

**Universidad de las Ciencias**

**Informáticas**

**Facultad 4**

**TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE  
INGENIERO EN CIENCIAS INFORMÁTICAS**

***“Sistema de Gestión de Información del Área de  
Servicios de la Facultad 4”.***

**Autores:**

Jessica Contreras Rodríguez  
Carlos Manuel Padilla Rodríguez

**Tutores:**

Ing. Yasirys Terry González  
MsC. Rosalba Carralero Medina

“La Habana, 2015”.  
“Año 57 de la Revolución”.

## Dedicatoria

*Jessica:*

*Dedico este trabajo a esas personitas que siempre han estado a mi lado desde que nací, brindándome todo su amor y cariño, quienes apoyaron cada una de mis decisiones, quienes vivieron mis tristezas y alegrías, quienes nunca me dejaron caer y siempre fueron un ejemplo a seguir con sus enseñanzas y exigencias. A ustedes, por creer siempre*

*en mí, todo mi corazón, gracias:*

*Mamá y Papá, Mima y Abuelo.*

## Agradecimientos

*Jessica:*

*Estar agradecido y no expresarlo, es como envolver un regalo y no darlo, por lo que aprovecho para agradecerle:*

*A mis padres por su apoyo incondicional, por haber confiado en mí, por todo su amor y cariño. Este título es para ustedes. Los amo.*

*A mis abuelos por estar siempre junto a mí y demostrarme que sí se puede.*

*A mi hermanita, que a pesar de nuestras interminables peleas, no sé qué sería de mí sin ella. Espero ser su ejemplo a seguir. Te quiero mucho.*

*A mis primos-hermanos Raybe y Jenny por darme fuerzas para seguir adelante y mostrarme siempre el lado positivo de las cosas.*

*A mis tías Níurka y Raquel por ser mi locura más cuerda, mis heroínas, por enseñarme a no depender de nadie, a que si quiero algo, es solo ir a por ello.*

*A mi sobrinito y a mi ahijado que solo con su presencia me alegraban el día, espero ser su, "cuando yo sea grande quiero ser como mi tía o como mi madrina".*

*A mi amigo, Alexis, que él sabe que es más que un amigo, gracias por estar siempre a mi lado y por nunca soltarme la mano ni cuando estaba insoportable.*

*A mi compañero de tesis. Gracias.*

*A mis amigos Yosle, Danay, Claudio que nunca me dejaron estudiar sola, quienes me acompañaban y apoyaban en cada una de mis locuras. A mi Albertico, aún no me he ido y ya me estás haciendo falta.*

*A Iván, a mis compañeros de aula, en especial a Oniel, con quienes compartí momentos inolvidables.*

*A mis tutores por toda la paciencia que se requiere para tratar conmigo, por todo su tiempo y dedicación, luchando siempre por obtener buenos resultados. A Yasirys, gracias por convertirse en más que una tutora, una amiga.*

*A todas esas personas que de una forma u otra contribuyeron a mi formación como profesional.*

## Dedicatoria

*Carlos Manuel*

*Quiero dedicar la realización de este proyecto especialmente a mi abuelo Pancho y espero que haya sentido en vida lo especial que era para mí, a mi abuela Elia que sé, estaría muy contenta de poder verme cumplir mi sueño.*

## Agradecimientos

*Carlos Manuel*


*Siento que debo agradecer especialmente a mi abuelo Francisco que aun después de no estar a mi lado su recuerdo me sigue dando la fuerza necesaria para continuar con mi proyecto de vida y tratar de ser tan especial para mi familia como lo fue él.*

*Son muchas las personas a la cuales les debo agradecer por haberme ayudado tanto durante todo este tiempo, primero a mis padres por estar siempre apoyándome y por ser mis mejores amigos, a mi hermanita preciosa que tanto le insistió a mi madre para que le diera un hermano, a mi cuñado Elvito, a mis tíos Felipe y Wilson, a mi tía Vilma y en general a toda mi familia.*

*Quiero agradecer también a mi novia Dayany por su paciencia y por su apoyo durante los últimos 4 años, a sus padres Neisi y Lázaro los cuales me han hecho sentir como un hijo más y agradecer también a toda su familia, a mis amigos Lisandra Roger, Pavel, Anelis, Kelvis, El Moro, Lisniel, Joan, El Pina, Javier Acosta, Alexis Borges, Alberto Torres, Armando Ávila, Daniel, Arai, Raúl, Luis, Bladimir, Malidia, Silfredo, Nolberto, Yaniel, Yosleidy, Oniel, Yoandris, y todos los demás que siempre me han ayudado, a todos mis profesores durante esta larga travesía especialmente a Misbel mi profesora de historia de quinto grado que siempre confió en mí.*

*Agradezco a mis tutoras Yasirita y Rosalba y a mi compañera de tesis Jessica que tanto trabajo pasaron conmigo.*

*Por ultimo quiero agradecer a la Revolución por darme la oportunidad de realizar este sueño tanpreciado.*



*“Si no existe la organización, las ideas, después del primer momento de impulso, van perdiendo eficacia”.*

## **Declaración de autoría**

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estime pertinente con el mismo.

Para que así conste firmamos la presente a los días \_\_\_\_ del mes de \_\_\_\_\_ del año\_\_\_\_\_.

\_\_\_\_\_  
Jessica Contreras Rodríguez

Autor(es)

\_\_\_\_\_  
Carlos M. Padilla Rodríguez

\_\_\_\_\_  
Ing. Yasirys Terry González

Tutor (es)

\_\_\_\_\_  
MsC. Rosalba Carralero Medina

## **Resumen**

El Sistema de Gestión de Información del Área de Servicios tiene como objetivo permitir el almacenamiento, de forma centralizada y segura, de la información que se genera en el área de servicios administrativos de la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI). Durante el progreso de la investigación se justifican las principales características que debe poseer el sistema, cuyo proceso de desarrollo fue guiado por la metodología ágil, Programación Extrema. Como lenguajes de programación se utilizan Groovy y Java, Grails como marco de desarrollo web, IntelliJ IDEA como entorno de desarrollo y como servidor de aplicaciones, Apache Tomcat (7.0), Visual Paradigm (8.0) como herramienta CASE y PostgreSQL (9.1) como servidor de base de datos. Una vez desarrollado el sistema se efectuaron las pruebas de software a la solución propuesta, las cuales certifican la calidad de las funcionalidades y la satisfacción por parte del cliente, solucionando de esta manera el problema planteado y garantizando el cumplimiento de los objetivos propuestos.



**Índice de ilustraciones**

Ilustración 1 Modelo conceptual .....	27
Ilustración 2 Patrón Modelo-Vista-Controlador .....	40
Ilustración 3 Representación del patrón arquitectónico en la propuesta de solución ...	41
Ilustración 4 Patrón Decorator en Grails .....	43
Ilustración 5 Diagrama Entidad Relación .....	43
Ilustración 6 Interfaz "Adicionar local" .....	45
Ilustración 7 Interfaz "Adicionar evaluación" .....	46
Ilustración 8 Interfaz "Listar turno" .....	46
Ilustración 9 Interfaz "Autenticarse" .....	47
Ilustración 10 Diagrama de Despliegue .....	54
Ilustración 11 Resultado del testNullable de la clase Asignatura en el campo nombre	57
Ilustración 12 Resultado del testUnique de la clase Asignatura en el campo nombre..	57

## **Índice de tablas**

Tabla 1 Requisitos funcionales y no funcionales .....	29
Tabla 2 Usuarios del sistema .....	32
Tabla 3 HU1. Gestionar persona .....	32
Tabla 4 HU4. Gestionar turnos .....	33
Tabla 5 HU6. Gestionar incidencias .....	34
Tabla 6 HU2. Gestionar evaluaciones .....	35
Tabla 7 Planificación de las iteraciones .....	36
Tabla 8 Plan de entregas .....	37
Tabla 9 Funcionalidades disponibles para la entrega del producto.....	37
Tabla 10 Tarjeta CRC PersonaController .....	38
Tabla 11 Tarjeta CRC TurnoController .....	38
Tabla 12 Tarjeta CRC IncidenciaController .....	38
Tabla 13 Tarjeta CRC EvaluacionController .....	39
Tabla 14 Descripción de las tablas que conforman la base de datos .....	44
Tabla 15 Tarea de ingeniería # 1.....	50
Tabla 16 Tarea de ingeniería # 2.....	50
Tabla 17 Tarea de ingeniería # 3.....	51
Tabla 18 Tarea de ingeniería # 4.....	51
Tabla 19 Tarea de ingeniería #5.....	52
Tabla 20 Tarea de ingeniería #6.....	52
Tabla 21 Tarea de ingeniería #7.....	52
Tabla 22 Tarea de ingeniería #8.....	53
Tabla 23 Ambiente de pruebas.....	56
Tabla 24 Cantidad de no conformidades detectadas por iteración.....	56
Tabla 25 HU.8 Gestionar reporte.....	66
Tabla 26 HU5. Autenticar usuario.....	66
Tabla 27 HU7. Gestionar local.....	67
Tabla 28 HU3. Gestionar usuario.....	67
Tabla 29 HU 9 Exportar.....	68
Tabla 30 HU 10 Listar.....	68
Tabla 31 Tareas de ingeniería #8.....	69
Tabla 32 Tareas de ingeniería #9.....	69

Tabla 33 Tareas de ingeniería #10.....	70
Tabla 34 Tareas de ingeniería #11.....	70
Tabla 35 Tareas de ingeniería #12.....	71
Tabla 36 Caso de prueba de aceptación #1.....	71
Tabla 37 Caso de prueba de aceptación #2.....	72
Tabla 38 Caso de prueba de aceptación #3.....	73
Tabla 39 Caso de prueba de aceptación #4.....	73
Tabla 40 Caso de prueba de aceptación #5.....	74
Tabla 41 Caso de prueba de aceptación #6.....	74
Tabla 42 Caso de prueba de aceptación #7.....	75
Tabla 43 Caso de prueba de aceptación #9.....	76
Tabla 44 Caso de prueba de aceptación #10.....	76
Tabla 45 Caso de prueba de aceptación #11.....	77
Tabla 46 Caso de prueba de aceptación #12.....	78
Tabla 47 Caso de prueba de aceptación #13.....	78
Tabla 48 Tarjeta CRC ReporteController.....	79
Tabla 49 Tarjeta CRC LocalController.....	79
Tabla 50 Tarjeta CRC DireccionController.....	79

## **Índice**

<b>INTRODUCCIÓN</b> .....	1
<b>CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA</b> .....	6
1.1 Introducción.....	6
1.2 Conceptos relacionados con la investigación .....	6
1.3 Sistemas similares .....	7
1.3.1 Valoración de los sistemas existentes .....	8
1.4 Metodología, lenguajes de programación, marco de trabajo, y gestor de bases de datos. ....	8
1.4.1 Metodología .....	8
1.4.2 Lenguajes y herramienta CASE.....	15
1.4.3 Marco de trabajo .....	16
1.4.4 Lenguajes de Programación.....	19
1.4.5 Entorno de Desarrollo Integrado.....	22
1.4.6 Sistema Gestor de Bases de Datos (SGBD) .....	23
Conclusiones parciales .....	25
<b>CAPÍTULO 2. EXPLORACIÓN Y PLANIFICACIÓN</b> .....	26
2.1 Introducción.....	26
2.2 Gestión de los procesos del área de servicios administrativos de la Facultad 4. ...	26
2.3 Modelo conceptual .....	27
2.3.1 Descripción de los conceptos del modelo.....	27
2.4 Requisitos .....	29
2.5 Usuarios del sistema .....	32
2.6 Historia de Usuario (HU) .....	32
2.6.1 Historias de Usuario de la propuesta de solución .....	32
2.7 Planificación .....	35

2.7.1 Plan de Iteraciones.....	36
2.7.2 Plan de entregas .....	37
2.8 Tarjetas CRC (Clase-Responsabilidad-Colaborador) .....	38
2.9 Patrones de diseño y de arquitectura .....	39
2.9.1 Patrón arquitectónico.....	39
2.9.2 Patrones de diseño.....	41
2.10 Diagrama Entidad Relación (DER) .....	43
2.11 Interfaces del sistema.....	45
Conclusiones parciales .....	48
<b>CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA .....</b>	<b>49</b>
3.1 Introducción.....	49
3.2 Implementación .....	49
3.2.1 Programación en pareja .....	49
3.2.2 Tareas de ingeniería.....	50
3.3 Diagrama de despliegue.....	53
3.4 Tratamiento de errores .....	54
3.5 Pruebas.....	55
3.5.1 Resultados de las pruebas .....	56
Conclusiones parciales .....	58
<b>CONCLUSIONES .....</b>	<b>59</b>
<b>RECOMENDACIONES .....</b>	<b>60</b>
<b>REFERENCIAS BIBLIOGRÁFICAS .....</b>	<b>61</b>
<b>ANEXOS.....</b>	<b>65</b>
Anexo 1 Historias de usuario.....	65
Anexo 2 Tareas de ingeniería .....	68
Anexo 3 Casos de pruebas de aceptación .....	70

Anexo 4. Tarjetas CRC ..... 78

## **INTRODUCCIÓN**

Las Tecnologías de la Información y la Comunicación (TIC), son el conjunto de tecnologías desarrolladas para gestionar información y enviarla de un lugar a otro. Incluye las tecnologías para almacenar y recuperar información, enviarla y recibirla y procesarla (1).

El desarrollo de las TIC se ha expandido a todos los sectores de la sociedad. Su uso impulsa todos los campos del conocimiento y provoca una revolución tecnológica que influye en los más disímiles aspectos de la vida cotidiana, la cultura, la economía y los procesos sociales en general; lo que facilita a la mayoría de las organizaciones de tipo empresarial e instituciones, optar por la informatización de sus procesos de trabajo, con el fin de que estos se realicen de forma más rápida y eficiente.

En los Lineamientos de la Política Económica y Social del Partido y la Revolución se plantea que en Cuba se pretende “sostener y desarrollar los resultados alcanzados en el campo de la industria del software y el proceso de informatización de la sociedad...” (2) y “ejecutar inversiones en la industria electrónica y de informática y comunicaciones que permitan mantener lo logrado y su desarrollo, según las posibilidades de la economía del país, con vistas a incrementar las exportaciones, la soberanía tecnológica y los servicios, de acuerdo con las prioridades que se establezcan” (3).

Con el propósito de llevar adelante la producción de software en Cuba, teniendo como base el alto nivel de su capital intelectual, surgieron diferentes alternativas, una de las más importantes fue la creación de la Universidad de las Ciencias Informáticas (UCI), denominada en sus principios "Proyecto Futuro", con dos objetivos: informatizar la sociedad cubana y desarrollar la industria del software para contribuir al desarrollo económico del país.

La UCI cuenta con un modelo flexible de centro docente-productor que le permite formar profesionales altamente calificados y comprometidos con su país, así como producir aplicaciones y servicios informáticos a partir de la vinculación estudio-trabajo, para servir de soporte a la industria cubana de la informática. La universidad ha alcanzado importantes resultados (4):

- Más del 60% de los estudiantes incorporados a proyectos productivos e investigativos, en interés de la informatización de la sociedad cubana sobre diversas líneas de desarrollo.
- Prestación de asistencia técnica y capacitación en el país y en el exterior en diversos proyectos de informatización, formación y entrenamiento de los usuarios y clientes.
- Participación en eventos científicos nacionales e internacionales, destacándose las Cumbres Mundiales de la Sociedad de la Información, convocadas por la Unión Internacional de Telecomunicaciones (UIT), realizadas en Ginebra (2003) y Túnez (2005); y en las diferentes ediciones de las Convenciones y Feria Internacional de Informática; Congresos Internacionales de Universidad y Pedagogía.

