Facultad 6

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

Aplicación web para la gestión y apoyo al proceso de enseñanza - aprendizaje de la Matemática Discreta (GECMA v2.0)

Autor: Arlen Neira Falcón

Tutor: MSc. Yidian Yosbel Castellanos Sabarí

"Año 58 de la Revolución"
La Habana, Cuba
Julio 2016

…" Lo fundamental es que seamos capaces de hacer cada día algo que perfeccione lo que hicimos el día anterior"…

Ernesto Che Guevara

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser autora de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo la presente a los 7 días del mes de julio del año 2016.

Firma de la autora:

_____
Arlen Neira Falcón

Firma del tutor:

_____
MSc. Yidian Yosbel Castellanos Sabarí

DATOS DE CONTACTO

Autora
Arlen Neira Falcón
Universidad de las Ciencias Informáticas, La Habana, Cuba
Correo: arlenfalcon@uci.cu

Tutor
MSc. Yidian Yosbel Castellanos Sabarí
Universidad de las Ciencias Informáticas, La Habana, Cuba
Correo: yycastellanos@uci.cu

AGRADECIMIENTOS

A mi tutor por su ayuda y comprensión en cada momento que lo necesité.

A mis compañeros Katia Roselló, Dagmar Mir,Gloria Leyva,Osmel Pérez y Yidier Romero por brindar su colaboración para aclarar todas mis dudas.

DEDICATORIA

A mis padres, especialmente a mi madre por su apoyo y ayuda incondicional.

A mis abuelos por sus buenos consejos y estar siempre ahí.

A toda mi familia por su apoyo incondicional.

RESUMEN

Las Tecnologías de la Información y las Comunicaciones están cada vez más presentes en el mundo moderno con implicaciones en cada una de las ramas de la sociedad. La esfera de la educación no se encuentra ajena a este fenómeno y precisamente la tendencia de las universidades a incorporar el uso de la tecnología en sus distintos procesos ha aumentado en los últimos años.

La Universidad de las Ciencias Informáticas es uno de los centros que aboga por informatizar todos sus procesos, siendo la docencia uno de ellos. Por esta razón en el presente trabajo se realizó el análisis, diseño e implementación de la aplicación web para la gestión y apoyo al proceso de enseñanza-aprendizaje de la Matemática Discreta (GECMA v2.0) con el objetivo de desarrollar una herramienta informática que le permita al profesor gestionar adecuadamente los procesos para el control docente de esta asignatura.

El software resultante permite gestionar grupos, estudiantes, asistencia, evaluaciones y llevar a cabo el análisis estadístico de los resultados de los estudiantes en dicha materia.

Para el desarrollo de esta aplicación se utilizó la metodología AUP, definida por la universidad para la actividad productiva con el objetivo de guiar el proceso de desarrollo de software; de igual forma se describieron las características del sistema. Se generaron los artefactos que propone la metodología utilizada y se realizaron pruebas de caja negra para garantizar el correcto funcionamiento del sistema.

Palabra clave: control docente, gestión del conocimiento, Matemática Discreta.

ABSTRACT

The Information and Communications Technology are present in the modern world with implications for each of the fields of society. The area of education is not excepted to this phenomenon and precisely the tendency of universities to incorporate the use of technology in its different processes has increased in recent years.

The University of Informatics Science is one of the centers which advocates computerize all processes teaching being one of them. Then, in this research the analysis, design and implementation of web application for managing and supporting the process of teaching and learning of Discrete Mathematics (GECMA v2.0) with the purpose of developing a software tool that was made allows the teacher to properly manage processes control for teaching this subject.

This software allows you to manage groups, students, attendance, evaluations and carry out statistical analysis of the results of students in this discipline.

For the development of this application was used AUP methodology, defined by the university for productive activity to guide the software development process; equally the system characteristics described, artifacts proposed for the methodology used and tests were conducted to ensure the appropriate functioning of the system.

Keyword: teaching control, knowledge management, Discrete Mathematic.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) están cada vez más presentes en el mundo moderno con implicaciones en cada una de las ramas de la sociedad. La esfera de la educación no se encuentra ajena a este fenómeno y la tendencia de las universidades a incorporar el uso de la tecnología en sus distintos procesos ha aumentado en los últimos años. Evidencia de ello es la implantación de varios sistemas computacionales como son:

- los Sistemas de Información del Estudiante (SIE), diseñados para colectar datos socio-demográficos de cada uno de los ingresados a una universidad;
- los Sistemas de Gestión Académica (SGA), encargados del manejo de las notas y trayectorias de los estudiantes en los diferentes cursos;
- Sistemas de Gestión del Aprendizaje (LMS, por sus siglas en inglés) utilizados para la creación de entornos virtuales que sirvan de apoyo a los procesos tradicionales de enseñanza.

Los sistemas de gestión como aporte de las TIC, en el contexto educacional, han contribuido al ejercicio de la enseñanza, el aprendizaje de calidad y el desarrollo profesional de los docentes, así como a la gestión, dirección y administración más eficiente del sistema educativo. Muestra de la importancia que le ha concedido Cuba a la imperativa necesidad de informatizar sus instituciones educativas universitarias, lo constituyen las palabras expuestas por el ministro de Educación Superior, Dr. Rodolfo Alarcón Ortiz, al explicitar que es necesario: *"Informatizar la educación superior como acompañamiento imprescindible a las transformaciones y perfeccionamientos constantes de los procesos sustantivos universitarios*" (Ortiz, 2008).

La Universidad de las Ciencias Informáticas (UCI) es una de las 26 universidades del sistema universitario cubano, fundada con el objetivo de participar en la vanguardia de la informatización del país y desarrollar la industria del software. En un intercambio con los estudiantes, el 19 de julio de 2003 el líder histórico de la Revolución cubana explicaba refiriéndose a la UCI que: "… *Una institución de Excelencia tiene que ser capaz de ver cómo se proyecta el mundo, cómo se proyecta la humanidad y qué contribución pueda hacer en primer lugar para salvarla"* (Durán , 2014). Fidel, como estratega fundacional de la UCI, recomendó que la universidad fuese concebida como un centro de nuevo tipo, de alcance nacional, de características atípicas y tareas concretas en el proyecto de informatización de la sociedad cubana, con énfasis en la producción de software.

La Ingeniería en Ciencias Informáticas, carrera única que se imparte en la UCI se rige por el Plan de Estudios D. Dentro de este plan, específicamente en la disciplina de Matemática, se imparten las asignaturas Matemática Discreta 1 (MD1) y Matemática Discreta 2, durante el primer año de la carrera.

En el curso 2014-2015 surge la tesis de pregrado “GECMA: Aplicación para la gestión y apoyo al control del proceso de enseñanza-aprendizaje de la Matemática Discreta 1” en la cual se expone que: “la MD1 constituye una piedra angular en el proceso de formación de ingenieros informáticos en no pocas universidades del mundo, sin embargo, los resultados docentes que muestran los alumnos al ser evaluada la misma, evidencian el grado de dificultad con que estos se enfrentan a dicha asignatura ” (Rivero, 2015).

A pesar del surgimiento de GECMA, la cual tuvo como objetivo desarrollar una aplicación web para apoyar el control del proceso de enseñanza-aprendizaje de la Matemática Discreta 1, los resultados docentes en la asignatura continúan siendo desfavorables. De ahí que la autora del presente trabajo de investigación indagara en las causas, desde un enfoque informático, que favorecen los resultados deficientes de dicha asignatura. Se pudo constatar que esta aplicación web al tener en su primera versión requisitos limitados, en función del tiempo en la que se construyó, presenta en las condiciones actuales las siguientes deficiencias:

1. Se hace engorroso en su base de datos la correlación de información referente a las evaluaciones sistemáticas y parciales.
2. Inexistencia de diagramas de diversos formatos tales como: diagramas de barras, pasteles e histogramas, que permitan una mayor usabilidad al sistema.
3. Imposibilidad de obtener informes concretos y fáciles de comprender de cada uno de los alumnos, así como de un grupo en general.
4. La fórmula asociada a la propuesta de corte evaluativo es inflexible en función de los parámetros comparables en cuanto a asistencia y evaluaciones, para ofrecer una propuesta a la medida de las características particulares de cada grupo docente.
5. El registro que se ofrece está confeccionado de tal forma que impide agregar nuevas actividades docentes (clases de laboratorios, seminarios, conferencias, clases prácticas y consultas) cuando el profesor lo estime conveniente.
6. Los reportes que ofrece GECMA actualmente se realizan de forma no estructurada, lo que impide que posteriormente sean utilizados en otras plataformas o bases de datos.

Hasta el presente curso, 2015-2016, la universidad cuenta con la aplicación web GECMA; sin embargo, debido a las limitaciones antes expuestas, el colectivo de profesores con esta aplicación no ha podido

realizar un adecuado control docente de la asignatura. Al analizar la situación problemática previamente expuesta, se evidencia la necesidad de implementar GECMA v2.0 para solucionar dichas deficiencias.

Por lo que se formula el siguiente **problema a resolver**: ¿Cómo gestionar adecuadamente el control docente en el proceso de enseñanza-aprendizaje de la Matemática Discreta desde la aplicación web GECMA?

A partir del problema enunciado, se define como **objeto de estudio**: las aplicaciones para la gestión del conocimiento en la educación, cuyo **campo de acción** se enmarca en la gestión del conocimiento en el control docente del proceso de enseñanza-aprendizaje de la Matemática Discreta con la aplicación web GECMA. Para resolver el problema planteado se propone como **objetivo general** de la investigación:

Desarrollar una aplicación web que permita gestionar adecuadamente el control docente en el proceso de enseñanza-aprendizaje de la Matemática Discreta desde la aplicación web GECMA.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de investigación**:

- Revisión de la bibliografía asociada a las aplicaciones desarrolladas, para contribuir a gestionar el conocimiento que se obtiene al controlar del proceso de enseñanza-aprendizaje en la Matemática Discreta.
- Definición de las herramientas a utilizar para la implementación de la aplicación web.
- Análisis, diseño e implementación de la aplicación web informática que ofrezca solución al problema planteado.
- Diseño de los casos de prueba para la evaluación del sistema.
- Validación de la aplicación web.
- Corrección de los posibles errores surgidos en la aplicación web o base tecnológica.
- Autenticar usuario mediante un servidor LDAP (Lightweight Directory Access Protocol o Protocolo Ligero de Acceso a Directorios).

A continuación, se detallan los **métodos de investigación científica** empleados en la investigación:

Métodos teóricos

- **Histórico-lógico:** permitió mediante un minucioso estudio, conocer el proceso de control de aprendizaje en GECMA y su funcionamiento, donde la información obtenida fue aplicada en la solución del problema presentado.

- **Análisis y síntesis:** se empleó para el análisis de la aplicación web GECMA para realizar una descomposición de este y estudiarlo minuciosamente hasta comprender su funcionamiento y su relación intrínseca, complementándose con la parte sintética.

Métodos empíricos

- **Encuesta:** se empleó para obtener una percepción real del fenómeno que se investiga teniendo en cuenta los criterios de los profesores de asignatura en la UCI (ver Anexo 6).

Con el cumplimiento de las tareas antes planteadas, se consideran como posibles resultados:

- La versión 2.0 de la aplicación GECMA para gestionar adecuadamente el control docente en el proceso de enseñanza-aprendizaje de la Matemática Discreta.
- La base teórica conceptual de la investigación para apoyar la comprensión del sistema en el desarrollo de versiones posteriores.

El presente documento consta de tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos.

Capítulo 1. En la **Fundamentación teórica** se define la base teórica de la presente investigación. Se describen las herramientas y tecnologías actuales sobre las cuales se apoya la propuesta del sistema informático, así como la metodología de desarrollo de software a emplear en el proceso de solución.

Capítulo 2. En la **Propuesta de solución** se realiza una descripción detallada de la solución, donde se definen los requisitos funcionales y además se realizan actividades de análisis y diseño.

Capítulo 3. En la **Construcción y validación del sistema** se describe la implementación de la solución, se ejecutan los casos de pruebas y se corrigen las no conformidades encontradas para garantizar el buen funcionamiento de la solución propuesta.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

El presente capítulo aborda los principales conceptos que servirán de apoyo para lograr un mayor entendimiento del problema a resolver. Se realiza un estudio de GECMA, aplicación web para la gestión y apoyo al control del proceso de enseñanza-aprendizaje de la Matemática Discreta 1, debido a que se toma como punto de partida para el diseño de la propuesta de solución. Se define la metodología, herramientas y lenguaje a emplear durante el proceso de desarrollo.

1.1 Marco conceptual

En este acápite se ofrecen algunos conceptos que permiten una mejor comprensión de la investigación por parte del lector.

1.1.1 Análisis del sistema GECMA: aplicación para la gestión y apoyo al control del proceso de enseñanza-aprendizaje de la Matemática Discreta 1

La aplicación GECMA es una aplicación web para la gestión y apoyo al control del proceso de enseñanza-aprendizaje de la MD1. La aplicación está compuesta por un banner superior, un panel lateral izquierdo y un panel central donde se le brindan al usuario todas sus opciones.

Las funcionalidades que brinda el software actualmente se describen a continuación:

Inicio: muestra información referente a la asignatura.

Grupos: visualiza los diferentes grupos que atiende el profesor, mostrando por cada grupo cada uno de sus miembros y una descripción de estos. Permitiendo además la gestión (insertar, modificar, eliminar, mostrar) de grupos en la aplicación.

Estudiantes: posibilita la gestión (insertar, modificar, eliminar, mostrar) de todos los aspectos relacionados con los educandos, mostrándose inicialmente una síntesis de los estudiantes de cada grupo que atiende el profesor. De igual forma se pueden realizar comparaciones entre alumnos mediante filtros para los resultados docentes y la asistencia.

Asistencia: permite gestionar (insertar, modificar, eliminar, mostrar) toda la información estudiantil referente a la asistencia. Muestra una gráfica con la asistencia de los estudiantes en el presente curso.

Evaluaciones: brinda las opciones de gestión de evaluaciones que se les realicen a los estudiantes.

Sin embargo, después de haberse desplegado este software para el uso de los profesores de Matemática Discreta se pudo confirmar que eran muy restringidos los requisitos para los cuales se había implementado

y que limitaba además la usabilidad de GECMA. Se defiende la idea de que la nueva aplicación deba tener un módulo de Registros de evaluaciones y asistencia donde se pueda crear o modificar su estructura según las necesidades del profesor. Debe, además, generar reportes y gráficos de diversos tipos para detallar la información cuyo conocimiento luego ayudará a realizar una ayuda diferenciada a grupos o estudiantes.

Todo esto permitió a la autora del presente trabajo confirmar que no debían hacerse modificaciones a la aplicación web anterior, sino desarrollar una nueva versión.

1.1.2 Servicios web

Los autores P. J. Deitel y H. M. Deitel consideran que:

"Un servicio Web es un componente de software almacenado en una computadora, el cual se puede utilizar mediante llamadas a métodos desde una aplicación (u otro componente de software) en otra computadora, a través de una red. Los servicios Web se comunican mediante el uso de tecnologías como XML[1] y HTTP[2]."
"… Los servicios Web y SOAP son independientes de la plataforma y del lenguaje, por lo que las compañías pueden colaborar a través de servicios Web sin tener que preocuparse por la compatibilidad de sus tecnologías de hardware, software y comunicaciones" (Deitel, 2008).

Marco Besteiro y Miguel Rodríguez expresan que:
"Un servicio web o WebService es un servicio ofrecido por una aplicación que expone su lógica a clientes de cualquier plataforma mediante una interfaz accesible a través de la red utilizando tecnologías (protocolos) estándar de Internet" (Besteiro, 2002).

Ethan Cerami plantea:
"Un servicio web es cualquier servicio que está disponible a través de Internet, utiliza un sistema de mensajería XML estandarizada, y no está ligada a ningún sistema operativo o de un lenguaje de programación" (Cerami, 2002).

La autora de la presente investigación asume las posiciones de los autores anteriormente citados y considera que un servicio web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML estandarizada y que a su vez es independiente de la plataforma.

[1] Lenguaje de Marca Extensible, del inglés *eXtensible Markup Language.*
[2] Protocolo de Transferencia de Hiper-Texto, del inglés *Hipertext Transference Protocol*.

1.1.3 Pasarela de autenticación

Es un servicio web que se desarrolló en la UCI y que varios sistemas emplean para llevar a cabo el proceso de autenticación.

1.2 Soluciones similares

Los sistemas informáticos facilitan las actividades de cualquier empresa o compañía, estos contribuyen a la optimización de los resultados y mejor aprovechamiento de los recursos, buscando mayor eficiencia en los procesos que se realizan y mejores resultados en la producción. En busca de una solución al problema de la investigación planteado se realizó un estudio de sistemas de gestión para el control de diversos procesos, en el ámbito internacional y nacional. A continuación, se detallan algunos de los sistemas analizados:

1.2.1 Sistemas existentes en el ámbito internacional

Sistema de Información de Gestión Académica (SIGA)

SIGA es un producto desarrollado en la Universidad de Chile. Es un sistema realizado para instituciones de educación superior con el propósito de organizar el manejo de la información. Con SIGA las entidades universitarias pueden ofrecer nuevos servicios y mejorar la atención a la comunidad educativa. Genera indicadores y reportes de análisis en docencia de pregrado y postgrado. Brinda posibilidades como:

- La matrícula en línea: Permite a los estudiantes hacer su matrícula desde cualquier computadora con acceso a Internet y puede imprimir una tabla en la cual tiene definidos sus horarios de acuerdo con la matrícula.
- Expediente académico: Brinda la posibilidad de organizar y administrar la información de los diferentes historiales académicos del estudiante gestionando aspectos como: asignaturas matriculadas, calificaciones obtenidas, distinciones, bajo rendimiento, entre otros.
- Hoja de vida del estudiante: Maneja la información de carácter personal del estudiante, permitiendo la actualización de la misma.
- Gestión de la información estudiantil: Facilita los procesos relacionados con la trayectoria académica del estudiante, desde su ingreso hasta la obtención del título (Universidad de Chile, 2015).

