Universidad de las Ciencias Informáticas

Facultad 4

Título: Sistema de gestión para la evaluación de los profesores de la Facultad 4.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: *Yaiselín Ramos Rodríguez*
José Alberto Fonseca Calistre

Tutores: *MSc. Mailin Carballosa Infante*
Ing. Yordankis Matos López

Co-Tutor: *Ing. Orlando Grabiel Toledano López*

"Año 58 de la Revolución"
La Habana, junio 2016

Declaración de Autoría

Declaramos ser autores del presente trabajo y autorizamos a la Facultad 4 de la Universidad de las Ciencias Informáticas de hacer uso del mismo en su beneficio.

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

Yaiselín Ramos Rodríguez

_____

Firma del autor

José Alberto Fonseca Calistre

_____

Firma del autor

MSc. Mailin Carballosa Infante

_____

Firma del tutor

Ing. Yordankis Matos López

_____

Firma del tutor

Dedicatoria

De Yaiselín:

Dedico este trabajo a la persona más importante de mi vida, a la que me vio nacer, crecer, quien me educó con amor, quien me quiso hasta el último momento, a esa que me dejó sola cuando más la necesitaba, nunca te voy a perdonar que te pierdas este día, estés donde estés debes estar orgullosa, soy la mujer que soñaste gracias abuela, hoy entiendo más que nunca lo mucho que me amabas.

De José:

A mi mamá y a mi tío Omar que durante estos cinco años han mostrado su preocupación y apoyo incondicional.

Agradecimientos

De Yaiselín:

Agradezco a mi mami por su paciencia, amor, incondicionalidad, me diste la vida y me enseñaste a vivir, gracias por todo lo que te quitaste para dármelo, gracias por estar ahí.

A mi papito, que es más que mi papá es un amigo, es mi confianza y a pesar de malentendidos siempre lo voy a querer.

A mis tías gracias por escucharme, por entenderme, por apoyarme, realmente son mis otras mamás.

A mis abuelas que a las dos las amo mucho son y serán mi fuente de experiencia y sabiduría.

A mi hermano que aparte de agradecerle porque es mi guía y razón de existir quiero que este sea un comienzo para él, que me tome de ejemplo y que salga a luchar por sus sueños que también son los míos.

A mi hermana Pochi gracias por estar siempre conmigo, te quiero mucho pase lo que pase eres mi hermana.

A mis compañeros de aula y apartamento, que muchos no están aquí, pero estuvieron cuando los necesité, vale mencionar a algunos que han sido un eslabón fundamental en este logro:

A mi novio que estuvo conmigo desde el principio y espero que hasta el final.

A Tania, Edelin y Wendys quiero agradecerles cada gota de paciencia conmigo, no soy fácil, lo sé, gracias por compartir sus conocimientos, experiencias, y sus cosas conmigo, no lo voy a olvidar.

A Henry y Lester, son muchos los compañeros de aula, pero ustedes dos siempre han sido una de las personas más importantes de mi grupo, los admiro y respeto, no me olviden y gracias por escucharme.

A Yeni y Andy, que sería de la vida sin la amistad verdadera, entraron en mi vida para quedarse, gracias por todos los buenos momentos que pasé a su lado, los espero en Cárdenas con las puertas abiertas.

A Arlenys gracias por estar ahí siempre, nunca voy a olvidar tu amistad porque hay pocas personas como tú, eres lo máximo.

A moni, gracias por ser mi amigo y mi confidente, cuenta conmigo siempre, te adoro.

A los amigos de toda la vida que, aunque no están aquí mantuvieron su apoyo constante Kassandra, Irianna, Denisse, Gueordys, Luisa, Wendys, Glenys.

A mis vecinos, que realmente son la extensión de mi familia.

A mis primos muchas gracias por su hermandad incondicional, los quiero pueden contar conmigo siempre.

A mis tutores, que diga a mis segundos padres, con gracias no podría agradecerles todo lo que han hecho por mí, ustedes han sido mis guías, sin ustedes esto no sería posible, los quiero tanto que le daría mi casa, si fuera mía.

A mi cotutor Orlando, mil gracias por todos los momentos donde escogiste ayudarme, sé que tenías muchas cosas que hacer.

De José

A mi madre por su dedicación y entrega.

A mi tío que me ha dado el amor de padre, gracias.

A mi primo Omarito que es como un hermano para mí.

A mi hermano que ha estado conmigo en las buenas y en las malas.

A mis compañeros de aula y apartamento, gracias por soportarme.

A mi novia que ha tenido paciencia y me ha dado su amor incondicional cuando más lo he necesitado.

A mis abuelos por ser fuente de amor y experiencia.

A mis tutores que más que la tesis le agradezco mi carrera. Muchas gracias.

Resumen

Los sistemas de gestión de información permiten a las organizaciones mejorar el procesamiento de la información que se obtiene a partir de los procesos que realizan. La Universidad de las Ciencias Informáticas como parte de su estrategia, exige la evaluación de todos sus docentes. En los departamentos de la Facultad 4 se desarrolla el proceso de evaluación ya sea, mensual, trimestral o anual. Este genera un gran volumen de datos, que en la actualidad son procesados manualmente o almacenados en correo electrónico, lo cual dificulta su almacenamiento y mantenimiento. Además, consume tiempo y esfuerzo de los encargados de este proceso, al no existir una forma sencilla de monitorizar las evidencias y evaluaciones de los profesores. Esta investigación se planteó como propósito desarrollar un sistema informático que contribuya al control de las evaluaciones de los profesores de la Facultad 4. El mismo fue desarrollado utilizando la metodología AUP, basándose en el framework de desarrollo Vaadin y el lenguaje de programación Java. Se realizaron los flujos de Modelado, Implementación y Prueba, indicados por la metodología elegida, obteniéndose un sistema que cumple con los requerimientos identificados. Con el software desarrollado, se permite la gestión de evidencias, actividades, planes de trabajo y otras funciones que contribuyen con la mejora del proceso de evaluación profesoral.

Palabras clave: evaluación profesoral, sistema de gestión de información.

Índice

Introducción

La Universidad de las Ciencias Informáticas (UCI) desde su creación ha formado profesionales de alta calidad y competencia. Por muchos años la UCI entre sus principales estrategias, utiliza la evaluación de sus trabajadores para reconocerles su trabajo y participación en las actividades planificadas, así como otras que surgen debido al dinamismo de la universidad. El Ministerio de Educación Superior (MES) establece la evaluación de todos los docentes en el año fiscal. A partir de la planificación, en el momento de elaboración del plan individual de trabajo del profesor, se reflejará su superación, con acciones específicas de acuerdo con su rama, papel que desempeña, categoría docente y científica, carga de trabajo integral, necesidades específicas de superación y cambio de categoría docente previsto, entre otros. (1) Para la evaluación profesoral anual el docente realiza una autoevaluación indicando los resultados alcanzados en cada una de las áreas de resultados clave, que a continuación se relacionan:

- ✓ Trabajo docente-educativo en pregrado y posgrado.
- ✓ Trabajo político-ideológico.
- ✓ Trabajo metodológico.
- ✓ Trabajo de investigación e innovación.
- ✓ Superación.
- ✓ Extensión universitaria.

El jefe del departamento comprueba la veracidad de la autoevaluación, otorga una evaluación por cada una de las áreas de resultado clave y emite las observaciones que estime pertinente, así como la evaluación general.

Como parte de la estimulación a los profesores, la UCI aplica la evaluación de desempeño, donde se asigna un pago adicional al salario básico, de acuerdo a una evaluación trimestral de Deficiente, Adecuado o Superior. Esta evaluación se realiza a partir de una autoevaluación del profesor con las acciones y resultados del trimestre y la valoración del Jefe de Departamento, quien presenta las propuestas y son aprobadas por el Consejo de dirección de cada facultad.

Otro de los procesos en la universidad es el Balance de los indicadores de Ciencia, Tecnología e Innovación (CTI) por cada área, donde se mide el impacto de las investigaciones de los profesores y la participación en eventos, obtención de premios, entre otros. En este proceso, es necesario, recopilar todas las evidencias de publicación y participación en eventos de los docentes. Dicha actividad la dirige el Vicedecano de Investigación y Postgrado de las facultades quienes solicitan información a los Jefes de Departamento y responsables de CTI de cada área, a partir de los documentos presentados por el profesor que avalan dicho resultado.

En los procesos antes mencionados, se hace necesario el análisis de la participación de los docentes en diferentes actividades, así como los resultados obtenidos en diferentes períodos. Esto trae consigo que, en varios momentos del curso, se les solicite a los profesores las evidencias de los resultados en las áreas de resultado clave. La solicitud y recepción de información se realiza mediante el envío de correos electrónicos lo que no asegura el resguardo de esta para futuras consultas.

La Facultad 4 pretende presentarse ante la Junta de Acreditación Nacional para acreditar la carrera de Ingeniería en Ciencias Informáticas. En la Variable dos *"Colectivo revolucionario de excelencia"*, se hace necesario tener evidencias de las actividades y resultados alcanzados por los docentes de hasta cinco años atrás. En el proceso de autoevaluación, se realizó el levantamiento de la información necesaria a partir del currículum vitae y los certificados de constancia de participación en eventos y publicaciones científicas. Sin embargo, durante la recopilación, se han presentado algunas dificultades que limitan la calidad de este proceso:

- ✓ Existen profesores que han solicitado la baja de la institución y no se cuenta con las evidencias y currículum vitae.
- ✓ Los profesores no entregan su currículum actualizado.
- ✓ La entrega de las evidencias de participación en una actividad se realiza en formato duro o vía correo electrónico lo que puede ocasionar la pérdida de esta información.
- ✓ Las evidencias son solicitadas por los interesados solo en un momento específico del trimestre o el año lo que puede ocasionar la pérdida de la evidencia por los profesores o el olvido de su participación en ciertas actividades al no existir un espacio para recopilarlas.

Teniendo en cuenta la situación problemática anteriormente descrita, se plantea como **problema de investigación** la siguiente interrogante:

¿Cómo contribuir con el control de las evidencias de los profesores de la Facultad 4 con vistas a la evaluación profesoral?

Por lo cual el **objeto de estudio** son los procesos de evaluación de desempeño laboral.

Por tanto, el **campo de acción** está enmarcado en los procesos de gestión de evidencias del desempeño laboral de los profesores de la Facultad 4.

Se define como **objetivo general:** Desarrollar un sistema de gestión de evidencias para la evaluación de los profesores de la Facultad 4.

A partir del objetivo general se derivan los siguientes **objetivos específicos:**

- ✓ Analizar elementos teóricos y principales tendencias del desarrollo de sistemas en la actualidad, específicamente en el campo de sistemas de gestión de la información.
- ✓ Definir las tecnologías y herramientas necesarias para el desarrollo de la propuesta de solución.
- ✓ Implementar las funcionalidades de la propuesta de solución.
- ✓ Evaluar la propuesta de solución.

Como **Hipótesis** se plantea: El desarrollo de una aplicación informática de gestión de evidencias del desempeño laboral para la Facultad 4 contribuirá a la gestión de la información en los procesos de otorgamiento de evaluación a los profesores.

En la investigación se utilizó los siguientes **métodos científicos**:

Métodos teóricos

Histórico-lógico: Para el estudio del desarrollo y evolución de los diferentes sistemas de gestión de información similares, nacionales e internacionales, así como las herramientas y tecnologías para el

desarrollo del software, entre ellos los lenguajes de programación, framework de desarrollo, metodologías y herramientas CASE.

Analítico-sintético: En la realización del análisis de la información empleada para la investigación, identificando así, conceptos, definiciones y avances acerca de los sistemas de gestión de información existentes.

Modelación: Se utiliza en la modelación de los diagramas dentro de la metodología de desarrollo de software seleccionada para llevar a cabo la solución.

Métodos Empíricos

Observación: Se utilizó para obtener información de las necesidades existentes en los departamentos de la Facultad 4.

El presente trabajo de diploma cuenta con la siguiente estructura: Introducción, tres capítulos, conclusiones, recomendaciones y por último referencias bibliográficas. A continuación, se presenta un resumen de las diferentes temáticas que se abordan en los capítulos.

Estructura capitular

Capítulo 1: Fundamentación Teórica: En este capítulo se exponen los elementos teóricos utilizados en la investigación, describiendo además las tecnologías, metodologías, herramientas y el lenguaje de programación utilizado en el desarrollo de la solución, así como los principales conceptos involucrados, para una mejor comprensión.

Capítulo 2: Propuesta de solución: En este capítulo se exponen los elementos que permiten describir la propuesta de solución, tales como: modelo de dominio, requerimientos funcionales y no funcionales, diagramas de casos de uso y la descripción textual de cada uno de ellos incluyendo los prototipos de interfaz de usuario para lograr un entendimiento claro del sistema a desarrollar.

Capítulo 3. Construcción de la propuesta de solución: En este capítulo se realiza el modelado de diseño de las funcionalidades y la implementación de la propuesta de solución, donde se obtiene un recurso

informático que brinda los servicios requeridos por el cliente. Se describen además las pruebas realizadas al sistema, para comprobar el correcto funcionamiento de la propuesta de solución.

Capítulo 1 Fundamentación teórica

En la actualidad los sistemas informáticos se han convertido en parte de la vida cotidiana del ser humano, aumentando su productividad personal, académica y laboral. Debido al avance tecnológico durante los últimos años han surgido nuevas herramientas, tecnologías, metodologías que permiten que cada día se desarrolle un software con más calidad, software que facilita el procesamiento de los datos en disímiles escenarios. El presente capítulo aborda los diferentes conceptos asociados al problema a resolver y se describen las diferentes herramientas de software, metodologías y lenguajes de programación que se emplearán en el proceso de diseño y desarrollo de la solución planteada.

1.1 Conceptos asociados a la investigación

Para la comprensión total del trabajo se debe tener en cuenta una serie de conceptos y definiciones asociadas al dominio del problema.

Sistema de gestión de información

Sistema informático que proporciona a directivos de todos los niveles de la organización un conjunto mínimo de información para la planificación, operación, monitoreo, evaluación y la toma de decisiones. (2) Los sistemas de gestión de información permiten la comprensión de cada organización desde un enfoque analítico, evaluador y creativo que permite develar las oportunidades que merezcan ser explotadas y contrarrestar las amenazas, además de establecer los factores que resulten críticos y buscar nuevas oportunidades. (3)

Evidencia

Prueba determinante en un proceso. (4) Una evidencia es una muestra verificada y certera obtenida en una investigación. En la presente investigación las evidencias constituyen la prueba de la participación de los profesores en las diferentes actividades.

Actividad

Facultad de obrar. Prontitud en el obrar. Conjunto de operaciones o tareas propias de una persona o entidad. (4) Se trata de acciones que desarrolla una persona o institución de manera cotidiana.

1.2 Estudio de soluciones similares

Con el desarrollo de las tecnologías y el avance de la informática en el mundo ha surgido la necesidad de informatizar procesos de forma más rápida y sencilla. Los sistemas de gestión de información constituyen hoy día un avance necesario para disímiles organizaciones. Como punto de partida de la investigación fue necesario analizar sistemas similares a nivel nacional e internacional para el desarrollo de la propuesta de solución. A continuación se exponen los resultados de la investigación:

Sistema de información para la gestión del proceso de prácticas profesionales de la Universidad de Bío-bío, institución de educación chilena.