La UCI cuenta con varias Facultades y cada una de ellas con un área de servicios administrativos en la que se realizan diferentes procesos como los relacionados con la entrega y recepción de las aulas, salones y teatros, donde se desarrollan actividades docentes e investigativas, con la higienización de los locales del docente y con la atención a los recursos humanos que trabajan en esta área. Cada uno de estos procesos genera documentación manejada por los Auxiliares Generales de Servicio (AGS), el Administrador del docente, el Segundo Administrador y el Vicedecano Administrativo (5) y (6).

Los servicios administrativos de la Facultad 4 de la UCI están encaminados a la protección, organización y mantenimiento de todos los medios básicos del Docente 5; además de todo lo relacionado con la limpieza de las aulas, los salones, las oficinas, los laboratorios y los baños para un mejor funcionamiento del docente en general. Tales características permiten identificar los objetivos que a continuación se mencionan:

- Lograr la mayor calidad y eficacia en el proceso de entrega y recepción de las aulas, salones y teatros.
- Contabilizar diariamente el 100% de los medios bajo la responsabilidad de los AGS, dejando evidencia documental de los resultados.



- Contribuir a minimizar las causas y condiciones que propician la ocurrencia de anomalías contra los bienes del Estado ubicados en los locales.

En el área de servicios administrativos se genera un gran cúmulo de información que se gestiona manualmente, la cual no está almacenada de manera segura y se encuentra en diferentes formatos, lo que dificulta su manejo, además de que se pueda extraviar y ser accedida y modificada por personas no autorizadas.

La situación problemática antes expuesta permite identificar el siguiente **problema a resolver**: ¿Cómo contribuir al almacenamiento, de forma centralizada y segura, de la información que se genera en el área de servicios administrativos de la Facultad 4 de la UCI?

El problema se enmarca en el **objeto de estudio** el proceso de gestión de la información en áreas de servicios administrativos. Se determina para la investigación como **campo de acción** la gestión de la información que se genera en el área de servicios administrativos de la Facultad 4 de la UCI.

Para la solución del problema se plantea como **objetivo general** desarrollar un sistema informático que permita el almacenamiento, de forma centralizada y segura de la información que se genera en el área de servicios administrativos de la Facultad 4 de la UCI.

Para dar cumplimiento a los objetivos específicos se sustenta la siguiente **hipótesis**: el desarrollo de un sistema informático permitirá el almacenamiento, de forma centralizada y segura de la información que se genera en el área de servicios administrativos de la Facultad 4 de la UCI.

Para dar cumplimiento al objetivo planteado se proponen los siguientes **objetivos específicos**:

- Fundamentar los conceptos y características relacionados con los procesos que se desarrollan en el área de servicios administrativos de la Facultad 4 mediante el estudio bibliográfico.

- Seleccionar la metodología y las herramientas adecuadas para el desarrollo del sistema.
- Implementar el sistema informático que cumpla con los requisitos especificados para su adecuado funcionamiento.
- Validar la solución implementada mediante las pruebas de software.

A lo largo del proceso investigativo, se utiliza una serie de métodos teóricos y empíricos, los cuales se describen en los párrafos siguientes:

Dentro de los **métodos teóricos**:

- **Análítico-Sintético:** Se utiliza para analizar los procesos y documentos del área de servicios administrativos, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio y para determinar las características que poseerá la propuesta de solución y definir las tecnologías y herramientas más adecuadas para su desarrollo.
- **Histórico-Lógico:** Se emplea para analizar el desarrollo del software en el transcurso del tiempo y de las posibles herramientas a utilizar para la implementación del sistema.

Dentro de los **métodos empíricos**:

- **Entrevista:** Se utiliza para conocer los aspectos esenciales referentes al área de servicios administrativos de la Facultad 4, así como las características del monitoreo y control de cada uno de los procesos que se efectúan en ella. Se realizarán entrevistas no estructuradas o libres en las que se trabajan con preguntas abiertas, sin un orden preestablecido, adquiriendo características de la conversación (7).

El presente trabajo está estructurado en 3 capítulos, distribuidos de la siguiente forma:

- **Capítulo 1. Fundamentación teórica.** En este capítulo se exponen los elementos teóricos que sustentan el presente trabajo. Se estudian las soluciones que dan respuesta a los problemas similares. Además, se justifica la

selección de las herramientas y las tecnologías adecuadas para el desarrollo de la solución.

- **Capítulo 2. Exploración y Planificación.** En este capítulo se abordan las dos primeras fases de la metodología de desarrollo Programación Extrema: Exploración y Planificación. En la Exploración el cliente define las historias de usuario que son de interés para la entrega de la primera versión del producto, al mismo tiempo que los desarrolladores se familiarizan con las herramientas, tecnologías y prácticas que se emplearán en el proyecto. En la Planificación el cliente establece la prioridad para cada historia de usuario y los desarrolladores realizan una estimación del esfuerzo necesario para el desarrollo de la arquitectura. También contiene una descripción de los patrones de diseño utilizados en la propuesta de solución.
- **Capítulo 3. Implementación y prueba.** En este capítulo se exponen las tareas de ingeniería necesarias para llevar a cabo el proceso de desarrollo y se realiza un análisis del funcionamiento del sistema. Se define la estrategia de pruebas a seguir, empleando los métodos de caja blanca y caja negra, en los niveles de pruebas unitarias y de aceptación y se valida la solución propuesta mediante pruebas de software.

## **CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA**

### **1.1 Introducción**

Se pretende desarrollar un sistema de gestión administrativa, el cual debe proveer información basada en registros técnicos, de las operaciones realizadas en la Facultad con el fin de demostrar sus resultados. En el presente capítulo se realiza un análisis de la información relacionada con el proceso de gestión del área de servicios administrativos de la Facultad 4 de la UCI y se hace un estudio detallado de las herramientas y tecnologías seleccionadas para el desarrollo del sistema.

### **1.2 Conceptos relacionados con la investigación**

Para un buen entendimiento de la investigación, se enunciarán en este epígrafe, varios de los conceptos que se emplearán durante su desarrollo:

**Gestión:** es llevar a cabo diligencias que hacen posible la realización de una operación comercial o de un anhelo cualquiera. Es el conjunto de trámites que se llevan a cabo para resolver un asunto (8) y (9).

**Sistema de gestión:** es un conjunto de reglas y principios relacionados entre sí de forma ordenada, para contribuir a la gestión de procesos generales o específicos de una organización. Proceso continuo, que permite trabajar ordenadamente una idea hasta lograr mejoras y su continuidad (10), (11) y (12).

**Gestión de la información:** se trata de la explotación de la información para la obtención de los objetivos de la entidad. Es la creación, adquisición, procesamiento y difusión de la información (13) y (14).

**Sistemas administrativos:** conjunto de partes independientes que actúan en forma conjunta para llevar a cabo una función determinada, que forma un todo unitario. Los componentes que interactúan entre sí y se encuentran interrelacionados recibe el nombre de sistema. Administrativo, está vinculado a la administración, es el acto de administrar, organizar o gestionar recursos (15) y (16).

### **1.3 Sistemas similares**

Durante el estudio de la presente investigación, se identificaron tres trabajos de diplomas desarrollados en la UCI que presentan cierta similitud con la propuesta de solución. A continuación se detallan los mismos:

En el trabajo de diploma, “Desarrollo del componente Puesto de Trabajo durante la Gestión del Capital Humano”, elaborado en Junio del 2010 en la Facultad 1 se describen distintos procesos entre los que se encuentran organización del trabajo y dentro de este se identificó como un subproceso “definir puesto de trabajo”. Este subproceso permite la caracterización y modificación adecuada de los cargos, para diferenciarlos unos de otros, trayendo como ventaja que la información referente a los mismos esté digitalizada. Este sistema fue desarrollado en el framework Sauxe, que tiene como objetivo fundamental agilizar la construcción y desarrollo de aplicaciones web con mayor calidad, usando las tecnologías DOM y DHTML, los lenguajes JavaScript y PHP, PostgreSQL como gestor de bases de datos y un Servidor web (Apache) (17).

“Sistema de Información para la Gestión del Capital Humano en el Centro de Tecnologías de Gestión de Datos”, desarrollado en la Facultad 6 en Junio del 2011, es otro de los trabajos de diplomas identificados, enfocado a los procesos de la Dirección de Capital Humano. Este sistema se basa en gestionar un grupo de procesos que interactúan entre sí, como son: organización del trabajo, evaluación de desempeño, capacitación y desarrollo, seguridad y salud del trabajo, estimulación moral y material y comunicación institucional. Se utilizó como framework de desarrollo Symfony, ExtJS, Servidor web Apache, PostgreSQL (8.4) como gestor de bases de datos y JavaScript y PHP como lenguaje de programación (18).

El trabajo de diploma “Desarrollo del Módulo de Capital Humano para el Sistema de Gestión de Evaluaciones de Software de la UCI”, desarrollado en Junio del en la Facultad 4 tuvo como objetivo fundamental desarrollar el módulo de gestión de evaluaciones de software del Centro de Calidad para Soluciones Informáticas (CALISOFT). A partir del software implementado se permite gestionar el capital humano que allí labora, las habilidades de cada revisor auditor, evaluar el desempeño

del personal, los cursos de capacitación que se imparten por la entidad, generar reportes de evaluaciones, entre otras funcionalidades específicas. Se utilizaron en este sistema: PHP como lenguaje de programación, Symfony como framework de desarrollo y PostgreSQL (3.8) como gestor de bases de datos (19).

### **1.3.1 Valoración de los sistemas existentes**

Los sistemas anteriormente descritos no pueden ser usados en su totalidad para dar solución al problema que se plantea en la presente investigación porque carecen de funcionalidades con las que debe contar el sistema de gestión del área de servicios administrativos de la Facultad 4, como son: gestión de los estudiantes que realizan las actividades del componente laboral, gestión de los reportes de mantenimiento, gestión de los locales del docente. Sin embargo, varias de las funcionalidades que están implementadas en los sistemas estudiados servirán de guía para el desarrollo de este trabajo, como son: definir puesto de trabajo y registrar datos de la evaluación del desempeño, ya que en la presente investigación se pretende gestionar la valoración sobre el trabajo de los AGS y delimitar el área laboral.

## **1.4 Metodología, lenguajes de programación, marco de trabajo, y gestor de bases de datos.**

### **1.4.1 Metodología**

La metodología es la parte del proceso de investigación o método científico, que sigue a la instrucción, y permite sistematizar los métodos y las técnicas necesarias para llevarla a cabo. Los métodos elegidos por el investigador facilitan el descubrimiento de conocimientos seguros y confiables que, potencialmente, solucionarán los problemas planteados (20) y (21).

Las metodologías tradicionales o no ágiles son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo; donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema (22).

Las metodologías ágiles se basan en un ciclo de vida de desarrollo del software iterativo e incremental, corporativo (cliente y desarrolladores trabajan juntos

constantemente con una cercana comunicación), sencillo y adaptable (permite realizar cambios de último momento) (22).

Se decide trabajar con un enfoque ágil ya que este tiene como prioridad la satisfacción al cliente, la interacción constante entre los desarrolladores y el cliente, se centra en la calidad técnica y en responder a los cambios que puedan surgir a lo largo del proyecto, antes de seguir estrictamente un plan.

Entre las metodologías ágiles se encuentran Scrum, Crystal Methodologies, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Feature-Driven Development (FDD) y Extreme Programming (XP). A continuación se describe con mayor detalle cada una de ellas.

**Scrum:** es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio sobre la manera de trabajar de equipos altamente productivos (23).

Scrum está caracterizado por (23):

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.

En Scrum cada iteración del proyecto tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite (23).

Roles Principales (23):

- Propietario del producto: representa la voz del cliente. Se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio.
- Facilitador: su trabajo primario es eliminar los obstáculos que impiden que el equipo alcance su objetivo, se asegura de que el proceso Scrum se utiliza

como es debido, hace que las reglas se cumplan.

- Equipo de desarrollo. El equipo tiene la responsabilidad de entregar el producto. Un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo.

**Crystal Methodologies:** Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo (24).

Crystal Methodologies permite a los usuarios finales acceder e interactuar con los reportes a través de portales web, dispositivos móviles y documentos de Microsoft Office. Por ello, se recomiendan para el buen desarrollo de proyectos de software, la aplicación de estas metodologías (24).

Características (24):

- Crystal aconseja que el tamaño del equipo sea reducido (pocos componentes).
- La mejora de la comunicación entre los miembros del equipo del proyecto:
  1. Mismo lugar de trabajo, disminuye el costo de la comunicación.
  2. Mejora individual, mejora global del equipo.

Roles (24):

- Patrocinador ejecutivo
- Jefe de proyecto
- Experto en el dominio
- Experto de uso
- Programador diseñador
- Diseñador
- Realizador de pruebas



- Programador técnico

**DSDM:** Es un método que provee un *framework* para el desarrollo ágil de software, apoyado por su continua implicación del usuario en un desarrollo iterativo y creciente que sea sensible a los requerimientos cambiantes, para desarrollar un sistema que reúna las necesidades de la empresa en tiempo y presupuesto. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos (25).

Principios (25):

- Involucrar al cliente es la clave para llevar un proyecto eficiente y efectivo.
- El equipo del proyecto debe tener el poder para tomar decisiones que son importantes para el progreso del proyecto.
- Se centra en la entrega frecuente de productos, asumiendo que entregar algo temprano es siempre mejor que entregar todo al final.
- El principal criterio de aceptación de entregables en DSDM reside en entregar un sistema que satisface las actuales necesidades de negocio.
- Todos los cambios durante el desarrollo son reversibles.
- Las pruebas son realizadas durante todo el ciclo de vida del proyecto.
- La comunicación y cooperación entre todas las partes interesadas en el proyecto es un prerrequisito importante para llevar un proyecto efectivo y eficiente.

**ASD:** Es una metodología de desarrollo de software, que surgió de una metodología de desarrollo rápido para aplicaciones, cuyo principio es adaptarse al cambio en lugar de luchar contra él.

Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda se desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente (26).

Ventajas (26):

- Se utiliza para poder aprender de los errores e iniciar nuevamente el ciclo de desarrollo.
- Utiliza información disponible acerca de todos los cambios para poder mejorar el comportamiento del software.
- Utiliza información disponible acerca de cambios para mejorar el comportamiento del software.
- Promulga colaboración, la interacción de personas.
- Anticipa cambios y trata automáticamente con ellos dentro de un programa en ejecución, sin la necesidad de un programador.

**FDD:** Es un enfoque ágil para el desarrollo de sistemas. Se enfoca en iteraciones cortas que entregan funcionalidades tangibles (27).

Características (28):

- Se preocupa por la calidad, por lo que incluye un monitoreo constante del proyecto.
- Ayuda a contrarrestar situaciones como el exceso en el presupuesto, fallas en el programa o el hecho de entregar menos de lo deseado.
- Propone tener etapas de cierre cada dos semanas. Se obtienen resultados periódicos y tangibles.
- Se basa en un proceso iterativo con iteraciones cortas que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorear.
- Define formas de evaluación del progreso del proyecto.
- No hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción.

Ventajas (28):

- Cada componente del producto final ha sido probado y satisface los requerimientos.
- Rápida respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos cortos de software funcional.

- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Importancia de la simplicidad, al eliminar el trabajo innecesario.
- Atención continua a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.

Desventajas (28):

- Problemas derivados de la comunicación oral. Este tipo de comunicación resulta difícil de preservar cuando pasa el tiempo y está sujeta a muchas ambigüedades.
- Fuerte dependencia de las personas. Como se evita en lo posible la documentación y los diseños convencionales, los proyectos ágiles dependen críticamente de las personas.

**XP:** Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (29).

XP tiene como objetivos establecer las mejores prácticas de Ingeniería de software en los desarrollos de proyectos, mejorar la productividad y garantizar la calidad del software que se está desarrollando, haciendo que este supere las expectativas del cliente (29).

