar al final de la línea que comienza la sentencia compuesta; la llave de cierre debe estar en una nueva línea y estar tabulada al nivel del principio de la sentencia compuesta.
- Las llaves se usan en todas las sentencias compuestas, incluidas las sentencias únicas, cuando forman parte de una estructura de control, como una sentencia if-else o un bucle for. Esto hace más fácil introducir nuevas sentencias sin provocar errores accidentales al olvidarse añadir las llaves.

Líneas en blanco

Las líneas en blanco mejoran la legibilidad resaltando secciones de código que están relacionadas lógicamente.

- En las siguientes circunstancias, siempre se deben usar dos líneas en blanco:
- Entre secciones de un archivo de código fuente.

- Entre definiciones de clases e interfaces.
- En las siguientes circunstancias, siempre se debería usar una línea en blanco:
- Entre métodos.
- Entre las variables locales de un método y su primera sentencia.
- Antes de un comentario de bloque o de una sola línea.
- Entre las secciones lógicas de un método, para mejorar la legibilidad.

Espacios en blanco:

Los espacios en blanco deberían usarse en las siguientes circunstancias:

- Una palabra reservada seguida por un paréntesis Por ejemplo:

```
while (true) {
        sentencias;
}
```

- En las listas de argumentos, debe haber un espacio después de cada coma.
- Todos los operadores binarios, excepto el operador punto (.) deben estar separados de sus operandos por espacios. Los operadores unarios (incremento ++, decremento --, negativo -) nunca deben estar separados de sus operandos. Por ejemplo:
- Las expresiones de una sentencia for deben estar separadas por espacios en blanco. Por ejemplo:

```
for (expr1; expr2; expr3);
```

- Las conversiones de tipo (cast) deberían estar seguidas de un espacio en blanco. Por ejemplo:

```
variableEntera = (int) variableCadena;
```