Sistema informático que apoya la gestión de prácticas profesionales de la universidad de Bío-Bío. Dicho sistema contribuye a la gestión de los directivos, profesores, jefes de carrera, además de controlar el desarrollo y evaluación de los estudiantes en las distintas áreas de estudio que realizan en su proceso formativo. El sistema brinda los siguientes beneficios:

- ✓ Mantener el control del desarrollo de los estudiantes.
- ✓ Es una plataforma educativa para el desarrollo de las prácticas profesionales.
- ✓ Propiciar acercamiento con el entorno laboral.
- ✓ Contribuir al desarrollo de competencias profesionales en los estudiantes.
- ✓ Realizar un seguimiento académico de las prácticas.

La plataforma fue desarrollada en PHP5, los clientes pueden acceder con interfaces web desde cualquier punto de la red global de internet. (5)

Sistema de gestión de la información de un Departamento Docente, Universidad de Matanzas

En la Universidad de Matanzas “Camilo Cienfuegos”, se implementó un sistema de gestión de información de un departamento docente desarrollando una Aplicación web y utilizando PHP como lenguaje de programación. Este sistema está constituido por cuatro módulos:

- ✓ Módulo de Capital Humano.
- ✓ Módulo de Formación del Profesional.
- ✓ Módulo de Educación de Postgrado.
- ✓ Módulo de Ciencia y Técnica.

Estos módulos facilitan la gestión de la información del capital humano de un departamento docente. Dentro de ellos quedan registrados los datos personales de los trabajadores del departamento, el cumplimiento del Plan de Trabajo, los objetivos del profesor, sus evaluaciones, entre otros. Se controlan los datos referentes a los asesoramientos a otros profesores. También se desarrolla todo el trabajo de planificación de la carga docente de los profesores y se gestiona la documentación de la educación posgraduada: cursos, maestrías, diplomados, doctorados que se ofertan o son recibidos por profesores del departamento, obteniendo reportes como el plan de postgrados del departamento, el estado de la superación del claustro, su planificación y control. (6)

Sistema de gestión de los recursos humanos (RRHH) del Departamento Productivo Señales Digitales de la Facultad 9

Sistema desarrollado en la Universidad de las Ciencias Informáticas en el año 2010 con el objetivo de gestionar, controlar y centralizar la información de los recursos humanos del departamento productivo Señales Digitales de la Facultad 9. Entre las principales funcionalidades del sistema se encuentran:

- ✓ Gestionar recursos humanos: El sistema permite que se lleve a cabo la inserción, eliminación, la modificación o la simple búsqueda de los recursos humanos dentro del mismo.
- ✓ Generar reportes: El sistema permite que se lleve a cabo la generación de reportes.

- ✓ Gestionar documentos: El sistema permite que sean insertados, eliminados, modificados, revisados o buscados todos los documentos pertinentes.

Como lenguaje de programación del lado del servidor se utilizó PHP y Symfony como marco de trabajo. Se utiliza como gestor de bases de datos PostgreSQL.

En el estudio realizado a los sistemas similares desarrollados, se tuvo en cuenta indicadores tales como, las plataformas compatibles, funcionalidades y licencias de uso. El sistema de información para la gestión del proceso de prácticas profesionales de la Universidad de Bío-bío posee funcionalidades como las descritas anteriormente, que podrán ajustarse a los requisitos del cliente. Sin embargo, este es un software privativo lo que trae consigo gastos por concepto de uso. El sistema de gestión de la información de un Departamento Docente de la Universidad de Matanzas entre sus funcionalidades, se encuentran la gestión de la información del capital humano, entre otras explicadas con anterioridad, sin embargo, este sistema se diseñó específicamente para gestionar información referente a los profesores de los departamentos que trabajan en la Universidad “Camilo Cienfuegos” de Matanzas, además en este sistema no se gestionan las evidencias, no se verifican la participación de un profesor en una actividad, los profesores no pueden hacer reportes personales de sus evaluaciones, actividades o evidencias, entre otros. En el caso del sistema de gestión de los recursos humanos del Departamento Productivo Señales Digitales de la Facultad 9, garantiza la gestión de recursos humanos y la generación de reportes, pero no ofrece otras funcionalidades tales como gestionar Plan de Trabajo, Plan de Resultado, tampoco permite gestionar evidencias, gestionar actividades, además el sistema se diseñó para gestionar información de ese departamento, sus procesos son diferentes a los que necesita gestionar la facultad. A pesar de ello, estos sistemas ofrecen características que servirán de guía y apoyo al sistema que se desea implementar, tales como:

- ✓ Gestionar recursos humanos.
- ✓ Generar reportes.
- ✓ Mantener el control del desarrollo de los profesores.
- ✓ Cumplimiento del Plan de Trabajo.

1.3 Metodologías de desarrollo de software

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo ha evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. Una metodología de desarrollo de sistemas no tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo. (7)

Según la filosofía de desarrollo las metodologías se pueden clasificar en dos grupos. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado. (7)

Metodologías Tradicionales

Las metodologías tradicionales son denominadas, a veces, de forma peyorativa, como metodologías pesadas. Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto. Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, entre otros). (7) Algunas de las metodologías tradicionales son RUP (Rational Unified Process), MSF (Microsoft Solution Framework) entre otras.

Metodologías Ágiles

Este enfoque nace como respuesta a los problemas que puedan ocasionar las metodologías tradicionales. Basan su fundamento en la adaptabilidad de los procesos de desarrollo. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. (7)

El equipo de trabajo seleccionó para el desarrollo de la propuesta de solución trabajar con las metodologías ágiles, ya que estas son adaptables a proyectos pequeños, generan pocos artefactos y el producto final se acerca más a lo que desea el cliente. Algunas de las metodologías ágiles son las siguientes:

SCRUM: Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos.(8) El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración máxima de 30 días. Es una metodología que se usa para minimizar los riesgos durante la realización de un proyecto. Entre las ventajas se encuentran la productividad, calidad y que se realiza un seguimiento diario de los avances del proyecto, logrando que los integrantes estén unidos, comunicados y que el cliente participe en los avances.

Programación extrema (XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP, se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y esfuerzo para enfrentar los cambios.(8) Esta metodología es eficaz en un ambiente dinámico donde persisten altos riesgos técnicos su principal artefacto son las historias de usuarios y aunque estas pueden no estar completamente definidas, no traen consigo grandes dificultades, pues se pueden modificar o sustituir por otras.

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 (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos.(8) La familia de este tipo de metodología requiere del uso de pocos recursos y está enfatizada en la mejora de las competencias del personal, enmarcando sus políticas según el tamaño del equipo de proyecto.

Proceso Unificado Ágil (AUP): Es una versión simplificada de la metodología tradicional Rational Unified Process (Proceso Unificado Racional). Esta define un flujo de trabajo con disciplinas diferentes a las de RUP, aunque conserva sus fases en cada una. La disciplina de modelación incluye la Modelación del Negocio, Requisitos, Análisis y Diseño. Por otra parte, se integran además la Gestión de Cambios y Gestión de Configuración en una sola disciplina.(9) AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Para ello, se crean y mantienen listas identificando los riesgos desde etapas iniciales del proyecto. El proceso AUP establece un Modelo más simple que el que aparece en RUP por lo que reúne en una única disciplina el Modelado de Negocio, Requisitos y Análisis y Diseño.

El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP. Entre las ventajas que brinda está su apoyo a un proceso formalizado evitando las improvisaciones, tiene bien definido los roles y fases, es ágil estando basada en un proceso iterativo evolutivo, incrementa la productividad y facilita el trabajo en proyectos de pequeña envergadura.

De las metodologías ágiles anteriormente mencionadas el equipo de trabajo selecciona AUP, debido a la familiarización que posee con los artefactos que son manejados en las diferentes fases de la misma, al ser los mismos que genera RUP. Entre las ventajas que brinda se encuentran su apoyo a un proceso formalizado evitando las improvisaciones, tiene bien definido los roles y fases, es ágil estando basada en un proceso iterativo evolutivo, incrementa la productividad y facilita el trabajo en proyectos de pequeña envergadura, además describe de una manera simple y fácil la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Entre las ventajas que ofrece está la colaboración con el cliente y el desarrollo incremental del software con iteraciones muy cortas.

1.4 Lenguaje de modelado

UML2.0: Prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan; posibilitando así visualizar, especificar y documentar los artefactos o toda información que se obtiene o modifica durante un proceso de desarrollo de software además de poder utilizarse para modelar distintos tipos de sistemas de software, hardware y organizaciones del mundo real. (10)

Por las características antes explicadas se decide utilizar UML como soporte de la modelación de la solución propuesta dado que los diagramas que provee permiten documentar de manera completa el software a realizar.

1.5 Herramienta CASE

Para el uso de un lenguaje de modelado existen varias herramientas CASE (Ingeniería de Software Asistida por Computadoras) estas son aplicaciones informáticas que traen grandes impactos al proyecto al reducir de forma significativa costes y tiempo de producción al trabajar directamente en cada fase del ciclo de vida de desarrollo del software. Para el desarrollo de la solución se propone Visual Paradigm en su versión 8.0.

Visual Paradigm 8.0: Es una herramienta UML profesional que soporta el ciclo de vida completo de desarrollo de software, es independiente de la plataforma y está dotada de una buena cantidad de módulos para facilitar el trabajo durante la confección de un software. Esta herramienta propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (11)

1.6 Tecnologías y lenguajes

1.6.1 Lenguajes de procesamiento de datos

Son aquellas estructuras formales que, con una cierta base sintáctica y semántica, imparten distintas instrucciones a un programa de computadora. Está formado por un conjunto de símbolos y reglas que definen su estructura y el significado de sus elementos.(12) Para el desarrollo de aplicaciones web existe un gran número de lenguajes de programación, divididos en dos grupos, lenguajes del lado del cliente y lenguajes del lado del servidor. Para el primer grupo son muy utilizados en la capa de presentación o las vistas de los usuarios los lenguajes: XHTML, CSS y JavaScript. El segundo grupo es utilizado en la capa de negocio, la cual se encarga de la implementación de los servicios o funcionalidades que el sistema proveerá al usuario. Los más utilizados son: Python, Ruby, PHP, ASP.NET, Java entre otros. (13)

Java: Es un lenguaje de programación creado por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan en Sun Microsystems en 1991 basado en el lenguaje C++. Es orientado a objetos, interpretado por una máquina virtual, multiplataforma y fuertemente tipado. (14) Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Está diseñado para estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.

Python: Es un lenguaje de alto nivel simple, muy expresivo y legible. En tiempo de ejecución detecta muchos errores y proporciona abundante información para detectarlos y corregirlos. Puede ser usado tanto como lenguaje imperativo procedimental o como lenguaje orientado a objetos. Posee un rico juego de estructuras de datos que se pueden manejar con facilidad. (15)

PHP: Es un lenguaje de programación libre, sencillo, de sintaxis cómoda y similar a otros lenguajes. Es rápido, interpretado, orientado a objetos y multiplataforma. Para este lenguaje se encuentran disponibles una multitud de librerías. (16) Posee una alta capacidad de conexión con la mayoría de los Sistemas Gestores de bases de datos. No requiere de definición de tipos de datos para las variables.

Para el desarrollo de la solución se utilizará Java como lenguaje de programación del lado del servidor debido a que es un lenguaje fácil de utilizar, práctico y con una larga trayectoria, llegando a realizar grandes proyectos en un tiempo relativamente corto. (17) Posee un paradigma puramente orientado a objetos, al ser un lenguaje ya maduro dispone de una amplia comunidad de desarrollo donde se pueden adquirir tutoriales y libros para su aprendizaje. Por otra parte, el equipo de trabajo cuenta con conocimientos sobre el mismo y ha desarrollado varias aplicaciones escritas en este lenguaje.

1.6.2 Framework del lado del cliente

Framework Javascript

JQuery: Es un framework JavaScript, implementa una serie de clases (Programación Orientada a Objeto (POO)) que permite programar sin preocuparse por el navegador con que está visitando el usuario. Ofrece una infraestructura con la que se obtendrá mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. (18)

YUI: El framework o librería YUI (*Yahoo! User Interface*) es un conjunto de utilidades y controles escritos en JavaScript que se utilizan para crear aplicaciones web dinámicas complejas. Además, la librería YUI incluye varias utilidades relacionadas con CSS. Yahoo! distribuye gratuitamente la librería YUI en forma de software libre y bajo la licencia BSD, que permite utilizar YUI para proyectos de cualquier tipo, incluso comerciales. Además, YUI cuenta con una gran documentación y se ejecuta correctamente en todos los navegadores modernos e incluso en algún navegador obsoleto. (19)

Para el desarrollo de la propuesta de solución se selecciona jQuery como framework Javascript en su versión 1.10.2. jQuery es más rápido, ligero y flexible que otros frameworks Javascript, además, el equipo de trabajo posee conocimientos previos de este framework.

Framework CSS

Bootstrap 3.3.1: Es el framework para CSS más popular que existe para el desarrollo web, posee una amplia documentación, haciéndose extensible su uso. Provee de forma fácil las mejores prácticas en el diseño de plantillas, se generan vistas muy agradables y ligeras para el cliente. Es de código abierto, con una comunidad que lo mantiene a través de GitHup. (20)

Blueprint: Es un framework CSS, que tiene como objetivo reducir su tiempo de desarrollo. Proporciona una base sólida para construir proyectos, con una tabla fácil de usar, tipografía sensible, plugins útiles, e incluso una hoja de estilo para la impresión.(21)

Se selecciona Bootstrap en su versión 3.3.1, debido a que está implementado usando jQuery, además, es uno de los frameworks CSS más usados por su amplia documentación.

1.6.3 Framework del lado del servidor

Framework (plataforma, entorno, marco de trabajo): Desde el punto de vista del desarrollo de software, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Estos incluyen las mejores prácticas ya probadas para problemas comunes que surgen en el desarrollo de sistemas, centrándose en lo que realmente le importa al equipo. (22) Para Java, existen varios marcos de trabajo entre los que se destacan Spring MVC, JSF, Vaadin, GWT (Google Web Toolkit), Grails, Play 2, entre otros. (23)

Vaadin 7.6.5: Marco de trabajo para el desarrollo de aplicaciones web escritas en Java, está diseñado para la creación y mantenimiento de sistemas de alta calidad y con interfaces de usuario fáciles de construir. Soporta los modelos de programación del lado del cliente y del servidor, siendo este último su principal fortaleza. Se encarga desde el lado del servidor de la administración de las interfaces de usuario y la comunicación AJAX (JavaScript asíncrono y XML) entre el navegador y el servidor.