Características (30):

- Metodología liviana de desarrollo de software
- Conjunto de prácticas y reglas empleadas para desarrollar software
- En vez de planificar, analizar y diseñar para el futuro distante, hacer todo esto un poco cada vez, a través de todo el proceso de desarrollo

Ventajas (30):

- Programación organizada
- Menor tasa de errores
- Satisfacción del programador

Desventajas (30):

- Es recomendable emplearlo solo en proyectos a corto plazo
- Altas comisiones en caso de fallar

XP propone los siguientes roles para un equipo de desarrollo (29):

- Programador
- Cliente
- Encargado de pruebas
- Encargado de seguimiento
- Entrenador
- Consultor
- Gestor

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la entrega, Iteraciones, Producción, Mantenimiento y Muerte del proyecto. Sin violar este ciclo, el equipo de desarrollo decidió agrupar las seis fases en cuatro, para un mejor entendimiento y estructuración de la investigación, estas etapas son:

- Etapa I: Exploración
- Etapa II: Planificación
- Etapa III: Implementación
- Etapa IV: Pruebas

A continuación se mencionan las prácticas propuestas por XP, que son la concentración de los principios y valores que fundamentan las metodologías ágiles. (29):

- El juego de planificación
- Entregas pequeñas
- Metáforas
- Diseño simple

- Pruebas
- Refactorización (*Refactoring*)
- Programación en parejas
- Propiedad colectiva del código
- Integración continua
- 40 horas por semana
- Cliente in-situ
- Estándares de programación

Por todo lo antes expuesto, se decidió trabajar con la metodología ágil XP, ya que trae como beneficio un desarrollo incremental del sistema además el proyecto se adapta a los requisitos de esta metodología, como promover el trabajo en equipo, crear un sistema nuevo y necesario para la Facultad 4 y la retroalimentación continua del cliente y los desarrolladores.

#### **1.4.2 Lenguajes y herramienta CASE**

##### **Lenguajes de modelado**

Un lenguaje de modelado es el conjunto de símbolos estandarizados y de modos de utilizarlos para modelar parte de un diseño de software Orientado a Objetos (OO). Generalmente son empleados en combinación con una metodología de desarrollo de software para llegar a una especificación inicial de la implementación (31) y (32).

##### **UML 2.0**

Lenguaje Unificado de Modelado (UML, del inglés *Unified Modeling Language*) es un lenguaje de modelado para visualizar, especificar, construir y documentar un sistema. Incluye aspectos esenciales tales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño OO. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios (33).

Es importante destacar que UML es un lenguaje de modelado para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar sus

artefactos y para documentar y construir. Es el lenguaje en el que está descrito el modelo.

### **Herramienta CASE**

Las herramientas CASE se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Aumentan la productividad en el desarrollo de software reduciendo el costo del mismo en términos de tiempo y de dinero, facilitan la reutilización de componentes, de prototipos y el desarrollo conjunto de aplicaciones. Permiten la aplicación práctica de metodologías estructuradas que agilizan el trabajo (34).

### **Visual Paradigm 8.0**

Visual Paradigm es una herramienta para el desarrollo de aplicaciones informáticas, permite aumentar la calidad del software a través de la mejora de la productividad en el desarrollo y mantenimiento del software (35).

Visual Paradigm ofrece (35):

- Navegación intuitiva entre la escritura del código y su visualización
- Potente generador de informes en formato PDF/HTML
- Ambiente visualmente superior de modelado

Para la elaboración de los diagramas se utilizará Visual Paradigm 8.0, por todo lo anteriormente expuesto, además de ser una herramienta ya conocida por los desarrolladores que permite representar los diagramas necesarios para el análisis y diseño de la propuesta de solución.

### **1.4.3 Marco de trabajo**

Un marco de trabajo (*framework*) en el desarrollo de software, no es más que una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un marco de trabajo representa una arquitectura de software que modela las relaciones generales de las entidades del

dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio (36).

Para el desarrollo del sistema se analizaron los *frameworks*: Ruby on Rails y Grails, los cuales se detallan a continuación.

### **Ruby on Rails**

Ruby on Rails, también conocido como (Rails o RoR) es un entorno de desarrollo de código abierto que está optimizado para la satisfacción de los programadores y para la productividad sostenible. Permite escribir un buen código bajo los paradigmas de desarrollo “convención sobre configuración” y “no te repitas” (37).

- Primer principio: No te repitas (DRY, del inglés *Don't Repeat Yourself*), es uno de los paradigmas más novedosos que se puede encontrar en este *framework*. Esta regla permite tener un formulario y llamarlo las veces que se requiera y desde donde se necesite, simplemente con una línea de código. Permite manipular los registros como un objeto y a sus campos como un atributo en una tabla en la base de datos, sin necesidad de declarar nada.
- Segundo principio: Convención sobre Configuración (*Convention Over Configuration*), es un principio que plantea, que solo debe existir configuración para aquello que no escape de lo convencional. Para lograr esto el *framework* plantea un camino convencional con el cual se evita la configuración, cuando el comportamiento deseado escapa del convencional debe aplicar una configuración especial.

Sus dos piezas principales son Ruby y Rails. Ruby es un lenguaje de programación, dirigido a la simplicidad y a la productividad, totalmente OO; mientras que Rails es un *framework* creado para el desarrollo de aplicaciones web (38).

Características (38):

- Está basado en el patrón arquitectónico Modelo-Vista-Controlador (MVC).
- Es independiente de la base de datos.
- Puede ser extendido mediante el uso de plugins.

### **Grails**

Grails es un *framework* libre para aplicaciones web desarrollado sobre el lenguaje de programación Groovy (el cual a su vez se basa en Java). Grails pretende ser un marco de trabajo altamente productivo siguiendo paradigmas tales como “mejoremos la rueda, no la reiventemos”, “convención sobre configuración” y “no te repitas”, proporcionando un entorno de desarrollo estandarizado y ocultando gran parte de los detalles de configuración al programador (39) y (40).

Grails permite emplear el menor tiempo necesario en tareas repetitivas y presenta una gran capacidad adaptativa ante la presencia de un cambio parcial o total de requisitos, lo cual permite en un tiempo reducido hacer las modificaciones necesarias sin tener que rescribir la aplicación totalmente (39) y (40).

Este *framework* funciona bajo un modelo muy conocido MVC y utiliza una cuarta capa llamada servicio para separar la inteligencia de negocio logrando así un mejor entendimiento a la hora de analizar la aplicación. Grails también ofrece la posibilidad de crear plantillas no solo CSS sino que también incluyen código HTML mediante los *layouts*. Además cuenta con un lenguaje llamado *Groovy Server Pages* (GSP) para poder programar directamente sobre la aplicación web en código HTML (41).

Características (39):

- Ofrece un *framework* web de alta productividad para la plataforma Java.
- Ofrece un *framework* consistente que reduzca la confusión y que sea fácil de aprender.
- Ofrece documentación para las partes del *framework* relevantes para sus usuarios.
- Proporciona lo que los usuarios necesitan en áreas que a menudo son complejas e inconsistentes.
- *Framework* de persistencia potente y consistente.
- Patrones de visualización potente y fácil de usar con GSP.
- Bibliotecas de etiquetas dinámicas para crear fácilmente componentes web.

Se decidió trabajar con el *framework* Grails 2.1.1 porque es un entorno de desarrollo de aplicaciones web nacido en un contexto muy particular: el de las metodologías de desarrollo de software ágiles, donde cada una de estas técnicas tienen una clara orientación a gestionar el cambio; sin embargo, XP establece una forma de trabajo en



la que la comunicación fluida garantiza que si se produce un cambio total o parcial en los requisitos, se podrá adaptar el sistema en un tiempo razonable. Además de que este *framework* permite la reutilización del código, la construcción de soluciones robustas sin poner en riesgo la calidad del producto final y simplifica el desarrollo de sistemas.

#### **1.4.4 Lenguajes de Programación**

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana (42). Para el desarrollo de la solución propuesta se decidió trabajar con los lenguajes de programación Groovy y Java, porque permiten escribir el programa una vez y ejecutarlo en cualquier dispositivo. Además, Java es uno de los lenguajes más populares en uso, particularmente para sistemas de cliente-servidor, por lo que existe abundante documentación. Posteriormente se referencian los lenguajes de programación empleados para el desarrollo de la aplicación.

##### **Groovy**

Groovy es un lenguaje de programación OO que se basa en Java. Tiene características similares a Python, Ruby, Perl y Smalltalk. Es simple de entender y dinámico lo cual hace que sea muy cómodo para los desarrolladores. Además es un lenguaje muy compacto lo cual permite escribir menos código para una misma sintaxis que en Java y de esta manera se convierte en un salto natural para los programadores que desarrollan en Java.

Desde Groovy se puede acceder directamente a todas las Interfaces de Programación de Aplicaciones (API, del inglés *Application Programming Interface*) existentes en Java. El *bytecode* (clase Java) generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto puede usarse directamente en cualquier aplicación Java. Todo lo anterior unido a que la mayor parte de código escrito en Java es totalmente válido en Groovy hacen que este lenguaje sea de muy fácil adopción para programadores Java; la curva

de aprendizaje se reduce mucho en comparación con otros lenguajes que generan *bytecode* para la JVM, tales como Jython o JRuby. Groovy puede usarse también de manera dinámica como un lenguaje de *scripting* (43).

### **Java**

Java es un lenguaje de programación de propósito general, concurrente, OO y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

Java *Enterprise Edition* es un conjunto de componentes que forman una plataforma de programación para el desarrollo y ejecución de aplicaciones desarrolladas en lenguaje de programación Java con arquitectura de N capas distribuidas, que se apoya ampliamente en componentes de software modulares ejecutándose en un servidor de aplicaciones.

La sintaxis del lenguaje de programación Java deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a *bytecode* que puede ejecutarse en cualquier máquina virtual Java sin importar la arquitectura de la computadora subyacente (44).

Características (44):

- La orientación a objeto, se refiere a un método de programación y al diseño del lenguaje.
- La independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware.
- En Java el problema fugas de memoria se evita en gran medida gracias a la recolección de basura. El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (*Java runtime*) es el responsable de gestionar el ciclo de vida de los objetos.

Entornos de funcionamiento:

- En dispositivos móviles y sistemas empujados
- En el navegador web

- En aplicaciones de escritorio
- En sistema servidor

### **Bootstrap**

Bootstrap fue desarrollado como marco de trabajo para fomentar la consistencia a través de herramientas internas, además facilita la construcción de aplicaciones que utilicen el patrón *responsive design* mediante tecnologías CSS3, HTML5 y JavaScript (45) y (46).

Este marco de trabajo permite varias formas de poner un texto en énfasis, otorgando estilos a las abreviaturas y los formatos de negrita y cursiva y alineando los textos.

Algunas de sus características son:

- Permite definir diferentes *layouts*
- Amplio conjunto de componentes para el desarrollo
- Posibilidad de compilar el *framework* con diferentes valores
- Soporte HTML5 y CSS3

La selección de Bootstrap como *framework* se basa en una mejor personalización de la vista de la propuesta solución.

### **JavaScript**

JavaScript es un lenguaje de programación de secuencia de comandos (*scripts*) OO. Estos scripts consisten en funciones que son llamados desde el propio HTML cuando algún evento sucede. JavaScript es actualmente utilizado en internet, junto con las páginas web (HTML o XHTML), está directamente incluido en ellas y mejora una página HTML, añadiendo interacción del usuario, animación, ayudas a la navegación, entre otros. Los programas escritos en JavaScript se pueden probar en cualquier navegador sin necesidad de procesos intermedios (47) y (48).

La selección de JavaScript como lenguaje de script del lado del cliente está basada en que utiliza la librería JQuery, que es empleada por el framework Bootstrap.

### **HTML**

HTML es el lenguaje, de marcas de hipertexto, básico de la web para organizar su contenido, compartir información, crear documentos y aplicaciones que puedan ser utilizadas en cualquier lugar. Es un estándar a cargo de la W3C (*World Wide Web*

*Consortium*), organización dedicada a la estandarización de la mayoría de las tecnologías ligadas a la web (49).

La última versión, HTML5, que provee básicamente tres características: estructura, estilo y funcionalidad, pretende corregir los problemas con los que los desarrolladores web se encuentran, así como rediseñar el código, actualizándolo a las nuevas necesidades que demanda la web. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas (50) y (51).

## **CSS**

La hoja de estilo en cascada es un lenguaje, que trabaja junto con HTML para proveer estilos visuales a los elementos del documento (52). Este lenguaje es un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML (51).

La versión 3 de CSS está dividida en varios documentos separados, llamados “módulos”, cada módulo incorpora nuevas funcionalidades a las definidas en la versión anterior para conservar la compatibilidad. Algunas de estas funcionalidades son:

- Incluye la posibilidad de aplicar sombra a los elementos
- Nuevas alternativas para dibujar bordes
- Capacidad de rotación de elementos
- Opciones de transformación de elementos

CSS3 fue seleccionado como lenguaje para escribir el estilo visual de la propuesta solución porque es un estándar determinado por la W3C, además permite la separación de la estructura del documento de su presentación.

Para la necesaria evolución de las web HTML, CSS y JavaScript se convirtieron en la más perfecta combinación, donde HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista y JavaScript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales (49) y (51).

### **1.4.5 Entorno de Desarrollo Integrado**

Un entorno de desarrollo integrado, (IDE, del inglés *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas

de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (53) .

Se decide utilizar la herramienta IntelliJ IDEA durante el ciclo de desarrollo porque entre los diferentes *frameworks* que soporta, está Grails, además permite la compilación y depuración del código, construcción de interfaz gráfica. También crea métodos estáticos, mueve una clase interna a un nivel superior y reemplaza los códigos de métodos duplicados.

#### **1.4.6 Sistema Gestor de Bases de Datos (SGBD)**

##### **PostgreSQL**

PostgreSQL es un sistema gestor de bases de datos objeto-relacional de carácter libre muy utilizado últimamente por su fácil comprensión. PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos mediante un sistema denominado Acceso Concurrente Multiversión (MVCC, del inglés *Multiversion Concurrency Control*) (54).

Principales características (54):

- Alta concurrencia: Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.
- Amplia variedad de tipos nativos: PostgreSQL provee nativamente soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas (con una variedad de funciones asociadas) y direcciones IP (IPv4 e IPv6). Adicionalmente los usuarios pueden crear sus propios tipos de datos.

Ventajas (54):

- Fácil de administrar
- Su sintaxis SQL es estándar y fácil de aprender
- Multiplataforma

Se estará trabajando con PostgreSQL 9.1 para la confección de la base de datos, pues

los autores ya han interactuado con esta herramienta anteriormente, además permite la interacción con el software de forma simultánea y es fácil de manejar y de administrar.

### **Conclusiones parciales**

El área de servicios administrativos es de gran utilidad para la Facultad 4 pues en ella se concentran todos los procedimientos que hacen posible el buen funcionamiento del docente; a partir de este estudio se realizó un levantamiento de los procesos que en el área se realizan y las ventajas que traería la implementación de un sistema de gestión de información. En el presente capítulo se realizó un análisis de las principales herramientas y tecnologías a utilizar para el desarrollo, además de las principales metodologías de desarrollo de software, justificando el uso de XP, a partir de las características del sistema a desarrollar y del equipo de desarrollo.

## **CAPÍTULO 2. EXPLORACIÓN Y PLANIFICACIÓN**

### **2.1 Introducción**

Un producto de software tiene calidad si su proceso de desarrollo la tiene, por lo que en este capítulo se trabajará siguiendo la filosofía de la metodología XP, la cual construye un proceso de diseño evolutivo el cual se basa en obtener un sistema simple al final de cada iteración. En este capítulo se determinan las iteraciones desarrolladas durante la etapa de construcción del sistema definiéndose las Historias de Usuario (HU) para especificar los requisitos del software, las tarjetas CRC (Clase-Responsabilidad-Colaborador) para la representación de los sistemas OO y los patrones arquitectónicos y de diseño de la propuesta solución.

### **2.2 Gestión de los procesos del área de servicios administrativos de la Facultad 4.**

En el área de servicios administrativos de la Facultad 4 se realizan una serie de procedimientos, a continuación se detallan algunos.