ISOCLOUD

ISOCLOUD es una aplicación que permite adaptarse a las empresas de una forma rápida y a través de la web. Contiene tres servicios fundamentales: Sistema de Gestión Integrado, Sistema de Gestión de Calidad y Sistema de Gestión Medio Ambiental. Algunas de sus características fundamentales son:

- Generación de informes de incidencias, objetivos, metas, y resultados de encuestas de satisfacción.

- Gestión documental y proceso de control de firmas.
- Permite el registro de consumos, residuos peligrosos y no peligrosos con el fin de llevar a cabo el control operacional en las organizaciones.
- Gestión de diferentes tipos de incidencias (no conformidades internas, no conformidades de proveedor, acciones correctivas, entre otras), relacionadas con distintas procedencias (encuestas de satisfacción, planes de formación y auditorías).
- Definición y evaluación de aspectos ambientales, a través de una metodología que permite establecer distintas variables y valoraciones, tanto para los aspectos ambientales reales como potenciales (Carvajal, 2013).

Sistema de Estudios de Postgrado (SEP)

SEP fue desarrollado en la Universidad de Costa Rica, se dedica específicamente a organizar, orientar y facilitar el desarrollo de los programas de maestrías, doctorados y especialidades para una mejor preparación de los profesionales. SEP ha desarrollado programas de postgrado y multidisciplinarios de carácter interinstitucional. Entre las funcionalidades que brinda se encuentran:

- Permitir al alumno inscribirse en los diferentes postgrados que están en oferta.
- Presentar un programa detallado acerca de las convocatorias de admisión, incluyendo la modalidad de los postgrados.
- Brindar un calendario a los estudiantes sobre el proceso de matrícula y los costos por etapas, así como la documentación necesaria para dicho proceso.
- Los estudiantes tienen la posibilidad de informarse acerca de los horarios relacionados con los diferentes postgrados de su interés.
- Brindar información referida a los costos y pagos para el correcto ingreso a los postgrados ofertados (Universidad de Costa Rica, 2012).

Dentro de las principales desventajas de SEP se encuentra que el proceso de gestión de inscripción no se lleva a cabo de forma automática, ya que tanto los profesores como los estudiantes deben realizar este proceso a través de correo, una vía a veces no segura para este tipo de actividades.

1.2.2 Sistemas existentes en el ámbito nacional

Sistema de Gestión Académica de la UCI (Akademos)

Sistema desarrollado en la UCI y tiene como objetivo permitir la gestión automatizada de los elementos en la labor académica, que pueda enfrentar los cambios de forma natural, adaptándose a las nuevas condiciones y formas de hacer con el menor costo. Fue desarrollado en Visual Studio .NET 2003 utilizando

SQL Server 2000 como gestor de base de datos. A continuación, se nombran y describen algunos de los módulos que contiene:

- Personal: gestiona todos los datos de las personas en la base de datos.
- Pregrado: gestiona el expediente de los estudiantes el que abarca el plan de estudios, trámites docentes, registro de asignaturas y un resumen de evaluaciones. El registro de asignaturas lleva a cabo el control de las evaluaciones por tipo de evaluación y registro de asistencia de cada semestre.
- Eventos: gestiona la información referente a los diferentes eventos que se desarrollan en la universidad.
- Notificaciones y alertas: brinda avisos a roles específicos sobre las acciones que se han llevado a cabo en el sistema, estos se pueden configurar para que envíen correos (Beatón, 2009).

Aplicación para la gestión de información de investigación y postgrado

Este sistema permite concentrar toda la información referente a investigación y postgrado de la Facultad 4. Es una aplicación web que contiene los módulos Investigación, Postgrado y Balance; los cuales permiten el manejo adecuado y la gestión de toda la información referente a estos procesos. Dispone de publicaciones de artículos interesantes para la comunidad de profesores, así como las convocatorias tanto a eventos como a cursos. El sistema tiene habilitado un foro que posibilita el intercambio de información entre todos los usuarios alrededor de los temas propuestos, permite además la gestión de documentos útiles, la actualización sobre la actividad postgraduada de los profesores, la revisión de trabajos realizados, así como la generación de reportes sobre la labor investigativa y postgraduada por cada profesor, por cada departamento y de la Facultad en general. Se habilitan tres tipos de usuarios con roles diferentes los cuales van a ser los encargados de formalizar toda la información referente al sistema, limitar el acceso, dar soporte, mantenimiento e interactuar con el sistema (Espinosa, 2012).

Versión 2.0 del módulo Resultados de la colección “La Caja Mágica”

La versión 2.0 del módulo Resultados de la colección “La Caja Mágica” tiene la responsabilidad de llevar un control de las acciones realizadas por los estudiantes en los distintos módulos que lo componen, obteniéndose un conjunto de reportes que le permiten al profesor tener buena visibilidad sobre los resultados de las evaluaciones de los estudiantes, estos son: trayectoria del estudiante, análisis de temas, historial de los estudiantes y análisis integral. Se obtienen las opciones de búsqueda y filtrado para organizar los datos de los estudiantes y una serie de tablas y gráficas que miden la efectividad de los resultados evaluativos de los estudiantes al realizar actividades en las distintas materias.

- Trayectoria del estudiante: presenta información de la interacción del estudiante en una sesión de trabajo.

- Análisis de temas: se realizan consultas de las calificaciones que los estudiantes van obteniendo de forma paulatina, con los cuestionarios interactivos del módulo ejercicios, en función de los temas.
- Historial de los estudiantes: se presenta un análisis de los resultados de la interacción de los estudiantes con los temas, tomando en cuenta sus resultados en función de las fechas en que los estudiantes interactuaron con los ejercicios.
- Análisis integral: se analizan a nivel general los resultados de la interacción de los estudiantes con los ejercicios de todos los productos de la colección en función de las asignaturas (Cruz, 2013).

1.2.3 Resultados

Después de analizados los sistemas antes citados y estudiar las posibles funcionalidades que se desean tener en la nueva versión de GECMA, se decide crear una solución propia para complementar las funcionalidades que aún no realiza Akademos debido al objetivo para el que fue creado. El estudio de las aplicaciones ISOCLUOD, La caja mágica, SEP, SIGA y Aplicación para la gestión de información de investigación y postgrado demostró que no solventaban la problemática debido que se especializaban en otros negocios. Luego del análisis exhaustivo de GECMA en su primera versión se evidenció la necesidad de transformar desde la estructura de la base de datos hasta las interfaces que se les brindaba a los clientes debido al alcance que se pretende lograr con la nueva aplicación.

1.3 Metodología de desarrollo

Las metodologías para el desarrollo del software imponen un proceso sobre el desarrollo de aplicaciones informáticas con el fin de hacerlo más predecible y eficiente. Tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo, haciendo énfasis en la calidad y menor tiempo de construcción del software o lo que es lo mismo “producir lo esperado en el tiempo esperado y con el coste esperado” (Pressman, 2002).

La metodología AUP (Proceso Unificado Ágil de Scott Ambler, del inglés Agile Unified Process) es una versión simplificada de RUP (Proceso Unificado de Rational, del inglés Rational Unified Process). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. La metodología de desarrollo institucionalizada por la universidad para su actividad productiva es AUP-UCI. Su definición se basa en una variación de la metodología AUP en unión con el modelo CMMI-DEV[3] v 1.3, de tal forma que se adapta al ciclo de vida definido para la actividad productiva de la UCI.

[3] Integración de modelos de madurez de capacidades para el desarrollo, del inglés *Capability Maturity Model Integration for Development.*

En su adaptación se especifican tres fases (Inicio, Ejecución y Cierre) y 8 disciplinas (Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación, Pruebas de Aceptación y Despliegue (Opcional)) (Sánchez, 2015).

Para definir la manera de encapsular los requisitos funcionales se hizo uso de la técnica Modelado ágil que utiliza AUP, donde teniendo en cuenta las características del negocio o del equipo de desarrollo se encapsularán los requisitos funcionales en Historias de usuarios (HU).

En el desarrollo de la propuesta de solución se decide optar por encapsular los requisitos en HU debido a que el proceso de desarrollo está orientado a un ciclo de vida de corta duración. Una sola persona debe adoptar varios roles para llevar a cabo todos los flujos de trabajo requeridos en el tiempo asignado y a su vez obtener los artefactos correspondientes de cada disciplina. Por otra parte, el estudio realizado traza el desarrollo de una nueva versión por lo que a este punto de la investigación se considera que la autora ya ha adquirido conocimientos del negocio y por ende tal como se estipula para este caso, no se requiere llevar a cabo el Modelado de negocio, aunque se haga uso del Modelo de dominio para una mejor comprensión programador-cliente.

1.4 Herramientas y tecnologías

El estudio de herramientas análogas permitió realizar el análisis de los lenguajes y herramientas que serán empleadas, así como la existencia de una solución adecuada a la problemática que se investigaba. Para el desarrollo de la presente investigación se utilizaron herramientas de código abierto para cumplir con las políticas de migración a software libre que defiende el país.

1.4.1 Lenguaje de modelado

UML 2.0: UML (Lenguaje Unificado de Modelado, del inglés *Unified Modeling Language*) es lenguaje estándar en el análisis y diseño de sistemas de cómputo que permite especificar, visualizar, construir y documentar todos los elementos que forman un sistema de software, desde una perspectiva orientada a objetos. Este modelado visual es independiente del lenguaje de implementación, los diseños realizados usando UML se pueden implementar en cualquier lenguaje que soporte las posibilidades de UML, principalmente lenguajes orientados a objetos (Hamilton, 2006).

1.4.2 Herramienta CASE[4]

Visual Paradigm para UML 8.0: herramienta para desarrollo de aplicaciones que utiliza el modelado UML y soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación. Presenta licencia gratuita y comercial. Es fácil de

[4] Ingeniería de software asistida por computadora, del inglés Computer Aided Software Engineering

instalar y actualizar, y compatible entre ediciones. Genera código para Java y exportación como HTML (Oscar, 2013).

1.4.3 Plataforma de desarrollo

JEE 1.7: JEE (Edición Empresarial de Java, del inglés *Java Enterprise Edition*) es un entorno independiente de la plataforma centrado en Java para desarrollar, crear e implementar en línea aplicaciones empresariales basadas en web. Consta de un conjunto de servicios, APIs (Interfaz de programación de aplicaciones, del inglés Application Programming Interface) y protocolos que proporcionan la funcionalidad necesaria para desarrollar aplicaciones basadas en web de varios niveles (Oracle, 2013).

1.4.4 IDE de desarrollo [5]

Netbeans 8.0: herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Es de código abierto, escrito completamente en Java, pero puede utilizarse para desarrollar en cualquier otro lenguaje de programación. Soporta el desarrollo de Aplicaciones empresariales con Java EE, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies, y modelado UML (Gimeno, 2011).

1.4.5 Lenguaje de programación

Java: lenguaje de programación orientado a objeto, de código abierto e independiente de la plataforma. Se distingue por ser capaz de gestionar la memoria automáticamente, posee mecanismos de seguridad incorporados, los cuales limitan el acceso a recursos de las máquinas donde se ejecuta e incorpora herramientas de documentación. Es multihilos, habilidad que consiste en dividir el trabajo de un programa en diferentes hilos de ejecución (Weitzenfeld, 2005).

1.4.6 Frameworks

Spring 4.2.2: contenedor[6] y *framework*[7] de peso ligero, basado en inyección de dependencias y orientación a aspecto, promoviendo el bajo acoplamiento pues los objetos reciben pasivamente sus dependencias en lugar de crearlas o buscar los objetos dependientes por sí mismos. Brinda facilidades para desarrollar una aplicación empresarial en JEE (Walls, 2008).

PrimeFaces 5.3: librería de componentes visuales de código abierto desarrollada y mantenida por Prime Technology, una compañía turca de Tecnologías de la Información especializada en consultoría ágil, JSF y Java EE. Entre las principales características destacan: soporte nativo de Ajax, es de código libre, activo, bastante estable entre versiones y compatible con otras librerías de componentes (Çivici, 2015).

---

[5] *Entorno de Desarrollo Integrado, del inglés Integrated Development Environment*

[6] Interfaz entre el componente y la plataforma sobre la que se ejecuta. Facilita los servicios que éste necesita para su funcionamiento.

[7] Arquitectura reusable que brinda la estructura genérica y de comportamiento para un grupo de abstracciones de software.

JSF (Del inglés *Java Server Faces*) **2.2.9:** tecnología y *framework* para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF incluye un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas, dar soporte para internacionalización y accesibilidad (Geary, 2010).

Hibernate 4.3.7: capa de persistencia objeto/relacional y un generador de sentencias *sql*[8]. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se pueden generar bases de datos en cualquiera de los entornos soportados: Oracle[9], DB2[10], MySql[11], etc. Es de código abierto (González, 2003).

1.4.7 Herramientas

Apache Tomcat 8.0: servidor web multiplataforma, flexible, rápido y eficiente que se desarrolla de forma abierta. Es modular, ya que puede ser adaptado a diferentes entornos y necesidades (Fundation, The Apache Software).

1.4.8 Sistema de gestión de base de datos

PostgreSQL 9.4: gestor de bases de datos orientadas a objetos, muy conocido y usado en entornos de software libre. Puede funcionar en múltiples plataformas. Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario. Su administración se basa en usuarios y privilegios. Es altamente confiable en cuanto a estabilidad se refiere. Puede extenderse con librerías externas para soportar encriptación (Ginesta, 2005).

1.5 Consideraciones parciales del capítulo

En este capítulo tras un estudio de herramientas análogas y el análisis del sistema existente, se constató la necesidad de desarrollar una segunda versión de la aplicación web GECMA para integrar nuevas funcionalidades y solucionar las deficiencias antes expuestas. Luego del estudio de las tecnologías y tendencias actuales en cuanto a desarrollo de aplicaciones web y a partir de las nuevas funcionalidades a implementar se seleccionaron las herramientas a emplear en la construcción de la solución. La investigación se alinea con las políticas de migración a software libre del país, seleccionando herramientas de código abierto y multiplataforma.

[8] Lenguaje de Consulta Estructurado, de inglés *Structured Query Language*. Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

[9] Sistema de gestión de base de datos objeto-relacional desarrollado por Oracle Corporation.

[10] Es un motor de base de datos relacional que integra XML de manera nativa que permite almacenar documentos completos dentro del tipo de datos xml para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.

[11] Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

En este capítulo se procede a caracterizar y diseñar el sistema con el propósito de planificar, organizar y ejecutar el desarrollo de la solución. Se describe el proceso a automatizar, los requisitos funcionales y artefactos que se generan en cada hito de la construcción del software. Para guiar el proceso de desarrollo se hace uso de la metodología AUP-UCI.

2.1 Descripción del proceso a informatizar

Para la gestión y apoyo al control del proceso de enseñanza-aprendizaje de la Matemática Discreta 1 fue desarrollada la aplicación web GECMA. Este software lleva a cabo la gestión de los estudiantes, grupos, asistencia y evaluaciones. Sin embargo, no gestiona los datos de los profesores, asignaturas y el registro que se ofrece como valor agregado está confeccionado de tal forma que impide agregar nuevas actividades docentes cuando el profesor lo estime conveniente. Además, no muestra diagramas de barras que permitan una mayor usabilidad al sistema, así como la obtención de informes concretos y fáciles de comprender, de los resultados individuales de los alumnos y del grupo en general, en función del tiempo.

Por otra parte, para ofrecer un corte evaluativo a la medida de cada uno de los grupos, la fórmula asociada a la propuesta es inflexible en función de los parámetros comparables en cuanto a asistencia y evaluaciones. La base de datos presenta deficiencias para correlacionar la información referente a las evaluaciones sistemáticas y parciales, lo cual impide realizar un análisis adecuado. Al no alcanzar esta aplicación los resultados esperados, se requiere desarrollar una nueva versión para informatizar los procesos antes expuestos.

2.2 Modelo de dominio

El modelo de dominio permite comprender y describir las clases más importantes dentro del contexto del sistema, para ayudar a utilizar un vocabulario común entre los usuarios, clientes, desarrolladores y demás interesados. También contribuye a la comprensión de los requisitos del sistema que se desprende de este contexto (Rumbaugh y Jacobson, 2000).