Es un framework puramente RIA (Aplicaciones de Internet Enriquecidas, por sus siglas en inglés), elemento que lo hace muy usado hoy día. Posee una arquitectura por capas, donde se separan los elementos del navegador, el servidor y el Back-end (Acceso a datos, Persistencia, Lógica de negocios y Servicios web), permitiendo así una programación segura. (24)

Spring MVC: Es un framework de código abierto, siendo el más utilizado para la creación de aplicaciones empresariales. El objetivo de Spring es simplificar el desarrollo de aplicaciones empresariales Java. A grandes rasgos, un framework es un conjunto de clases que permiten resolver un problema en específico. En el caso particular de Spring, permite resolver muchos de los problemas que se presentan al desarrollar aplicaciones con tecnología JEE (Java Enterprise Edition). Una de las mayores ventajas de Spring, es la forma modular en el que fue creado, permitiendo habilitar/deshabilitar las características a utilizar según se requiera. Spring es utilizado en proyectos muy diversos, como puede ser en instituciones bancarias, aseguradoras, instituciones educativas y de gobierno, entre muchos otros tipos de proyectos y empresas. (25)

JSF: Java Server Faces (JSF) es el marco estándar de interfaz de usuario orientada a componentes para la plataforma Java EE (Enterprise Edition). Es un framework de desarrollo web basado en Java que establece el estándar para la construcción de interfaces de usuario del lado del servidor, establecidas en el patrón MVC (Modelo, Vista, Controlador). JSF permite desarrollar rápidamente aplicaciones de negocio dinámicas en las que toda la lógica de negocio se implementa en Java, o es llamada desde Java, creando páginas para las vistas muy sencillas. (26)

Para el desarrollo de la propuesta de solución el equipo de trabajo decide como framework a utilizar Vaadin ya que es una tecnología moderna para hacer aplicaciones web HTML5, además garantiza la seguridad. Todo el código Vaadin es Java lo que significa que el equipo no necesita aprender nuevos lenguajes ni nuevas API (Interfaz de Programación de Aplicaciones).

1.6.4 Tecnologías para el maquetado

HTML5: Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance. (27)

Es un estándar para HTML. Navegadores como Safari, Chrome, Firefox, Opera, Internet Explorer son compatibles con muchos de los nuevos elementos de HTML5.

CSS3: Las **hojas de estilo en cascada** o **Cascading Style Sheets** (**CSS**) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento. (28)

CSS3 se divide en varios documentos separados llamados módulos. Cada módulo añade nuevas capacidades o amplía funciones definidas en CSS2, sobre la preservación de compatibilidad con versiones anteriores. (29)

Para el maquetado y diseño de la propuesta de solución se utilizará **HTML5 y CSS3.**

1.6.5 Framework para la capa de acceso a datos

Estas tecnologías están centradas en la conversión de las entidades del sistema hacia tablas de las bases de datos y también funciona de forma inversa. A este proceso se le denomina mapeo ORM (Mapeo Relacional de Objetos, por sus siglas en inglés). Entre los marcos de trabajo más destacados para este lenguaje se encuentran Hybernate y JPA (Java Persistence API). (30)

JPA 2.0: Construida por Sun Microsystems para la plataforma Java EE, se utiliza como ORM. Permite elegir cualquier proveedor de persistencia: sea Hybernate, Eclipse Link, entre otros según las necesidades, pero todos trabajarán bajo el mismo API que este provee. Se realiza el mapeo de las entidades definiendo anotaciones sobre sus atributos, indicando las restricciones que estos tendrán a la hora de almacenarse en las tablas correspondientes de la base de datos o por un archivo de configuración en formato XML. Tiene definido un gran número de anotaciones para distintos tipos de datos e incluye soporte para expresiones regulares. (30)

Hibernate: Es una herramienta de Mapeo objeto-relacional para la plataforma Java, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL (Licencia Pública General Reducida de GNU).(31)

Para el desarrollo de la solución será utilizado JPA debido a su facilidad de uso, su alta compatibilidad con otras herramientas, es la API construida para Java EE y está incluida en el estándar EBJ3 (Enterprise JavaBean, estándar de aplicaciones empresariales). (32)

1.6.6 Gestor de base de datos

Se define un Sistema Gestor de Bases de Datos o SGBD, también llamado DBMS (Data Base Management System) como una colección de datos relacionados entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina Base de Datos o BD, (DB Data Base). Para seleccionar un SGBD que sea el más adecuado para desarrollar la propuesta de solución es necesario analizar algunos sistemas, teniendo en cuenta sus principales ventajas y desventajas.

Microsoft SQL Server: Es un gestor creado por la gran transnacional informática Microsoft, posee licencia EULA (Licencia de usuario final, por sus siglas en inglés), la cual es propietaria y utiliza como lenguaje de consultas T-SQL y ANSI-SQL. (33)

Principales aspectos

- ✓ Diseñado para trabajar en entornos híbridos, tanto de forma local o en la nube.
- ✓ Fue creado para soportar bases de datos relacionales.
- ✓ Posee un cliente gráfico que soporta sentencias para lenguaje DML (Lenguaje de Manipulación de Datos) y DDL (Lenguaje de Definición de Datos).
- ✓ Administra información de otros servidores de datos.

Desventajas

- ✓ En comparación con otros gestores, posee una limitada configuración de la memoria compartida para sistemas operativos de 32 bits, pues solo permite hasta 64 GB.
- ✓ No es multiplataforma pues solo funciona sobre Microsoft Windows.

Por los aspectos mencionados, no puede ser elegido para el desarrollo de la solución por las políticas de seguridad que sigue la universidad, al necesitarse un software con licencia libre y que se pueda ejecutar sobre GNU Linux.

MySQL: Es un software libre patrocinado por ORACLE y desarrollado en el lenguaje C/C++. Se utiliza comúnmente por los desarrolladores de aplicaciones web, se puede manejar remotamente desde un cliente web y utiliza para las consultas lenguaje SQL (Lenguaje Estructurado de Consultas). (34)

Principales aspectos

- ✓ Servidores multicapas con diferentes módulos.
- ✓ Implementa funciones SQL usando librerías de clase altamente optimizadas.
- ✓ Para el manejo de los datos utiliza tablas hash, las cuales son usadas como tablas temporales.
- ✓ Soporte B-Tree para el manejo de índices.
- ✓ Posee motores transaccionales y mono transaccionales.
- ✓ Replicación y conectividad segura.

Actualmente es uno de los gestores más robustos, posee pocas desventajas y se han registrado pocos errores. (34) Algunas de las desventajas es que el planificador no actualiza los privilegios cuando se elimina una tabla. Esto puede no parecer peligroso, si el desarrollador se percata y actualiza los privilegios manualmente, pero puede traer errores futuros. Por otra parte, la universidad no cuenta con una comunidad dedicada a la misma.

PostgreSQL: Gestor de base de datos desarrollado por PostgreSQL Global Development Group (PGDG), es multiplataforma y posee licencia BSD, la cual es una licencia libre permisiva. Permite no solo la

implementación de bases de datos relacionales, también simula las multidimensionales. Se puede manejar mediante el cliente de entorno gráfico PGAdmin o mediante un cliente en forma de terminal de texto llamado PSQL, el cual es muy efectivo para servidores con pocos recursos de hardware. (35)

Principales aspectos

- ✓ Números de precisión larga.
- ✓ Para tipos de datos en forma de cadena, permite cadenas de longitud ilimitada.
- ✓ Soporta IP versión 6.
- ✓ Incluye varios métodos para el manejo de índices, utilizando por defecto B-Tree.
- ✓ Permite a los usuarios definir su propio tipo de datos.
- ✓ Utiliza nativamente el lenguaje procedural PL/PGSQL para implementar funciones, además de soportar otros, tales como: C++, Java PL, Java web, PL/Perl, PL/PHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl y PL/Scheme.

Ventajas

- ✓ Seguridad en términos generales.
- ✓ Integridad referencial y afirmaciones (Assertions) y restricciones de dominio.
- ✓ Disparadores (Triggers), autorizaciones, transacciones y respaldos.
- ✓ Existe una amplia comunidad de desarrollo.

Para el desarrollo del trabajo de diploma será necesario el uso del gestor PostgreSQL, debido a que es un software con licencia libre, se dispone de una amplia documentación tanto por parte del equipo de trabajo y la universidad, además de que el equipo está bien adiestrado en el uso de dicha herramienta, al estar incluida en su programa de estudio de la disciplina de Bases de Datos y haber desarrollado varios sistemas de gestión haciendo uso de la misma.

1.6.7 Entorno de desarrollo

Para el trabajo con el framework Vaadin se pueden utilizar tres entornos de desarrollo que lo soportan: NetBeans, IntelliJ IDEA y Eclipse. Estos sistemas son los recomendados para trabajar con el framework y existe amplia documentación para su configuración y uso. (24)

NetBeansIDE 7.3.1

Su aprendizaje se ha convertido en fundamental para quienes están interesados en el desarrollo de aplicaciones multiplataforma. Mediante NetBeans es posible diseñar aplicaciones con solo arrastrar y soltar objetos sobre la interfaz de un formulario. Con NetBeansIDE no solo es posible elaborar potentes aplicaciones de escritorio, también para la Web y para dispositivos portátiles, es un entorno de desarrollo gratuito y de código abierto. Da soporte a las siguientes tecnologías, entre otras: Java, PHP, Groovy, C/C++, HTML5. Además puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS. (36)

Eclipse

Es un IDE desarrollado por la prestigiosa compañía ORACLE basado en código abierto y multiplataforma, permite la creación de múltiples proyectos, herramientas en Java y el desarrollo de complementos (plug-ins), además posee una plataforma de herramientas web. Tiene como principales características la disponibilidad de un editor de texto con resaltado de sintaxis, las compilaciones las realiza en tiempo real y las pruebas unitarias son con JUnit. (37)

Se decide utilizar NetBeans 7.3.1 para el desarrollo del sistema, ya que es libre, multiplataforma y soporta los lenguajes a utilizar para la implementación de la solución. Además, el equipo de desarrollo posee conocimientos y amplia documentación de este IDE.

1.6.8 Herramienta de gestión de proyectos

Maven es una herramienta de gestión de proyectos de software. Basado en el concepto de un modelo de objeto de proyecto, Maven puede gestionar la generación de informes y documentación de un proyecto a partir de una pieza central de la información.(38)

Apache Maven 3.1.0: Repositorio local que se encarga de gestionar librerías, complementos y drivers, permite desde la web incorporar dichos elementos a utilizar mediante la inserción de dependencias al

archivo de configuración en formato XML del proyecto (pom.xml), una vez que está configurado al repositorio institucional. (39)

1.6.9 Servidores de aplicación web

Un servidor web permite procesar aplicaciones de este tipo del lado del servidor, donde se establece una conexión bidireccional entre el cliente que realiza peticiones y el servidor que se encarga de dar respuestas o ejecutarlas. Para Java existen varios servidores, tales como: GlassFish, JBoss y Apache Tomcat.

JBoss: Este servidor actualmente es el que lidera el mercado y al igual que GlassFish posee un certificado de compatibilidad de Java EE, permite el desarrollo de aplicaciones web para sistemas empresariales y servicios de SOA (Arquitectura Orientada a Servicios) y está pensado para la obtención del máximo rendimiento y productividad. (40)

GlassFish: Es un servidor de aplicaciones web basado en código abierto y asistencia gratuita, fue creado inicialmente por Sun Microsystems y luego adquirido por ORACLE. Posee arquitectura modular basada en OSGi (Open Services Gateway Initiative), es multiplataforma y posee un certificado de compatibilidad con Java EE. (41)

Apache Tomcat 7.0.34: Es un servidor de aplicaciones web basado en código abierto para tecnologías Java Servlet 3.0 y Java Server Pages 2.2 (JSP). Está publicado bajo licencia Apache versión 2. Posee una comunidad de desarrollo muy selectiva, utiliza Catalina como contenedor de servlets a partir de la versión 4. Es multiplataforma, siendo compatible con todo sistema operativo que posea instalada la máquina virtual de Java. Permite el monitoreo y la administración remota de las aplicaciones mediante una conexión segura, servicios de virtual hosting, clustering y equilibrado de cargas en las aplicaciones. (42)

Se decide optar por Apache Tomcat debido a que está recomendado para Vaadin 7

1.6.10 Generador de reportes

En los proyectos de informatización de los últimos años, la generación de reportes ha constituido una característica esencial, con gran peso para la toma de decisiones. Compañías de gran prestigio han invertido en desarrollar herramientas que, de una forma amigable, permiten al usuario personalizar sus reportes e

informes. Con estas bondades lo usuarios desconocen qué hay dentro de esas herramientas o cuáles son las librerías más importantes que fueron utilizadas.(43)

iReport: La herramienta iReport es un constructor / diseñador de informes visual, poderoso, intuitivo y fácil de usar para JasperReports, escrito en Java. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, entre otros. iReport está además integrado con JFreeChart, una de las bibliotecas gráficas más difundida para Java. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, TableModels, JavaBeans, XML, entre otros. (44)

iText: Es una biblioteca de código abierto para crear y manipular archivos PDF, RTF, y HTML en Java. Fue escrita por Bruno Lowagie, Paulo Soares, y otros; está distribuida bajo la Mozilla Public License con la LGPL como licencia alternativa. El mismo documento puede ser exportado en múltiples formatos, o múltiples instancias del mismo formato. Los datos pueden ser escritos a un fichero o, por ejemplo, desde un servlet a un navegador web. (45)

iText, una de las librerías Java más utilizadas para la generación de reportes e informes. Se destacan las bondades siguientes: creación del documento, formas de visualización, organización por secciones, capítulos, utilización de diferentes tipologías de letras, inserción de tablas, imágenes y otros. iText es una de las librerías que, sin intermedio de herramientas de diseño, al ser utilizada directamente mantiene gran parte de sus potencialidades. (43)

Para la generación de reportes se seleccionó iText por todas las bondades que brinda especificadas anteriormente.

1.7 Patrón de arquitectura

Cuando se menciona patrón de arquitectura primero se debe analizar el concepto de arquitectura de software, a partir de este se tendrá una visión más clara, para llegar a la definición de patrón de arquitectura y cómo emplearla en el desarrollo de la solución. La arquitectura de software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente y las relaciones entre ellos. (46)

Patrón de arquitectura: Descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma de cómo estos colaboran entre sí. (47)

Para la elección de un determinado patrón de arquitectura se deben tener en cuenta los diferentes atributos de calidad, cada estilo o patrón propicia atributos de calidad en el sistema, pero con la diferencia de que cada uno se enfoca en grupos de atributos diferentes mejorando o disminuyendo algunos de estos.

Arquitectura de tres capas: Es una arquitectura para la construcción de aplicaciones basadas en un entorno cliente-servidor. Se encarga de separar de manera lógica la interfaz de usuario (capa de presentación), el negocio (capa de negocio) y las entidades (capa de datos). La comunicación entre dichas capas se realiza de forma lineal, comenzando desde la capa de presentación donde el usuario realiza las peticiones, estas son procesadas en la capa de negocio y luego los datos son almacenados, consultados y actualizados en la última capa. (48)

Descripción de cada una de las capas

Capa de presentación: Se encarga de mostrar los datos mediante una interfaz gráfica o terminal de texto al usuario, incluyendo además notificaciones de error, formateo de datos, ordenación y validación de los mismos.

Capa de lógica de negocio: Se encarga de la publicación de los servicios del negocio, control de acceso, implementación de las funcionalidades e invoca la persistencia.

Capa de datos: Asegura la integridad y disponibilidad de los datos y gestiona la persistencia.