El registro de las incidencias es uno de los procesos que se realizan en el área. En estas se almacenan los certificados médicos que presentan los trabajadores, sus ausencias a los turnos laborales junto a la causa que las provoca.

En el área también se efectúa un control a los estudiantes que realizan el Trabajo Socialmente Útil (TSU) y las actividades del componente laboral, de los últimos, se registra la cantidad de asignaturas que cursan y dependiendo de estas y del horario en que las reciben, se les asigna un turno y un área para efectuar el trabajo; al concluir el primer semestre se les otorga una evaluación parcial y al culminar el curso, la final. A los estudiantes del TSU, se les comunican las tareas que deben desarrollar y el horario de su jornada laboral y posteriormente se les asigna su evaluación final.

El control de reportes es otro proceso que se realiza en el área, de cada reporte se almacena su número, la descripción y local de la rotura o afectación, el lugar a donde se reporta, la fecha y hora en que se efectúa el mismo. También se guardan los datos de quién recibe el reporte y de quién lo reporta junto con la fecha y hora de restablecimiento de la rotura o afectación.



El monitoreo de los datos de las personas (trabajadores y estudiantes) que laboran en el área es otro de los procesos que se realiza. De cada uno se registra su nombre y apellidos, carnet de identidad, solapín, número de expediente, número telefónico y padecimientos; si la persona es un trabajador se le especifica además su dirección, si por el contrario, es un estudiante se verifica la clasificación y a partir de esta se detallan otros datos de interés.

### 2.3 Modelo conceptual

La metodología XP no requiere de un modelo conceptual, pero los autores de la presente investigación consideran que resulta valioso para descomponer el dominio del problema la identificación de los conceptos, los atributos y las asociaciones del dominio que se juzgan importantes (55). El resultado se expresa en un modelo conceptual, el cual se presenta a continuación, que describe y representa los principales conceptos que se manipulan en el área de servicios administrativos de la Facultad 4.

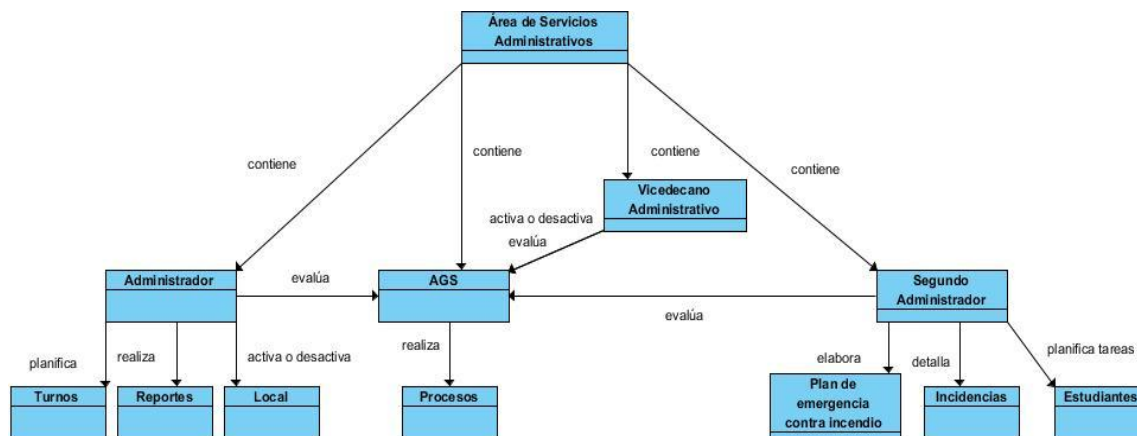


Ilustración 1 Modelo conceptual

#### 2.3.1 Descripción de los conceptos del modelo

- Área de servicios administrativos: Representa el área de trabajo de los AGS, el Administrador, el Segundo Administrador y el Vicedecano Administrativo de la Facultad 4.
- Segundo Administrador: Es el responsable de notificar las incidencias que

ocurren en el docente, planificar las tareas y horario de trabajo de los estudiantes que realizarán el TSU y las actividades del componente laboral, elaborar el Plan de emergencia contra incendio y evaluar frecuentemente a las AGS.

- AGS: Son los encargados de realizar varios de los procesos que se realizan en el área de servicios administrativos.
- Vicedecano Administrativo: Se encarga de otorgar una evaluación frecuente, mensual y anual a los trabajadores del área de servicios y monitorea los datos de sus subordinados.
- Administrador: Es el responsable de asignar un turno laboral a cada AGS, realiza los reportes de mantenimiento del área, activa o desactiva los locales del docente y evalúa frecuentemente a sus subordinados.
- Turnos: Se refiere a los turnos laborales de cada trabajador del área de servicios administrativos.
- Procesos: Describe los procedimientos que se realizan en el área de servicios administrativos, como el control de los estudiantes que realizan las actividades del componente laboral y TSU, control de reportes, locales y turnos laborales.
- Reportes: Detalla los reportes de mantenimiento elaborados por el Administrador.
- Local: Especifica los locales del Docente 5.
- Estudiantes: Se refiere a los estudiantes que estarán realizando las actividades del componente laboral y TSU.
- Plan de emergencia contra incendio: Contiene el inventario de los medios de protección, la cantidad de extintores que hay disponibles en el docente, su clasificación y estado, las medidas de prevención de incendios y los riesgos más significativos en las condiciones y características del Docente 5. En el plan también se registran los datos de las personas que conforman la brigada contra incendios y si están certificados o no.
- Incidencias: Se refiere a las incidencias que impiden que los trabajadores

cumplan con su jornada laboral.

## 2.4 Requisitos

A continuación se muestran los requisitos funcionales y no funcionales del sistema:

**Tabla 1 Requisitos funcionales y no funcionales**

No.	Requisitos funcionales
1.	Gestionar persona
<p>El sistema debe permitir adicionar, editar, consultar y eliminar los datos de la persona que se registren en él, ya sea un trabajador o estudiante. De cada uno se registra su nombre y apellidos, carnet de identidad, solapín, padecimientos, número de expediente, número telefónico y turno laboral. Si es un trabajador, se le especifica además, su dirección; si por el contrario es un estudiante que realiza las actividades del componente laboral, se detallan las asignaturas que está cursando, el grupo, el curso escolar y las horas a cumplir y a los del TSU se les define la fecha de inicio y fecha fin.</p>	
2.	Gestionar evaluación
<p>El sistema debe permitir adicionar, editar, consultar y eliminar las evaluaciones de los trabajadores y estudiantes que realizan las actividades del componente laboral y las del TSU. De cada una se registra la evaluación otorgada, las observaciones junto con las evidencias que conllevaron a ese resultado y la fecha en que se asignó.</p>	
3.	Gestionar usuario
<p>El sistema debe permitir crear, editar y eliminar usuario. De cada usuario se registrará su nombre y apellidos y rol que desempeña.</p>	
4.	Gestionar turno
<p>El sistema debe permitir adicionar, editar, consultar y eliminar los turnos laborales. De</p>	

cada turno se almacenarán los trabajadores involucrados, junto con el responsable, la hora de entrada y de salida, la fecha y las observaciones.	
5.	Autenticar usuario
El sistema debe permitir que un usuario se autentique insertando su identificador y contraseña. Este usuario debe tener previa autorización del Vicedecano Administrativo para acceder al sistema.	
6.	Gestionar incidencia
El sistema debe permitir adicionar, editar, consultar y eliminar las incidencias. Estas se refieren a las causas que impiden a los trabajadores cumplir con su jornada laboral.	
7.	Gestionar local
El sistema debe permitir adicionar, editar, consultar y eliminar los locales del Docente 5. De cada local se almacena su nombre y estado (activo o inactivo).	
8.	Gestionar reporte
El sistema debe permitir adicionar, editar, consultar y eliminar los reportes que se originan en el área. De cada reporte se almacenará su número, descripción de la rotura o afectación, local donde se localiza, fecha y hora de elaboración y de restablecimiento de la rotura, quién lo recibe y quién lo reporta y lugar donde se entrega.	
9.	Exportar
El sistema debe permitir exportar en formato pdf y en exe los listados que se generan en el sistema. Estos son: lista de personas, lista de evaluaciones, lista de turnos, lista de reportes, lista de incidencias y lista de locales.	
10.	Listar

El sistema debe permitir listar la información generada a partir de la gestión de una persona, una evaluación, un turno, un local, un reporte y una incidencia.	
<b>Requisitos no funcionales</b>	
1	<b>Rendimiento:</b> El sistema debe tener una velocidad de procesamiento óptimo y un buen aprovechamiento de los recursos, debe tardarse en procesar la información y en dar una respuesta no más de 5 segundos.
2	<b>Disponibilidad:</b> El sistema deberá tener una disponibilidad total para ser usado durante toda la jornada laboral, permitiendo que los usuarios previamente definidos puedan acceder a la información y recursos cuando lo necesiten.
3	<b>Soporte:</b> En caso de cambios en el negocio, el sistema debe permitir posteriores modificaciones y actualizaciones.
4	<b>Software:</b> Para las computadoras cliente: Sistema operativo: Windows, Linux. Navegador: Mozilla Firefox: 35. 0 superior.  Para los servidores: Sistema operativo: Windows, Linux. Servidor web Apache 7.0 o superior. PostgreSQL versión 9.1 o superior como gestor de base de datos.
5	<b>Hardware:</b> Para las computadoras cliente: 512 MB de Memoria de Acceso Aleatorio (RAM, del inglés <i>Random-Access Memory</i> ) o superior, 80 GB de disco duro o superior y velocidad del procesador 1.86 GHz o superior.  Para los servidores: 1 GB de RAM o superior, 150 GB de disco duro o superior y velocidad del procesador 2.0 GHz o superior.
6	<b>Seguridad:</b> Garantizará que el sistema muestre las funcionalidades de acuerdo al nivel de acceso del usuario activo y lo protegerá contra acciones que puedan afectar la integridad de los datos.

## 2.5 Usuarios del sistema

Los usuarios del sistema son las roles que interactúan directamente con la aplicación, a continuación se describen los mismos con sus responsabilidades.

**Tabla 2 Usuarios del sistema**

<b>Usuarios del sistema</b>	<b>Descripción</b>
Administrador	Es el encargado de gestionar los turnos laborales y los datos de un trabajador, hacer los reportes de mantenimiento, activar o desactivar los locales del Docente 5 y en conjunto con el Segundo Administrador, otorga evaluaciones frecuentes a sus subordinados.
Segundo Administrador	Es el encargado de gestionar las incidencias que ocurren en el área y los datos de los estudiantes que realizan las actividades del TSU y componente laboral.
Vicedecano Administrativo	Tiene la responsabilidad de asignar una evaluación frecuente, mensual y anual a los trabajadores del área, activa y desactiva los mismos y gestiona los usuarios del sistema.

## 2.6 Historia de Usuario (HU)

El cliente describe las características que el sistema debe poseer a través de las HU, que son empleadas por la metodología XP como técnica para especificar tanto sus requisitos funcionales como los no funcionales (56).

### 2.6.1 Historias de Usuario de la propuesta de solución

Fueron detalladas por parte del cliente un total de 10 HU, en este epígrafe sólo se exponen las principales, las restantes se encuentran en el Anexo 1 de la investigación.

**Tabla 3 HU1. Gestionar persona**

<b>Historia de Usuario</b>
----------------------------

<b>Número:</b> 1	<b>Nombre del requisito:</b> Gestionar persona
<b>Programador:</b> Jessica Contreras Rodríguez	<b>Iteración asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 5 días
<b>Riesgo en desarrollo:</b>	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá adicionar, editar, listar y eliminar los datos de un trabajador y de los estudiantes que realizan las actividades del componente laboral y del TSU.	
<b>Observaciones:</b> El usuario debe estar autenticado con el rol Vicedecano Administrativo o Segundo Administrador y el sistema les indicará los campos que son obligatorios para gestionar los datos de un trabajador y los de un estudiante. El Vicedecano Administrativo, el Administrador y el Segundo Administrador, podrán consultar los datos de la persona una vez creado el mismo. Se podrá generar un listado con los datos de los trabajadores, en formato pdf y exe así como también, el sistema permitirá subir las evidencias referentes a los certificados médicos y a las evaluaciones.	

**Tabla 4 HU4. Gestionar turnos**

Historia de Usuario	
<b>Número:</b> 4	<b>Nombre del requisito:</b> Gestionar turno
<b>Programador:</b> Carlos Manuel Padilla Rodríguez	<b>Iteración asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 4 días

<b>Riesgo en desarrollo:</b>	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá adicionar, editar, listar y eliminar los turnos laborales de un AGS.	
<b>Observaciones:</b> El usuario deberá ser autenticado con el rol Administrador y el sistema le indicará los campos que son obligatorios para gestionar los turnos laborales. El Vicedecano Administrativo, el Administrador y el Segundo Administrador, podrán consultar los datos de los turnos, una vez creado. Se podrá generar un listado con los datos de los turnos, en formato pdf y exe.	

**Tabla 5 HU6. Gestionar incidencias**

Historia de Usuario	
<b>Número:</b> 6	<b>Nombre del requisito:</b> Gestionar incidencia
<b>Programador:</b> Carlos Manuel Padilla Rodríguez	<b>Iteración asignada:</b> 3
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 4 días
<b>Riesgo en desarrollo:</b>	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá adicionar, editar, listar y eliminar las incidencias que ocurren en el área.	
<b>Observaciones:</b> El usuario deberá ser autenticado con el rol Vicedecano Administrativo y el sistema le indicará los campos que son obligatorios para registrar las incidencias. El Vicedecano Administrativo, el Administrador y el Segundo Administrador, podrán consultar las incidencias registradas, una vez creadas las mismas. Se podrá generar un listado con las incidencias insertadas en formato pdf y exe. El sistema permitirá subir la evidencia que se relacione con la incidencia registrada.	



**Tabla 6 HU2. Gestionar evaluaciones**

Historia de Usuario	
<b>Número:</b> 2	<b>Nombre del requisito:</b> Gestionar evaluación
<b>Programador:</b> Jessica Contreras Rodríguez	<b>Iteración asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 7 días
<b>Riesgo en desarrollo:</b>	<b>Tiempo real:</b> 1 semana
<p><b>Descripción:</b> El sistema permitirá adicionar, editar, listar y eliminar las evaluaciones de los trabajadores y estudiantes que realizan las actividades del componente laboral y las del TSU.</p>	
<p><b>Observaciones:</b> Una vez autenticado el usuario, el sistema indicará los campos que son obligatorios para gestionar las evaluaciones. Los trabajadores reciben evaluaciones frecuentes, que son otorgadas por el Administrador o el Segundo Administrador o el Vicedecano Administrativo y las mensuales y anuales, que las asigna el Vicedecano Administrativo. Los estudiantes que realizan el TSU, solo reciben una evaluación final al culminar su labor y los que ejecutan las actividades del componente, se les asigna una parcial, al culminar el primer semestre y la final cuando culmina el curso, estas evaluaciones las asigna el Segundo Administrador. Se podrá generar un listado con las evaluaciones insertadas en formato pdf y exe. El sistema, también permitirá subir la evidencia asociada a la evaluación otorgada.</p>	

## **2.7 Planificación**

La planificación es una fase corta de la metodología XP, en la que el cliente establece el orden en que deberían implementarse las HU y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los desarrolladores utilizando como medida el punto. Un punto equivale a una semana ideal de

programación. Las historias generalmente requieren de 1 a 3 puntos (57) y (58). El resultado de esta fase es un Plan de Iteraciones y un Plan de Entregas, que son de suma importancia para el desarrollo del proyecto.