2.2.1 Diagrama de clases del dominio

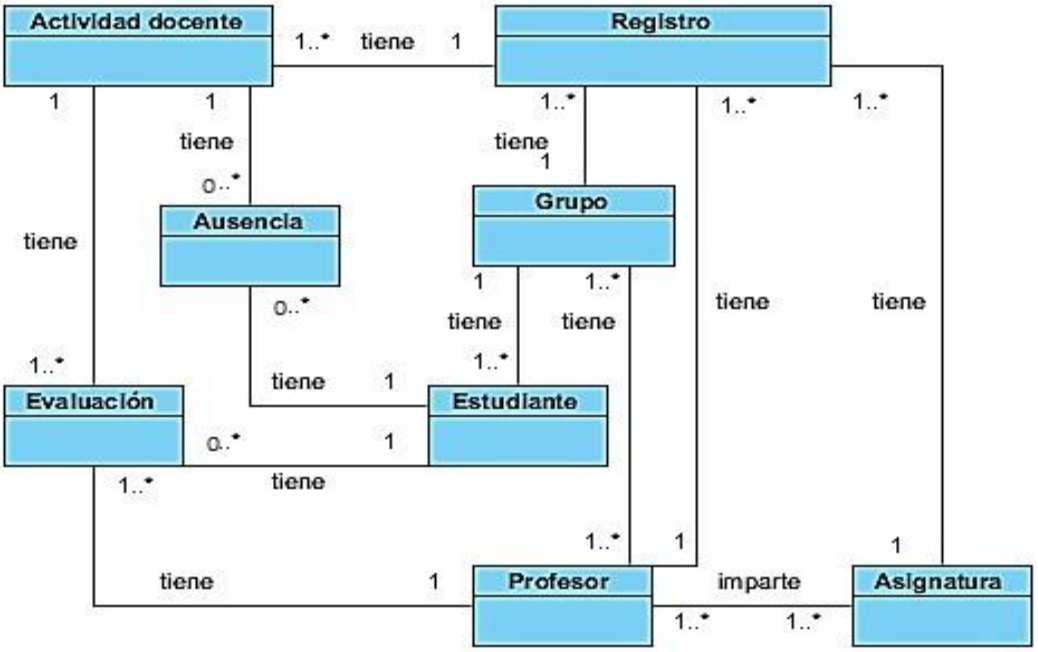


Figura 1 Modelo de dominio

Fuente: elaboración propia

2.2.2 Descripción de las clases

Para una mejor comprensión del modelo de dominio, se proporciona a continuación una breve descripción de las clases que lo integran:

Actividad docente: es el concepto encargado de gestionar las formas básicas de organización del proceso de enseñanza tales como: seminarios, conferencias, clases prácticas y consultas.

Registro: describe a la organización de la información del cuerpo académico de cada asignatura.

Grupo: representa un conjunto docente de estudiantes que interactúa con un profesor.

Estudiante: encargado de almacenar toda la información concerniente al estudiante.

Profesor: encargado de almacenar toda la información concerniente al profesor.

Evaluación: representa el desempeño de estudiantes y profesores en las diferentes actividades docentes.

Asignatura: encargado de gestionar las diferentes materias impartidas por los profesores.

Ausencia: encargado de manejar las inasistencias de los estudiantes en las actividades docentes.

3 Especificación de Requisitos de software

Los requerimientos para un sistema lo constituyen la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Los requisitos reflejan la necesidad de los clientes, que ayuda a resolver problemas como el control de un dispositivo, hacer un pedido o encontrar información. El proceso

que permite descubrir, analizar, documentar y verificar esos servicios y restricciones se denomina ingeniería de requisitos (Sommerville, 2005).

Existen varias técnicas para llevar a cabo el proceso de **captura de requisitos**. Es tarea del equipo de desarrollo determinar cuál o cuáles son las más factibles a utilizar para el producto a desarrollar. Entre algunas de estas técnicas se pueden mencionar:

- **Entrevistas:** es un medio tradicional de obtención de requisitos. La entrevista es un método muy efectivo que permite conocer los problemas de los clientes y encontrar requisitos generales. Para aplicar este método es necesario conocer la forma en que se debe realizar una entrevista para lograr una buena comunicación entre el entrevistador y el entrevistado.
- **Prototipos:** es la representación o visualización de parte del sistema. Es una herramienta valiosa para clarificar requisitos confusos. Proveen a los usuarios un contexto para entender mejor qué información necesitan proporcionar.
- **Tormenta de ideas:** esta técnica es usada para generar nuevas ideas y encontrar la solución a cuestiones específicas. Es muy común en los comienzos del proceso de ingeniería de requisitos.
- **Observación:** este método consiste en la identificación de requisitos cuando se observan a las personas realizar su trabajo diario. Es muy usado para encontrar requisitos adicionales cuando el usuario es incapaz de explicar los requisitos que necesita para el nuevo sistema (Pressman, 2002).

Para realizar un buen proceso de captura de requisitos del sistema a desarrollar, se empleó una combinación de estas técnicas. Proceso que arrojó los requisitos que a continuación se describen.

3.1.1 Requisitos funcionales

Los requisitos funcionales de un sistema describen las capacidades o funciones que el sistema debe cumplir, los servicios que de él se esperan, o los que proveerá (Sommerville, 2005).

Los requisitos funcionales identificados se enuncian a continuación:

RF 1 Insertar estudiante.
RF 2 Modificar estudiante.
RF 3 Eliminar estudiante.
RF 4 Mostrar estudiante.
RF 5 Graficar estado de cada estudiante.
RF 6 Insertar profesor.
RF 7 Modificar profesor.
RF 8 Eliminar profesor.

RF 9 Mostrar profesor.

RF 20 Insertar grupo.

RF 11 Modificar grupo.

RF 12 Eliminar grupo.

RF 13 Mostrar grupo.

RF 14 Graficar información de un grupo.

RF 15 Asignar grupos a un profesor.

RF 36 Insertar actividad docente.

RF 17 Modificar actividad docente.

RF 18 Eliminar actividad docente.

RF 19 Mostrar actividad docente.

RF 20 Insertar ausencia.

RF 21 Modificar ausencia.

RF 22 Eliminar ausencia.

RF 23 Mostrar ausencia.

RF 24 Graficar asistencia de un grupo.

RF 25 Exportar registro de asistencia (excel o pdf).

RF 26 Insertar asignatura.

RF 27 Modificar asignatura.

RF 28 Eliminar asignatura.

RF 29 Mostrar asignatura.

RF 30 Insertar registro de asistencia.

RF 31 Modificar registro de asistencia.

RF 32 Eliminar registro de asistencia.

RF 33 Mostrar registro de asistencia.

RF 34 Insertar evaluación.

RF 35 Modificar evaluación.

RF 36 Eliminar evaluación.

RF 37 Mostrar evaluación.

RF 38 Generar corte evaluativo.

RF 39 Graficar corte evaluativo.

RF 40 Mostrar listado de corte evaluativo.

RF 41 Mostrar gráfico sobre el estado de evaluaciones de un grupo.

RF 42 Obtención desde el LDAP del estudiante o el profesor por el usuario UCI.

RF 43 Autenticar usuario con LDAP.

RF 44 Asignar estudiantes a un grupo.

3.1.2 Requisitos no funcionales

Los requerimientos no funcionales definen las restricciones del sistema, son propiedades o cualidades que el sistema debe poseer. Representan las características del producto (Sommerville, 2005).

Para la definición de los requisitos no funcionales se hizo uso de la norma ISO – IEC 25010. A continuación, se especifican los atributos de calidad con los subatributos respectivamente.

Usabilidad

RNF 1 Protección contra errores de usuarios: el sistema debe proporcionar mensajes de error que sean informativos y orientados al usuario final para proteger a los usuarios de cometer errores cuando realizan alguna acción en el sistema.

Fiabilidad

RNF 2 Disponibilidad: la aplicación debe estar disponible a tiempo completo, permitiendo el trabajo de los profesores y la revisión de las actividades en el momento que se necesite.

Eficiencia en el rendimiento

RNF 3 Utilización de recursos: el sistema deberá funcionar correctamente ante las siguientes características de hardware y software:

Software

- PC cliente y PC servidor de base de datos.
- Computadoras con sistemas operativos Windows o Linux.
- Usar navegador web Mozilla Firefox versión 45.0.1 o superior.

Hardware

PC cliente:

- Microprocesador a 1.0GHz o superior, 256Mb de memoria RAM o superior, 1Gb disponible en disco duro.

PC servidor:

- PC con 3Gb de memoria RAM o superior, 4Gb disponibles en disco duro.

Seguridad

RNF 4 Confidencialidad: la información deberá protegerse de accesos no autorizados utilizando autenticación con LDAP. La autenticación deberá ser la primera acción del usuario en la aplicación. La contraseña debe ser de conocimiento exclusivo de la persona que se autentica. Si el usuario autenticado no se encuentra registrado se debe reportar un error de acceso.

RNF 5 Integridad: la información podrá ser modificada solamente por el personal autorizado según el rol asignado a su usuario.

Restricciones del sistema

- Usar el lenguaje de programación JAVA.
- Emplear como gestor de base de datos PostgreSQL versión 9.4.
- Utilizar como servidor web Apache Tomcat versión 8.0.
- Desarrollar bajo el *framework* de componentes visuales PrimeFaces versión 5.3

3.2 Propuesta de solución

Luego del análisis realizado y la problemática existente se propone desarrollar una aplicación web que permita el acceso a toda la información relacionada con los estudiantes de un grupo dado. La aplicación está compuesta por un banner superior, un panel central donde se le brinda al usuario la posibilidad de realizar las operaciones y un menú donde se agrupan los siguientes módulos:

- **Asignatura:** posibilita la gestión de todos los aspectos relacionados con las asignaturas y permite filtrar por nombre.
- **Profesor:** posibilita la gestión de todos los aspectos relacionados con los profesores.
- **Estudiantes:** posibilita la gestión de todos los aspectos relacionados con los estudiantes.
- **Grupos:** posibilita la gestión de todos los aspectos relacionados con los grupos y admite filtrar por nombre y por curso.
- **Registro:** posibilita la gestión de todos los aspectos relacionados con los registros y permite filtrar por nombre, asignatura y grupo.
- **Actividad docente:** posibilita la gestión de todos los aspectos relacionados con las actividades docentes. Además, permite filtrar por nombre, tipo de actividad docente y fecha.
- **Evaluación:** posibilita la gestión de todos los aspectos relacionados con las evaluaciones docentes. También, permite filtrar por estudiante, actividad docente y nota.
- **Ausencia:** permite gestionar toda la información estudiantil referente a la ausencia, soporta filtrar por estudiante, actividad docente y tipo de ausencia.

Igualmente, le permitirá al usuario autenticarse mediante la comprobación de la clave introducida, y verificar que coincide con la almacenada en la base de datos y que se corresponde a la del servidor LDAP.

3.3 Historias de usuario

Las HU constituyen una forma rápida de administrar los requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. La HU es una representación de un requisito de software donde se utiliza el lenguaje común del usuario (Suaza, 2013).

3.3.1 Descripción de las historias de usuario

Se describen a continuación las HU de los RF9 y RF13.

Tabla 1 HU9 Gestionar actividad docente

Historias de Usuario	
Número: HU13	**Nombre del requisito:** Gestionar actividad docente
Programador: Arlen Neira Falcón	**Iteración Asignada:** 1ra
Prioridad: Media	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: • Rotura de la estación de trabajo. • Problemas eléctricos.	**Tiempo Real:** 40 horas
Descripción: El sistema debe mostrar la opción “Actividad Docente”, al elegir esta opción se debe mostrar la vista “Gestionar actividades docentes” la cual debe permitir realizar la operación de “Adicionar”. En esta vista adicionalmente el sistema deberá listar todas las actividades docentes almacenadas en la base de datos y para cada actividad docente debe permitir la realización de las acciones “Actualizar” y “Eliminar”. Para adicionar una actividad docente, el sistema debe listar los registros que se tienen almacenados, permitir al usuario especificar el nombre, una descripción y la fecha, y listar los tipos de actividades para que el usuario pueda escoger una. Al “Adicionar”, el sistema debe: agregar la nueva actividad docente a la base de datos, notificar al usuario que la actividad docente fue adicionada satisfactoriamente y actualizar en tiempo real el listado de actividades docentes almacenadas. En cambio, si la actividad docente se encuentra registrada se debe notificar al usuario con un mensaje de error que la actividad docente especificada ya existe para que defina otra. La opción “Cancelar” deberá limpiar los campos, mientras que la opción “Cerrar” deberá cerrar el formulario. Al realizar la acción “Actualizar” sobre la actividad docente seleccionada, el sistema debe abrir la ventana “Actualizar actividad” y brindar al usuario la posibilidad de modificar los datos de la actividad docente. En esta ventana, se deben cargar los datos previamente almacenados y mostrar las opciones “Actualizar” y “Cancelar” para que sea actualizada la actividad docente y cerrada la ventana respectivamente. El sistema con la opción “Eliminar” debe suprimir de la base de datos la actividad docente seleccionada y	

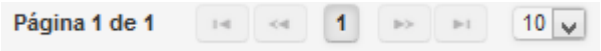
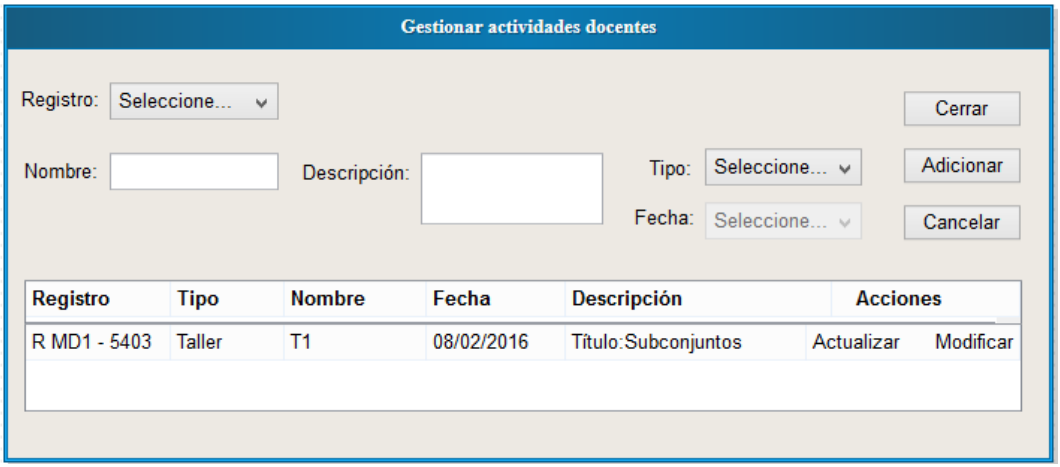
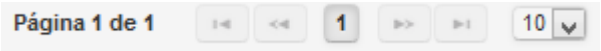
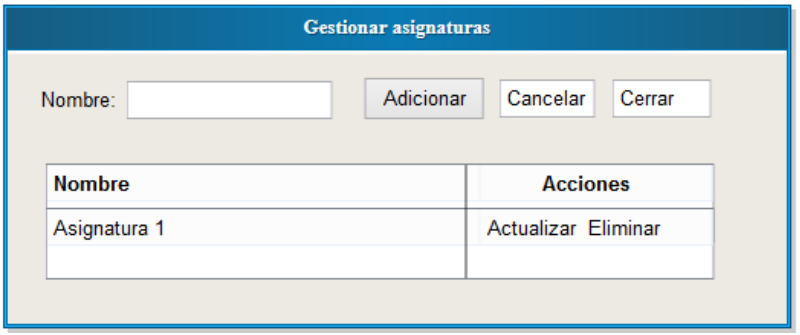
actualizar el listado de actividades docentes que se encuentran almacenadas en la base de datos.
Observaciones: Para listar las asignaturas se debe hacer uso de la paginación en la que se determina además el número de filas a mostrar en cada página. 
Prototipo de interfaz: 

Tabla 2 HU13 Gestionar asignatura

Historias de Usuario	
Número: HU4	**Nombre del requisito:** Gestionar asignatura
Programador: Arlen Neira Falcón	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: • Rotura de la estación de trabajo. • Problemas eléctricos.	**Tiempo Real:** 40 horas
Descripción: El sistema debe mostrar la opción “Asignatura”, al elegir esta opción se debe mostrar la vista “Gestionar asignaturas” la cual debe permitir realizar la operación de “Adicionar”. En esta vista adicionalmente el sistema deberá listar todas las asignaturas almacenadas en la base de datos y para cada una debe permitir la realización de las acciones “Actualizar” y Eliminar. Para adicionar una asignatura, el sistema debe permitir al usuario que especifique de la asignatura el nombre. Al “Adicionar”, el sistema debe añadir la nueva asignatura a la base de datos, notificar al usuario que la asignatura fue adicionada satisfactoriamente y actualizar en tiempo real el listado de asignaturas almacenadas. En cambio, si la asignatura se encuentra registrada se debe notificar al usuario con un mensaje de error que la asignatura especificada ya existe para que defina un nuevo nombre. La opción “Cancelar” deberá limpiar el campo nombre mientras que la opción “Cerrar” deberá cerrar el formulario.	