1.8 Conclusiones parciales

El análisis del estado del arte de los principales conceptos asociados al dominio del problema permitió una mejor comprensión del objeto de estudio. Los sistemas de gestión de información analizados no cumplen con las expectativas del cliente sin embargo se tomaron características comunes para ayudar el desarrollo del mismo. Se seleccionó como metodología de desarrollo ágil AUP, que proporcionará la guía de todo el

proceso de desarrollo. Las tecnologías seleccionadas para llevar a cabo la implementación de la propuesta de solución fueron: Java como lenguaje de programación del lado del servidor, HTML5 y CSS3 para el maquetado. Se selecciona como servidor web Apache Tomcat en su versión 7.0.34 y como framework de desarrollo Vaadin 7.6.5. Además, como IDE se elige NetBeans 7.3.1 y como sistema gestor de base de datos PostgreSQL 9.3.

Capítulo 2 Propuesta de solución

La metodología AUP define en cada una de sus fases un grupo de artefactos que permiten una mejor comprensión del proceso y el sistema a desarrollar, al tener la misma un enfoque ágil permite que el equipo elabore la documentación necesaria que facilite la comunicación entre los desarrolladores y las partes interesadas. En el presente capítulo se exponen los conceptos fundamentales del dominio y se presentarán los siguientes artefactos: Modelo de dominio, la especificación de requisitos funcionales y no funcionales que tendrá la propuesta de solución, se modelarán los casos de uso y la interacción de los actores con el sistema, así como una descripción detallada de los mismos.

2.1 Modelo de dominio

Un modelo de dominio o modelo conceptual es una representación de conceptos del dominio de un problema determinado. Mediante el uso de UML este se puede modelar utilizando un grupo de diagramas de estructura estática que no representa ninguna operación. Permite representar elementos del mundo real y no componentes del software. Provee conocimiento de la nomenclatura del dominio al comunicarle a las partes interesadas los términos importantes y cómo se relacionan entre sí, siendo un artefacto fundamental para realizar el análisis orientado a objetos. (26)

2.1.1 Conceptos del dominio

Para un mayor entendimiento del problema se procede a esclarecer las terminologías asociadas al dominio, definiendo los términos más importantes que le serán de utilidad al equipo de trabajo para la posterior confección del modelo.

Actividad: Se trata de acciones que desarrolla el profesor de manera cotidiana, en este caso de todas aquellas que son previamente coordinadas y que exigen su participación.

Evidencia: Constituyen la prueba de la participación de los profesores en las diferentes actividades.

Plan de Trabajo: Documento donde se registran todas las actividades del mes de un profesor.

Plan de Resultado: Actividades que el profesor debe hacer durante un año fiscal que serán luego evaluadas.

Evaluación profesoral: Evaluación que se realiza anualmente, según los resultados alcanzados durante el año fiscal.

Evaluación de desempeño: Evaluación que se realiza por trimestre, de acuerdo a los resultados alcanzados en el período.

2.1.2 Diagrama del modelo de dominio

El siguiente diagrama muestra la relación entre los conceptos identificados del problema y anteriormente descritos.

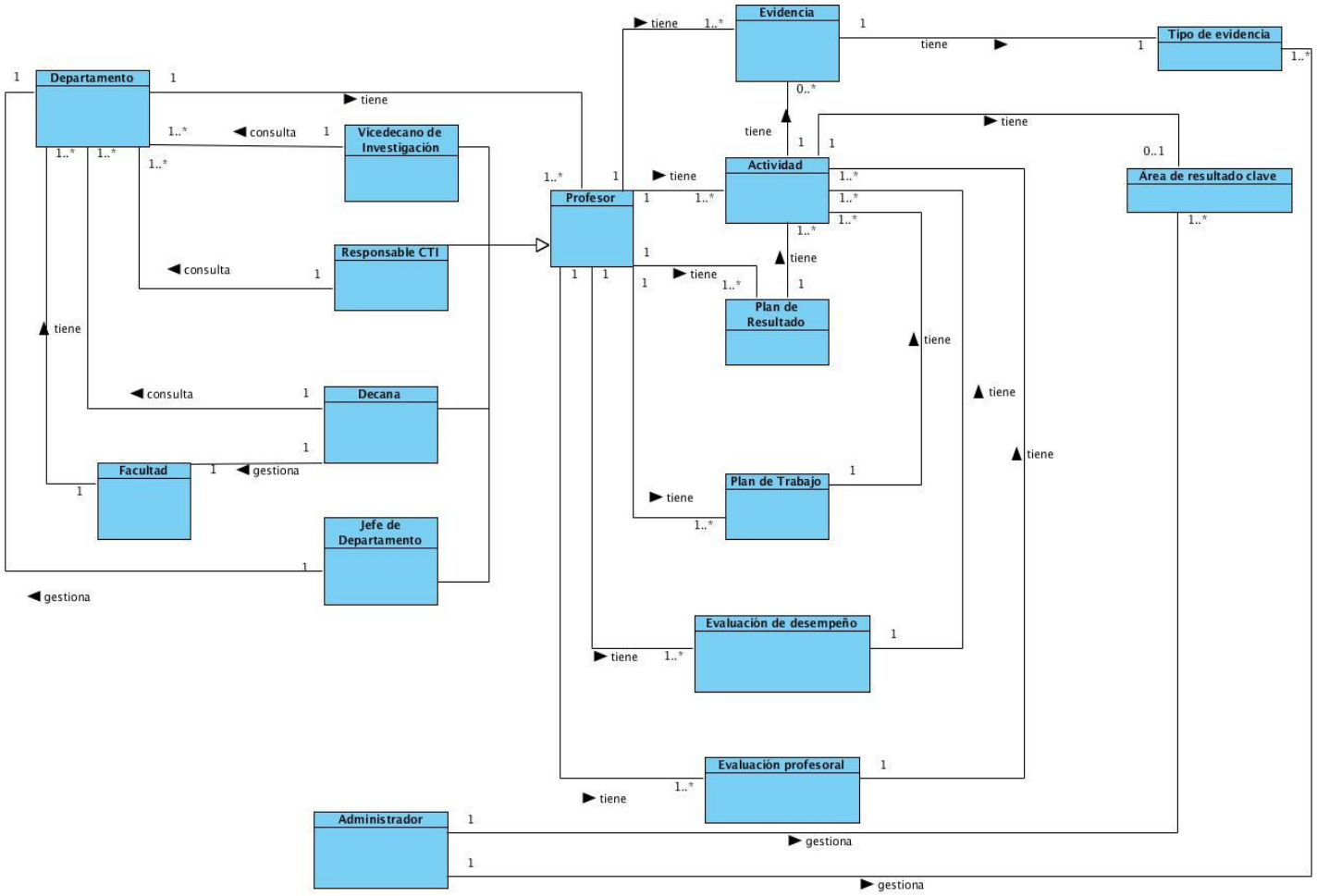


Ilustración 1 Modelo de dominio.

2.2 Requisitos funcionales

Las funcionalidades de un sistema son la descripción de los servicios que ofrecerá el mismo y sus restricciones operativas. Estos son planteados con el objetivo de lograr un entendimiento entre el cliente y el equipo de desarrollo, con el fin de satisfacer sus necesidades.

A continuación, se presentan los requisitos funcionales del sistema:

RF1 Gestionar evidencia:

RF1.1 Insertar evidencia

RF1.2 Modificar evidencia

RF1.3 Visualizar evidencia

RF1.4 Eliminar evidencia

RF2 Gestionar actividad:

RF2.1 Insertar actividad

RF2.2 Modificar actividad

RF2.3 Visualizar actividad

RF2.4 Eliminar actividad

RF3 Gestionar Plan de Trabajo Individual:

RF3.1 Insertar actividad en Plan de Trabajo Individual

RF3.2 Modificar Plan de Trabajo Individual

RF3.3 Visualizar Plan de Trabajo Individual

RF4 Gestionar Plan de Trabajo del Departamento:

RF4.1 Insertar actividad en el Plan de Trabajo

RF4.2 Modificar Plan de Trabajo

RF4.3 Visualizar Plan de Trabajo

RF5 Gestionar Plan de Resultado:

RF5.1 Insertar Plan de Resultado

RF5.2 Modificar Plan de Resultado

RF5.3 Visualizar Plan de Resultado

RF6 Gestionar departamento:

RF6.1 Modificar departamento

RF6.2 Visualizar departamento

RF7 Gestionar facultad:

RF7.1 Modificar facultad

RF7.2 Visualizar facultad

RF8 Insertar área

RF9 Generar reporte CTI

RF10 Generar reporte Evaluación de desempeño

RF11 Generar reporte Evaluación Profesoral

RF12 Generar reporte del Departamento

RF13 Verificar actividad

RF14 Autenticar usuario

RF15 Modificar contraseña

RF16 Importar Plan de Trabajo Individual

RF17 Exportar Plan de Trabajo Individual

RF18 Importar Plan de Trabajo del Departamento

RF19 Descargar archivo de la evidencia

RF20 Visualizar perfil

RF21 Gestionar Evaluación de desempeño:

RF21.1 Adicionar Evaluación de desempeño

RF21.2 Modificar Evaluación de desempeño

RF21.3 Visualizar Evaluación de desempeño

RF22 Gestionar Evaluación profesoral:

- **RF22.1** Adicionar Evaluación profesoral
- **RF22.2** Modificar Evaluación profesoral
- **RF22.3** Visualizar Evaluación profesoral

RF23 Listar departamentos

RF24 Listar evidencia

RF25 Listar actividad

RF26 Listar Plan de Resultado

RF27 Listar Evaluación de desempeño

RF28 Listar Evaluación profesoral

RF29 Gestionar área de resultado clave:

- **RF29.1** Insertar área de resultado clave
- **RF29.2** Modificar área de resultado clave
- **RF29.3** Listar área de resultado clave

RF30 Gestionar tipo de evidencia:

- **RF30.1** Insertar tipo de evidencia
- **RF30.2** Modificar tipo de evidencia
- **RF30.3** Visualizar tipo de evidencia

RF31 Exportar Plan de Trabajo Individual a PDF

RF32 Cerrar Evaluación profesoral

2.3 Requerimientos no funcionales

Los aspectos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. (49) A continuación, se presentan los requisitos no funcionales del sistema:

Usabilidad

El sistema debe poder ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente web en sentido general. También debe satisfacer las necesidades del usuario, de forma tal que se cumpla con las funcionalidades descritas anteriormente.

Seguridad

- ✓ El acceso a la información debe estar restringido por usuario, contraseña y rol.
- ✓ Cuando un usuario se autentique o registre en el sistema se cifrará la contraseña del usuario mediante el algoritmo MD5, tanto para almacenarla en la base de datos como para compararla cuando se inicie sesión.

Portabilidad

El sistema debe ser independiente de plataforma. Debe ejecutarse tanto en Microsoft Windows como en GNU/Linux.

Apariencia o interfaz externa:

- ✓ Debe mostrar claridad y buena organización de la información, que permita la interpretación completa de la misma.
- ✓ El sistema debe notificar al usuario ante la presencia de un error.

Software

PC Cliente

- ✓ Sistema operativo Linux en cualquiera de sus distribuciones y Windows en cualquiera de sus versiones.
- ✓ Navegador Mozilla Firefox versión 19 o superior, Google Chrome versión 28 o superior.

Servidor de aplicación web

- ✓ JRE versión 7 o superior.
- ✓ Apache Tomcat versión 7 o superior.

Servidor de Base de Datos

- ✓ PostgreSQL versión 9.3.

Hardware

PC Cliente

- ✓ Procesador Intel Pentium 4 o superior.
- ✓ Memoria RAM: 512 (mínimo).

Servidor de aplicación web

- ✓ Procesador Intel Core i3 o superior.
- ✓ Memoria física 4GB.
- ✓ Disco Duro 500GB.

Servidor de Base de Datos

- ✓ Procesador Intel Core i3 o superior.
- ✓ Memoria física 4GB.

2.4 Modelo de casos de uso del sistema

El modelado de casos de uso permite documentar el comportamiento del sistema desde el punto de vista del usuario, posibilitando representar los requisitos funcionales que debe ejecutar el software y los usuarios que interactúan con este. Resultan muy útiles en la comunicación con el cliente, debido a su fácil interpretación y sirven de base para la ejecución de futuras fases en el proceso de desarrollo de software. (50)

2.4.1 Descripción de los actores del sistema

Tabla 1 Actores del sistema y su descripción

Actor	Descripción
Administrador	Actor que tendrá acceso a algunas funcionalidades como Gestionar tipo de evidencia, Gestionar área de resultado clave, Insertar área.

Profesor	Actor que tendrá acceso a algunas de las funcionalidades del sistema tales como: Autenticarse, Gestionar evidencia, Gestionar actividad, Gestionar Plan de Trabajo Individual, Generar reporte CTI, Generar reporte Evaluación de desempeño, Generar reporte Evaluación Profesoral, Importar Plan de Trabajo Individual, Exportar Plan de Trabajo Individual, Visualizar Perfil.
Encargado de CTI	Actor que tendrá acceso a las funcionalidades del profesor y también podrá Visualizar departamento y generar reportes de CTI de cualquier integrante del departamento.
Jefe de Departamento	Actor que tendrá acceso a las funcionalidades del encargado de CTI y también podrá Gestionar departamento, Gestionar Plan de Trabajo del Departamento, Gestionar Plan de Resultado, Verificar actividad, generar reportes de cualquier integrante del departamento, Importar Plan de Trabajo del Departamento y Cerrar evaluación profesoral.
Vicedecano de Investigación y Posgrado	Actor que tendrá acceso a las funcionalidades del profesor además podrá Visualizar departamento y Listar departamentos.
Decano	Actor que tendrá acceso a las funcionalidades de Vicedecano de Investigación y Postgrado además, podrá Gestionar la facultad y generar reportes de cualquier profesor de la facultad.

2.4.2 Diagrama de casos de uso del sistema

Este artefacto permite mostrar la relación entre los actores y los casos de uso del sistema, siendo el actor el que inicializa estos casos de uso.