Ejemplo de documentación básica de un EJB de Sesión.

```
package co.edu.uis.siocid.Administracion.Mantenimiento.EstadoPrueba;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Remove;
import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import org.hibernate.Criteria;
import org.hibernate.criterion.Restrictions;
import org.jboss.seam.ScopeType;
import org.jboss.seam.annotations.Destroy;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;
import org.jboss.seam.annotations.datamodel.DataModel;
import org.jboss.seam.international.StatusMessages;
import org.jboss.seam.international.StatusMessage.Severity;
import co.edu.uis.excepciones.DSIException;
import co.edu.uis.servicios.FormatoWeb;
import co.edu.uis.siocid.entidades.EstadoPrueba;
/**
 * @autor Silvia Bermudez, Tatiana Tarazona Este EJB Administra la
 *Gestion( Crear , Modificar, eliminar ) de los Estado Prueba
 */
@Stateful
@Scope(ScopeType.CONVERSATION)
@Name("AdministrarEstadoPrueba")
public class AdministrarEstadoPruebaEJB implements AdministarEstadoPrueba
  { @In
  private EntityManager em;
  /**
   Estado toma los valores de 1: editando|2: viendo| 3:eliminando
  */
  private int estado;
  private boolean exito;
  /**
   * tipos de datos objeto
  */
  private EstadoPrueba estadoPrueba;
  /**
   * tipos de datos estructuras
  */
  @DataModel
  private List<EstadoPrueba> estadoPruebas;
  /**
   * Este metodo asigna los valores iniciales que seran usados en la
   *pagina inicial de Estado prueba
  */
  public void asignarEstadoPrueba() throws DSIException {
        this.estadoPrueba = new EstadoPrueba();
    try {
        consultarEstadoPruebas();
    }
```

```
    catch(Exception e) {
         throw new DSIException(e);
    }
  }
  /**
   * Este metodo consulta los estados prueba disponibles en la BD
   */
  public void consultarEstadoPruebas() throws DSIException {
            estadoPruebas = new ArrayList<EstadoPrueba>();
    try {
         String jpql = "FROM EstadoPrueba";
         Query query = em.createQuery(jpql);
         this.estadoPruebas = query.getResultList();
    }
    catch(Exception e) {
         throw new DSIException(e);
    }
  }
  /**
   * Este metodo Crea un nuevo estadoPrueba, previa validacion del nombre
   *para no crear uno repetido
   */
  public void crearEstadoPrueba() throws DSIException {
            EstadoPrueba aEstadoPrueba = null;
  /**
   * Este segmento hace la Verificacion en la BD
   */
 org.hibernate.Session
 sesion=(org.hibernate.Session)this.em.getDelegate();
 Criteria criterios = sesion.createCriteria(EstadoPrueba.class);
 criterios.add(Restrictions.eq("nombreEstadoPrueba",
 estadoPrueba.getNombreEstadoPrueba()));
 aEstadoPrueba= (EstadoPrueba)criterios.uniqueResult();
 try {
     if(aEstadoPrueba == null) {
     em.persist(this.estadoPrueba);
     em.flush();
     StatusMessages.instance().addToControl("mensajesValidacion",
     Severity.INFO,FormatoWeb.get("creacionExitosa", true));
      }
     else {
     StatusMessages.instance().addToControl("mensajesValidacion",
     Severity.INFO,FormatoWeb.get("falloCreacion", false));
      }
 }
  catch(Exception e) {
      throw new DSIException(e.getMessage());
  }
}
/**
 * Este metodo permite editar el estado prueba validando que no exista un
 *registro igual.
*/
public void editarEstadoPrueba() throws DSIException{
```

```
List<EstadoPrueba> consulta;
String jpql= "SELECT p                                                    ";
       jpql+="FROM EstadoPrueba p                                         ";
       jpql+="WHERE p.codigoEstadoPrueba<>:codigoEstadoPrueba  ";
       jpql+="AND p.nombreEstadoPrueba=:nombreEstadoPrueba     ";
Query query= em.createQuery(jpql).setParameter("codigoEstadoPrueba ",
this.estadoPrueba.getCodigoEstadoPrueba()).setParameter("nombreEstadoPrue
ba", this.estadoPrueba.getNombreEstadoPrueba());
consulta = query.getResultList();
this.exito=true;
try {
    if(consulta.isEmpty()){
    this.em.merge(estadoPrueba);
    this.em.flush();
    StatusMessages.instance().addToControl("mensajesValidacion",
    Severity.INFO,FormatoWeb.get("edicionExitosa", true));
     }
    else {
         this.exito=false;
        StatusMessages.instance().addToControl("mensajesValidacion",
        Severity.INFO,FormatoWeb.get("falloCreacion", false));
         estadoPruebas = new ArrayList<EstadoPrueba>();
         }
 }
 catch (Exception e) {
       this.exito=false;
       throw new DSIException(e);
 }
}
/**
 * Este metodo permite editar o eliminar el registro de Estado Prueba
*/
 public void modificarEstadoPrueba(EstadoPrueba aEstadoPrueba,int
 estado)throws DSIException{
   this.estadoPrueba=aEstadoPrueba;
   setEstado(estado);
   if(estado==3){
      StatusMessages.instance().addToControl("mensajesValidacion",
      Severity.INFO,FormatoWeb.get("seguroBorrar", false));
    }
 }
 /**
  * Este metodo permite eliminar el estado Prueba
 */
 public void eliminarEstadoPrueba()throws DSIException{
   this.exito=true;
   try {
       this.em.remove(this.em.merge(estadoPrueba));
       this.em.flush();
       StatusMessages.instance().addToControl("mensajesValidacion",
       Severity.INFO,FormatoWeb.get("eliminacionExitosa", true));
   }
   catch(Exception e) {
        this.exito=false;
```

```
            throw new DSIException(e);
        }
    }
    /**
     *Este metodo refresca y reinicializa el objeto estado Prueba
    */
    public void cancelarEstadoPrueba() {
        this.em.refresh(estadoPrueba);
        this.estadoPrueba = new EstadoPrueba();
    }
    /**
     * GETTERS AND SETTERS VBLES PRIMITIVAS
    */
    public int getEstado() {
        return estado;
    }
    public void setEstado(int estado) {
        this.estado = estado;
    }
    public boolean isExito() {
        return exito;
    }
    public void setExito(boolean exito) {
        this.exito = exito;
    }
    /**
     * GETTERS AND SETTERS OBJETOS
    */
    public EstadoPrueba getEstadoPrueba() {
        return estadoPrueba;
     }
    public void setEstadoPrueba(EstadoPrueba estadoPrueba) {
        this.estadoPrueba = estadoPrueba;
    }
   /**
     * GETTERS AND SETTERS ESTRUCTURAS
    */
    public List<EstadoPrueba> getEstadoPruebas() {
        return estadoPruebas;
    }
    public void setEstadoPruebas(List<EstadoPrueba> estadoPruebas) {
        this.estadoPruebas = estadoPruebas;
    }
    public EntityManager getEm() {
        return em;
    }
    public void setEm(EntityManager em) {
        this.em = em;
    }
    @Remove
    @Destroy
    public void destroy(){
    }
}
```