Al realizar la acción “Actualizar” sobre la asignatura seleccionada, el sistema debe mostrar la ventana “Actualizar asignatura” y brindar al usuario la posibilidad de modificar los datos de la asignatura. En esta ventana, se deben cargar los datos previamente almacenados y mostrar las opciones “Actualizar” y “Cancelar” para que sea actualizada la asignatura y cerrada la ventana respectivamente. El sistema con la opción “Eliminar” debe suprimir de la base de datos la asignatura seleccionada y actualizar el listado de asignaturas.
Observaciones: Para listar las asignaturas se debe hacer uso de la paginación en la que se determina además el número de filas a mostrar en cada página. 
Prototipo de interfaz: 

3.3.2 Planificación de las iteraciones

Luego de identificar, describir las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas. El paso a seguir es especificar las HU que serán implementadas para cada iteración del sistema. La planificación de las iteraciones se realizó teniendo en cuenta la prioridad de las HU.

Iteración 1

En esta iteración se implementarán primero las HU que por su alta prioridad deben ser construidas antes, debido a que las restantes HU dependen de ellas para su realización.

Tabla 3 Planificación de las iteraciones

Iteración	Prioridad	Orden de las HU a implementar		Duración
1ra	Alta	1. Gestionar estudiante 2. Gestionar profesores	3. Gestionar grupos 4. Gestionar asignaturas	4 semanas
1ra	Normal	5. Mostrar información de los estudiantes.	13. Gestionar actividad docente	12 semanas

		6. Graficar estado de cada estudiante 7. Mostrar información de un grupo 8. Graficar información de un grupo 9. Asignar grupos a un profesor 10. Gestionar registro de asistencia 11. Exportar registro de asistencia (excel o pdf) 12. Graficar asistencia de un grupo	14. Gestionar ausencia 15. Gestionar evaluación 16. Generar corte evaluativo 17. Graficar corte evaluativo 18. Mostrar gráfico sobre el estado de evaluaciones de un grupo 19. Mostar listado de corte evaluativo 20. Obtención desde el LDAP del grupo según el profesor	

3.4 Arquitectura del sistema

La IEEE 1471-2000 define la arquitectura de software como: *"La organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente, y los principios que orientan su diseño y evolución"* (IEEE, 2000).

Pressman define la arquitectura del software de un programa o sistema de cómputo como: "*la estructura o las estructuras del sistema, que incluye los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos. La arquitectura no es el software operativo. En cambio, es una representación que permite que un ingeniero de software: 1) analice la efectividad del diseño para cumplir con los requisitos establecidos, 2) considere opciones arquitectónicas en una etapa en que aún resulta relativamente fácil hacer cambios al diseño, y 3) reduzca los riesgos asociados con la construcción del software* (Lantigua, 2013).

A partir de los elementos anteriormente expuestos se puede concluir que la arquitectura en el desarrollo de software es necesaria para comprender el sistema, organizar el desarrollo, fomentar la reutilización y hacer evolucionar el sistema. Para definirla es necesario seleccionar y combinar patrones. Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Cada patrón describe un problema que ocurre una y otra vez, y luego describe el núcleo de la

solución a ese problema, de tal manera que puedes usar esa solución varias veces más, sin hacer la misma acción dos veces. Según la escala o nivel de abstracción presentan tres categorías:

- **Patrones arquitectónicos**: aquellos que expresan un esquema organizativo estructural fundamental para sistemas informáticos, también conocidos como estilos.
- **Patrones de diseño**: aquellos que expresan esquemas para definir estructuras de diseño o sus relaciones con las que construir software.
- **Idiomas**: patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto. (Larman, 2003).

3.4.1 Estilos y patrones arquitectónicos

El estilo arquitectónico **Llamada y retorno** permite que un diseñador de software (arquitecto del sistema) obtenga una estructura del programa que resulta relativamente fácil modificar y cambiar de tamaño (Lantigua, 2013). Existen diferentes patrones arquitectónicos, entre ellos se encuentran las arquitecturas en capas, el modelo-vista-controlador, los sistemas orientados a objeto y los sistemas basados en componentes.

Características del estilo arquitectónico **Llamada y retorno**:

- El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Admite muy naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización.
- Ayuda a controlar y encapsular aplicaciones complejas.

Arquitectura en capas

La arquitectura en tres capas es muy utilizada tanto en aplicaciones web como en aplicaciones de escritorio, esta ofrece diversas ventajas: los datos y servicios aparecen separados y el cliente recibe los datos así como la información de forma indirecta. Además, esta arquitectura describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.

La arquitectura para el desarrollo de la solución propuesta está concebida en tres capas: capa de presentación, negocio y datos.

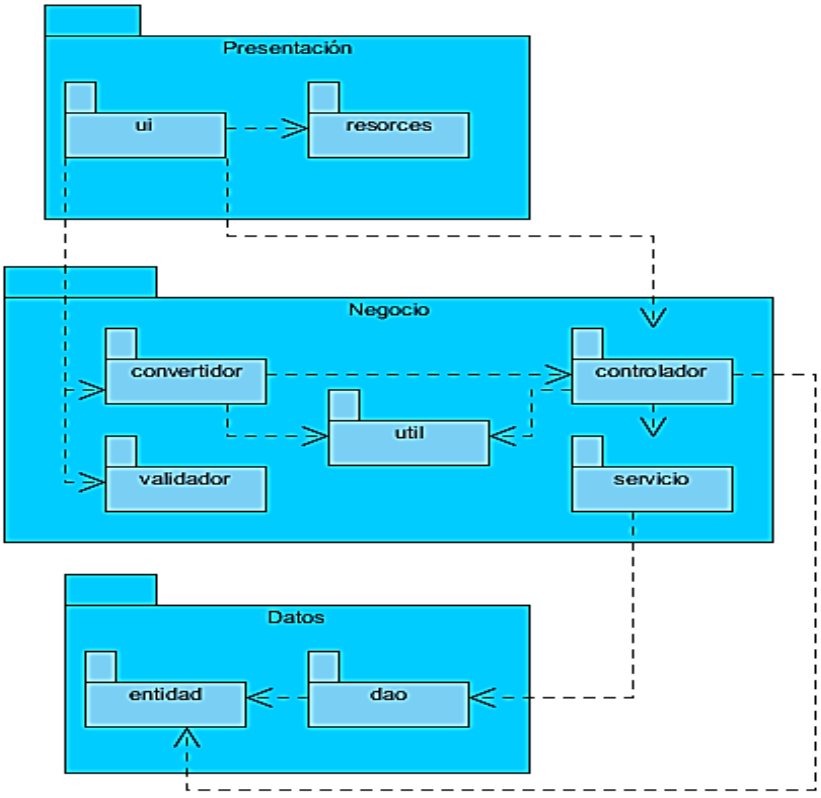


Figura 2 Arquitectura del sistema

Fuente: elaboración propia

Capa de presentación: es la que ve el usuario denominada también como "capa de usuario", presenta el sistema al usuario, comunica y captura la información del usuario en un mínimo de proceso. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio e incluso de lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la de presentación, para recibir las solicitudes y presentar los resultados, y con la de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

3.4.2 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño; facilitan la reusabilidad, extensibilidad y mantenimiento. Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades (Guerrero, 2013).

Patrón DAO

El patrón DAO (Objeto de Acceso a Datos, del inglés *Data Access Object.*) independiza la aplicación de la forma de acceder a la base de datos, o cualquier otro tipo de repositorio (base de datos, archivos, servicios externos, etcétera). Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento. Un DAO define la relación entre la lógica de presentación y empresa por una parte y por otra los datos. El DAO tiene un interfaz común, sea cual sea el modo y fuente de acceso a datos. No es imprescindible, pero en proyectos de cierta complejidad resulta útil que el DAO implemente una interfaz. De esta forma los objetos cliente tienen una forma unificada de acceder a los DAO (ver Anexo 1).

Fuera de las clases DAO no debe haber ningún tipo de código que acceda al repositorio de datos. El DAO accede a la fuente de datos y la encapsula para los objetos clientes. Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente, la cual podría ser JDBC (Dao, 2011).

Patrones GRASP

A continuación, se muestran los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades, del inglés *General Responsibility Assignment Software Patterns*) utilizados:

- **Experto**: las responsabilidades de las entidades han sido debidamente asignadas. Cada clase cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan. Por ejemplo, la clase *EstudianteControlador* cuenta con la información necesaria para cumplir la responsabilidad de manejar los datos del estudiante, mientras que la clase *ProfesorControlador* se responsabiliza de manejar la información concerniente a los profesores.
- **Creador**: cada instancia de un objeto es creada por el objeto que tiene la información necesaria para ello. Consiste en asignarle a una clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:
    - B agrega los objetos A.
    - B contiene los objetos A.
    - B registra las instancias de los objetos A.

- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.

Por ejemplo, a la clase *EstudianteControlador* se le asigna la responsabilidad de crear una instancia de *Estudiante* porque:

- *EstudianteControlador* agrega los objetos Estudiante.
- *EstudianteControlador* contiene los objetos Estudiante.
- *EstudianteControlador* registra las instancias de los objetos Estudiante.
- *EstudianteControlador* utiliza específicamente los objetos Estudiante.
- *EstudianteControlador* tiene los datos de inicialización que serán transmitidos a Estudiante cuando este objeto sea creado.

Por tanto, *EstudianteControlador* es un creador de los objetos de *Estudiante* (ver Anexo 2).

- **Controlador**: se encarga de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas, y de facilitar la centralización de actividades. Se han diseñado clases controladoras para asignar las responsabilidades de controlar cada flujo de eventos del sistema.
- **Alta Cohesión**: como cada clase tiene un conjunto de funcionalidades relacionadas directamente con la entidad que representan, no realizan un trabajo enorme y por tanto pueden ser calificadas como de alta cohesión.
- **Bajo acoplamiento**: las clases se comunican solamente con las necesarias para desarrollar cada flujo de evento. Por ejemplo, la clase *EstudianteControlador* se comunica con la menor cantidad de clases posibles para el flujo de evento que adiciona a un estudiante en la base de datos (Larman, 2003) (ver Anexo 3).

Patrones GOF

Se utilizaron además los patrones GoF (Banda de los Cuatro, del inglés *Gang-of-Four*) los cuales se clasifican en dependencia del propósito para los que hayan sido definidos: creación, estructurales y de comportamiento (Guerrero, 2013).

- **Patrón Singleton:** es utilizado en la aplicación para garantizar que solo haya una instancia a la vez de los formularios.

3.5 Modelo de diseño

El modelo de diseño describe la realización de los casos de uso y se utiliza como una abstracción del modelo de implementación y el código fuente. Se usa como una entrada inicial en las actividades de implementación y prueba (ISW, 2011).

3.5.1 Diagrama de clases del diseño

Los diagramas de clases exponen un conjunto de interfaces, colaboraciones y sus restricciones. Se utilizan para modelar la vista de diseño estática de un sistema. Son importantes para visualizar, especificar, documentar modelos estructurales y construir sistemas ejecutables con la aplicación de ingeniería directa e inversa. Es una representación concreta de lo que debe ser implementado (Colectivo ISW, 2011) (Larman, 2003).

Tabla 4 Descripción de los estereotipos de clases del diseño

Clases	Función
Páginas clientes, del inglés Client Page	Representa una página Web, con formato HTML. Son interpretadas por el navegador.
Formularios	Colección de elementos de entrada que son parte de una página cliente. Su función es visualizar, interactuar y mostrar lo que el usuario necesita.
Páginas servidoras, del inglés Server Page	Representa la página Web que tiene código que se ejecuta en el servidor. Su función principal es construir la página cliente.

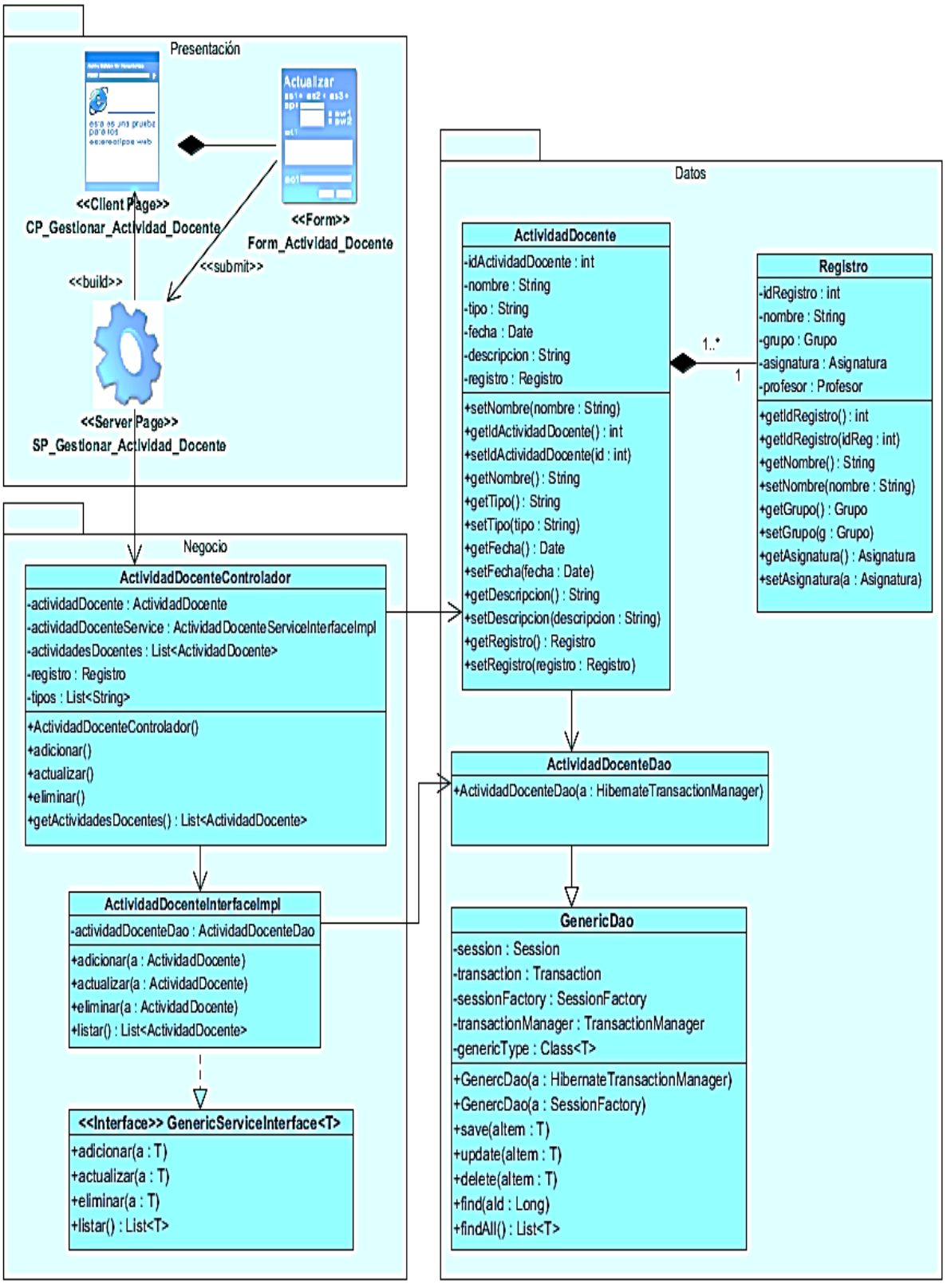


Figura 3 Diagrama de clases de la HU13

Fuente: elaboración propia

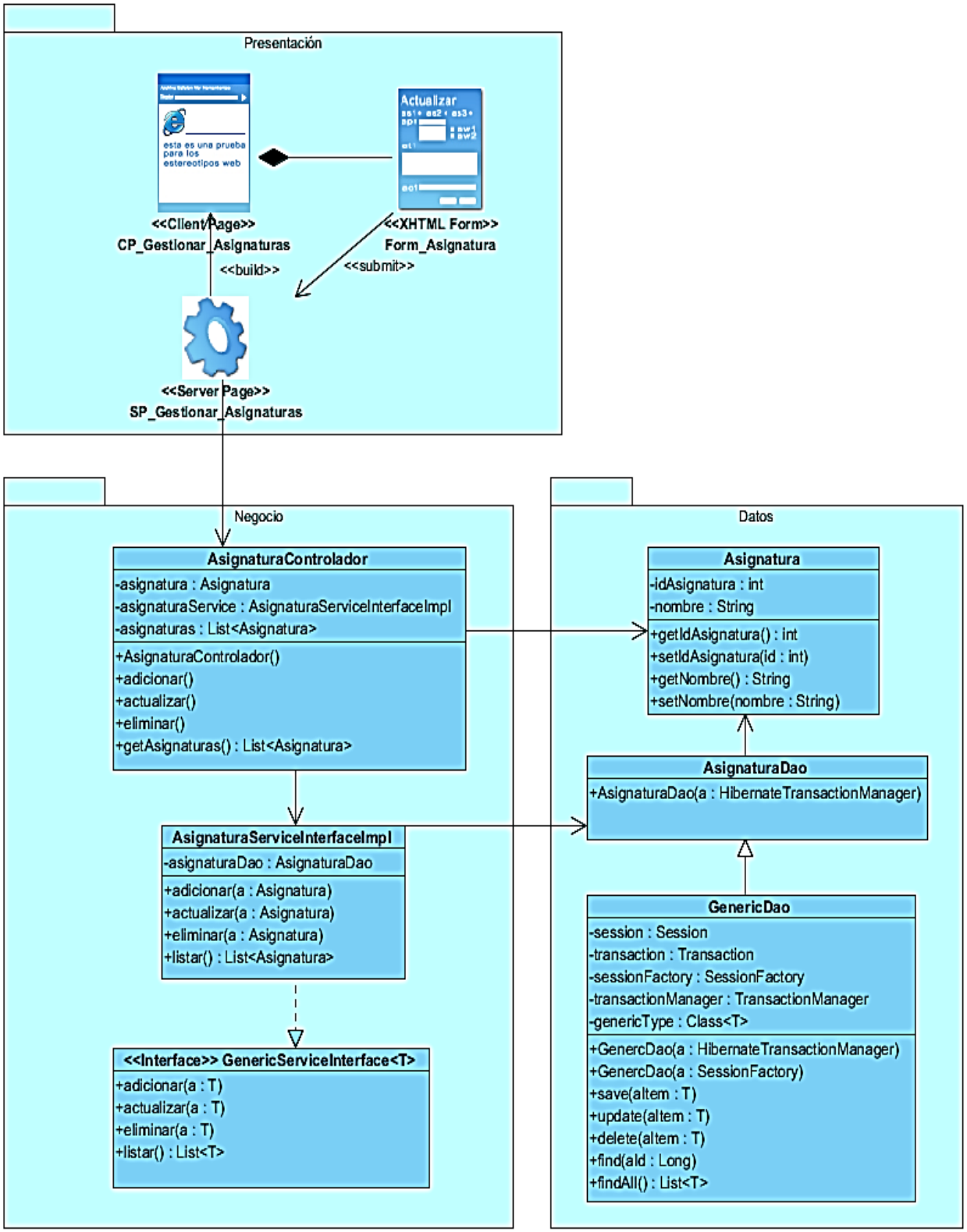


Figura 4 Diagrama de clases de la HU4

Fuente: elaboración propia

3.5.2 Descripción de las clases del diseño

Tabla 5 Descripción textual de la clase Asignatura

Descripción de la clase	
Nombre: Asignatura	
Tipo de clase: Modelo	
Atributos	**Tipos**
idAsignatura	private
nombre	private
Responsabilidades	
Nombre:	Asignatura()
Descripción:	Constructor de la clase, responsable de inicializar los atributos definidos.
Nombre:	getIdAsignatura(): int
Descripción:	Responsable de obtener el identificador de la asignatura.
Nombre:	setIdAsignatura (id: int): void
Descripción:	Responsable de actualizar el identificador de la asignatura por el especificado por parámetro.
Nombre:	getNombre (): String
Descripción:	Responsable de obtener el nombre de la asignatura.
Nombre:	setNombre (nombre: String): void
Descripción:	Responsable de actualizar el nombre de la asignatura por el especificado por parámetro.