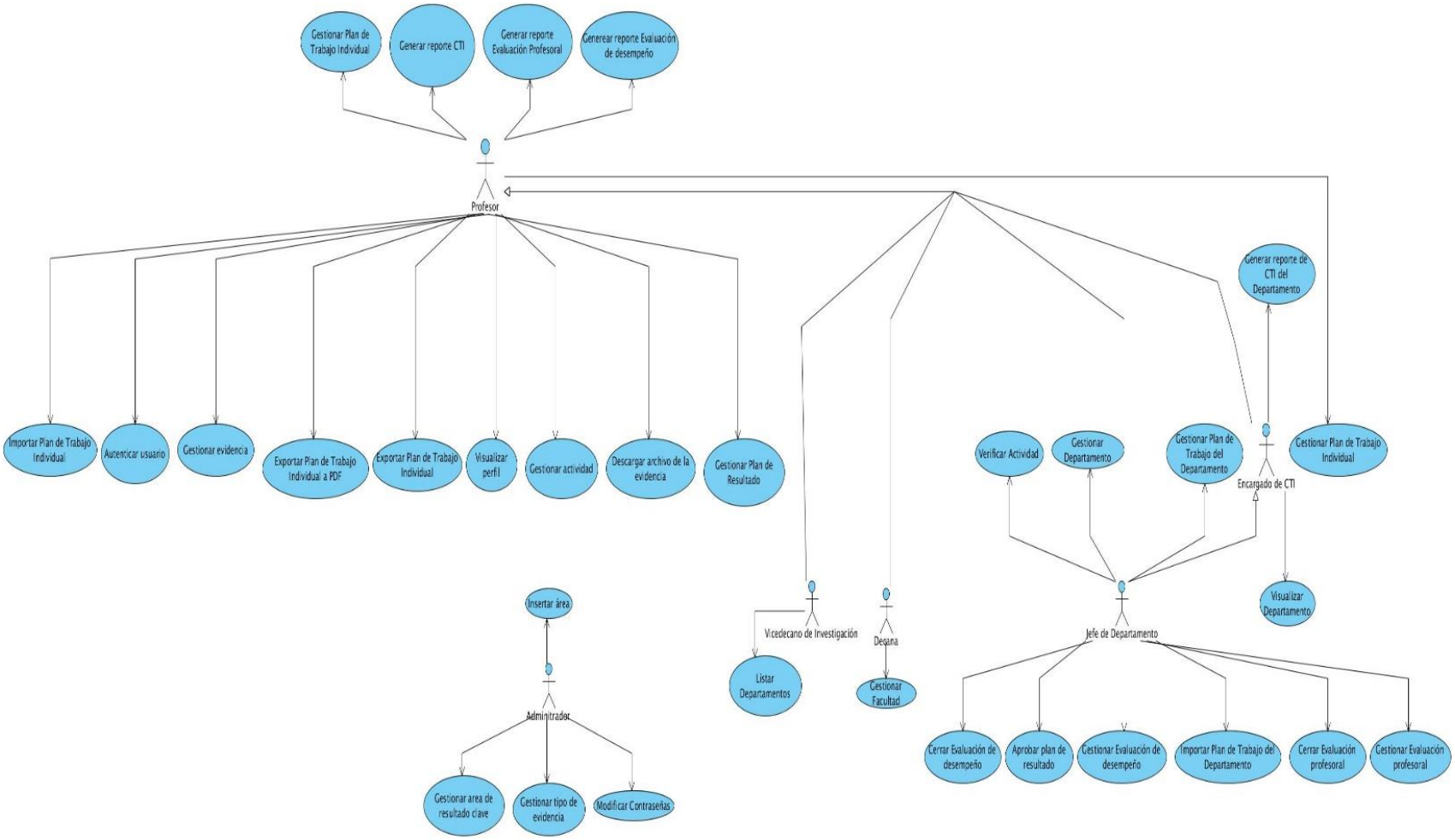


Ilustración 2 Diagrama de casos de uso del sistema.

Nota: Los casos de uso Listar entidad no se incluyeron en el diagrama, para una mejor comprensión del mismo.

2.4.3 Patrones de casos de uso

Al modelado por casos de uso se le aplican un grupo de patrones que permiten simplificar el diagrama y mejorar su comprensión, los mismos describen diferentes soluciones para el modelado de problemas específicos, atendiéndolos por separado. (51)

- ✓ **Patrón CRUD:** Es un patrón que permite agrupar las operaciones de crear, leer, actualizar y eliminar piezas de información de un tipo o entidad. El mismo puede ser completo o parcial, dependiendo si se agrupan o no todas estas operaciones en un mismo caso de uso. (51)
- ✓ **CRUD completo:** Se puede observar su aplicación en los casos de uso Gestionar evidencia y Gestionar actividad, donde se incluyen todas las operaciones descritas anteriormente.
- ✓ **CRUD parcial:** Se puede observar su aplicación en los casos de uso Gestionar Plan de Trabajo del Departamento, Gestionar Plan de Resultado, entre otros.

Patrón múltiples actores:

- ✓ **Rol común:** Permite que dos actores desempeñen el mismo rol, para que a través de este puedan acceder a un determinado caso de uso. (51) Se puede apreciar este patrón en los roles Profesor y los roles Decana, Vicedecano de Investigación y Posgrado, Jefe de Departamento.

2.4.4 Descripción textual de los casos de uso del sistema

A continuación, se muestra la descripción textual del caso de uso Gestionar evidencia. Las restantes descripciones textuales están presentes en el Anexo 1.

Tabla 2 Descripción textual del caso de uso Gestionar evidencia.

Caso de uso	Gestionar evidencia
Actores	Usuarios
Resumen	El caso de uso inicia cuando el actor decide insertar, editar o eliminar una evidencia. En caso de que el actor seleccione la opción Adicionar evidencia, el

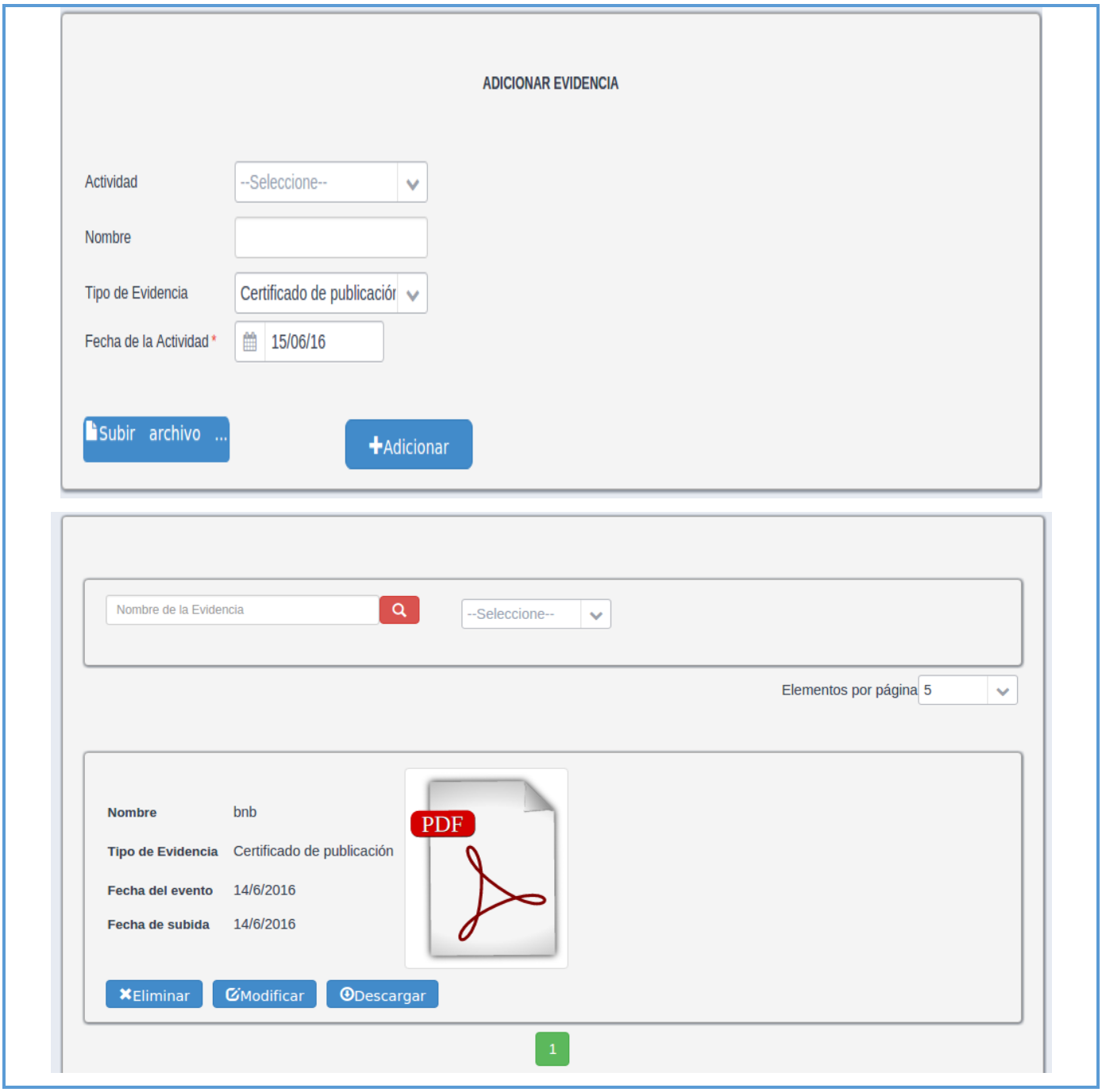
	sistema proveerá un formulario con los campos necesarios para llenar los datos de una evidencia, además permite cargar la evidencia en forma de archivo, permitiendo completar la operación. Permite además editar los datos de una evidencia si el usuario lo desea y brinda la posibilidad de eliminar una evidencia haciendo clic en el botón eliminar disponible en cada evidencia, terminando así el caso de uso.
Precondiciones	El Usuario debe estar autenticado.
Postcondiciones	Se creó, se eliminó, se visualizó o se editaron los datos de una evidencia.
Referencias	RF1, RF1.1, RF1.2, RF1.3, RF1.4
Flujo de eventos	
Flujo básico	
Acciones del Actor.	Respuesta del sistema.
1. El caso de uso inicia cuando el usuario selecciona la opción Adicionar evidencia del menú Evidencia.	
	2-Se muestra un formulario con los siguientes campos: • Nombre del evento Permite elegir: • Tipo de evidencia • Fecha de la actividad

	• Actividad del Plan de Trabajo Y las opciones: • Subir archivo • Adicionar
3-Llena los campos requeridos, selecciona el tipo de evidencia y la fecha de la actividad y ejecuta la opción subir archivo.	
	4-Abre una ventana que permite explorar y elegir unidades y archivos disponibles en el ordenador.
5-Selecciona el archivo deseado y ejecuta la opción Aceptar.	
	6-Carga el archivo elegido a una carpeta local del servidor y lo muestra en el formulario.
7-Selecciona la opción Adicionar.	
	8-Valida los datos.
	9-Registra una nueva evidencia en el sistema.
	10-Muestra una vista con las evidencias adicionadas en el sistema por el usuario
	11-Termina el caso de uso.
Flujos Alternos	
8. a Existen campos vacíos.	
Acciones del Actor	Respuesta del sistema
	8. a.1 Muestra un mensaje informando: “El campo no puede ser vacío”.
	8. a.2 Muestra el indicador de los campos que no pueden ser vacíos.

	8. a.3 Retorna al paso 2.
8.b No se ha elegido Fecha de la actividad	
Acciones del Actor	Respuesta del sistema
	8. b.1.Muestra el indicador de los campos requeridos.
	8. b.3 Retorna al paso 2.
8.c No se ha subido ningún archivo	
Acciones del Actor	Respuesta del sistema
	8. c.1 Muestra un mensaje informando: “Debe subir un archivo”.
	8. c.2 Retorna al paso 2.
Sección1:Modificar datos de una evidencia	
Flujo básico	
Acciones del Actor	Respuesta del Sistema
1-Selecciona la opción Listar disponible en el menú Evidencias.	
	2-Muestra un listado con las evidencias disponibles en el sistema y por cada uno de ellos permite: • Eliminar Ver sección 2: “Eliminar evidencia”. • Modificar • Descargar Ver CU: “Descargar archivo”.
3. Selecciona la opción Modificar de una evidencia deseada.	

	4-Abre una vista con los datos de la evidencia correspondiente, permitiendo modificar los valores del mismo: • Nombre del evento Permite elegir: • Tipo de evidencias • Fecha de la actividad Y las opciones: • Actualizar • Descargar
5. Modifica los datos deseados y selecciona Actualizar.	
	6. Valida los cambios realizados.
	7. Actualiza la evidencia en el sistema.
	8. Muestra el mensaje: “Evidencia actualizada correctamente”.
	9. Termina el caso de uso.
Flujo alterno	
6. a Existen campos vacíos.	
Acciones del Actor	Respuesta del sistema
	6. a.1 Muestra un mensaje informando: “El campo no puede estar vacío”.
	6. a.2 Muestra el indicador de los campos que no pueden estar vacíos.

	6. a.3 Retorna al paso 4.
6.b No se ha elegido Fecha de la actividad	
Acciones del Actor	Respuesta del sistema
	6. b.1 Muestra el indicador de los campos requeridos.
	6. b.2 Retorna al paso 4.
Sección 2: Eliminar evidencia.	
Flujo básico	
Acciones del Actor	Respuesta del Sistema
1-Selecciona la opción Listar disponible en el menú Evidencias.	
	2-Muestra un listado con las evidencias disponibles en el sistema y por cada uno de ellos permite: • Eliminar • Modificar Ver sección 1: “Editar evidencia”. • Descargar Ver CU: “Descargar archivo”.
3. Selecciona la opción Eliminar de la evidencia deseada.	
	4. Elimina la evidencia del sistema. 5. Muestra el mensaje: “Evidencia eliminada satisfactoriamente”. 6. Termina el caso de uso.
Prototipos de interfaz	


ADICIONAR EVIDENCIA
Actividad
--Seleccione--
Nombre
Tipo de Evidencia
Certificado de publicació
Fecha de la Actividad *
15/06/16
Subir archivo ...
Adicionar
Nombre de la Evidencia
--Seleccione--
Elementos por página 5
Nombre
bnb
Tipo de Evidencia
Certificado de publicación
Fecha del evento
14/6/2016
Fecha de subida
14/6/2016
PDF
Eliminar
Modificar
Descargar
1

2.5 Conclusiones parciales

El estudio realizado de los conceptos asociados al negocio permitió comenzar a desarrollar la propuesta de solución del sistema. La confección del modelo de dominio tributó a la comprensión del entorno en que se relacionan las funcionalidades. Los requisitos funcionales y no funcionales ayudaron a definir las capacidades y cualidades que debe tener el sistema. Mediante la realización del diagrama de casos de usos del sistema se establecieron las relaciones entre los actores y los casos de uso. Los artefactos generados: el modelo de dominio, la especificación de casos de uso y otros mencionados anteriormente servirán de punto de partida para el posterior diseño.

Capítulo 3 Desarrollo de la propuesta de solución

La disciplina de modelado de la metodología AUP comprende las fases de modelado del negocio, requisitos, análisis y diseño. Este modelado es colaborativo, generando los artefactos necesarios para la comprensión del sistema con un mayor nivel de abstracción y una vez obtenidos dichos elementos se da paso a la implementación y las pruebas. En el presente capítulo se desarrollará la propuesta de solución, donde se elaborará el modelado de diseño del sistema debido a que el equipo de trabajo y el cliente poseen una total claridad de los requerimientos, por lo que se generarán los diagramas de clases, diagramas de secuencia y diagrama de despliegue. Para la ejecución de la implementación y las pruebas será necesario la obtención de los diagramas de componentes y el diseño de los casos de prueba, con esto se implementará el sistema y luego se someterá a evaluación para validarlo.

3.1 Patrones de diseño

Los patrones describen un problema que ocurre una y otra vez en el entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada en otras ocasiones. (52) Un patrón de diseño es una descripción de un problema, este no es más que la solución estándar para un problema común de programación. Para que la forma de resolver un problema sea considerada un patrón debe poseer ciertas características. Una de ellas es que se debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores.

El equipo de desarrollo decide emplear en el diseño, los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades, siglas en inglés), los cuáles facilitarán la implementación del sistema, la reutilización de código y la obtención de un software con alto grado de mantenibilidad.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, define un grupo de principios fundamentales para diseñar eficazmente un software. (10) Para la propuesta de solución se utilizan los siguientes:

Creador: Se emplea cuando se le aplica la responsabilidad a una clase determinada de crear una o más instancias de otra. Esto sucede en caso de que la clase creadora contenga, agregue, registre, utilice o posea

datos de inicialización de objetos de alguna clase determinada. (10) El empleo de este patrón se ve reflejado en las clases: **ControladorJPAGenericoImpl** y **ServiciosImpl**.