### **2.7.1 Plan de Iteraciones**

En esta fase se incluyen varias iteraciones del sistema antes de ser entregado. Cada iteración producirá un conjunto de casos de pruebas funcionales para cada una de las HU agendadas para la iteración. Las iteraciones son empleadas para medir la evolución del proyecto; una iteración terminada sin errores es una medida de avance (57). A continuación se muestra el Plan de iteraciones de la propuesta solución:

**Tabla 7 Planificación de las iteraciones**

<b>Iteraciones</b>	<b>Orden de las HU a implementar</b>	<b>Cantidad de tiempo de trabajo</b>
Iteración 1	HU1. Gestionar persona HU2. Gestionar evaluación HU3. Gestionar usuario	3 semanas
Iteración 2	HU4. Gestionar turno HU5. Autenticar usuario HU7. Gestionar local	3 semanas
Iteración 3	HU6. Gestionar incidencia HU8. Gestionar reporte HU9. Exportar HU10. Listar	3 semanas

### 2.7.2 Plan de entregas

El Plan de entregas establece qué HU serán agrupadas para conformar una entrega y el orden de las mismas. Este constituye un documento oficial por el cual los clientes le exigen a los desarrolladores la entrega de las distintas versiones del producto (58). El equipo de desarrollo definió el siguiente Plan de Entregas, a partir de una reunión con el cliente, teniendo en cuenta las prioridades planteadas por el mismo.

**Tabla 8 Plan de entregas**

<b>Entregable</b>	<b>1ra entrega (4ta marzo semana de marzo)</b>	<b>2da entrega (4ta semana de abril)</b>	<b>3ra entrega (3ra semana de mayo)</b>
Sistema de gestión de información del área de servicios de la Facultad 4.	Versión 0.1	Versión 0.2	Versión 0.3

**Tabla 9 Funcionalidades disponibles para la entrega del producto**

<b>Historia de Usuario</b>	<b>1ra entrega</b>	<b>2da entrega</b>	<b>3ra entrega</b>
HU1. Gestionar persona	X		
HU2. Gestionar evaluación	X		
HU3. Gestionar usuario	X		
HU4. Gestionar turno		X	
HU7. Gestionar local		X	
HU6. Gestionar incidencia		X	
HU5. Autenticar usuario			X
HU8. Gestionar reporte			X

HU9. Exportar			X
HU10. Listar			X

### 2.8 Tarjetas CRC (Clase-Responsabilidad-Colaborador)

Las tarjetas CRC (Clase-Responsabilidad-Colaborador) son una técnica para la representación de sistemas OO, que sirven para diseñar el mismo en conjunto entre todo el equipo. Su objetivo es hacer un inventario de las clases que se necesitarán para implementar el sistema y la forma en que van a interactuar (59) y (60). A continuación se muestran las tarjetas relacionadas con las HU descritas anteriormente, el resto se encuentra en el Anexo 4 de la investigación.

**Tabla 10 Tarjeta CRC PersonaController**

Clase: PersonaController	
Responsabilidad	Colaborador
Se encarga de adicionar, editar, consultar y eliminar los datos de los trabajadores y los estudiantes del sistema.	Componente laboral, Turno, Evaluación, Certificado, TSU

**Tabla 11 Tarjeta CRC TurnoController**

Clase: TurnoController	
Responsabilidad	Colaborador
Adiciona, edita, consulta y elimina los turnos laborales.	Persona

**Tabla 12 Tarjeta CRC IncidenciaController**

Clase: IncidenciaController	
-----------------------------	--

Responsabilidad	Colaborador
Tiene la responsabilidad de adicionar, consultar y eliminar los locales activos e inactivos del docente.	Persona

**Tabla 13 Tarjeta CRC EvaluacionController**

Clase: EvaluacionController	
Responsabilidad	Colaborador
Adiciona, edita, consulta y elimina las evaluaciones de los trabajadores y estudiantes que realizan las actividades del componente laboral y TSU.	Persona

## **2.9 Patrones de diseño y de arquitectura**

### **2.9.1 Patrón arquitectónico**

Un patrón arquitectónico especifica un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes (55) y (61).

**Modelo-Vista-Controlador:** Un sistema desarrollado con el *framework* Grails sigue la arquitectura MVC. Este patrón separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. También define componentes para la representación de la información y por otro lado para la interacción del usuario, se basa en las ideas de reutilización de código y la separación de conceptos y busca facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento (39) y (40).

**Modelo:** es el conjunto de clases que representan la información del mundo real que el sistema debe procesar, sin tomar en cuenta ni la forma en que esta va a ser mostrada ni los mecanismos que hacen que los datos estén dentro del modelo.

**Vista:** son el conjunto de clases que se encargan de mostrar al usuario la información

contenida en el modelo.

**Controlador:** se encarga de modificar el modelo o de abrir y cerrar vistas y tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador.

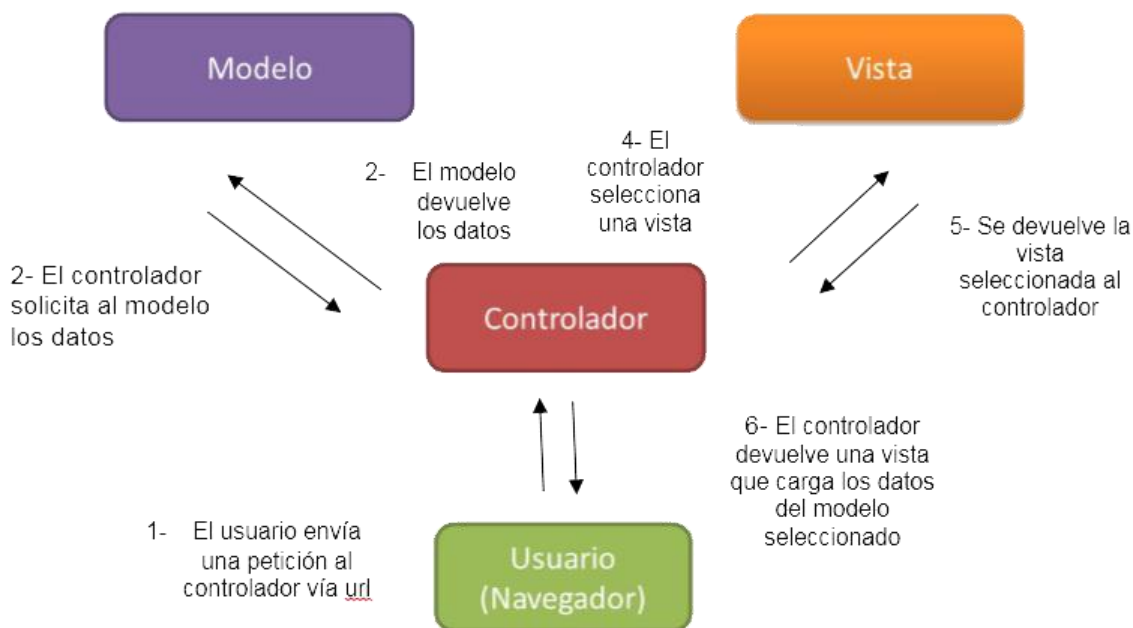


Ilustración 2 Patrón Modelo-Vista-Controlador

En la ilustración 3 se evidencia como el controlador recibe la notificación de la acción solicitada por el usuario, quien accede al sistema a través de una red interna o internet. El controlador accede al modelo, modificándolo de forma adecuada a la operación pedida por el usuario y delega a los objetos de la vista la tarea de desplegar la interfaz del usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo. La interfaz de usuario espera nuevas interacciones del usuario con el sistema, comenzando el ciclo nuevamente.

A continuación se evidencia el patrón MVC en la propuesta de solución:

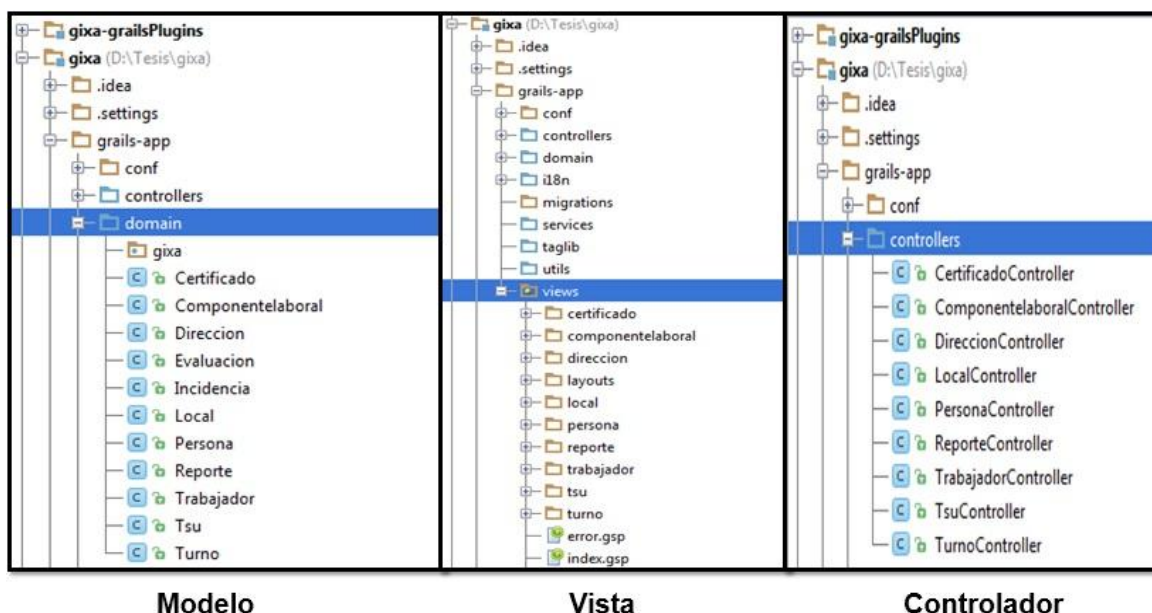


Ilustración 3 Representación del patrón arquitectónico en la propuesta de solución

## 2.9.2 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Los patrones brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (62).

- **Inversión de Control (IoC):** La Inversión de Control (IoC, del inglés *Inversion of Control*) es un patrón utilizado en Gails, según el cual las dependencias de un componente no deben gestionarse desde el propio componente para que éste solo contenga la lógica necesaria para hacer su trabajo. El objetivo de esta técnica es mantener los componentes lo más sencillos posible, incluyendo únicamente el código que tenga relación con la lógica del negocio y deja fuera todo el código de instalación; así la aplicación será más fácil de mantener y comprender (40) y (41).
- **Patrones GRASP:** Los patrones (GRASP, del inglés *General Responsibility Assignment Software Patterns*) o Patrones Generales de Software para

Asignación de Responsabilidades se consideran una serie de buenas prácticas de aplicación recomendable en el diseño de software (55) y (63). Los patrones básicos utilizados para aspectos fundamentales del diseño del sistema son:

**Experto:** permite la asignación de responsabilidades al experto en información, adquiriendo, de este modo, un diseño con mayor cohesión y menor acoplamiento de la información (63). Este patrón es utilizado en la clase TurnoController, que garantiza que la creación de un turno cuente con las funcionalidades que permiten realizar esta operación, mediante la información obtenida de las clases del modelo, con quienes se encuentra relacionada. También se evidencia en las clases PersonaController y LocalController.

**Creador:** indica quién es el responsable de la creación o instanciación de nuevos objetos o clases del proyecto (63). Es utilizado en las clases controladoras, donde se encuentran las acciones definidas para el sistema, por ejemplo, en la clase TurnoController, en el método *def create()*, cuando se crea el objeto de tipo turno.

**Controlador:** este patrón es el intermediario entre una determinada interfaz del producto y el algoritmo que implementa. Además permite a los desarrolladores la reutilización del código y a la vez tener un mayor control (63). Este patrón se evidencia en las clases que forman la capa Controlador del patrón arquitectónico MVC: TurnoController, PersonaController, LocalController, ReporteController, entre otras.

- **Patrones GoF:** Los patrones (GOF, del inglés *Gang of Four*) o Banda de los Cuatro se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “*Design Patterns*” (Patrones de Diseño) de Erich gamma, Richard Helm, Ralph Jonson y John Vlissides (64), (65) y (55). El patrón empleado en el diseño del proyecto fue:

**Decorator:** patrón Estructural que añade funcionalidades a un objeto individual de forma dinámica y transparente, sin afectar a otros objetos (64), (65) y (55). Este patrón es utilizado en la capa Vista del MVC y se emplea para incorporar modificaciones a partir de cada vista particular. Se definen en un único archivo independiente de las vistas GSP con lo que se simplifica enormemente la gestión del *look&feel* de la aplicación (43). Ejemplo de este patrón lo constituye la clase Main que es padre de



todas las vistas de la aplicación. A continuación se evidencia el patrón Decorator en Grails.

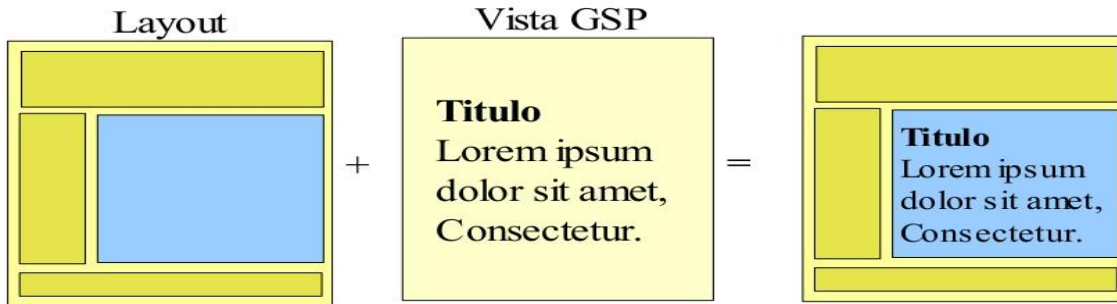


Ilustración 4 Patrón Decorator en Grails

## 2.10 Diagrama Entidad Relación (DER)

El Modelo Entidad Relación es un modelo de datos que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos, implementándose en forma gráfica a través del Diagrama Entidad Relación (66).

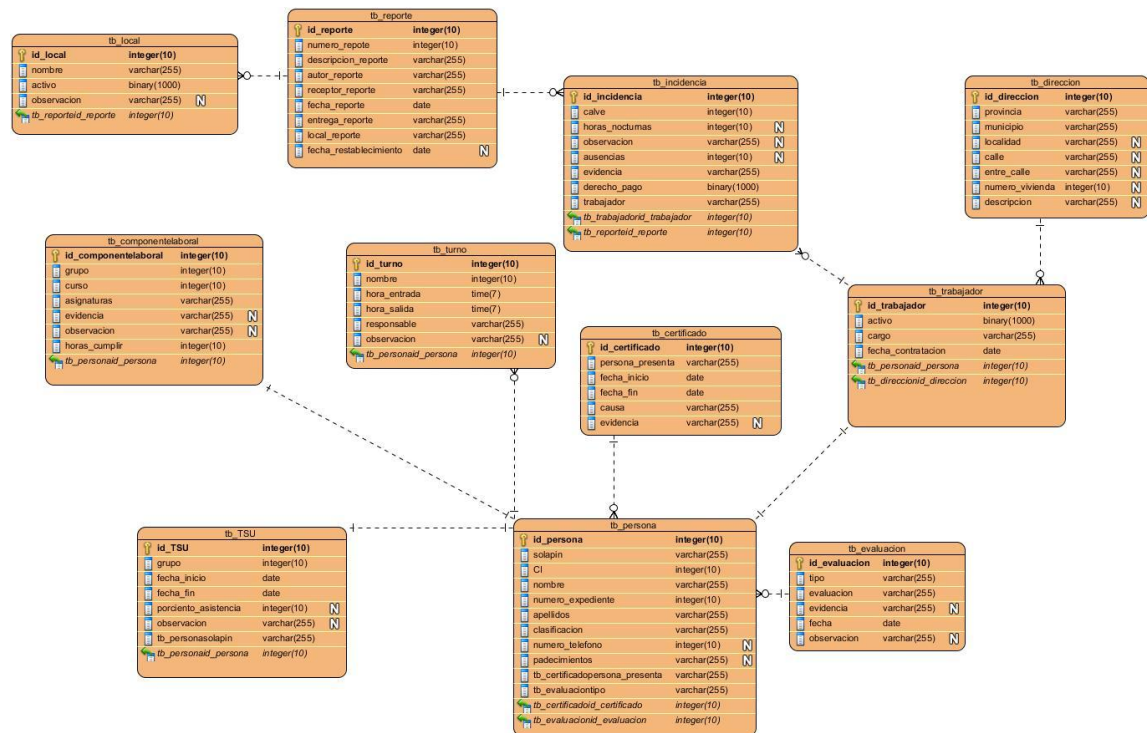


Ilustración 5 Diagrama Entidad Relación

**Tabla 14 Descripción de las tablas que conforman la base de datos**

<b>Tablas</b>	<b>Descripción</b>
tb_persona	Es la encargada de almacenar la información de un trabajador y de los estudiantes que realizan las actividades del componente laboral y TSU. Posee una relación de uno a uno con cada una de estas tablas.
tb_trabajador	En esta tabla se guardan los datos de los trabajadores.
tb_componentelaboral	En ella se almacenan los datos fundamentales de los estudiantes que realizan las actividades del componente laboral.
tb_TSU	Esta tabla contiene la información referente a los estudiantes del TSU.
tb_direccion	Es la tabla encargada de recopilar los datos relacionados con la dirección de un trabajador.
tb_local	Almacena los locales del Docente 5.
tb_incidencia	Esta tabla se emplea para almacenar los datos que conllevaron al no cumplimiento de la jornada laboral de los trabajadores.
tb_turno	Se encarga de almacenar los datos de los turnos laborales.
tb_reporte	Se utiliza para guardar los datos de cada reporte.

tb_certificado	Recoge los certificados médicos que presentan los trabajadores y estudiantes.
tb_evaluacion	Recoge las evaluaciones que les son otorgadas a los trabajadores y estudiantes que realizan las actividades del componente laboral y TSU.