Tabla 6 Descripción textual de la clase AsignaturaControlador

Descripción de la clase	
Nombre: AsignaturaControlador	
Tipo de clase: Controladora	
Atributos	**Tipos**
asignatura	private
asignaturaService	private
asignaturas	private
Responsabilidades	
Nombre:	AsignaturaControlador()
Descripción:	Constructor de la clase, responsable de inicializar los atributos definidos.
Nombre:	adicionar(): void
Descripción:	Responsable de adicionar al sistema las asignaturas.
Nombre:	actualizar(): void
Descripción:	Responsable de actualizar satisfactoriamente los datos de una determinada asignatura.
Nombre:	eliminar(): void

Descripción:	Responsable de eliminar del sistema una asignatura determinada por el usuario.
Nombre:	getAsignaturas(): List<Asignaturas>
Descripción:	Responsable de obtener las asignaturas existentes en el sistema.

Tabla 7 Descripción textual de la clase GenericServiceInterface

Descripción de la clase	
Nombre: GenericServiceInterface	
Tipo de clase: Interfaz	
Responsabilidades	
Nombre:	adicionar(a:T): void
Descripción:	Responsable de invocar a la funcionalidad del dao que salva el objeto parametrizado.
Nombre:	actualizar(a:T): void
Descripción:	Responsable de invocar a la funcionalidad del dao que actualiza el objeto parametrizado.
Nombre:	eliminar(a:T): void
Descripción:	Responsable de invocar a la funcionalidad del dao que elimina el objeto parametrizado.
Nombre:	listar(): List<T>
Descripción:	Responsable de invocar a la funcionalidad del dao que lista los objetos del tipo parametrizado almacenados en la base de datos.

Tabla 8 Descripción textual de la clase AsignaturaServiceInterfaceImpl

Descripción de la clase	
Nombre: AsignaturaServiceInterfaceImpl	
Tipo de clase: Auxiliar	
Atributos	**Tipos**
asignaturaDao	private
Responsabilidades	
Nombre:	adicionar(a: Asignatura): void
Descripción:	Responsable de invocar a la funcionalidad del dao que salva la asignatura especificada por parámetro.
Nombre:	actualizar(a: Asignatura): void
Descripción:	Responsable de invocar a la funcionalidad del dao que modifica los datos de la asignatura especificada por parámetro.
Nombre:	eliminar(a: Asignatura): void
Descripción:	Responsable de invocar a la funcionalidad del dao que elimina la asignatura especificada por parámetro.
Nombre:	listar(): List<Asignatura>
Descripción:	Responsable de invocar a la funcionalidad del dao que lista las asignaturas que se tienen almacenadas en la base de datos.

Tabla 9 Descripción textual de la clase GenericDao

Descripción de la clase	
Nombre: GenericDao	
Tipo de clase: Auxiliar	
Atributos	**Tipos**
session	protected
transaction	protected
sessionFactory	protected
transactionManager	protected
genericType	protected
Responsabilidades	
Nombre:	save(aItem: T): void
Descripción:	Salva en la base de datos el objeto parametrizado.
Nombre:	update(aItem: T): void
Descripción:	Actualiza en la base de datos el objeto parametrizado.
Nombre:	delete(aItem: T): void
Descripción:	Elimina de la base de datos el objeto parametrizado.
Nombre:	find(aId: Long): List<T>
Descripción:	Busca en la base de datos los objetos del tipo parametrizado cuyo identificador coincide con el especificado.
Nombre:	findAll(): List<T>
Descripción:	Busca en la base de datos los objetos del tipo parametrizado que se tienen almacenados.

Tabla 10 Descripción textual de la clase AsignaturaDao

Descripción de la clase	
Nombre: AsignaturaDao	
Tipo de clase: Auxiliar	
Responsabilidades	
Nombre:	AsignaturaDao(a: HibernateTransactionManager): void
Descripción:	Constructor de la clase.

3.5.3 Modelo de datos

Un modelo de datos, es un conjunto de conceptos que permiten describir los datos, las relaciones que existen entre ellos, la semántica y las restricciones de consistencia.

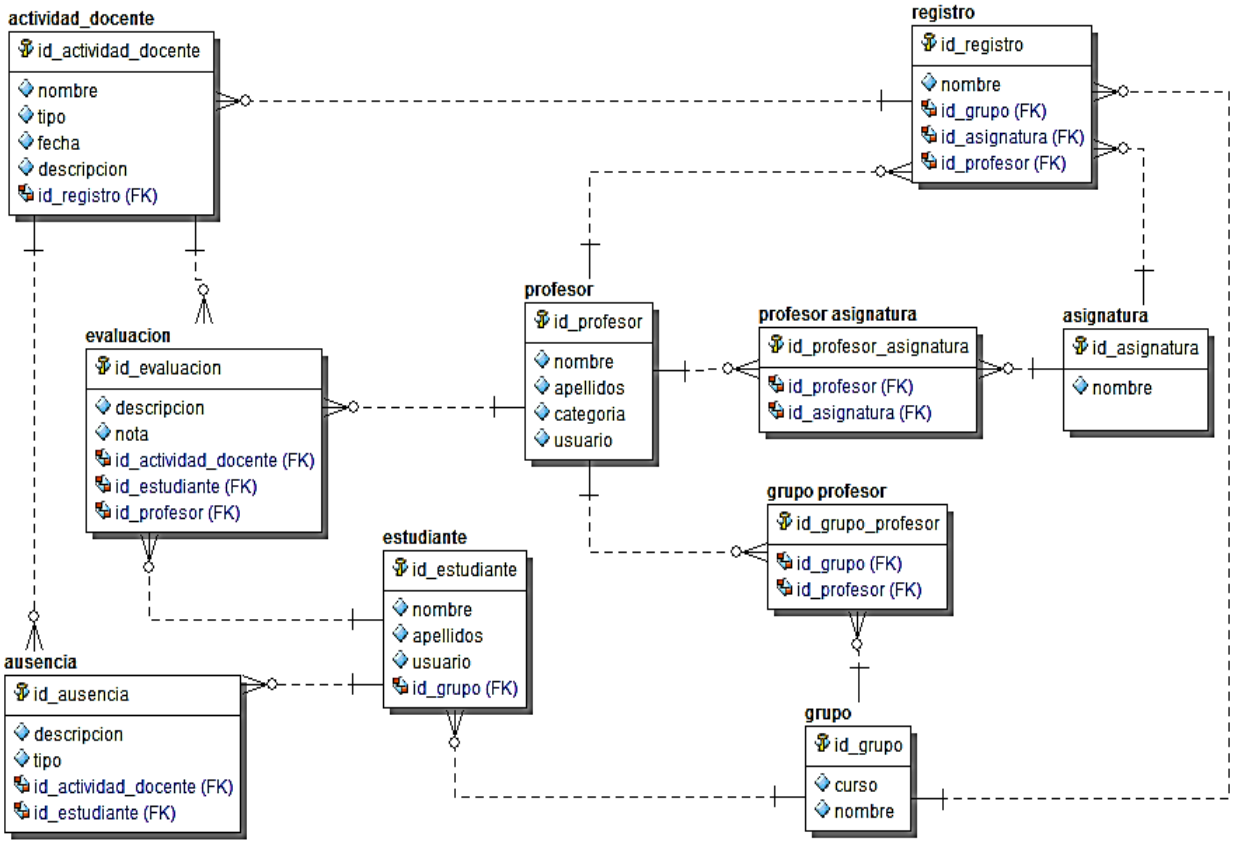


Figura 5 Modelo de datos

Fuente: elaboración propia

3.6 Modelo de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los enlaces de comunicación entre ellos, las instancias de los componentes y objetos que residen en ellos. El modelo de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos (Colectivo ISW, 2011).

El diagrama de despliegue está compuesto por una computadora (PC cliente) que se conecta al servidor web a través de los protocolos HTTPS[12] y el servidor web a su vez se conecta al servidor de bases de datos mediante el protocolo TCP/IP[13]. En la siguiente figura se modela la distribución física de los nodos necesarios para la implantación del sistema.

[12] Protocolo Seguro de Transferencia de Hiper-Texto, del inglés *Hipertext Transference Protocol Security*.

[13] Protocolo de Control de Transmisión/Protocolo de Internet, del inglés *File Transference Protocol / Internet Protocol*.

Figura 6 Diagrama de despliegue

Fuente: elaboración propia

Tabla 11 Observaciones importantes de los nodos

Nodos	Observaciones
PC Cliente	Computadora que contará con un navegador actualizado a través del cual se accederá al sistema.
Servidor Web	Representa una estación donde estará montado el Servidor Apache en el cual correrá la aplicación.
Servidor de Base de datos	Representa el servidor donde estará el Sistema Gestor de Bases de Datos PostgreSQL que dará respuesta a las peticiones hechas por la aplicación.

3.7 Seguridad del sistema propuesto

Con el uso del framework Spring se facilita el trabajo con usuarios y roles, protección de las URL[14], autenticación básica y aseguramiento de métodos de la capa de negocio. Para evitar ataques a la base de datos (BD) el framework Hibernate hace sus consultas mediante la sintaxis HQL[15] para construir una consulta dinámica de SQL, la cual no es vulnerable a las inyecciones de código SQL.

Para garantizar la seguridad en el sistema propuesto se concibieron dos roles: administrador del sistema y profesor. El administrador puede acceder a los módulos Asignatura, Profesor, Estudiante y Grupo y es el responsable de crear cada uno de ellos. El usuario del profesor deberá ser insertado en la BD de la aplicación por el usuario administrador para que después este pueda autenticarse mediante su usuario del LDAP en el sistema, de no ser agregado por el administrador no podrá acceder a la aplicación. El profesor solo podrá acceder y modificar los registros creados por él mismo y gestionar los grupos que le han sido previamente asignados. Otra medida de seguridad utilizada es que inmediatamente de transcurrir 15 minutos de inactividad, el sistema cerrará la sesión.

[14] Localizador de recursos uniforme (*URL*, del inglés *Uniform Resource Locator*)

[15] Lenguaje de consulta de datos llamado Hibernate Query Language (HQL por sus siglas en inglés).

2.10 Consideraciones parciales del capítulo

En este capítulo se identificaron los requisitos funcionales y no funcionales del sistema que dieron paso a las historias de usuario, las cuales permitieron describir cada uno de los requisitos funcionales según su prioridad y estimado de tiempo. Con la implementación de los requisitos especificados, se construirá un sistema que brindará solución a la problemática planteada.

CAPÍTULO 3. CONSTRUCCIÓN Y VALIDACIÓN DEL SISTEMA

En el presente capítulo se describe cómo los elementos del modelo de diseño se implementan en términos de componentes mediante la representación del Modelo de implementación. Posteriormente se realiza el proceso de pruebas en el cual se describe la estrategia de pruebas seleccionada para validar la calidad del sistema.

3.1 Modelo de implementación

El modelo de implementación describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Rumbaugh y Jacobson, 2000).

3.1.1 Diagramas de componentes

Un diagrama de componente describe la descomposición física del sistema de software en componentes (código fuente, binario, ejecutable, así como librerías dinámicas, páginas web, interfaces, un conjunto de relaciones de dependencia, generalización, asociación y realización), así como la relación que existen entre ellos en el sistema (Pressman, 2002).

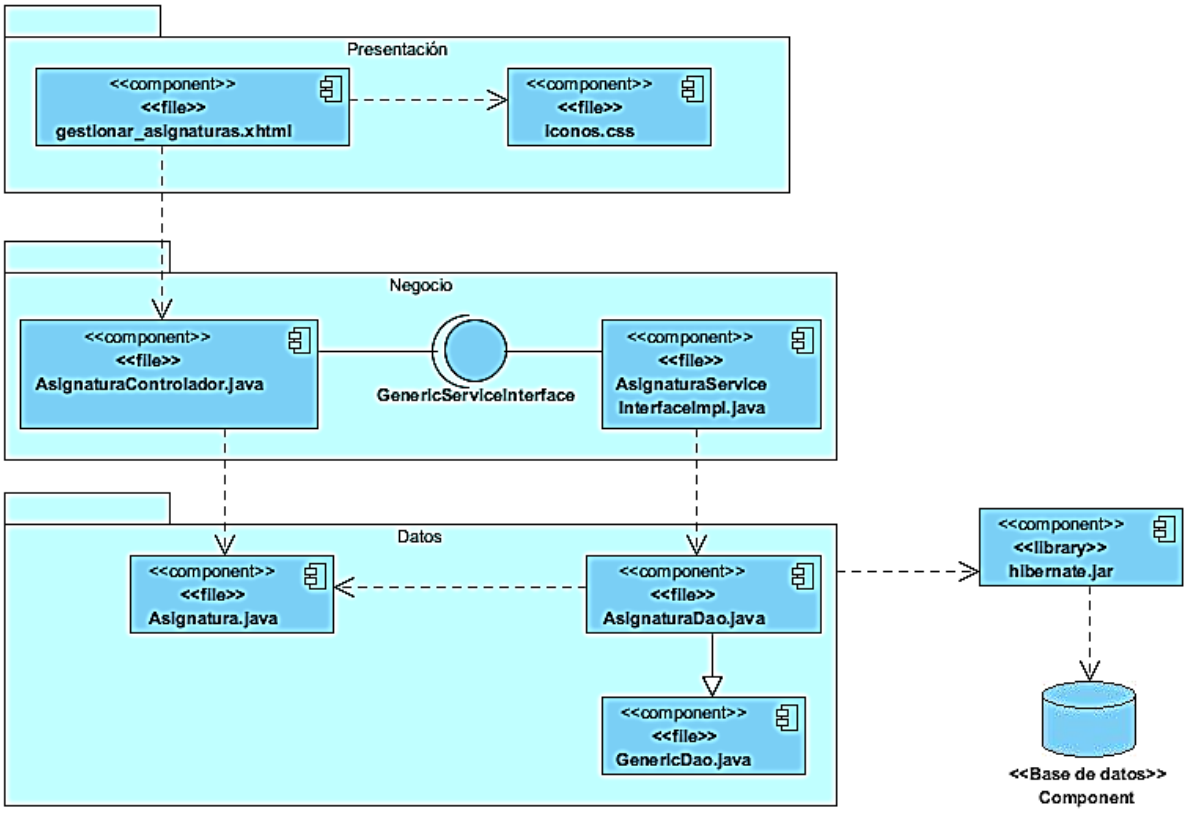


Figura 7 Diagrama de componentes del diagrama de clases de la HU13

Fuente: elaboración propia

3.2 Código fuente

Se define como código fuente al conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por humanos. Es un programa en su forma original, tal y como fue escrito por el programador, no es ejecutable directamente por el computador, debe convertirse en lenguaje de máquina mediante compiladores, ensambladores o intérpretes (Deitel, 2004).