Experto: Se basa en asignar la responsabilidad a una clase que se encargará de ser experto de la información, la misma cuenta con la información necesaria para cumplir con la responsabilidad. (10) Este patrón se evidencia en las clases que extienden de la clase Entidad, las cuales son expertas en su propia información, tales como: **Actividad, Departamento y Evidencia**.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. (10) Este patrón se observa en la clase **ContoladorJPAGenericoImp**l.

Controlador: Asigna la responsabilidad del manejo de mensajes de los eventos de un sistema a una clase que represente el sistema global, organización, elemento activo del mundo real o manejador artificial de los eventos del sistema. (10) Este patrón se evidencia en las clases controladoras creadas para realizar las operaciones de gestión sobre las entidades. En este caso se observa en la clase **ControladorJPAGenerico**.

3.1.1 Patrones estructurales

Estos patrones de diseño fueron concebidos para resolver problemas de agregación y composición de las relaciones de clases. (53)

Patrón Fachada: Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema, se define una interfaz de alto nivel para facilitar el uso del subsistema, sirviendo de intermediaria entre el cliente y un grupo de interfaces más complejas. (54) Este patrón se evidenciará en la implementación, facilitando el uso de las clases controladoras de la capa del negocio y los controladores genéricos para JPA, concentrando el acceso a sus métodos hacia la clase **FachadaImpl**. Esto permite una mejor comprensión a los programadores sobre el código que usan, las librerías y clases, haciendo más flexible el desarrollo.

3.1.2 Patrones de creación

Singleton: Garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella.(55) Este patrón se evidencia en la clase **EviPro.java.**

3.2 Modelado de diseño

El modelo de diseño crea un modelo de software enfocado en la representación de los datos, las funciones y el comportamiento requerido. Permite al ingeniero de software modelar el sistema o producto que se va a construir, posibilitando evaluar su calidad y efectuar mejoras antes de generar el código. Con este se obtiene una representación arquitectónica, de interfaz y despliegue del sistema. (46)

3.2.1 Diagramas de clases del diseño

Estos diagramas representan la vista estática del sistema y modelan los conceptos asociados al dominio de la aplicación, así como los conceptos internos definidos para la parte de la implementación. No se describe el comportamiento del sistema dependiente del tiempo y se representa mediante clases y sus relaciones. (56)

Se presenta a continuación el diagrama de clases del diseño del caso de uso **Gestionar evidencia**.

Ilustración 3 Gestionar evidencia.

3.2.2 Diagramas de secuencia

Un diagrama de secuencia muestra un conjunto de mensajes, dispuestos en una secuencia temporal, donde cada rol en la secuencia se muestra como una línea de vida que representa el rol durante cierto plazo del tiempo, con la interacción completa. Los mensajes se muestran como flechas entre las líneas de vida. Un diagrama de secuencia puede mostrar un escenario, es decir una historia individual de una transacción. (56)

A continuación, se muestran los diagramas de secuencia del caso de uso Gestionar evidencia, donde se observan las interacciones entre los objetos mediante transferencias de mensajes. El resto de los artefactos se encuentran en los Anexos.

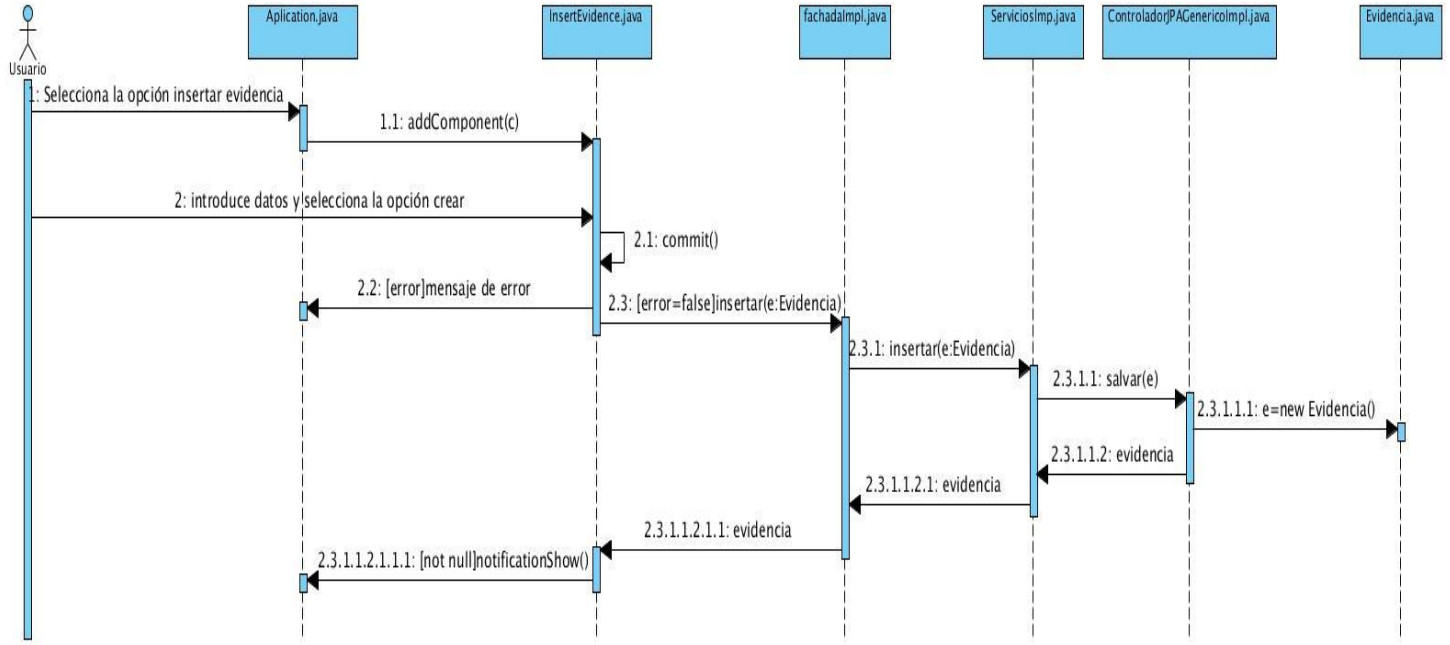


Ilustración 4 Gestionar evidencia-Modificar.

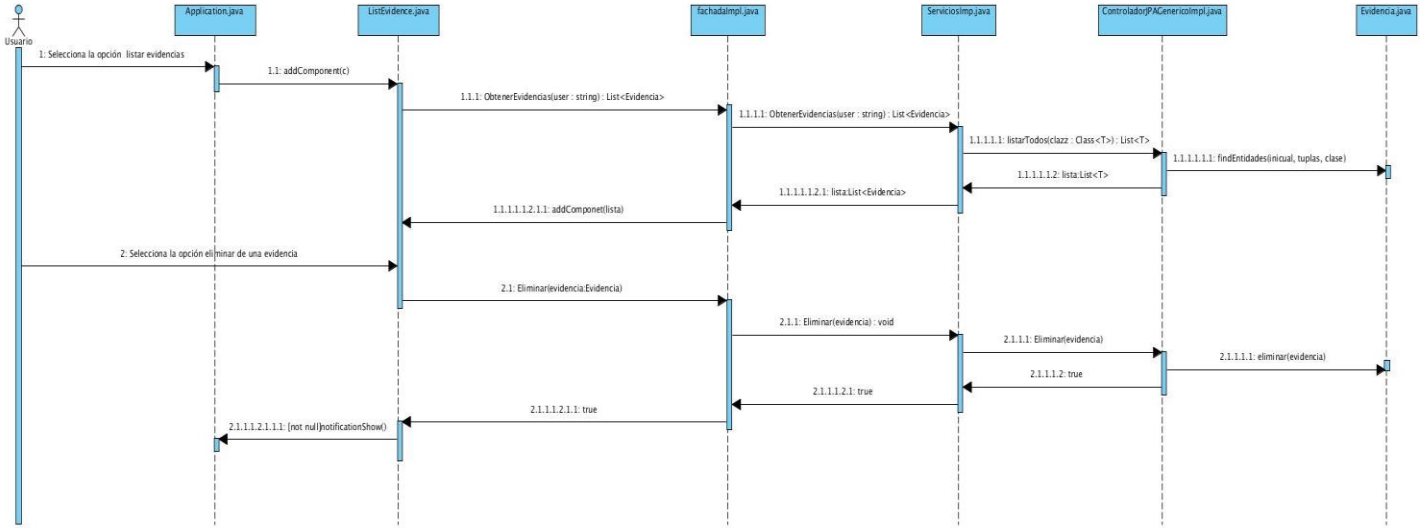


Ilustración 5 Gestionar evidencia-Eliminar.

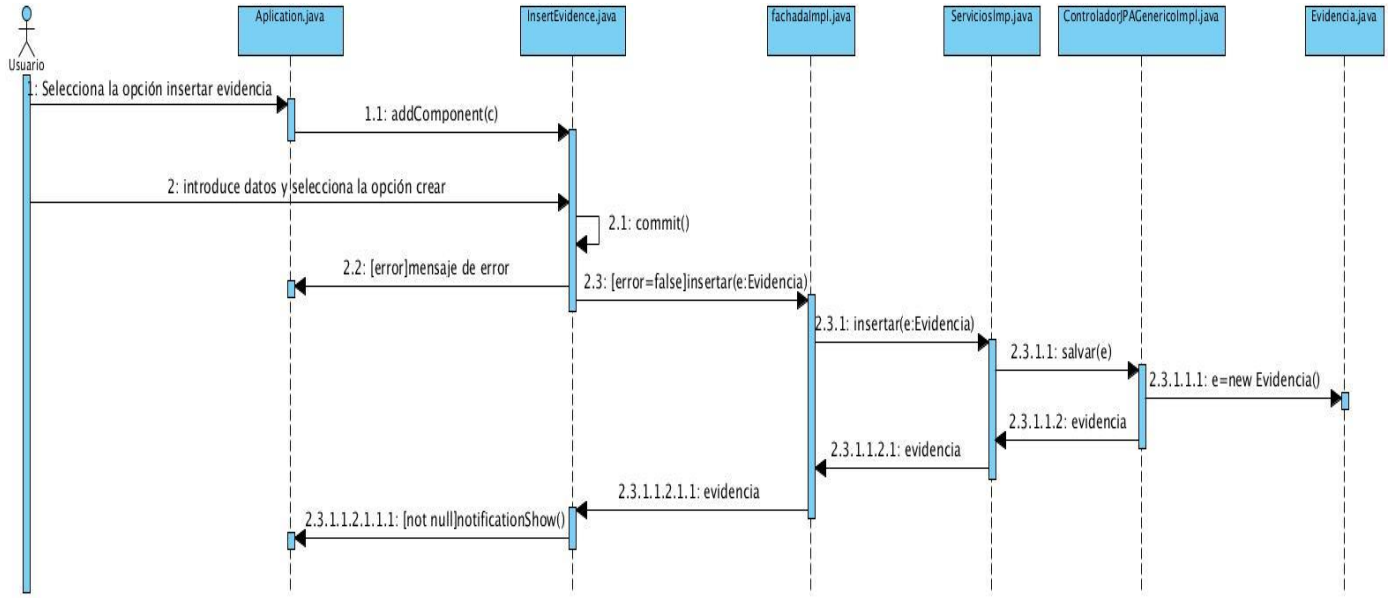


Ilustración 6 Gestionar evidencia-Insertar.

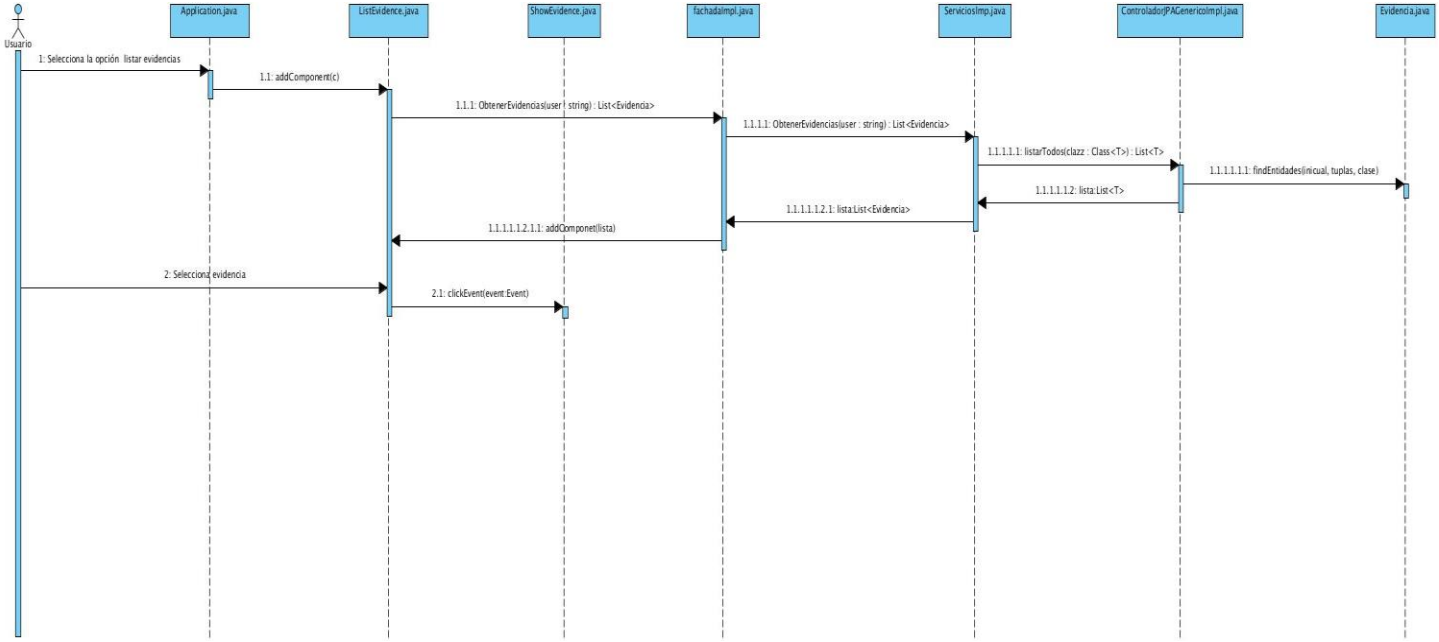


Ilustración 7 Gestionar evidencia-Visualizar.