### 2.11 Interfaces del sistema

The screenshot shows a web interface for adding a local. At the top is a grey header with the text "Adicionar local". Below the header is a light grey bar containing a blue link icon and the text "Lista de locales". The main form area has three fields: "Nombre \*" with a text input box, "Activo" with a checkbox, and "Observación" with a large text area. At the bottom of the form is a blue button labeled "Crear".

Ilustración 6 Interfaz "Adicionar local"

### Adicionar evaluación

[Lista de trabajadores](#)

**Persona** Carlos

**Tipo** Frecuente

**Evaluación** Adecuado

**Evidencia**  No se ha seleccionado ningún archivo.

**Fecha \*** 16/06/2015

**Observación**

Ilustración 7 Interfaz "Adicionar evaluación"

### Lista de turnos

[+ Adicionar turno](#)

Nombre	Fecha	Hora entrada	Hora salida	Personas
Mañana	2015-06-16	07:30 A.M.	12:00 P.M.	0
Tarde	2015-06-16	02:00 P.M.	6:30 P.M.	1
Noche	2015-06-16	08:00 P.M.	11:30 P.M.	0

EXCEL PDF

Ilustración 8 Interfaz "Listar turno"



The image shows a login form with a blue border. At the top, the title "Autenticarse" is displayed. Below the title are two input fields: the first is labeled "Usuario" with an envelope icon, and the second is labeled "Contraseña" with a magnifying glass icon. Below these fields is a checkbox labeled "Recordarme". At the bottom of the form is a blue button with the text "Iniciar sesión".

Ilustración 9 Interfaz "Autenticarse"

### **Conclusiones parciales**

En este capítulo se abordaron las características fundamentales del sistema a desarrollar, a través de la exposición de los artefactos generados por las dos primeras fases de la metodología XP. La confección del modelo conceptual permitió una mejor comprensión de la lógica del negocio, los requisitos funcionales componen la base para la implementación del sistema, las HU representaron las características que el producto debe poseer y la identificación de los patrones, facilitó el trabajo de los desarrolladores. La realización del modelo de datos proporcionó una mejor visión de la estructura de la base de datos del sistema. El Plan de entrega y Plan de iteraciones permitieron evaluar el progreso del proyecto.

## **CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA**

### **3.1 Introducción**

En el presente capítulo se describe la etapa de implementación, donde se especifican las tareas de ingeniería generadas a partir del desarrollo de las HU agrupadas en las iteraciones planificadas. Se validan cada una de las funcionalidades mediante las pruebas de software para garantizar el cumplimiento de los requerimientos pactados y se realiza un análisis de sus resultados.

### **3.2 Implementación**

Durante esta fase se procede a implementar las HU agrupadas en cada uno de las iteraciones descritas. Este plan se lleva a cabo dividiendo las HU en tareas de ingeniería y asignando a cada una, las personas responsables de su implementación (67), (29), (56) y (57).

#### **3.2.1 Programación en pareja**

Toda la producción del código debe realizarse con trabajo en parejas de programadores. Así se incrementa la calidad del sistema desarrollado sin afectar el tiempo de entrega (29). Las principales ventajas de introducir este estilo de programación son (67):

- Inspecciones de código continuas, por lo que la tasa de errores del producto final es más baja.
- Se obtiene un código de menos tamaño, por la continua discusión de ideas de los programadores.
- Los problemas de programación se resuelven más rápido.
- Varias personas entienden las diferentes partes del sistema.
- Transparencia de conocimientos de programación entre los miembros del equipo.

Los programadores parten de la idea que quieren desarrollar, mientras uno piensa en la táctica con la que se va a abordar el problema, el otro se encarga de las estrategias que permiten llevar dicha táctica a su máximo exponente.

### 3.2.2 Tareas de ingeniería

Las tareas de ingeniería se definen como un conjunto de actividades en los cuales, utilizando técnicas y herramientas, se analiza un problema y se concluye con la especificación de una solución (68). Se implementaron un total de 28 tareas de ingeniería, en este epígrafe se muestran las principales que se relacionan con las HU que se describen en el Capítulo 2, las restantes se encuentran en el Anexo 2 de la investigación.

**Tabla 15 Tarea de ingeniería # 1**

Tarea	
<b>Número:</b> 1	<b>Número de HU:</b> 4
<b>Nombre:</b> Adicionar turno	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 2 días
<b>Fecha inicio:</b> 1 de Abril del 2015	<b>Fecha fin:</b> 2 de Abril del 2015
<b>Programador responsable:</b> Carlos Manuel Padilla Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad “Adicionar turno” de la propuesta solución.	

**Tabla 16 Tarea de ingeniería # 2**

Tarea	
<b>Número:</b> 2	<b>Número de HU:</b> 4
<b>Nombre:</b> Editar turno	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 2 días
<b>Fecha inicio:</b> 3 de Abril del 2015	<b>Fecha fin:</b> 4 de Abril del 2015



<b>Programador responsable:</b> Carlos Manuel Padilla Rodríguez
<b>Descripción:</b> Implementar la funcionalidad que permite modificar los turnos en el sistema.

**Tabla 17 Tarea de ingeniería # 3**

Tarea	
<b>Número:</b> 3	<b>Número de HU:</b> 1
<b>Nombre:</b> Adicionar persona	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 2 días
<b>Fecha inicio:</b> 2 de Marzo del 2015	<b>Fecha fin:</b> 3 de Marzo del 2015
<b>Programador responsable:</b> Jessica Contreras Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad “Adicionar persona” en el sistema.	

**Tabla 18 Tarea de ingeniería # 4**

Tarea	
<b>Número:</b> 4	<b>Número de HU:</b> 1
<b>Nombre:</b> Editar persona	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 1 día
<b>Fecha inicio:</b> 4 de Marzo del 2015	<b>Fecha fin:</b> 5 de Marzo del 2015
<b>Programador responsable:</b> Jessica Contreras Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad que permita modificar los datos de la persona en el sistema.	

**Tabla 19 Tarea de ingeniería #5**

Tarea	
Número: 5	Número de HU: 2
Nombre: Adicionar evaluación	
Tipo de tarea: Desarrollo	Estimación: 2 días
Fecha inicio: 11 de Marzo del 2015	Fecha fin: 12 de Marzo del 2015
Programador responsable: Jessica Contreras Rodríguez	
Descripción: Implementar la funcionalidad "Adicionar evaluación" en el sistema.	

**Tabla 20 Tarea de ingeniería #6**

Tarea	
Número: 6	Número de HU: 2
Nombre: Editar evaluación	
Tipo de tarea: Desarrollo	Estimación: 1 día
Fecha inicio: 13 de Marzo del 2015	Fecha fin: 14 de Marzo del 2015
Programador responsable: Jessica Contreras Rodríguez	
Descripción: Implementar la funcionalidad "Editar evaluación" en el sistema.	

**Tabla 21 Tarea de ingeniería #7**

Tarea	
Número: 7	Número de HU: 6

<b>Nombre:</b> Adicionar incidencias	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 2 días
<b>Fecha inicio:</b> 16 de Marzo del 2015	<b>Fecha fin:</b> 17 de Marzo del 2015
<b>Programador responsable:</b> Carlos Manuel Padilla Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad "Adicionar incidencias" en el sistema.	

**Tabla 22 Tarea de ingeniería #8**

Tarea	
<b>Número:</b> 8	<b>Número de HU:</b> 6
<b>Nombre:</b> Editar incidencias	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 1 día
<b>Fecha inicio:</b> 18 de Marzo del 2015	<b>Fecha fin:</b> 18 de Marzo del 2015
<b>Programador responsable:</b> Carlos Manuel Padilla Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad "Editar incidencias" en el sistema.	

### **3.3 Diagrama de despliegue**

La metodología XP no requiere diagramas de despliegue, pero para un mejor entendimiento de la distribución física de la propuesta de solución se diseñó un diagrama de despliegue.

El diagrama de despliegue muestra la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos del despliegue. Los artefactos representan los elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo (69). Los diagramas de despliegue simbolizan los nodos,

que son objetos físicos con recursos computacionales y sus relaciones, que se etiquetan con un estereotipo que identifican el protocolo de comunicación o la red utilizada.

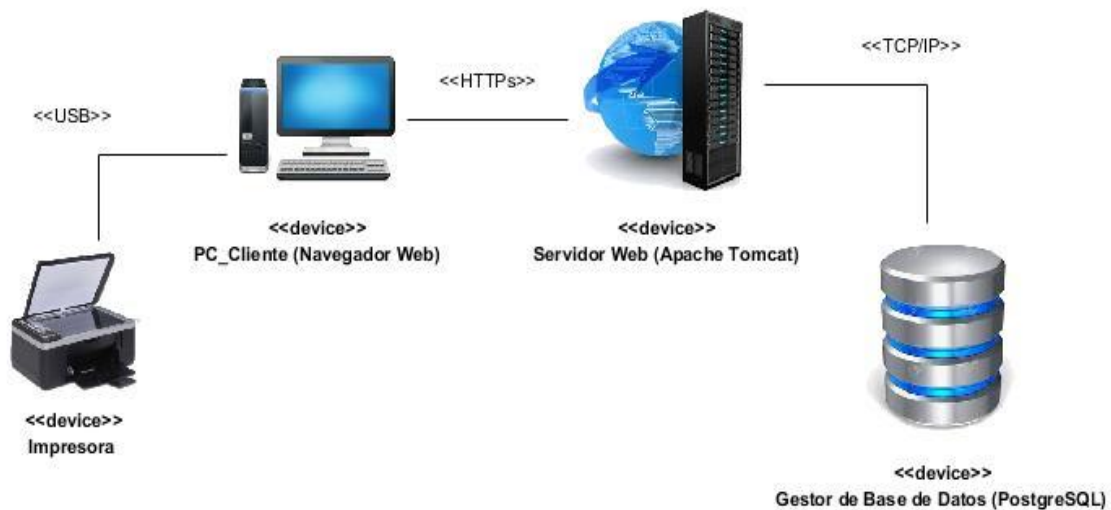


Ilustración 10 Diagrama de Despliegue

### **3.4 Tratamiento de errores**

En el sistema se deben analizar las diferentes situaciones que puedan ocurrir y los errores que estas puedan ocasionar. Las excepciones son identificadas en el proceso de implementación, aunque se debe definir con antelación un mecanismo efectivo para darle tratamiento a cada una de ellas.

En el caso del sistema de gestión de información del área de servicios administrativos de la F4, es necesario que el usuario que esté interactuando con el mismo especifique alguna información que se relacione con la futura validación o corrección de la excepción lanzada. Todas las excepciones identificadas fueron tratadas, evitando errores dentro del flujo de trabajo, por ejemplo el campo “Nombre” de la funcionalidad “Adicionar persona” no aceptaba caracteres extraños ni espacios, el campo “Calle” solo aceptaba letras y el “Número expediente” solo aceptaba números.

### **3.5 Pruebas**

Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos que se vayan implementando. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. Las pruebas se dividen en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y pruebas de aceptación destinadas a evaluar si, al culminar una iteración, se consiguió la funcionalidad requerida diseñada por el usuario final (70).

Las pruebas del sistema son fundamentales en XP, en la que se ha desarrollado un enfoque que reduce la probabilidad de producir nuevos incrementos del sistema que introduzcan errores en el software existente (71). A continuación se muestran los métodos, tipos y niveles de pruebas realizados al sistema, junto al ambiente en que se desarrollaron.

#### **Métodos de pruebas**

- Pruebas de caja blanca: estas pruebas comprueban los caminos lógicos del software, verificando todos los componentes internos, para determinar si el estado real coincide con el esperado (72) y (73).
- Pruebas de caja negra: estas pruebas se desarrollan sobre la interfaz del software comprobando todas las funcionalidades sin tener en cuenta la estructura interna del sistema (72) y (73).

#### **Tipos de pruebas**

- Pruebas unitarias: estas pruebas se encargan de probar cada uno de los métodos y verifica si dado unos parámetros de entrada, la salida es la esperada (72) y (73).
- Pruebas de funcionalidad: estas pruebas se realizan con el objetivo de verificar el cumplimiento de los requisitos funcionales (72) y (73).

#### **Niveles de pruebas**

- Pruebas de aceptación: Las pruebas de aceptación aseguran el comportamiento del sistema, son escritas por el cliente o por el usuario y

especifican los aspectos a probar cuando una HU ha sido correctamente implementada, garantizando que los requerimientos han sido cumplidos y que el sistema es aceptable. Significan la satisfacción del cliente con el producto desarrollado (70) y (72).

### **Ambiente de prueba**

Las pruebas se ejecutaron en un ambiente compuesto por la computadora del área de servicios administrativos de la Facultad 4.

**Tabla 23 Ambiente de prueba**

<b>Cantidad de PC</b>	<b>Sistema Operativo</b>	<b>RAM</b>	<b>Procesador</b>
1	Linux	2 GB	Intel® Core™2 Duo CPU E4500 @ 2.40GHz x 2

### **3.5.1 Resultados de las pruebas**

Al concluir las pruebas realizadas a cada una de las iteraciones planificadas, se efectúan las entregas pactadas con el cliente, la siguiente tabla muestra los resultados por cada iteración.

**Tabla 24 Cantidad de no conformidades detectadas por iteración**

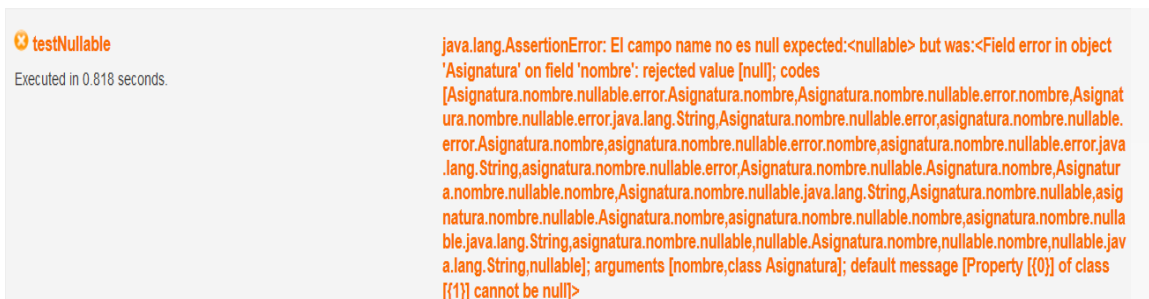
<b>No conformidades detectadas por iteración</b>			
	<b>Iteración 1</b>	<b>Iteración 2</b>	<b>Iteración 3</b>
<b>Cantidad de no conformidades detectadas</b>	<b>17</b>	<b>7</b>	<b>2</b>

Todas las no conformidades fueron resueltas, lo que garantiza la calidad de la propuesta de solución y la satisfacción del cliente. Los casos de pruebas que guiaron la realización de las pruebas se encuentran en el Anexo 3 de la investigación.