Con el objetivo de alcanzar un mayor entendimiento del código fuente se utilizó el estándar de codificación JAVA. Entre las reglas definidas se encuentran:

- Utilizar la variante *lowerCamelCase* para los identificadores del tipo variables y métodos. Las palabras comenzarán con minúsculas y si los identificadores están compuestos por varias palabras, las siguientes comenzarán con mayúscula.
- Utilizar la variante *UpperCamelCase* para los identificadores del tipo clase, enumeradores e interfaces. Todas las palabras que componen a dichos identificadores comenzarán con mayúscula.

A continuación, se muestran fragmentos de código fuente pertenecientes a implementaciones importantes de la clase *AsignaturaControlador*:

Tabla 12 Código fuente del método adicionar asignatura

AsignaturaControlador

```
public void adicionar() {
    try {
        asignaturaServiceInterfaceImpl.adicionar(asignatura);
        init();
        FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_INFO,
                "Adicionar Asignatura", "La asignatura ha sido adicionada satisfactoriamente.");
        FacesContext.getCurrentInstance().addMessage(null, msg);
    } catch (DatabaseAccessException ex) {
        Logger.getLogger(AsignaturaControlador.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

3.3 Validación del diseño

La evaluación del sistema a partir de métricas de software es una medida cuantitativa que permite tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar soluciones para rectificar y mejorar el proceso del software.

Con el objetivo de medir el grado en que las características del sistema cumplen con los requisitos planteados con anterioridad, se emplean las métricas Tamaño Operacional de la Clase (TOC) y Relaciones entre Clases (RC). Dichas métricas están diseñadas para evaluar los siguientes atributos de calidad:

- **Responsabilidad:** se le asigna a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación**: grado de dificultad en la implementación de un diseño de clases determinado.
- **Reutilización:** nivel de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento:** grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento:** grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirectamente, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas:** número o grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, entre otras) diseñado (Pressman, 2002).

En un lenguaje orientado a objetos como los es el JAVA, las clases constituyen la unidad básica y fundamental. Por tanto, debido a que la solución está realizada sobre este tipo de programación, su validación está centrada en la aplicación de métricas dirigidas a sus clases de forma individual, sus jerarquías y colaboraciones, haciendo uso de los atributos de calidad descritos anteriormente.

3.3.1 Métrica Tamaño operacional de clases

Tamaño Operacional de Clases (TOC): está dado por el número de métodos u operaciones (de instancia privada y heredada) que están encapsulados dentro o por una clase (Lantigua, 2013). Evalúa los atributos de calidad contenidos en la tabla 13.

Tabla 13 Atributos de calidad que evalúa TOC

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Para la evaluación de estos atributos de calidad se definieron criterios y categorías de evaluación. Para ver estos criterios consultar el Anexo 4 Criterios de evaluación de la métrica TOC.

La tabla que se muestra a continuación ofrece las clases del sistema a las que se le aplicó la métrica y los resultados obtenidos para cada atributo evaluado. Para determinar el valor de los atributos, se calcula el promedio de la columna cantidad de procedimientos, en este caso el promedio es 7,16.

Tabla 14 Resultados de la aplicación de la métrica TOC para cada clase del sistema

Clase	Cantidad de procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
ActividadDocente Controlador	15	Alta	Alta	Baja
AsignaturaControlador	11	Media	Media	Media
AusenciaControlador	11	Media	Media	Media
EstudianteControlador	14	Media	Media	Media
EvaluacionControlador	13	Media	Media	Media
FacesController	9	Media	Media	Media
GrupoControlador	11	Media	Media	Media
LoginController	26	Alta	Alta	Baja
ProfesorControlador	11	Media	Media	Media
RegistroControlador	11	Media	Media	Media
ActividadDocente Convertidor	2	Baja	Baja	Alta
AsignaturaConvertidor	2	Baja	Baja	Alta
EstudianteConvertidor	2	Baja	Baja	Alta
GrupoConvertidor	2	Baja	Baja	Alta
ProfesorConvertidor	2	Baja	Baja	Alta
RegistroConvertidor	2	Baja	Baja	Alta
GenericDao	15	Alta	Alta	Baja
DatabaseAccess Exception	1	Baja	Baja	Alta
ActividadDocenteDao	2	Baja	Baja	Alta
AsignaturaDao	1	Baja	Baja	Alta
AusenciaDao	1	Baja	Baja	Alta
EstudianteDao	1	Baja	Baja	Alta
EvaluacionDao	1	Baja	Baja	Alta
GrupoDao	1	Baja	Baja	Alta
GrupoProfesorDao	1	Baja	Baja	Alta
ProfesorAsignaturaDao	1	Baja	Baja	Alta
ProfesorDao	1	Baja	Baja	Alta
RegistroDao	1	Baja	Baja	Alta
ActividadDocente	16	Alta	Alta	Baja
Asignatura	7	Baja	Baja	Alta
Ausencia	12	Media	Media	Media

Clase	Cantidad de procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
Estudiante	13	Media	Media	Media
Evaluacion	14	Media	Media	Media
Grupo	10	Media	Media	Media
GrupoProfesor	7	Baja	Baja	Alta
Profesor	13	Media	Media	Media
ProfesorAsignatura	7	Baja	Baja	Alta
Registro	13	Media	Media	Media
GenericServiceInterface	4	Baja	Baja	Alta
ActividadDocenteService InterfaceImpl	6	Baja	Baja	Alta
AsignaturaService InterfaceImpl	6	Baja	Baja	Alta
AusenciaService InterfaceImpl	6	Baja	Baja	Alta
EstudianteService InterfaceImpl	6	Baja	Baja	Alta
EvaluacionService InterfaceImpl	6	Baja	Baja	Alta
GrupoService InterfaceImpl	6	Baja	Baja	Alta
ProfesorService InterfaceImpl	6	Baja	Baja	Alta
RegistroService InterfaceImpl	7	Baja	Baja	Alta
Ldap	2	Baja	Baja	Alta
PasarelaAutenticacion Util	1	Baja	Baja	Alta
AlphaValidator	4	Baja	Baja	Alta
EmailValidator	4	Baja	Baja	Alta
NumberValidator	4	Baja	Baja	Alta
RealValidator	4	Baja	Baja	Alta
StringValidator	4	Baja	Baja	Alta
FacesUtil	35	Alta	Alta	Baja

La figura 8 muestra la representación de los resultados obtenidos agrupados en los intervalos definidos. El gráfico refleja que la mayoría de las clases tienen de 1 a 5 procedimientos. Esto demuestra que el funcionamiento general del componente está distribuido equitativamente entre las diferentes clases.

Los resultados obtenidos luego de aplicar las métricas TOC arrojaron que el diseño propuesto tiene una calidad aceptable, teniendo en cuenta que aproximadamente el 67 % de las clases poseen una cantidad de procedimientos menor o igual al promedio general de 7.16, esto conlleva a que las evaluaciones sean positivas en los atributos de calidad involucrados.

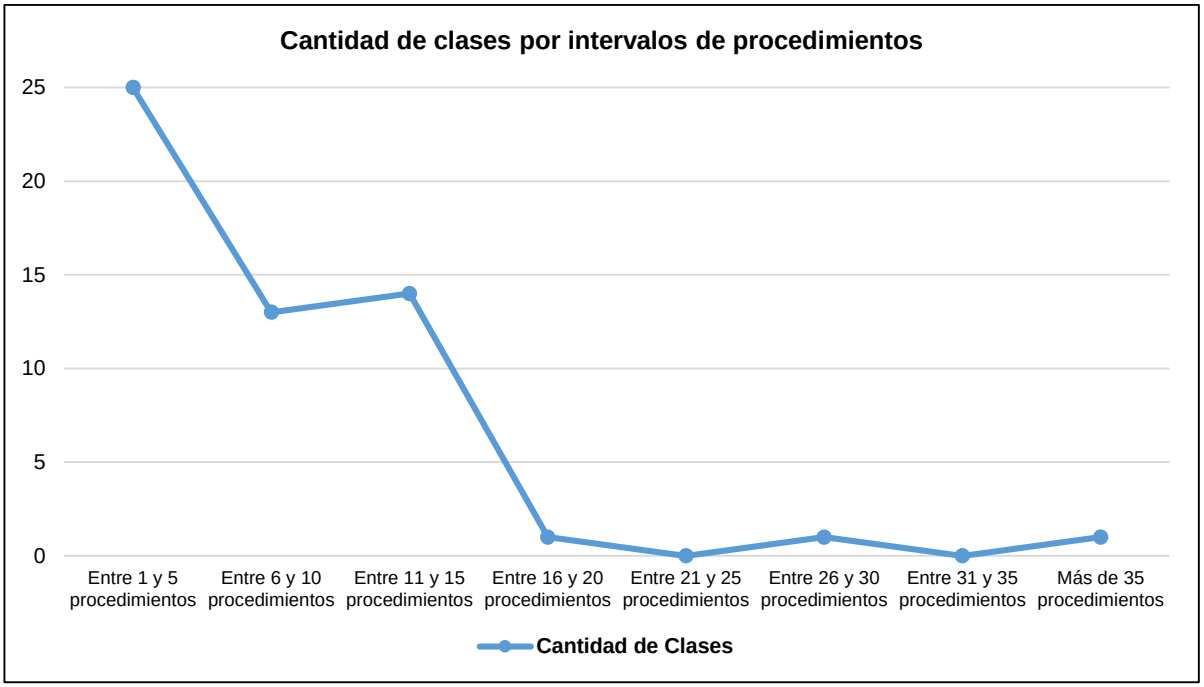


Figura 8 Representación de las clases según la cantidad de operaciones

La figura 9 muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad. Quedó demostrado que el 67% de las clases tienen una baja responsabilidad ya que este tributo se distribuyó equitativamente entre todas las clases del sistema. Esta característica permite que en caso de fallos como la responsabilidad está distribuida de forma equilibrada ningún componente sea demasiado crítico como para dejar fuera de servicio el sistema.

La figura 10 visualiza la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación. El gráfico muestra que el 67% de las clases tienen una baja complejidad, este atributo está distribuido equitativamente. Esta característica permite mejorar el mantenimiento y soporte de las clases.

La figura 11 deja ver la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización. Queda demostrado que el diseño de la solución es eficiente ya que el 67% de las clases tienen un alto grado de reutilización.

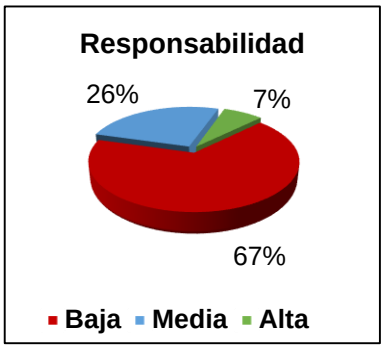


Figura 9 Resultados de la evaluación de la métrica TOC para el atributo de calidad Responsabilidad

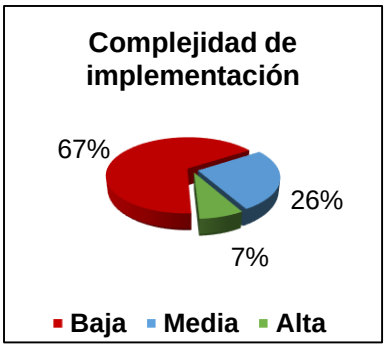


Figura 10 Resultados de la evaluación de la métrica TOC para el atributo de calidad Complejidad de implementación

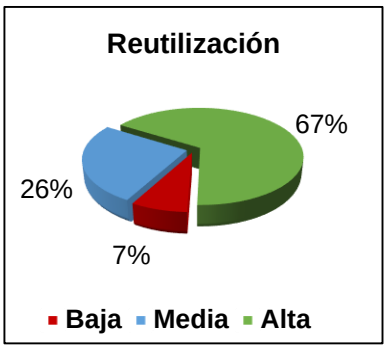


Figura 11 Resultados de la evaluación de la métrica TOC para el atributo de calidad Reutilización

Después de hacer un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño de la aplicación web GECMA v2.0 tiene una buena calidad.

3.3.2 Métrica Relaciones entre clases

Relaciones entre Clases (RC): está dado por el número de relaciones de uso de una clase con otra (Lantigua, 2013). Evalúa los atributos de calidad mostrados en la tabla 15.

Tabla 15 Atributos de calidad que evalúa RC

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Aumento del RC provoca aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Aumento del RC provoca aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para la evaluación de estos atributos de calidad se definieron criterios y categorías de evaluación. Para ver estos criterios consultar el Anexo 5 Criterios de evaluación de la métrica RC.

En la tabla 16 se muestran las clases del sistema a las que se le aplicó la métrica y los resultados obtenidos para cada atributo evaluado. El promedio utilizado para evaluar el criterio es el resultado del cálculo del promedio de la columna Cantidad de relaciones de uso, en este caso el promedio es 1,05.

Tabla 16 Resultados de la aplicación de la métrica RC para cada clase del sistema

Clase	Cantidad de relaciones de uso	Responsabilidad	Complejidad de implementación	Reutilización
ActividadDocente Controlador	3	Alta	Alta	Baja
AsignaturaControlador	2	Media	Media	Media
AusenciaControlador	2	Media	Media	Media
EstudianteControlador	2	Media	Media	Media
EvaluacionControlador	3	Alta	Alta	Baja
GrupoControlador	2	Media	Media	Media
LoginController	1	Baja	Baja	Alta
ProfesorControlador	2	Media	Media	Media
RegistroControlador	2	Media	Media	Media
ActividadDocente Convertidor	1	Baja	Baja	Alta
AsignaturaConvertidor	1	Baja	Baja	Alta
EstudianteConvertidor	1	Baja	Baja	Alta
GrupoConvertidor	1	Baja	Baja	Alta
ProfesorConvertidor	1	Baja	Baja	Alta
RegistroConvertidor	1	Baja	Baja	Alta
GenericDao	5	Alta	Alta	Baja
ActividadDocente	1	Baja	Baja	Alta
Ausencia	2	Media	Media	Media
Estudiante	1	Baja	Baja	Alta
Evaluacion	3	Alta	Alta	Baja
GrupoProfesor	2	Media	Media	Media
ProfesorAsignatura	2	Media	Media	Media
Registro	3	Alta	Alta	Baja
ActividadDocenteService InterfaceImpl	1	Baja	Baja	Alta
AsignaturaService InterfaceImpl	1	Baja	Baja	Alta
AusenciaService InterfaceImpl	1	Baja	Baja	Alta
EstudianteService InterfaceImpl	1	Baja	Baja	Alta
EvaluacionService InterfaceImpl	1	Baja	Baja	Alta
GrupoService InterfaceImpl	1	Baja	Baja	Alta
ProfesorService InterfaceImpl	1	Baja	Baja	Alta
RegistroService InterfaceImpl	2	Media	Media	Media
AlphaValidator	1	Baja	Baja	Alta
EmailValidator	1	Baja	Baja	Alta

Clase	Cantidad de relaciones de uso	Responsabilidad	Complejidad de implementación	Reutilización