3.2.3 Vista de despliegue

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos. Un nodo es un recurso de ejecución, tal como una computadora, un dispositivo o una memoria. Esta vista permite determinar la secuencia de distribución y asignación de recursos. Para su representación se emplea el artefacto diagrama de despliegue. (56)

A continuación, se muestra el diagrama de despliegue del sistema:

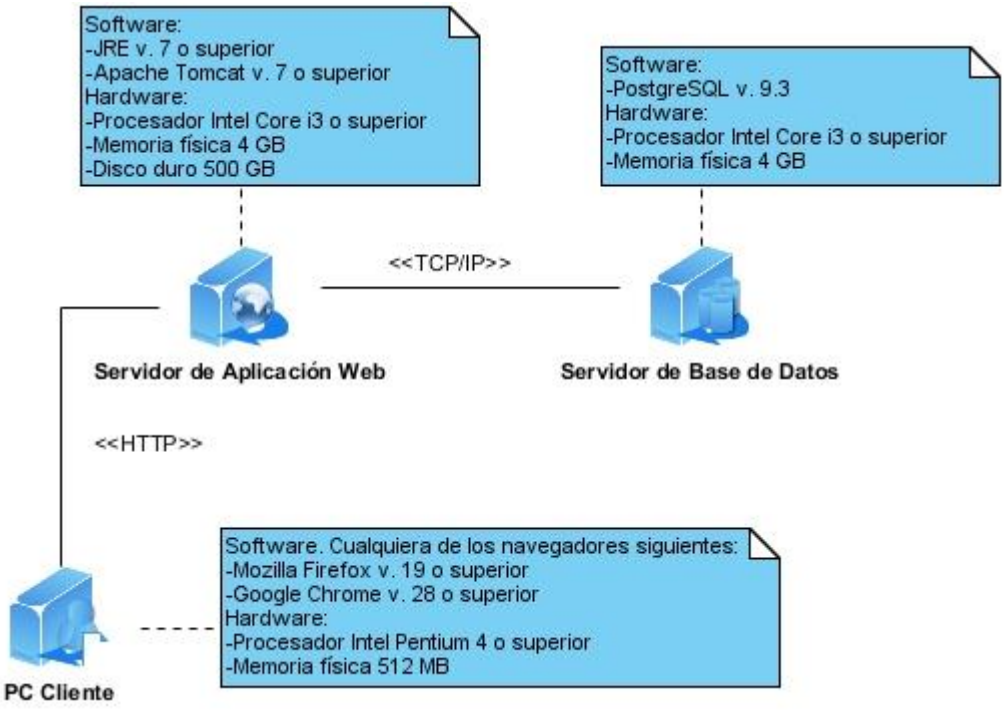


Ilustración 8 Diagrama de despliegue del sistema.

3.2.4 Modelo de implementación

El modelo de implementación describe cómo los elementos del diseño se implementan en términos de componentes; representa ficheros de código abierto, ficheros de código binario, scripts, ejecutables, entre otros. El modelo de implementación es la entrada principal de las etapas de prueba que siguen a la implementación. Más concretamente, durante la etapa de prueba cada construcción generada durante la implementación es sometida a pruebas de integración, y eventualmente también a pruebas de sistema. (57)

Como parte del modelo de implementación se obtiene el diagrama de componentes que se presenta a continuación.

3.2.5 Diagrama de componentes

El diagrama de componentes muestra cómo el sistema está dividido en componentes y las dependencias entre ellos. Estos pueden ser utilizados para documentar y modelar cualquier arquitectura del sistema. Los

componentes son implementados por varias clases u objetos del sistema, se considera una unidad autónoma que provee una o más interfaces las cuales representan un contrato de servicios que el componente ofrece. Los componentes pueden ser archivos, códigos fuentes, ejecutables y paquetes. El diagrama de componentes provee una vista arquitectónica de alto nivel del sistema. Ayuda a los desarrolladores a visualizar el camino de la implementación. Permite tomar decisiones respecto a las tareas de implementación. (57)

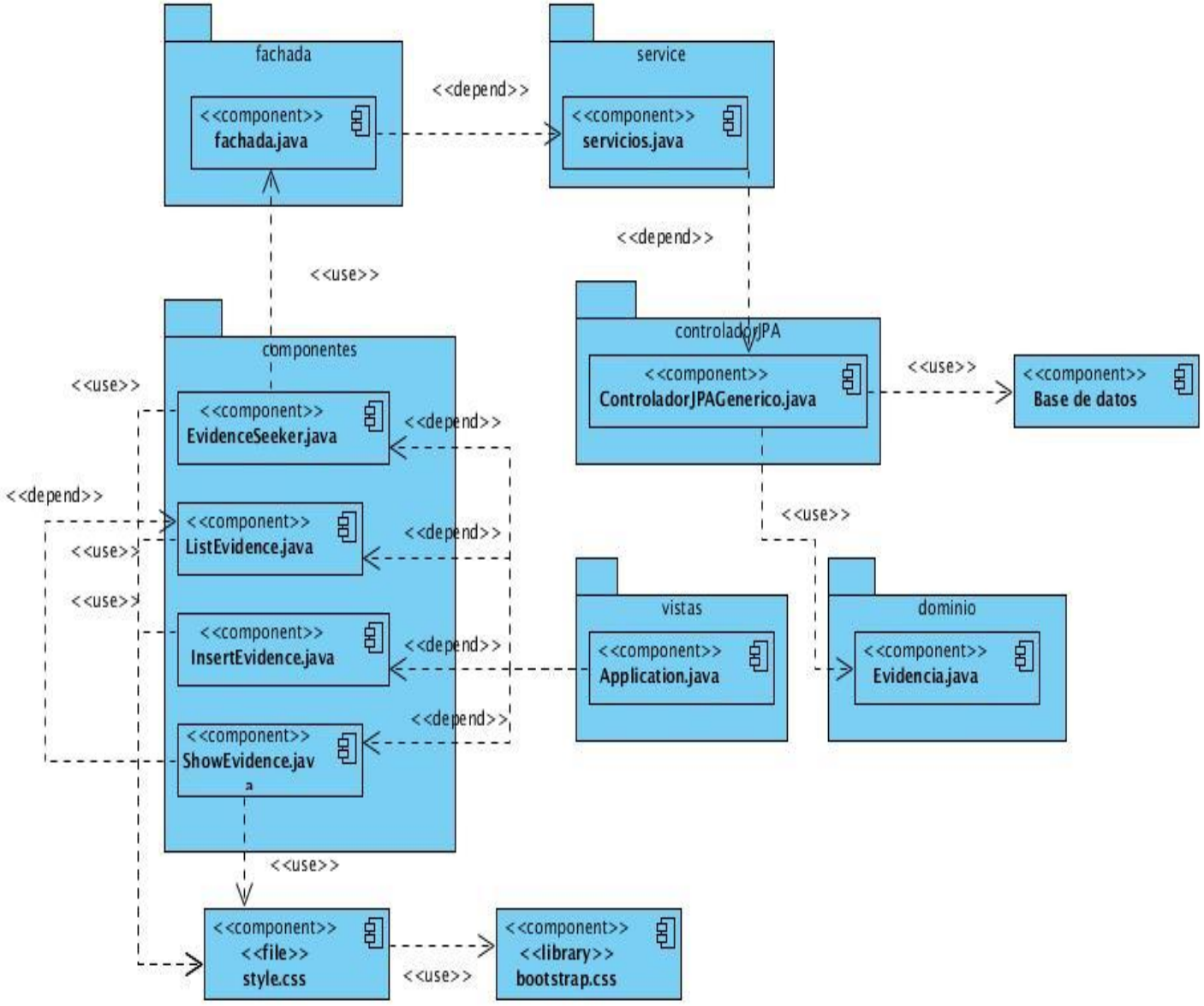


Ilustración 9 Gestionar evidencia.

3.3 Pruebas de software

Las pruebas del software son una técnica dinámica de validación y verificación (V&V), las cuales implican ejecutar una implementación del software con datos de prueba, examinando las salidas del software y su entorno operacional para comprobar que funciona tal y como se requiere. (58)

Una vez generado el código fuente, es necesario probar el software para descubrir y corregir la mayor cantidad de errores posible antes de entregarlo al cliente. Para lograr contribuir con la calidad del software es recomendable que el producto sea evaluado en la medida en que se va construyendo. Posteriormente, se hace necesario llevar a cabo, paralelo al proceso de desarrollo, un proceso de evaluación y comprobación de los distintos productos o modelos que se van generando, en el que participarán desarrolladores y clientes. Las pruebas de software constituyen un pilar indispensable para evaluar y determinar la calidad de un software.(57)

3.3.1 Niveles de prueba

El proceso de pruebas se realiza en varios niveles, estos se enfocan a determinados objetivos y están estrechamente relacionados con los tipos de pruebas. Seguidamente se muestran algunos tipos de niveles de pruebas para valorar cuáles se van a utilizar:

Prueba de sistema: Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. (59)

Pruebas unitarias: En este nivel se verifican por separados las piezas de software en un funcionamiento aislado, estas piezas pueden ser módulos individuales, subprogramas y componentes. Pueden llevarse a cabo con herramientas de depuración, acceso al código fuente y pueden participar en esta de forma opcional los programadores. (60)

Prueba de integración: Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, realizada en

conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes del software funciona junto. (57)

Pruebas de Aceptación: Son realizadas por el usuario final en lugar del responsable del desarrollo del sistema. Una prueba de aceptación puede ir desde un informal paso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. Además, se llevan a cabo con el objetivo de que el cliente valide todos los requisitos que debe poseer el sistema. (61)

Para la presente investigación, no se ejecutan pruebas de integración dado que al sistema no se le incorpora ningún módulo, componente o sección que se haya desarrollado de forma ajena a la solución. En el caso de las pruebas de unidad no se realizan mediante el uso de una herramienta informática pues solo se efectuaron revisiones del código por parte de los desarrolladores. Son las pruebas de sistema las que permitirán comprobar el cumplimiento de los requisitos funcionales y no funcionales por estar enfocadas a verificar cómo trabaja el software una vez concluido su desarrollo.

3.3.2 Métodos de prueba

Se puede probar cualquier producto de ingeniería de dos formas: Conociendo la funcionalidad específica para la cual fue diseñado, demostrando mediante pruebas que cada funcionalidad es operativa o conociendo el funcionamiento del producto, donde cada componente interno trabaja de la forma adecuada. Al primer enfoque se le denomina prueba de caja negra y al segundo prueba de caja blanca. (46) Para la validación de la propuesta de solución será aplicado el método de caja negra, debido a que este permitirá corregir problemas de rendimiento, en la interfaz del usuario y se comprobará que la propuesta de solución realiza las funciones requeridas por el usuario.

Prueba de caja negra: Se realizan las pruebas sobre la interfaz del programa, entiéndase por interfaz de entrada y salida del programa, no requieren del conocimiento de la lógica del programa y solamente se debe conocer la funcionalidad que este debe realizar. (62)

3.3.3 Partición equivalente

La partición equivalente es una técnica de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. Un caso de prueba

ideal de manejo simple descubre una clase de errores; como, el procesamiento incorrecto de todos los datos de caracteres; que, de otra forma, requeriría la ejecución de muchos casos antes de que se observe el error general. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse.

El diseño de casos de prueba para partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana. (57)

3.3.4 Diseño de casos de prueba

El diseño de casos de prueba es una parte de las pruebas de componentes y sistemas en las que se diseñan las entradas y las salidas esperadas para probar el sistema, el objetivo que persigue es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface los requerimientos.(58)

A continuación, se muestra el diseño de casos de prueba del caso de uso Gestionar evidencia.

Tabla 3 Diseño de casos de prueba del caso de uso Gestionar evidencia.

CP Gestionar evidencia
Descripción general El caso de uso inicia cuando el actor decide insertar, editar o eliminar una evidencia. En caso de que el actor seleccione la opción Adicionar evidencia, el sistema proveerá un formulario con los campos necesarios para llenar los datos de una evidencia, además permite cargar la evidencia en forma de archivo, permitiendo completar la operación. Permite además editar los datos de una evidencia si el usuario lo desea y brinda la posibilidad de eliminar una evidencia haciendo clic en el botón eliminar, disponible en cada evidencia, terminado así el caso de uso. **Condiciones de ejecución** El usuario debe estar autenticado.

SC 1 Insertar evidencia						
Escenario	**Descripción**	**Nombre del evento**	**Tipo de evidencia**	**Fecha de la actividad**	**Respuesta del sistema**	**Flujo Central**
EC1.1	Selecciona la opción Adicionar evidencia del menú Evidencia.				El sistema brinda la posibilidad de insertar los siguientes datos: ✓ Nombre del evento Permite seleccionar: ✓ Actividad del Plan del departamento ✓ Tipo de evidencia ✓ Fecha de la actividad Y las opciones: ✓ Subir archivo ✓ Adicionar	Evidencia/ Adicionar
EC1.2	Ejecuta la opción subir archivo.				Abre una ventana que permite explorar y elegir unidades y archivos disponibles en el ordenador.	Evidencia/ Adicionar / Subir archivo
EC1.3	Selecciona el archivo deseado y ejecuta la				Carga el archivo elegido a una carpeta local del servidor y lo	Evidencia/ Adicionar/ Subir archivo

	opción Aceptar				muestra en el formulario	/Aceptar
EC 1.4	Introduce datos y selecciona la opción Adicionar	V	V	V	Valida los datos introducidos. Registra una nueva evidencia en el sistema. El sistema muestra un mensaje de éxito.	Evidencia/ Adicionar / Adicionar
EC 1.5	Selecciona la opción Eliminar				Elimina el archivo del servidor Retorna al 1.1	Evidencia/ Listar/ Eliminar
EC1.6	No se ha elegido fecha de la actividad	N/A	N/A	I	El sistema muestra el indicador de campo requerido. Regresa al 1.1	Evidencia/ Adicionar
EC1.7	No se ha subido ningún archivo				El sistema muestra un mensaje informando: “Debe subir un archivo”. Regresa al 1.1	Evidencia/ Adicionar

EC1.8	Existen campos vacíos.	I	N/A	N/A	Muestra un mensaje informando: “El campo no puede estar vacío”. Muestra el indicador de los campos que no pueden estar vacíos. Regresa al 1.1	Evidencia/ Adicionar

SC2 Eliminar evidencia

Escenario	Descripción	Nombre	Tipo de evidencia	Fecha de la actividad	Respuesta del sistema	Flujo Central
EC2.1	Selecciona la opción Listar disponible en el menú Evidencias.				Muestra un listado con las evidencias disponibles en el sistema y por cada uno de ellos permite: • Eliminar • Modificar Ver sección 1: “Editar evidencia”. • Descargar Ver CU: “Descargar archivo”.	Evidencia/Listar
EC2.2	Selecciona la opción Eliminar de la evidencia deseada.				Elimina la evidencia del sistema. Muestra un mensaje de éxito.	Evidencia/Listar /Eliminar/

SC3 Editar datos de una evidencia

Escenario	Descripción	Nombre	Tipo de evidencia	Fecha de la actividad	Respuesta del sistema	Flujo Central
EC3.1	Selecciona la opción Listar disponible en el menú Evidencias.				Muestra un listado con las evidencias disponibles en el sistema y por cada uno de ellos permite: • Eliminar Ver sección 2: “Eliminar evidencia”. • Modificar • Descargar Ver CU: “Descargar archivo”.	Evidencia/Listar
EC3.2	Selecciona la opción Editar de una evidencia deseada.				4-Abre una vista con los datos de la evidencia correspondiente, permitiendo modificar los valores del mismo: • Nombre	Evidencia /Listar /Editar

					del evento Permite elegir: • Tipo de evidencias • Fecha de la actividad Y las opciones: • Actualizar • Descargar	
EC3.3	Modifica los datos deseados y selecciona Actualizar.	V	V	V	Valida los cambios realizados. Actualiza la evidencia en el sistema. Muestra un mensaje de éxito.	Evidencia/ Listar /Editar/ Actualizar
EC3.4	Existen datos vacíos	I	N/A	N/A	Muestra un mensaje informando: “Verifique que no existan campos vacíos”.	Evidencia/ Listar /Editar

					Muestra el indicador de los campos que no pueden estar vacíos. Regresa al 3.2	
EC3.5	No se ha elegido fecha de la actividad	N/A	N/A	I	El sistema muestra indicador de campo Regresa al 3.2	Evidencia /Listar /Editar
EC3.6	Selecciona la opción Cerrar.				Cierra la ventana actual.	Evidencia /Listar /Editar /Cerrar

SC4 Ver datos de una evidencia

Escenario	Descripción	Nombre	Tipo de evidencia	Fecha de la actividad	Respuesta del sistema	Flujo Central
EC4.1	Selecciona la opción Listar disponible en el menú Evidencias				Muestra un listado con las evidencias disponibles en el sistema y por cada uno de ellos permite: • Eliminar Ver sección 2: "Eliminar evidencia"	Evidencia /Listar

					. • Modificar Descargar Ver CU: "Descargar archivo".	