5. CONCLUSIONES

- Con la creación del sistema de información para la oficina de Control Interno Disciplinario, se podrá manejar de manera más ágil y controlada, la información de los diferentes procesos que llegan a esta dependencia y se podrá consultar en tiempo real todas las actividades que están siendo programadas, además de que todos los documentos que se realicen estarán online, con la opción de imprimir si es el caso.

- El sistema de información creado cumple con todos los requisitos que fueron estipulados en el análisis de requerimientos que se hizo con los usuarios finales mediante reuniones periódicas.

- El sistema de Información cumple además con los estándares, lineamientos y directrices establecidos por la División de Servicios de Información para todos los sistemas que se desarrollan bajo su supervisión.

- Con la implementación del nuevo Sistema de Seguridad que se desarrolló en la División de Servicios de Información, dentro de este sistema, se garantiza que el acceso al mismo solo será para aquellos usuarios que estén autorizados por parte del personal de la oficina de Control Interno Disciplinario.

- El uso de herramientas para la creación de prototipos no funcionales, permitieron que los usuarios finales pudieran interactuar con el sistema, realizando las correcciones que ellos consideraban necesarias en el

momento oportuno, ahorrando tiempo en la programación definitiva del sistema, logrando una mayor satisfacción y obteniendo resultados más acordes a sus necesidades.

- Con la realización de este proyecto adquirimos conocimientos de la tecnología Java 5; así mismo se aplicaron los conocimientos aprendidos durante la carrera en relación con la obtención del levantamiento de requerimientos, diseño, desarrollo y pruebas que se deben tener en cuenta para la creación de un sistema.

6. RECOMENDACIONES

- Asignar un funcionario de la División de Servicios de Información de la Universidad Industrial de Santander que pueda dar soporte permanente a los usuarios del sistema y al mantenimiento al mismo.

- Incentivar al personal de la oficina de Control Interno Disciplinario para que use el cargue de archivos soporte escaneados de manera más rigurosa, para tener un seguimiento total de todo el proceso, ya que esto haría que en caso de extraviarse algún documento, se tenga soporte en el sistema de todos los documentos que se tienen relacionados dentro de un proceso investigativo.

- Promover el uso de los prototipos no funcionales para posteriores desarrollos teniendo en cuenta los buenos resultados obtenidos en este proyecto. Además hace que la fase de levantamiento de requisitos y diseño sea más comprendida tanto por el cliente como por el equipo desarrollador.

7. BIBLIOGRAFÍA

COBOS, Carlos Alberto; MENDOZA, Martha Eliana. Manual De Informix – SQL. Universidad Industrial de Santander, 1998.

FERRÉ GRAU, Xavier - SÁNCHEZ S. Maria Isabel. Desarrollo Orientado a Objetos con UML. Facultad de Informática - UPM

GROFF, James R. – WEINBERG, Paul N. APLIQUE SQL. Osborne- McGraw-Hill, 1991.

PRESSMAN, Roger. Ingeniería del Software. Un enfoque práctico. Quinta Edición. McGraw Hill. España. 2005.

SITIOS WEB

- APACHE-TOMCAT, http://jakarta.apache.org/tomcat
- MYSQL, www.mysql.com
- ORACLE,http://download.oracle.com/docs/cd/E17802_01/j2ee/javaee/javaserverfaces/2.0/docs/pdldocs/facelets/index.html
- PROGRAMACION JAVA,www.programacion.com/java/tutorial.
- RICHFACES,http://livedemo.exadel.com/richfaces-demo/richfaces/comboBox.jsf?tab=usage&cid=170149
- www.astalaweb.com.
- www.sparxsystems.com.ar/