NumberValidator	1	Baja	Baja	Alta
RealValidator	1	Baja	Baja	Alta
StringValidator	1	Baja	Baja	Alta
FacesController	0	Ninguno	Baja	Alta
DatabaseAccess Exception	0	Ninguno	Baja	Alta
ActividadDocenteDao	0	Ninguno	Baja	Alta
AsignaturaDao	0	Ninguno	Baja	Alta
AusenciaDao	0	Ninguno	Baja	Alta
EstudianteDao	0	Ninguno	Baja	Alta
EvaluacionDao	0	Ninguno	Baja	Alta
GrupoDao	0	Ninguno	Baja	Alta
GrupoProfesorDao	0	Ninguno	Baja	Alta
ProfesorAsignaturaDao	0	Ninguno	Baja	Alta
ProfesorDao	0	Ninguno	Baja	Alta
RegistroDao	0	Ninguno	Baja	Alta
Asignatura	0	Ninguno	Baja	Alta
Grupo	0	Ninguno	Baja	Alta
Profesor	0	Ninguno	Baja	Alta
GenericServiceInterface	0	Ninguno	Baja	Alta
Ldap	0	Ninguno	Baja	Alta
PasarelaAutenticacion Util	0	Ninguno	Baja	Alta
FacesUtil	0	Ninguno	Baja	Alta

La figura 12 muestra la representación de los resultados obtenidos agrupados en los intervalos definidos. En la figura 13 se observa la representación en porciento de los resultados obtenidos en el instrumento, agrupados en los intervalos definidos.

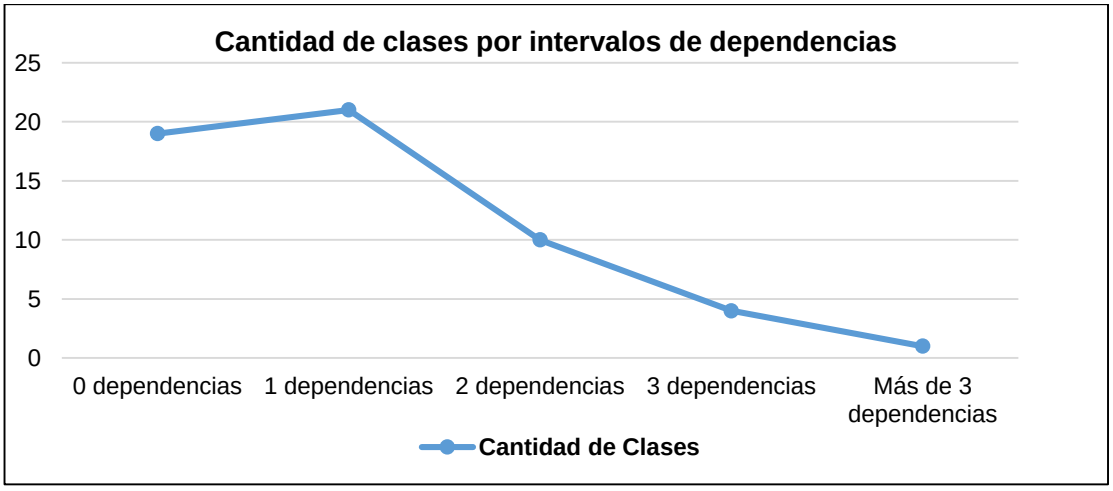


Figura 12 Representación de las clases según la cantidad de relaciones de uso

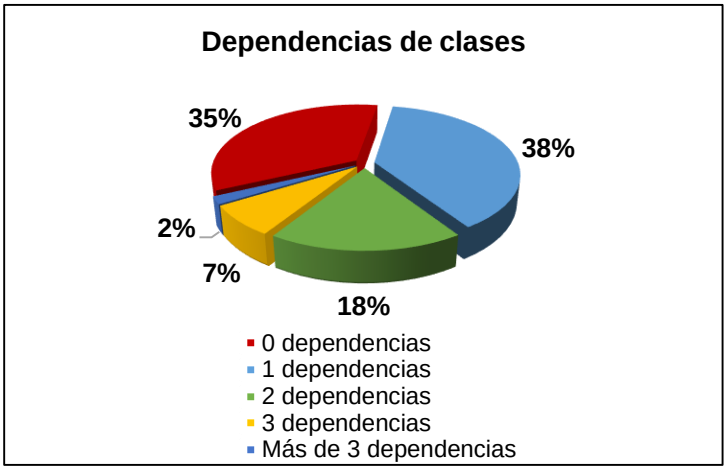


Figura 13 Representación en porciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

Los resultados obtenidos luego de aplicar las métricas RC arrojan que el diseño propuesto tiene una calidad aceptable. Tiene en cuenta que el 73% de las clases poseen una cantidad de relaciones de uso menor o igual al promedio general de 1,05, esto conlleva a que las evaluaciones sean positivas en los atributos de calidad involucrados.

La figura 14 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento. Se evidencia un diseño eficiente al quedar reflejado que un 58% de las clases cuentan con un bajo acoplamiento.

La figura 15 deja ver la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento. Queda demostrada la eficiencia de la arquitectura del sistema al evidenciarse que un 72% de las clases posee una baja complejidad de mantenimiento.

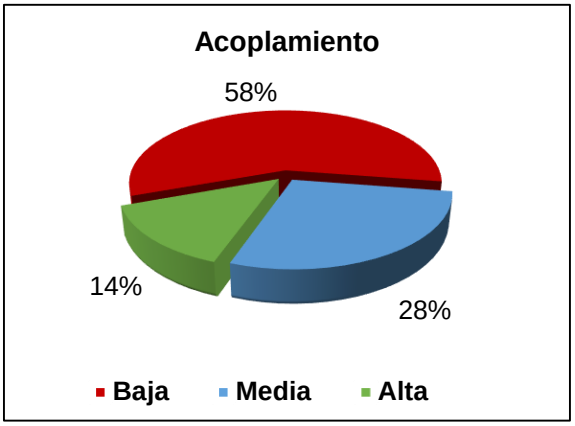


Figura 14 Resultados de la evaluación de la métrica RC para el atributo de calidad Acoplamiento

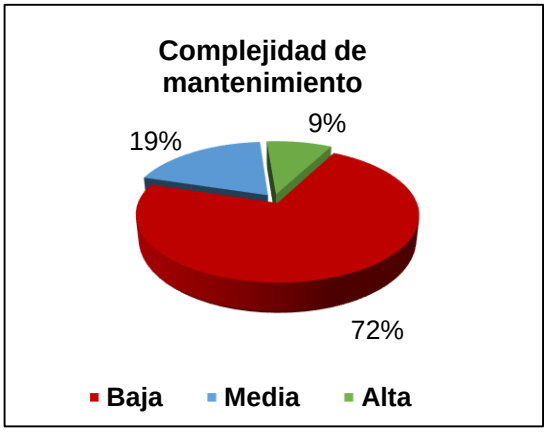


Figura 15 Resultados de la evaluación de la métrica RC para el atributo de calidad Complejidad de mantenimiento

La figura 16 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización. Esto evidencia que el 72% de las clases poseen una alta reutilización lo que es un factor fundamental que debe ser tenido en cuenta en el desarrollo de software.

En la figura 17 se observa la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas. Esto evidencia que un 72% de las clases posee bajas cantidad de pruebas, lo que representa valores favorables para el diseño realizado.

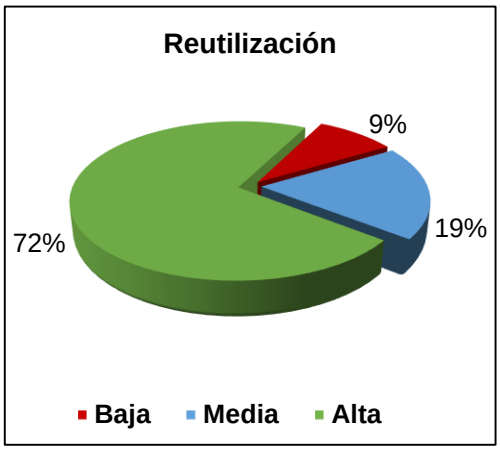


Figura 14 Resultados de la evaluación de la métrica RC para el atributo de calidad Reutilización

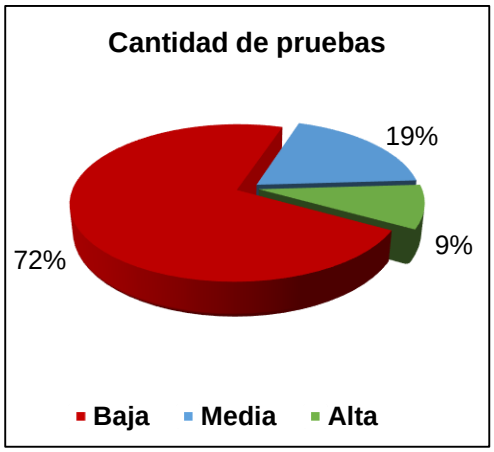


Figura 17 Resultados de la evaluación de la métrica RC para el atributo de calidad Cantidad de pruebas

3.3.3 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Dicha matriz permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica.

Si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guion simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas. A continuación, se muestran los resultados obtenidos (Baryolo, 2010).

Tabla 17 Rango de valores para la evaluación de la relación atributo/métrica

Atributo/ Métrica	TOC	RC	Promedio
Responsabilidad	1	-	1
Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de mantenimiento	-	1	1
Cantidad de pruebas	-	1	1

Tabla 18 Rango de valores para la evaluación de la relación categoría

Categoría	Rango de valores
Regular	>0.4 y <=0.7
Bueno	>0.7
Malo	<= 0.4

La figura 18 muestra la gráfica de los resultados obtenidos de los atributos de calidad evaluados en las métricas aplicadas anteriormente, donde todos los atributos de calidad mantienen un buen comportamiento.

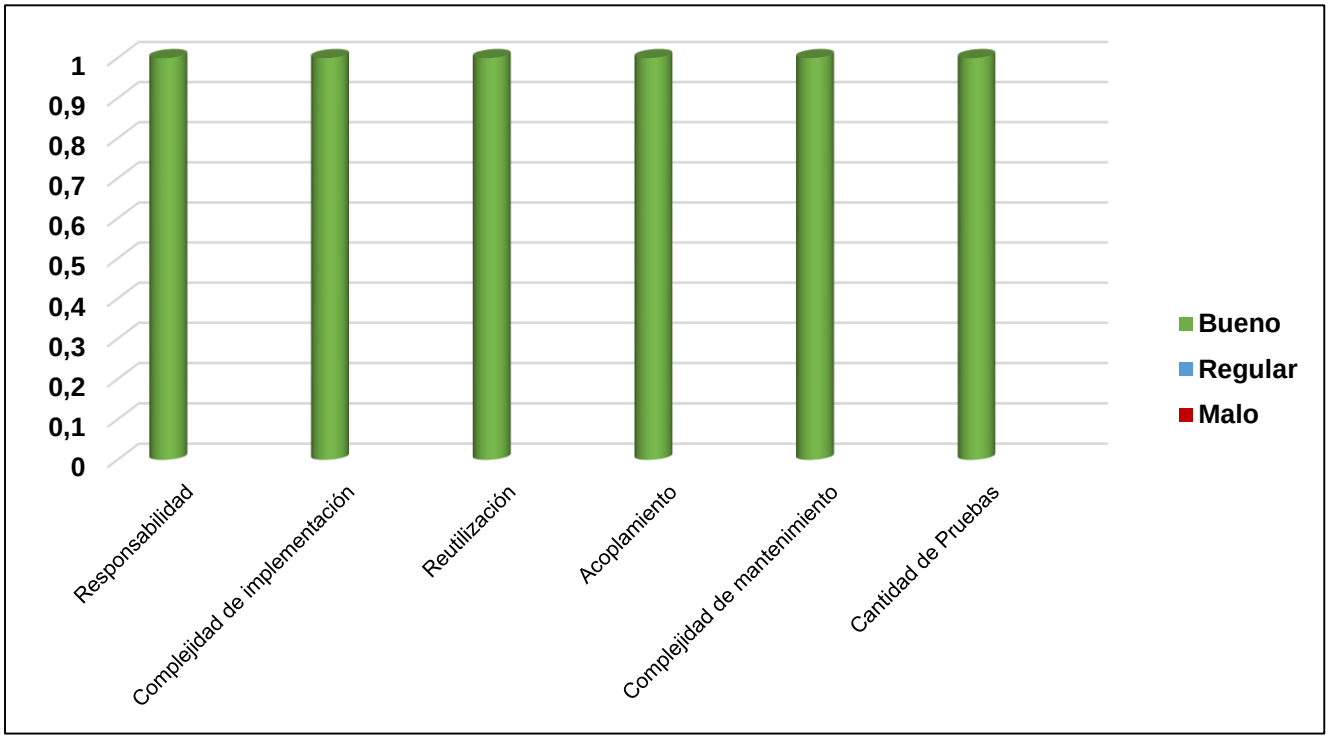


Figura 18 Resultados obtenidos de la evaluación de los atributos de calidad

3.4 Modelo de prueba

El instrumento adecuado para determinar la calidad de un producto de software es el proceso de pruebas. Pressman declara que las pruebas de software son un elemento crucial para garantizar la calidad del producto y permiten validar las especificaciones, el diseño y la programación. Estas tienen como objetivo, además de descubrir errores, medir el grado en que el software cumple con los requerimientos definidos (Pressman, 2002). "*Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente"*. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (Ramírez, 2014).

Niveles de prueba

Para evaluar dinámicamente un sistema de software se definen niveles de prueba que permiten probar desde los componentes más simples y pequeños e ir avanzando hasta probar todo el software en su conjunto. En total se dividen en 5 niveles de pruebas:

- **Pruebas unitarias:** se realizan en la primera fase de las pruebas dinámicas y sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, esté correctamente codificado. En estas pruebas cada módulo será probado por separado y lo hará, generalmente, la persona que lo creó.
- **Pruebas de integración:** son realizadas para probar el software, ensamblando todos los módulos probados previamente a partir del esquema del diseño.
- **Pruebas del sistema:** son efectuadas cuando el software se encuentra ensamblado totalmente e integrado con cualquier componente de hardware, permite comprobar que se cumplen los requisitos funcionales.
- **Pruebas de aceptación:** son aplicadas cuando el producto está listo para implantarse en el entorno del cliente y son realizadas por el usuario en conjunto con las personas del equipo de pruebas, siendo deseable, que sea el mismo usuario quien aporte los casos de prueba. Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las Historias de Usuario. Destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.
- **Pruebas de regresión:** consisten en la repetición selectiva de pruebas para detectar fallos introducidos durante la modificación de un sistema o componente de un sistema. Son efectuadas para comprobar que los cambios no han originado efectos adversos. (Vegas, 2005).

Tipos de pruebas

- **Prueba de seguridad:** consisten en verificar los mecanismos de control de acceso al sistema para evitar alteraciones indebidas en los datos.
- **Prueba de rendimiento:** consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.
- **Prueba de funcionalidad**: pruebas que se realizan fijando su atención en la validación de las funciones, métodos, servicios y casos de uso.
- **Prueba de usabilidad:** pruebas enfocadas a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente de documentación de usuarios y materiales de entrenamiento.
- **Pruebas de soportabilidad:** pruebas enfocadas a asegurar la instalación y funcionamiento en diferentes configuraciones de hardware y software.
- **Prueba de stress:** se proponen encontrar errores debidos a recursos bajos o completitud de recursos. El objetivo de esta prueba es investigar el comportamiento del sistema bajo condiciones que sobrecargan sus recursos. Se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

Métodos y técnicas de prueba

Cualquier producto de ingeniería se puede probar de dos formas: con el método de caja negra y/o método de caja blanca.

Pruebas de **caja blanca**: usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear sus métodos, se puede garantizar que, al menos una vez, se ejecuten todas las rutas independientes del módulo. En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y examinando el estado del programa en varios puntos (Solís, 2011).

- **Camino básico**: una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe. Permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. El empleo de este tipo de pruebas permitirá diseñar casos de prueba que comprueben que todas las sentencias del sistema se ejecuten al menos una vez, además de todas las condiciones, tanto verdaderas como falsas (Pressman, 2002).

- **Prueba de bucles**: técnica de prueba de caja blanca que se centra en la validez de las construcciones de los bucles (Sommerville, 2005).

Pruebas de **caja negra**: se aplican a la interfaz del software y se centran en los requisitos funcionales del sistema; que permiten derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. En la prueba de la caja negra, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta.

- **Partición de equivalencia:** técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Esta se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Se basa en una evaluación de las clases de equivalencia para una condición de entrada.
- **Análisis de valores al límite (AVL):** técnica de diseño de casos de prueba que complementa la partición equivalente. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.
- **Grafos de causa-efecto:** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.
- **Prueba de comparación:** este tipo de pruebas se emplea cuando la fiabilidad del software es crítica, (por ejemplo, cuando se desarrolla para aeronaves o plantas nucleares) varios equipos de ingeniería del software desarrollan versiones independientes de la misma aplicación, usando los mismos requisitos. Todas las versiones son probadas con los mismos datos, para asegurar que todas proporcionan una salida idéntica.
- **Prueba de la tabla ortogonal:** esta prueba puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño, pero demasiado grande para solicitar pruebas exhaustivas. El método de la tabla ortogonal es útil al encontrar errores asociados con fallos localizados (Pressman, 2002).

3.4.1 Estrategia de prueba diseñada

Para verificar el correcto funcionamiento del software y validar la calidad del producto desarrollado, fueron realizadas una serie de pruebas siguiendo la siguiente estrategia:

Tabla 19 Estrategia de prueba diseñada

Niveles de prueba	Tipos de prueba	Métodos de prueba	Técnicas de prueba
Pruebas del sistema	Funcionalidad	Caja negra	Partición de equivalencia
Pruebas de aceptación	Funcionalidad	Caja negra	Partición de equivalencia

Para realizar las pruebas según la estrategia definida, se elaboraron casos de prueba específicos para cada uno de los requisitos funcionales.

Tabla 20 Resumen de pruebas realizadas

No.	Caso de prueba	Resultado esperado
1.	DCP Gestionar estudiantes	El sistema adiciona, actualiza, elimina los datos de un determinado estudiante y lista los almacenados en la BD.
2.	DCP Mostrar información de los estudiantes	El sistema muestra de cada estudiante la cantidad de asistencias, de evaluaciones, aprobado o desaprobado.