Durante las pruebas unitarias se examinaron un total de 15 funcionalidades, a continuación se muestran ejemplos de los resultados arrojados:

El siguiente método muestra que el valor del campo “Nombre”, de la clase “Asignatura” no puede ser nulo.

```
class AsignaturaTests{
void testNullable() {
    def a = new Asignatura()
    assertFalse a.validate()
    assertEquals 'El campo name no es null', 'nullable', a.errors['nombre']
}
}
```



```
testNullable
Executed in 0.818 seconds.
java.lang.AssertionError: El campo name no es null expected:<nullable> but was:<Field error in object 'Asignatura' on field 'nombre': rejected value [null]; codes [Asignatura.nombre.nullable.error,Asignatura.nombre.nullable.error.nombre,Asignatura.nombre.nullable.error.java.lang.String,Asignatura.nombre.nullable.error,asignatura.nombre.nullable.error,Asignatura.nombre,asignatura.nombre.nullable.error.nombre,asignatura.nombre.nullable.error.java.lang.String,asignatura.nombre.nullable.error,Asignatura.nombre.nullable,Asignatura.nombre,Asignatura.nombre.nullable,Asignatura.nombre.nullable.java.lang.String,Asignatura.nombre.nullable,asignatura.nombre.nullable,Asignatura.nombre,asignatura.nombre.nullable.nombre,asignatura.nombre.nullable.java.lang.String,asignatura.nombre.nullable,nullable,Asignatura.nombre,nullable.nombre,nullable.java.lang.String,nullable]; arguments [nombre,class Asignatura]; default message [Property [{0}] of class [{1}] cannot be null]>
```

Ilustración 11 Resultado del testNullable de la clase Asignatura en el campo nombre El siguiente método muestra que el valor del campo “Nombre”, de la clase “Asignatura”, tiene que ser único.

```
class AsignaturaTests{
void testUnique() {
    def a = new Asignatura(nombre: 'P1')
    assertTrue a.validate()
    assertEquals 'El campo name no puede ser duplicado', 'unique', a.errors['nombre']
}
}
```



```
testUnique
java.lang.AssertionError: El campo name no puede ser duplicado expected:<unique> but was:<null>
```

Ilustración 12 Resultado del testUnique de la clase Asignatura en el campo nombre

### **Conclusiones parciales**

En este capítulo se desarrollaron las tareas de ingeniería correspondientes a la fase de implementación. Se evidenció la necesidad de aplicar pruebas al software para evitar comportamientos no deseados, permitiendo validar el correcto desempeño del sistema. También se mostraron los resultados de las pruebas de aceptación, lo que demuestran la obtención de un sistema con menor probabilidad de fallos.



## **CONCLUSIONES**

Una vez finalizada la investigación se arriba a las siguientes conclusiones:

- A través del estudio bibliográfico se pudieron identificar los principales conceptos y características que intervienen en los procesos de gestión de información del área de servicios administrativos y las ventajas que traería la implementación de un sistema de gestión de información.
- La metodología seleccionada para el desarrollo del sistema proporcionó el cumplimiento de los requerimientos establecidos. Las herramientas y lenguajes de programación elegidos permitieron el avance de la solución propuesta. Además se obtuvieron artefactos de ingeniería necesarios para la implementación de la propuesta de solución, tales como:
  - El modelo conceptual con 12 conceptos detallados
  - Historias de usuario
  - Tareas de ingeniería
  - Tarjetas Clase-Responsabilidad-Colaborador
  - El diagrama entidad relación que presenta 11 tablas
  - La descripción de la arquitectura
  - El diagrama de despliegue con 4 nodos computacionales identificados
- Se realizaron pruebas unitarias y pruebas de aceptación al sistema para garantizar su correcto funcionamiento. Todas las no conformidades fueron resueltas, lo que garantiza la calidad de la propuesta de solución y la satisfacción del cliente.

Por todo lo anteriormente expuesto se puede concluir que se han cumplido satisfactoriamente cada uno de los objetivos identificados para la confección del sistema, obteniéndose un producto informático que facilita la gestión de la información, que en el área de servicios administrativos se genera.

## **RECOMENDACIONES**

Teniendo como base la experiencia adquirida y el análisis de los resultados del trabajo, se recomienda:

- Crear un manual de usuario para mejorar el entendimiento del sistema.
- Extender el sistema hacia todas las áreas de servicios administrativos de la Universidad de las Ciencias Informáticas.

## **REFERENCIAS BIBLIOGRÁFICAS**

1. Servicios TIC. [En línea] <http://www.serviciostic.com/las-tic/definicion-de-tic.html>.
2. Lineamientos de la Política Económica y Social del Partido y la Revolución. 2011. V Política de Ciencia, Tecnología, Innovación y Medio Ambiente, Artículo 131.
3. Lineamientos de la Política Económica y Social del Partido y la Revolución. 2011. Lineamientos para las principales ramas. Artículo 226.
4. Sitio web de la Universidad de las Ciencias Informáticas. [En línea] 2014. <http://www.uci.cu>.
5. Pascual, Adis Pérez. [entrev.] Jessica Contreras y Carlos Manuel Padilla. 2014.
6. Parada, Yaritza Taquechel. [entrev.] Jessica Contreras y Carlos Manuel Padilla. 2014.
7. [En línea] 2015. [http://web.educastur.princast.es/proyectos/formadultos/unidades/lengua\\_3/ud2/11\\_1.html](http://web.educastur.princast.es/proyectos/formadultos/unidades/lengua_3/ud2/11_1.html).
8. [En línea] <http://definicion.de/gestion/#ixzz3YatF8NpU>.
9. WordReference.com. [En línea] <http://www.wordreference.com/definicion/gesti%C3%B3n>.
10. [En línea] 2015. <http://mejoratugestion.com/mejora-tu-gestion/que-es-un-sistema-de-gestion/>.
11. [En línea] 2015. <http://www.dnvba.com/es/Certificacion/Sistemas-de-Gestion/Pages/default.aspx>.
12. [En línea] 2015. <http://thinkandsell.com/servicios/consultoria/software-y-sistemas/sistemas-de-gestion-normalizados/>.
13. Arévalo, Julio Alonso. Gestión de la Información, gestión de contenidos y conocimientos. 2009.
14. Universidad Michoacana de San Nicolás de Hidalgo.
15. [En línea] <http://sistemadmi3511.blogspot.com/2012/09/concepto-de-sistemas-administrativos.html>.
16. [En línea] <http://definicion.de/sistema-administrativo/>.

**REFERENCIAS BIBLIOGRÁFICAS**

17. Fernández, Daily Nuñez y Bermudes García, Daimariys. Desarrollo del componente Puesto de Trabajo durante la Gestión del Capital Humano. 2010.
18. Hernández, Isneysusana Herrera. Sistema de Información para la Gestión del Capital Humano en el Centro de Tecnologías de Gestión de Datos. 2011.
19. Giró, Santiago Marrero. Desarrollo del Módulo de Capital Humano para el Sistema de Gestión de Evaluaciones de Software de la UCI. 2011.
20. [En línea] <http://www.wordreference.com/definicion/metodología>.
21. Definición .De. [En línea] <http://definicion.de/metodologia/>.
22. Gordillo, Elena Polo. Metodología de desarrollo Tradicional vs Ágil. [En línea] 2014. <http://inventtatte.com/metodologia-tradicional-vs-agil/>.
23. Proyectos ágiles.org. [En línea] <http://www.proyectosagiles.org/que-es-scrum>.
24. Ingeniería de Software. [En línea] [http://ingenieriadesoftware.mex.tl/59189\\_Metodologia-Crystal.html](http://ingenieriadesoftware.mex.tl/59189_Metodologia-Crystal.html).
25. Ingeniería de Software. [En línea] [http://ingenieriadesoftware.mex.tl/52827\\_DSDM.html](http://ingenieriadesoftware.mex.tl/52827_DSDM.html).
26. ASD (Adaptive Software Development). [En línea] <http://adaptivesoftwaredevelopment.blogspot.com/>.
27. Ingeniería de Software. [En línea] [http://ingenieriadesoftware.mex.tl/61162\\_FDD.html](http://ingenieriadesoftware.mex.tl/61162_FDD.html).
28. Desarrollo Basado en Funciones. [En línea] <http://metodologiafdd.blogspot.com/>.
29. Canós, José H., Penadés, María del Carmen y Letelier, Patricio. 2013.
30. Universidad Union Bolivariana. Ingeniería de Software. [En línea] [http://ingenieriadesoftware.mex.tl/52753\\_XP---Extreme-Programing.html](http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html).
31. El Lenguaje Unificado de Modelado (UML). [En línea] <http://www.disca.upv.es/enheror/pdf/ActaUML.PDF>.
32. [En línea] <http://www.conml.org/FAQ.aspx>.
33. El Lenguaje de Modelado Unificado (UML). [En línea] <http://www.docirs.com/uml.htm>.
34. Introducción a Herramientas CASE y System Architect. [En línea] [http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro\\_case\\_SA.pdf](http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf).

**REFERENCIAS BIBLIOGRÁFICAS**

35. Visual Paradigm para UML. [En línea] <http://www.software.com.ar/visual-paradigm-para-uml.html>.
36. [En línea] <http://www.tecnolinkinformatica.com/gestion/definicion/Framework>.
37. [En línea] <http://rubyonrails.org/>.
38. Tutorial de introducción a Ruby on Rails. [En línea] [http://asociacion-aoe.org/aoe/files/documentacio/volcanica/2007/introduccion\\_ruby\\_on\\_rails.pdf](http://asociacion-aoe.org/aoe/files/documentacio/volcanica/2007/introduccion_ruby_on_rails.pdf).
39. Grails.com. [En línea] <http://www.grails.com/>.
40. Calahorro, Nacho Brito. Manual de desarrollo web con Grails.
41. Grails. [En línea] <https://grails.org/2.1.1+Release+Notes>.
42. [En línea] <http://www.oreilynet.com/pub/au/446|Lutz,Mark>.
43. Calahorro, Nacho Brito. Manual de desarrollo web con Grails.
44. The Java Language Specification. [En línea] <http://books.google.com.cu>.
45. Spurlock, Jake. Bootstrap. Primera edición. 2013.
46. Bootstrap-Línea de código. [En línea] 2013. <http://lineadecodigo.com/categoria/bootstrap/>.
47. Menéndez-Barzanallana, Rafael Asensio. Desarrollo de aplicaciones web.
48. Nvarrete, Toni. Lenguaje de programación JavaScript. 2006.
49. World Wide Web Consortium. [En línea] diciembre de 2013. <http://www.w3.org/html/>
50. Castillo, Alejandro Cantón. Manual de HTML5 en español.
51. Gauchat, Juan Diego. El Gran libro de HTML5, CSS3 y JavaScript . Primera edición. 2012.
52. Cascading Style Sheets (CSS) Snapshot 2010. [En línea] 2011. <http://www.w3.org/TR/CSS/> .
53. [En línea] <http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/>.
54. PostgreSQL. [En línea] <http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html>.
55. UML y Patrones.
56. [En línea] <http://xprogramming.com/xpmag/whatisxp..>
57. Patricio, Letelier y Penadés, María del Carmen. 2013.
58. [En línea] <http://programacionextrema.tripod.com/fases.htm..>

*REFERENCIAS BIBLIOGRÁFICAS*

59. [En línea] <http://www.fing.edu.uy..>
60. [En línea] <http://c2.com/doc/oopsla89/paper.html..>
61. [En línea] <http://www.lsi.us.es/docencia..>
62. [En línea] <https://msdn.microsoft.com/es-es/library/bb972240.aspx..>
63. Visconti, Marcello y Astudillo, Hernán. Fundamentos de Ingeniería de Software.
64. [En línea] 2008. <https://infow.wordpress.com/category/patrones-de-disenogof/>.
65. [En línea] <http://www.scielo.cl>.
66. [En línea] [http://www.belgrano.esc.edu.ar/matestudio/carpeta\\_de\\_access\\_introduccion.pdf..](http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf..)
67. Joskowicz, José. Reglas y Prácticas en Extreme Programming. 2008.
68. [En línea] 2015. <http://insitutotec.blogspot.com/2013/03/21-tareas-de-la-ingenieria-de-requisitos.html>.
69. UML. [En línea] <http://umldiagramadespliegue.blogspot.com/>.
70. Ordóñez, Meylin Pérez y Rodríguez Corbea, Maite. La Metodología XP aplicable al desarrollo del software educativo en Cuba. 2007.
71. Ian, Sommerville. Ingeniería del software. [trad.] María Isabel Galapienso Alfonso , y otros. Séptima edición.
72. Pressman, Roger S. Ingeniería de software. Un enfoque práctico. Sexta Edición. 2005.
73. —. Ingeniería de software. Un enfoque práctico. Quinta Edición. 2001.
74. Zend Framework. [En línea] <http://framework.zend.com/>.
75. Analisis y Diseño de sistemas. [En línea] <http://www.adsfrank.blogspot.com/2007/06/metodologias-agiles-ld.html>.
76. Beck, Kent. Extreme Programming Explained. 1999.
77. Hernández Sampieri Roberto. Metodología de la Investigación. Segunda Edición. 1998, México.
78. [En línea] 2015. [www.inf.utfsm.cl/visconti/xp/Pruebas\\_Aceptacion\\_2](http://www.inf.utfsm.cl/visconti/xp/Pruebas_Aceptacion_2).

**ANEXOS**

**Anexo 1 Historias de usuario**

**Tabla 25 HU.8 Gestionar reporte**

<b>Historia de Usuario</b>	
<b>Número:</b> 8	<b>Nombre del requisito:</b> Gestionar reporte
<b>Programador:</b> Jessica Contreras Rodríguez	<b>Iteración Asignada:</b> 3
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 7 días
<b>Riesgo en Desarrollo:</b>	<b>Tiempo Real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá adicionar, editar, consultar y eliminar los reportes.	
<b>Observaciones:</b> El usuario deberá ser autenticado con el rol Administrador y el sistema le indicará los campos que son obligatorios para gestionar los reportes. El Vicedecano Administrativo, el Administrador y el Segundo Administrador, podrán ver las gestiones realizadas en los reportes. Se podrá exportar los datos generados en un documento exe.	

**Tabla 26 HU5. Autenticar usuario**

<b>Historia de Usuario</b>	
<b>Número:</b> 5	<b>Nombre del requisito:</b> Autenticar usuario
<b>Programador:</b> Carlos Manuel Padilla Rodríguez	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 5 días
<b>Riesgo en Desarrollo:</b>	<b>Tiempo Real:</b> 1 semana

**Descripción:** El sistema permitirá que un usuario se autentique insertando su identificador y contraseña.

**Observaciones:** El usuario podrá acceder a sus responsabilidades si introdujo correctamente su identificador y contraseña.

**Tabla 27 HU7. Gestionar local**

Historia de Usuario	
<b>Número:</b> 7	<b>Nombre del requisito:</b> Gestionar local
<b>Programador:</b> Carlos Manuel Padilla Rodríguez	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 7 días
<b>Riesgo en Desarrollo:</b>	<b>Tiempo Real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá adicionar, editar y eliminar los locales del Docente 5.	
<b>Observaciones:</b> El usuario deberá ser autenticado con el rol Administrador y el sistema le indicará los campos que son obligatorios para gestionar los locales. El Vicedecano Administrativo, el Administrador y el Segundo Administrador, podrán ver los locales una vez creados. Se podrá generar un listado con los datos de los locales en formato pdf y exe.	