3.3.5 Resultados de las pruebas de caja negra

Para la realización de las pruebas del sistema se aplicó el método de caja negra sobre las interfaces gráficas del sistema. Con la ejecución de las mismas se pudo constatar el cumplimiento de los requisitos funcionales y no funcionales del software obtenido.

A continuación, se presenta el resultado de las pruebas realizadas:

Tabla 4 Resultados de las pruebas por iteración.

Iteraciones	Cantidad de casos de prueba	No Conformidades detectadas			Total
		Alta	Media	Baja	
1	32	3	11	15	29
2	32	1	10	10	21
3	32	1	12	7	20
4	32	1	3	3	7

La tabla anterior muestra cuatro iteraciones, donde en cada iteración se les dan solución a las no conformidades detectadas, mejorando la calidad del software y preparándolo progresivamente para su uso final.

3.4 Conclusiones parciales

El uso de patrones de diseño permitió aplicar buenas prácticas en el proceso de modelado e implementación del software, logrando una mayor reutilización y mantenibilidad del código. La realización del modelado del sistema permitió obtener los diagramas de clases del diseño, secuencia y despliegue, los cuales le indicaron al equipo de desarrollo los objetos que se deben construir, los atributos y comportamientos de las clases, su relación y la forma en la que interactúan las mismas desde que el usuario realiza una acción. El proceso de implementación estuvo guiado por la elaboración de los diagramas de componentes de los casos de uso más representativos, con lo que se representó la vista de implementación del sistema y se obtuvo al finalizar el producto final con las funcionalidades requeridas. Para la validación del software se realizaron las pruebas al sistema mediante la técnica de caja negra, con esta se detectaron a tiempo un grupo de no conformidades en cada iteración, las cuales fueron resueltas al final de cada una.

Conclusiones generales

Con la realización del presente trabajo se arribaron a las siguientes conclusiones:

- ✓ Se realizó un estudio acerca del proceso de evaluación de los profesores de la Facultad 4 evidenciando la necesidad de informatizarlo, para de esta manera contribuir con el control de la información del mismo.
- ✓ Se analizaron elementos teóricos y principales tendencias del desarrollo de sistemas en la actualidad, específicamente en el campo de sistemas de gestión de la información.
- ✓ Se definieron las tecnologías y herramientas necesarias para el desarrollo de la propuesta de solución.
- ✓ Se implementó un sistema de gestión de información con los datos obtenidos en los estudios al proceso de evaluación de los profesores de la Facultad 4.
- ✓ Se hicieron pruebas de sistema y las no conformidades detectadas fueron resueltas en cuatro iteraciones.

Recomendaciones

Teniendo en cuenta los resultados obtenidos en el presente trabajo, se proponen las siguientes recomendaciones:

- ✓ Aplicar el software en todos los departamentos de la Facultad 4 y socializar su uso en el resto de las facultades de la universidad.
- ✓ Añadir otras funcionalidades acordes a las nuevas necesidades que surjan en el departamento.
- ✓ A los futuros desarrolladores del sistema, mantener actualizado el mismo con las nuevas tecnologías y herramientas informáticas.

Referencias bibliográficas

1. **Dr. Juan Vela Valdés.** Instrucción No. 3 /08 Sistema de superación de profesores (SSP) de los centros de Educación Superior adscritos al Ministerio de Educación Superior. 2008.

2. **Ibach, Michel O.Cifor´S**. Management information system. 1999.ISBN 979-8764-28-5 smt Grafika Desa Putera,Indonesia.

3. **Díaz Pérez, Maidelyn, Contreras Liz, Yimian, River Amador, Soleidys**. Características de los sistemas de información que permiten la gestión oportuna de la información y el conocimiento institucional. ACIMED. November 2009. Vol. 20, no. 5, p. 66–71.

4. **Diccionario de la lengua española** - Edición del Tricentenario. Diccionario de la lengua española [online]. [Accessed 10 February 2016]. Available from: http://dle.rae.es/ Versión electrónica del Diccionario de la lengua española, obra lexicográfica académica por excelencia.

5.**Universidad del Bío-Bío:** [online]. [Accessed 10 February 2016]. Available from: http://www.ubiobio.cl/w/#

6. **Ing. Irelys Baños Pinedo, Ing. Yeny Fírvida Martínez, Dra. María de Lourdes Artola Pimentel. and Dr. Julio Alfredo Telot González.** Sistema Automatizado para la Gestión de la Información Ciencia y Técnica de un Departamento Docente Universitario.". Universidad de Matanzas "Camilo Cienfuegos".2006

7. **Laboratorio Nacional de Calidad del Software**. Ingeniería del Software: Metodologías y Ciclos de Vida. March 2009.

8. **Letelier, Patricio and Penadés, M Carmen.** Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). www.cyta.com.ar/ta0502/v5n2a1.htm [online]. 15 April 2006. [Accessed 15 June 2016]. Available from: http://www.cyta.com.ar/ta0502/v5n2a1.htm

9. **The Agile Unified Process (AUP)** Home Page. [online]. 2009. [Accessed 10 February 2016]. Available from: http://www.ambysoft.com/unifiedprocess/agileUP.html

10. **Larman, Craig**. UML y Patrones. Introducción al análisis y diseño orientado a objetos. 1999. ISBN ISBN: 970-17-0261-1.

11. **Software Design Tools for Agile Teams, with UML, BPMN and More.** [online]. [Accessed 10 February 2016]. Available from: http://www.visual-paradigm.com/

12. **Los diferentes lenguajes de programación para la web.** Maestros del Web [online]. 2013. [Accessed 10 February 2016]. Available from: http://www.maestrosdelweb.com/los-diferentes-lenguajes-de-programacion-para-la-web/

13. **Lenguajes De Programación Del Lado Servidor** -Presentaciones de Google.2013 [online]. [Accessed 10 February 2016]. Available from: https://docs.google.com/presentation/d/1ZQgIsW4KCyIfH8NyiqH9ueanM83Ym7bjitaxyQKS0hs/edit?pli=1#slide=id.i0

14. **Herbert Shildt**. Java 2 Manual de referencia. 2001 ISBN: 9788448131739.S.A. MCGRAW-HILL / Interamericana de España.

15. **Marzal Andrés, García Isabel.**Introducción a la programación con Python. 2003.

16. **Mateu Andreu**. Php Debugging. 2004.

17. **¿Qué lenguaje de programación debería aprender para empezar?** 2013 [online]. [Accessed 8 June 2016]. Available from: http://es.gizmodo.com/que-lenguaje-de-programacion-deberia-aprender-para-emp-1479554075

18. **Álvarez,Miguel Ángel**. jQuery. 2009.

19. **YUI Library**. [online]. [Accessed 16 June 2016]. Available from: http://yuilibrary.com/

19. **Bootstrap.**2014 [online]. [Accessed 10 February 2016]. Available from: http://getbootstrap.com/2.3.2/

21. Blueprint: A CSS Framework | Spend your time innovating, not replicating. [online]. [Accessed 16 June 2016]. Available from: http://www.blueprintcss.org/

20. **Definicion de Framework**.2013 [online]. [Accessed 10 February 2016]. Available from: http://www.alegsa.com.ar/Dic/framework.php

21. **Pressman, Roger S.** javaHispano [online]. 2002. [Accessed 10 February 2016]. Available from: http://www.javahispano.org/portada/2014/8/20/los-4-framework-web-java

22. **Gronroos, Marko**. Book of Vaadin 7ma Edicion. s.l. : Creative Commons CC-BY-ND. 2014.

23. **Fernández,Norberto**.Introducción a Spring.2013.

24. **JavaServer Faces.org.** [online]. 2012 [Accessed 11 June 2016]. Available from: http://www.javaserverfaces.org/ JavaServer Faces - Rich Application Platform, Community Portal, Resource Center

25. **Html5 - Html | MDN**. [online] 2012. [Accessed 10 February 2016]. Available from: https://developer.mozilla.org/es/docs/HTML/HTML5

26. **Cascading Style Sheets**. [online]2012. [Accessed 10 February 2016]. Available from: https://www.w3.org/Style/CSS/

27. **Bos,Ber**t . Current Work on CSS at W3C.2009.

28. **Archivo - davidmarco.es.** [online]2014. [Accessed 10 February 2016]. Available from: http://www.davidmarco.es/archivo/jpa

29. **Hibernate ORM - Hibernate ORM.** [online] 2008. [Accessed 8 June 2016]. Available from: http://hibernate.org/orm/

30. **Olivas, Carlos Parreño**. Java Persistence API (JPA). [online] 2006. [Accessed 11 June 2016]. Available from: http://www.lab.inf.uc3m.es/~a0080802/RAI/jpa.html Explicación de JPA, persistencia y entidades

31. **Microsoft – Official Home Page.** [online]2010. [Accessed 10 February 2016]. Available from: http://www.microsoft.com/en-us/

32. **MySQL** . [online]2006. [Accessed 10 February 2016]. Available from: http://dev.mysql.com/doc/refman/5.7/en/introduction.html.

33. **Sobre PostgreSQL** | www.postgresql.org.es. [online] 2003. [Accessed 10 February 2016]. Available from: http://www.postgresql.org.es/sobre_postgresql#intro.

34. **Welcome to NetBeans.** [online] 2015. [Accessed 10 February 2016]. Available from: https://netbeans.org/

35. **Eclipse** - The Eclipse Foundation open source community website. [online] 2009. [Accessed 20 May 2016]. Available from: https://eclipse.org/

36**.** **Maven** – Welcome to Apache Maven. [online]. 2016. [Accessed 15 June 2016]. Available from: https://maven.apache.org/

37. **Maven** – Welcome to Apache Maven. [online]. [Accessed 10 February 2016]. Available from: http://maven.apache.org/

38. **Red Hat JBoss Middleware | Red Hat**. [online] 2008. [Accessed 10 February 2016]. Available from: http://www.redhat.com/en/technologies/jboss-middleware Red Hat JBoss Middleware offers high performance, scalability, stability, 24x7 support, and a low TCO. See how open source middleware from Red Hat works.

39. **Glassfish Project — Project Kenai.** [online]2007. [Accessed 10 February 2016]. Available from: https://java.net/projects/glassfish

40. **Apache Tomcat 7** (7.0.67) - Documentation Index. [online]2006. [Accessed 10 February 2016]. Available from: http://tomcat.apache.org/tomcat-7.0-doc/

41. **Hernández Marrero,Yasnay,Danilo Pina, DR. Joaquín , Socorro LLanes,MSC. Raisa and Jaime Puldón, ING. Joan**. Librería iText para la generación de PDF dinámicos. 2008. CUJAE.

42.**TEODORD. JasperReports® Library**. Jaspersoft Community [online]. 16 July 2008. [Accessed 26 May 2016]. Available from: http://community.jaspersoft.com/project/jasperreports-library.

43. **iText.** [online] 2009. [Accessed 8 June 2016]. Available from: http://itextpdf.com/

44. **Pressman, Roger S**. Ingeniería de software: Un efoque práctico 5ta edicion. s.l. : McGraw Hill. 2002.

45. **Camacho, Erika y Cardeso, Favio.** Arquitectura de software. 2004.

46. **Eastern Software Systems PVT. LTD**. Arquitectura de Tres Capas. 8 September 2006.

47. **James Rumbaugh.** El proceso unificado del desarrollo de software. Addison Wesley, 2005. ISBN 978-84-7829-036-9.

48. **Cáceres Tello, Jesús.** Diagramas de Casos de Uso. Universidad de Alcalá.2007.

49. **Gunnar Övergaard, Karin Palmkvis**t Use Cases Patterns and Blueprints. Addison Wesley, 2003.

50. **Alexander, Chistropher.** A Pattern Language. 1997.

51. **Gamma, Erich y Helm, Richard.** Design Patterns: Elements of Reusable Object-Oriented Software. 1995. ISBN 0-201-63361-2.

52. **Mulhrad, Daniel.** . Patrones de diseño. 2008.

53. **Eric Gramma**.Patrones de diseño. Adisson Wesley.2003.

54. **Rumbaugh, James, Jacopson, Ivar y Booch, Grady**. El Lenguaje Unificado de Modelado. Addison Wesley.1999.

55. **Jacobson, Ivar, Booch, Grady and Rumbaugh G, James.** El proceso unificado de desarrollo de software. Madrid. 2000.

56. **Sommervilles, Ian**. Ingeniería de software 7ma edición. Pearson-Addison Wesley, 2005.

57. **Departamento de Ingeniería.** Conferencia 7 de Ingeniería de Software ll. 2010.

58. **Blanco Bueno, Carlos** Ingeniería de Software - Construcción y pruebas 2002.

59. **Ruiz.** Las Pruebas de Software y su Importancia en las Organizaciones. 2010.

60. **Juristo, Natalia, Moreno, Ana M. y Vegas, Sira.** Técnicas de Evaluación de Software. 2005.

Glosario de términos

API: Interfaz de Programación de Aplicaciones. Representa la capacidad de comunicación entre componentes de software, poseen un conjunto de subrutinas, procedimientos, funciones o métodos ofrecidos por una biblioteca para ser utilizados por otro software.

CSS: Hoja de estilo en cascada. Es un mecanismo simple que permite definir y crear la presentación de un documento HTML o XML. La idea inicial de su creación era separar la estructura de un documento de su presentación. Permite a los desarrolladores el control total del estilo y el formato de múltiples páginas web al mismo tiempo.

Java EE: Java Empresarial. Es una plataforma de programación desarrollada inicialmente por *Sun Microsystems*, permite crear y ejecutar software escritos en lenguaje Java. Contiene varias API que incluyen servicios con diferentes fines y especificaciones que poseen tecnologías para la web.

JSP: *Java Server Pages*, siglas en inglés. Es una tecnología que permite a los desarrolladores mediante el uso del lenguaje Java crear documentos web dinámicos basados en HTML y XML. Para su ejecución requieren de servidores para Java web, tales como: *Apache Tomcat*, *JBoss* o *GlassFish.*

XHTML: Lenguaje de Marcadores e Hipertextos Extensible. Es una versión más estricta de HTML creada con el objetivo de reemplazar dicho lenguaje y avanzar hacia la web semántica. Combina la sintaxis de HTML 4.0 con XML, donde el primero está diseñado para mostrar datos y el segundo para describirlos.

XML: Lenguaje de Marcas Extensibles. Es un lenguaje de marcas que permite estructurar información en un documento o en cualquier fichero que contenga texto. Es una tecnología sencilla que propone un estándar para la comunicación de información entre diferentes plataformas, mejora la compatibilidad entre aplicaciones y provee soporte a las bases de datos.