**Tabla 28 HU3. Gestionar usuario**

Historia de Usuario	
<b>Número:</b> 3	<b>Nombre del requisito:</b> Gestionar usuario
<b>Programador:</b> Carlos Manuel Padilla Rodríguez	<b>Iteración asignada:</b> 1



<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 2 días
<b>Riesgo en desarrollo:</b>	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá crear, editar y eliminar usuario.	
<b>Observaciones:</b> El usuario deberá ser autenticado con el rol Vicedecano Administrativo. Las responsabilidades del usuario en el sistema dependerán del rol asignado.	

Tabla 29 HU 9 Exportar

Historia de Usuario	
<b>Número:</b> 9	<b>Nombre del requisito:</b> Exportar
<b>Programador:</b> Carlos Manuel Padilla Rodríguez	<b>Iteración asignada:</b> 3
<b>Prioridad:</b> Media	<b>Tiempo estimado:</b> 2 días
<b>Riesgo en desarrollo:</b>	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá exportar los listados que se originen.	
<b>Observaciones:</b> Los listados sólo se podrán exportar en los formatos pdf y exe.	

Tabla 30 HU 10 Listar

Historia de Usuario	
<b>Número:</b> 10	<b>Nombre del requisito:</b> Listar
<b>Programador:</b> Carlos Manuel Padilla Rodríguez	<b>Iteración asignada:</b> 3

<b>Prioridad:</b> Baja	<b>Tiempo estimado:</b> 2 días
<b>Riesgo en desarrollo:</b>	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El sistema permitirá, listar la información que se genera en el sistema tras gestionar las personas, las evaluaciones, los turnos, los locales, los reportes y las incidencias.	
<b>Observaciones:</b> El sistema permitirá exportar los listados en formato pdf y exe.	

## Anexo 2 Tareas de ingeniería

Tabla 31 Tareas de ingeniería #8

Tarea	
<b>Número:</b> 8	<b>Número de HU:</b> 6
<b>Nombre:</b> Listar incidencia	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 1 día
<b>Fecha inicio:</b> 19 de Marzo del 2015	<b>Fecha fin:</b> 19 de Marzo del 2015
<b>Programador responsable:</b> Carlos Manuel Padilla Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad "Listar incidencia".	

Tabla 32 Tareas de ingeniería #9

Tarea	
<b>Número:</b> 9	<b>Número de HU:</b> 6
<b>Nombre:</b> Adicionar local	

<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 1 día
<b>Fecha inicio:</b> 13 de Abril del 2015	<b>Fecha fin:</b> 14 de Abril del 2015
<b>Programador responsable:</b> Carlos Manuel Padilla Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad “Adicionar local” en el sistema.	

**Tabla 33 Tareas de ingeniería #10**

Tarea	
<b>Número:</b> 10	<b>Número de HU:</b> 8
<b>Nombre:</b> Editar reporte	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 1 día
<b>Fecha inicio:</b> 29 de Abril del 2015	<b>Fecha fin:</b> 29 de Abril del 2015
<b>Programador responsable:</b> Jessica Contreras Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad “Editar reporte”.	

**Tabla 34 Tareas de ingeniería #11**

Tarea	
<b>Número:</b> 11	<b>Número de HU:</b> 1
<b>Nombre:</b> Listar persona	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 1 día
<b>Fecha inicio:</b> 18 de Marzo del	<b>Fecha fin:</b> 18 de Marzo del 2015

2015	
<b>Programador responsable:</b> Jessica Contreras Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad "Listar persona".	

Tabla 35 Tareas de ingeniería #12

Tarea	
<b>Número:</b> 12	<b>Número de HU:</b> 8
<b>Nombre:</b> Adicionar reporte	
<b>Tipo de tarea:</b> Desarrollo	<b>Estimación:</b> 2 días
<b>Fecha inicio:</b> 27 de Abril del 2015	<b>Fecha fin:</b> 28 de Abril del 2015
<b>Programador responsable:</b> Jessica Contreras Rodríguez	
<b>Descripción:</b> Implementar la funcionalidad "Adicionar reporte" en el sistema.	

### Anexo 3 Casos de pruebas de aceptación

Tabla 36 Caso de prueba de aceptación #1

Caso de prueba de aceptación	
<b>Código:</b> HU1_CP1	<b>Historia de Usuario:</b> 1
<b>Nombre:</b> Adicionar persona	
<b>Condiciones de ejecución:</b> El usuario debe estar autenticado.	
<b>Entrada/Pasos de ejecución:</b> La entrada consiste en los datos de la persona que el	

<p>usuario debe introducir.</p> <p>Pasos de ejecución: Para introducir los datos, debe ir al menú y seleccionar la opción “Persona”, seleccionar la opción “Adicionar”. Una vez introducidos los datos de la persona, se selecciona la opción “Crear” y se muestran los datos de la persona añadida.</p>
<p><b>Resultado esperado:</b> Los datos de la persona fueron añadidos satisfactoriamente.</p>
<p><b>Evaluación de la prueba:</b> Prueba satisfactoria.</p>

**Tabla 37 Caso de prueba de aceptación #2**

Caso de prueba de aceptación	
<b>Código:</b> HU1_CP2	<b>Historia de Usuario:</b> 1
<b>Nombre:</b> Editar persona	
<b>Condiciones de ejecución:</b> Debe estar insertada al menos una persona en el sistema.	
<p><b>Entrada/Pasos de ejecución:</b></p> <p>Entrada: Los datos que el usuario desea modificar.</p> <p>Pasos de ejecución: Seleccionar en el menú la opción “Persona”, seleccionar la opción “Listar”, escoger la persona a la que se le quieren modificar los datos y seleccionar la opción “Editar”. Una vez modificados los datos, presionar el botón “Actualizar” y acto seguido se muestran los datos de la persona modificada.</p>	
<b>Resultado esperado:</b> Los datos de la persona fueron modificados satisfactoriamente.	
<b>Evaluación de la prueba:</b> Prueba satisfactoria.	

**Tabla 38 Caso de prueba de aceptación #3**

<b>Caso de prueba de aceptación</b>	
<b>Código:</b> HU2_CP3	<b>Historia de Usuario:</b> 2
<b>Nombre:</b> Adicionar evaluación	
<b>Condiciones de ejecución:</b> Debe estar insertada al menos una persona en el sistema.	
<p><b>Entrada/Pasos de ejecución:</b> La entrada consiste en la evaluación que el usuario desea asignar a un trabajador o estudiante.</p> <p>Pasos de ejecución: En el menú, seleccionar la opción “Persona”, seleccionar la opción “Listar”, en “Evaluaciones”, presionar el botón “(+)” o luego de haber listado a las personas, seleccionar a quién se le otorgará la evaluación y presionar el botón “Evaluar”. Una vez asignada la evaluación, presionar el botón “Crear” y se muestra la persona con la evaluación asignada.</p>	
<b>Resultado esperado:</b> La evaluación fue asignada satisfactoriamente.	
<b>Evaluación de la prueba:</b> Prueba satisfactoria.	

**Tabla 39 Caso de prueba de aceptación #4**

<b>Caso de prueba de aceptación</b>	
<b>Código:</b> HU2_CP4	<b>Historia de Usuario:</b> 2
<b>Nombre:</b> Editar evaluación	
<b>Condiciones de ejecución:</b> Al menos una evaluación debe haber sido otorgada.	
<p><b>Entrada/Pasos de ejecución:</b></p> <p>Entrada: Modificar la evaluación que el usuario desee.</p>	

<p>Pasos de ejecución: En el menú, seleccionar la opción “Persona”, seleccionar la opción “Listar”, seleccionar a quién se le modificará la evaluación, elegir el tipo de evaluación a editar, seleccionar la opción “Editar”. Una vez modificada la evaluación se presiona el botón “Actualizar” y se muestra la evaluación modificada.</p>
<p><b>Resultado esperado:</b> La evaluaciones fueron modificadas satisfactoriamente.</p>
<p><b>Evaluación de la prueba:</b> Prueba satisfactoria.</p>

**Tabla 40 Caso de prueba de aceptación #5**

Caso de prueba de aceptación	
<b>Código:</b> HU6_CP5	<b>Historia de Usuario:</b> 6
<b>Nombre:</b> Adicionar incidencia	
<b>Condiciones de ejecución:</b> El usuario debe estar autenticado con el rol Vicedecano Administrativo	
<p><b>Entrada/Pasos de ejecución:</b> La entrada consiste en los datos de una nueva incidencia que el usuario desea introducir.</p> <p>Pasos de ejecución: En el menú, presionar el botón “Incidencias” y seleccionar la opción “Adicionar”. Una vez introducidos los datos de la incidencia, seleccionar la opción “Crear” y se muestra la incidencia creada.</p>	
<b>Resultado esperado:</b> La incidencia fue añadida satisfactoriamente.	
<b>Evaluación de la prueba:</b> Prueba satisfactoria.	

**Tabla 41 Caso de prueba de aceptación #6**

Caso de prueba de aceptación
------------------------------

<b>Código:</b> HU4_CP6	<b>Historia de Usuario:</b> 4
<b>Nombre:</b> Adicionar turno	
<b>Condiciones de ejecución:</b> El usuario debe estar autenticado con el rol Administrador.	
<b>Entrada/Pasos de ejecución:</b>  Entrada: El usuario debe introducir los datos de un turno.  Pasos de ejecución: En el menú seleccionar la opción "Turnos", seleccionar la opción "Adicionar". Una vez insertado los datos del turno, seleccionar la opción "Crear" y se muestra el turno con los datos insertados.	
<b>Resultado esperado:</b> El turno fue añadido satisfactoriamente.	
<b>Evaluación de la prueba:</b> Prueba satisfactoria.	

**Tabla 42 Caso de prueba de aceptación #7**

Caso de prueba de aceptación	
<b>Código:</b> HU4_CP7	<b>Historia de Usuario:</b> 4
<b>Nombre:</b> Editar turno	
<b>Condiciones de ejecución:</b> Debe estar insertado al menos un turno en el sistema.	
<b>Entrada/Pasos de ejecución:</b> La entrada consiste en los datos del turno que el usuario desea modificar.  Pasos de ejecución: Pasos de ejecución: En el menú seleccionar la opción "Turnos", seleccionar la opción "Listar", escoger el turno que se desea modificar y presionar el botón "Editar". Una vez modificado el turno, se presiona el botón "Actualizar" y se	



muestra el turno con los datos modificados.
<b>Resultado esperado:</b> El turno fue modificado satisfactoriamente.
<b>Evaluación de la prueba:</b> Prueba satisfactoria.

**Tabla 43 Caso de prueba de aceptación #9**

Caso de prueba de aceptación	
<b>Código:</b> HU7_CP9	<b>Historia de Usuario:</b> 7
<b>Nombre:</b> Adicionar local	
<b>Condiciones de ejecución:</b> El usuario debe estar autenticado con el rol Administrador.	
<b>Entrada/Pasos de ejecución:</b> La entrada consiste en el local que el usuario desea añadir al sistema.  Pasos de ejecución: En el menú seleccionar la opción “Local”, seleccionar la opción “Adicionar”. Una vez llenado los campos, seleccionar la opción “Crear” y se mostrará el local insertado.	
<b>Resultado esperado:</b> Los locales fueron insertados satisfactoriamente.	
<b>Evaluación de la prueba:</b> Prueba satisfactoria.	

**Tabla 44 Caso de prueba de aceptación #10**

Caso de prueba de aceptación	
<b>Código:</b> HU8_CP10	<b>Historia de Usuario:</b> 8
<b>Nombre:</b> Adicionar reporte	

<p><b>Condiciones de ejecución:</b> El usuario debe estar autenticado con el rol Administrador.</p>
<p><b>Entrada/Pasos de ejecución:</b></p> <p>Entrada: El usuario debe introducir los datos de un nuevo reporte.</p> <p>Pasos de ejecución: En el menú seleccionar la opción “Incidencia”, seleccionar la opción “Adicionar reporte”. Una vez insertado los datos del reporte, seleccionar la opción “Crear” y se mostrará el reporte adicionado.</p>
<p><b>Resultado esperado:</b> Los reportes fueron insertados satisfactoriamente.</p>
<p><b>Evaluación de la prueba:</b> Prueba satisfactoria.</p>

**Tabla 45 Caso de prueba de aceptación #11**

Caso de prueba de aceptación	
<b>Código:</b> HU8_CP11	<b>Historia de Usuario:</b> 8
<b>Nombre:</b> Editar reporte	
<p><b>Condiciones de ejecución:</b> El usuario debe estar autenticado con el rol Administrador.</p>	
<p><b>Entrada/Pasos de ejecución:</b></p> <p>Entrada: El usuario debe introducir los datos que desea modificar del reporte.</p> <p>Pasos de ejecución: En el menú seleccionar la opción “Incidencia”, seleccionar la opción “Listar reporte”, seleccionar el reporte que se desea modificar. Una vez editado los datos del reporte, seleccionar la opción “Actualizar” y se mostrará el reporte modificado.</p>	

<b>Resultado esperado:</b> Los reportes fueron modificados satisfactoriamente.
<b>Evaluación de la prueba:</b> Prueba satisfactoria.

**Tabla 46 Caso de prueba de aceptación #12**

<b>Caso de prueba de aceptación</b>	
<b>Código:</b> HU2_CP12	<b>Historia de Usuario:</b> 2
<b>Nombre:</b> Listar evaluación	
<b>Condiciones de ejecución:</b> Deben existir evaluaciones en el sistema.	
<b>Entrada/Pasos de ejecución:</b>  Entrada: Las evaluaciones que el usuario desea listar.  Pasos de ejecución: En el menú seleccionar la opción “Persona”, seleccionar la opción “Listar”, seleccionar la persona deseada y se mostrarán todas sus evaluaciones otorgadas.	
<b>Resultado esperado:</b> Las evaluaciones fueron listadas satisfactoriamente.	
<b>Evaluación de la prueba:</b> Prueba satisfactoria.	

**Tabla 47 Caso de prueba de aceptación #13**

<b>Caso de prueba de aceptación</b>	
<b>Código:</b> HU4_CP13	<b>Historia de Usuario:</b> 4
<b>Nombre:</b> Listar turno	
<b>Condiciones de ejecución:</b> Deben existir turnos en el sistema.	

<p><b>Entrada/Pasos de ejecución:</b></p> <p>Entrada: Los turnos que el usuario desea listar.</p> <p>Pasos de ejecución: En el menú seleccionar la opción “Turno”, seleccionar la opción “Listar” y se mostrará todos los turnos listados.</p>
<p><b>Resultado esperado:</b> Los turnos fueron listados satisfactoriamente.</p>
<p><b>Evaluación de la prueba:</b> Prueba satisfactoria.</p>

#### **Anexo 4. Tarjetas CRC**

**Tabla 48 Tarjeta CRC ReporteController**

<b>Clase: ReporteController</b>	
<b>Responsabilidad</b>	<b>Colaborador</b>
Adiciona, edita, consulta y elimina los reportes que se generan en el área de servicios administrativos.	Local

**Tabla 49 Tarjeta CRC LocalController**

<b>Clase: LocalController</b>	
<b>Responsabilidad</b>	<b>Colaborador</b>
Tiene la responsabilidad de adicionar, editar, consultar y eliminar los locales activos e inactivos del docente.	Reporte

**Tabla 50 Tarjeta CRC DireccionController**

<b>Clase: DireccionController</b>	
-----------------------------------	--

<b>Responsabilidad</b>	<b>Colaborador</b>
Tiene la responsabilidad de adicionar, editar, consultar y eliminar las direcciones de los trabajadores.	Trabajador

**Tabla 51 Tarjeta CRC CertificadoController**

<b>Clase: CertificadoController</b>	
<b>Responsabilidad</b>	<b>Colaborador</b>
Tiene la responsabilidad de adicionar, editar, consultar y eliminar los certificados médicos de los trabajadores.	Persona