3.	DCP Graficar estado de cada estudiante.	El sistema grafica el estado del estudiante especificado.
4.	DCP Gestionar profesor.	El sistema adiciona, actualiza, elimina los datos de un determinado profesor y lista los almacenados en la BD.
5.	DCP Gestionar grupo.	El sistema adiciona, actualiza, elimina los datos de un determinado grupo, y lista los almacenados en la BD.
6.	DCP Mostrar información de un grupo.	El sistema muestra de cada grupo la cantidad de estudiantes, estado de las evaluaciones, aprobado o desaprobado que tiene.
7.	DCP Graficar información de un grupo.	El sistema grafica la información de un grupo.
8.	DCP Asignar grupos a un profesor.	El sistema asigna grupos al profesor especificado.
9.	DCP Gestionar actividad docente.	El sistema adiciona, actualiza, elimina los datos de una determinada actividad docente, y lista las almacenadas en la BD.
10	DCP Gestionar ausencia.	El sistema adiciona, actualiza, elimina ausencias, y lista las almacenados en la BD.
11	DCP Graficar asistencia de un grupo.	El sistema grafica la asistencia de un grupo.
12	DCP Exportar registro de asistencia (excel o pdf).	El sistema exporta registro de asistencia.
13	DCP Gestionar asignatura.	El sistema adiciona, actualiza, elimina los datos de una determinada asignatura, y lista las almacenados en la BD.
14	DCP Gestionar registro de asistencia.	El sistema adiciona, actualiza, elimina asistencia, y lista las almacenados en la BD.
15	DCP Gestionar evaluación.	El sistema adiciona, actualiza, elimina los datos de una determinada evaluación, y lista las almacenadas en la BD.
16	DCP Generar corte evaluativo.	El sistema genera los cortes evaluativos.

17	DCP Graficar corte evaluativo.	El sistema grafica los cortes evaluativos.
18	DCP Mostar listado de corte evaluativo.	El sistema muestra el listado de los cortes evaluativos registrados.
19	DCP Mostrar gráfico sobre el estado de evaluaciones de un grupo.	El sistema muestra el estado de las evaluaciones de un grupo.
20	DCP Obtención desde el LDAP del estudiante o el profesor por el usuario UCI.	El sistema obtiene los datos del estudiante y el profesor mediante el usuario UCI que obtiene del LDAP.
21	DCP Autenticar usuario con LDAP	El usuario se autentica con LDAP.

3.4.2 Resultados de las pruebas del sistema

Para las pruebas del sistema se realizaron tres iteraciones con la estrategia de pruebas propuesta. Como se muestra en la figura 19 las No Conformidades (NC) detectadas en cada iteración fueron resueltas.

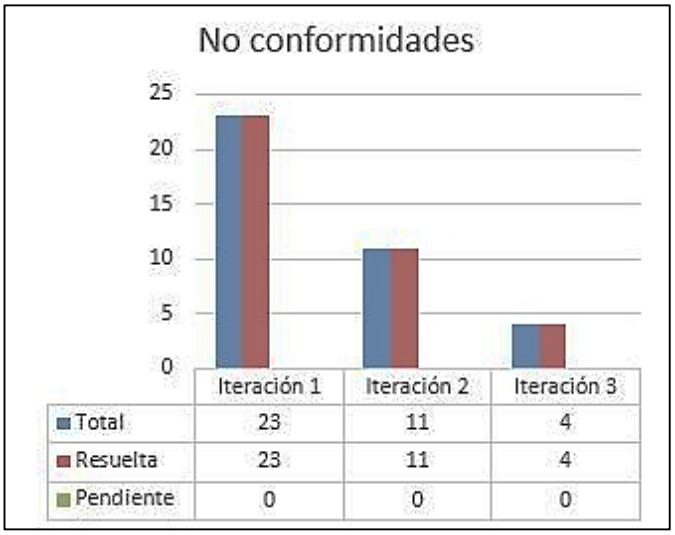
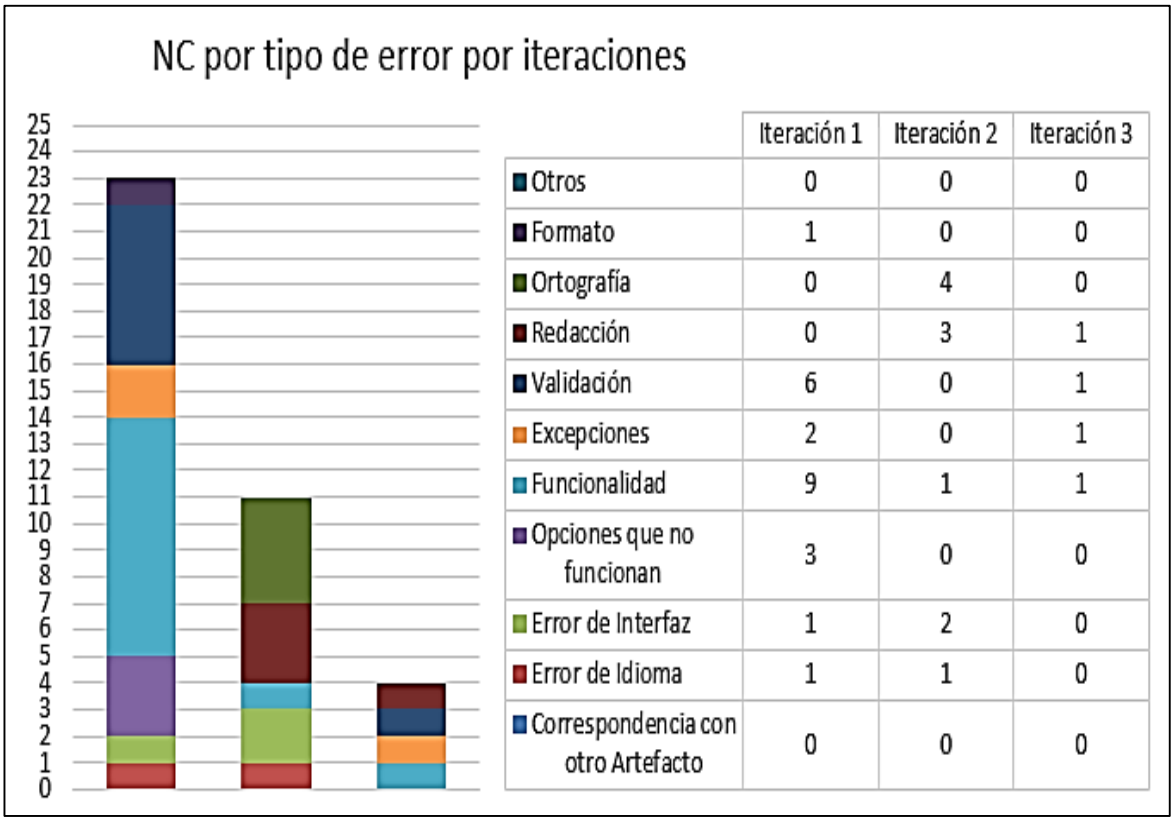


Figura 19 Cantidad de NC detectadas en cada iteración

En la figura 20 se muestran las NC por tipo de error. En la primera iteración las NC que predominaron estuvieron relacionadas con funcionalidades incorrectas y validaciones. En la segunda iteración se probó nuevamente con todos los casos de pruebas diseñados, centrándose más en errores de interfaz, redacción y ortografía, arrojando en estas categorías el mayor número de NC. Mientras que en la tercera iteración las NC estuvieron distribuidas.



	Iteración 1	Iteración 2	Iteración 3
Otros	0	0	0
Formato	1	0	0
Ortografía	0	4	0
Redacción	0	3	1
Validación	6	0	1
Excepciones	2	0	1
Funcionalidad	9	1	1
Opciones que no funcionan	3	0	0
Error de Interfaz	1	2	0
Error de Idioma	1	1	0
Correspondencia con otro Artefacto	0	0	0

Figura 20 Cantidad de NC por tipo de error por iteración

Al finalizar la última iteración se puede concluir que el software es completamente funcional pues brinda solución a todos los requisitos planteados. Se demuestra la solidez de la implementación y la valoración positiva de la calidad del producto desarrollado.

3.4.3 Resultados de las pruebas de aceptación

En las pruebas de aceptación se realizaron dos iteraciones de prueba. Todas las NC detectadas fueron resueltas de manera que el cliente corrobora que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales fue construido.

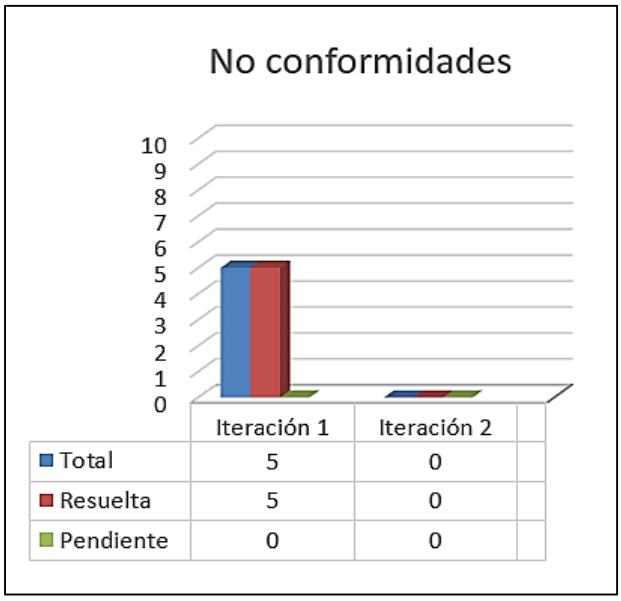


Figura 21 Cantidad de NC detectadas en cada iteración

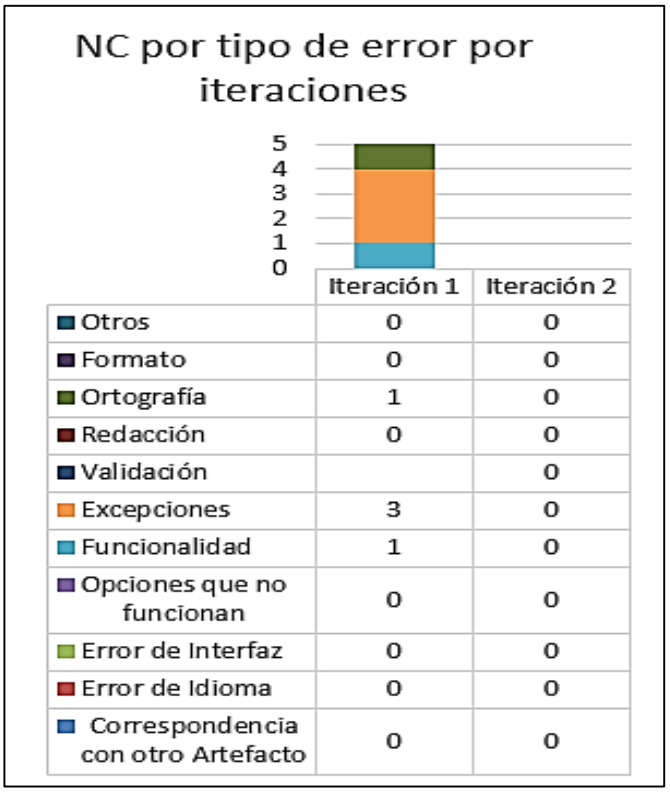


Figura 22 Cantidad de NC por tipo de error por iteración

3.5 Validación de requisitos

Existen varias **técnicas para validar los requisitos,** las cuales se aplican con el objetivo de examinar las especificaciones para asegurar que todos los requisitos han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones y que los errores detectados hayan sido corregidos. Algunas de estas técnicas son:

- **Revisiones:** los requisitos son analizados sistemáticamente por un equipo de revisores, en busca de anomalías y/u omisiones.
- **Prototipos:** se muestra un modelo ejecutable del sistema a los usuarios finales y los clientes, para que puedan ver si dicho modelo cumple con sus necesidades reales.
- **Generación de casos de prueba:** la elaboración de casos de prueba puede revelar los problemas con que puede contar un requisito y es parte fundamental de la programación externa. (Sommerville, 2005).

Para validar que todos los requisitos fueron establecidos con las condiciones anteriormente planteadas, se empleó la técnica de generación de casos de pruebas. Este proceso fue realizado de manera simultánea junto al proceso de pruebas del sistema y de aceptación.

3.6 Consideraciones parciales del capítulo

Una vez realizada la implementación y las pruebas del sistema, se comprobó que el sistema es funcional, y dio solución a todos los requisitos planteados. A partir de la metodología AUP-UCI, se diseñaron los casos de prueba que permitieron evaluar el funcionamiento del sistema desarrollado. Se demostró la solidez de la implementación del sistema mediante los resultados de las pruebas de software.

CONCLUSIONES GENERALES

Una vez finalizada la implementación de GECMA v2.0, se concluye que:

- ✓ El estudio de las soluciones similares en el ámbito nacional e internacional permitió comprender la estructura de estos sistemas y sus principales características, lo que hizo posible determinar la creación de una nueva versión de GECMA e incorporar funcionalidades que no fueron contempladas inicialmente por el cliente.
- ✓ La solución desarrollada permitió gestionar los datos de los profesores, grupos, estudiantes, asignaturas, actividad docente y brindar una herramienta para análisis estadístico de las evaluaciones, lo cual contribuyó a administrar adecuadamente los procesos de gestión del control docente en el proceso de enseñanza-aprendizaje de la Matemática Discreta.
- ✓ Las pruebas efectuadas al sistema erradicaron las deficiencias encontradas, contribuyendo a aumentar la calidad del producto desarrollado y constituyen la certificación de que el software soporta las funcionalidades requeridas.

RECOMENDACIONES

Al finalizar la presente investigación se recomienda:

- ✓ Elaborar el Manual de usuarios para un mejor manejo y comprensión del sistema por el cliente final.
- ✓ Poner el sistema a disposición de otras entidades que necesiten brindar el mismo servicio, colaborando de esta manera con la informatización de la sociedad cubana.

REFERENCIAS BIBLIOGRÁFICAS

1. AUTORES, C. D. Informe de la asignatura MD1 curso 2014-2015. La Habana: 2015.
2. BARYOLO, O. G. Solución Informática de autorización en entornos multientidad y multisistema. Tesis de Maestría en Informática Aplicada Universidad de las Ciencias Informáticas, 2010.
3. BESTEIRO, M. R. Y. M. Desarrollo de aplicaciones .NET con Visual C#. Edtion ed. Madrid: McGraw-Hill, 2002. 620 p. ISBN 84-481-3277-7. cap Web Services. [en línea] Disponible en:<http://www.ehu.eus/mrodriguez/archivos/csharppdf/ServiciosWeb/WebServices.pdf>.
4. BEATÓN, M. O. L. Y. Y. A. Análisis y Diseño de los Procesos de Gestión de Personal para Akademos v2.0. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2009.
5. CARVAJAL, D. A. S. Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnología de Gestión de Datos V2.0 (DRI MANAGEMENT SYSTEM V2.0). Trabajo de Diploma Universidad de las Ciencias Informáticas, 2013.
6. CERAMI, E. *Web Services Essentials*. Edtion ed.: O'Reilly, 2002. ISBN 0-596-00224-6.
7. ÇIVICI, Ç. Primefaces User Guide 5.3. 2016.Disponible en:< http://www.primefaces.org/docs/guide/primefaces_user_guide_5_3.pdf>
8. COMMITTEE, P. S. *A Guide to Project Management Body of Knowlegde*. Edtion ed. North Carolina, 1996. ISBN 1-880410-13-3.
9. CORDERO, E. F. Y. J. L. Metodologías ágiles Proceso Unificado Ágil (AUP). 2009.Disponible en:<ingenieriadesoftware.mex.tl/images/.../METODOLOGÍAS%20ÁGILES%20(AUP).ppt>.
10. CRUZ, A. R. Pruebas de caja negra. 2011. Disponible en:<http://www.buenastareas.com/ensayos/Pruebas-De-Caja-Negra/1477804.html>.
11. CRUZ, Z. C. Á. Y. R. C. D. L. Versión 2.0 del módulo Resultados de la colección “La Caja Mágica”. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2013.
12. CHILE, U. D. Sistema de Información de Gestión Académica 2015, [citado 20 de julio 2015]. Disponible en:<http://alpes.stg.uchile.cl/~srvcen01/censo/index_ie.html>.
13. DEITEL, H. M. D., P. J *Cómo programar en JAVA*. Edtion ed.: Pearson Education 2008. ISBN 978-970-26-1190-5.
14. DEITEL, P. J. D. Y. H. M. *Cómo programar en C/C++ y Java*. Edtion ed.: Pearson Education, 2004. ISBN 9702605318.
15. DURÁN, N. O. Universidad de Excelencia: 12 años de historia. 2014. [en línea] Disponible en:<http://www.uci.cu/universidad-de-excelencia-12-anos-de-historia-0>.

16. ESPINOSA, L. P. Diseño de una aplicación para la gestión de información de investigación y postgrado de la Facultad 4. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2012.
17. FUNDATION, T. A. S. Apache Tomcat. 2016 [en línea] [citado 5 de marzo 2016] Disponible en :< http://tomcat.apache.org/tomcat-8.0-doc/index.html >.
18. GEARY, C. H. Y. D. M. *Core JavaServer Faces 3ed. Boston* Edtion ed. Boston: Prentice Hall, 2010. ISBN 978-0-13-701289-3.
19. GINESTA, O. P. M. Y. M. G. Bases de datos en PostgreSQL. Edtion ed.: Universidad Oberta de Catalunya, 2005. ISBN 84-9788-269-5. Disponible en: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02152.pdf
20. GIMENO, J. L. G. Y. J. M. Introdución a Netbeans. 2011. [en línea] [citado 5 de febrero 2016].Disponible en:<http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/1-introduccioi81n-a-netbeans.pdf>.
21. GÓNGORA, L. R. R. Y. Y. L. Sistema para la evaluación de la Seguridad de un Software. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2013.
22. GONZÁLEZ, A. E. L. Y. G. F. Desarrollo de una herramienta para la evaluación de la seguridad. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2013.
23. GONZÁLEZ, H. S. Manual Hibernate. 2003. [citado 5 de marzo 2016] Disponible en:<http://static1.1.sqspcdn.com/static/f/923743/14427535/1317484934257/ManualHibernate.pdf?token=GAwxh16SQEWOlkFROTd3vZ%2Fxt7A%3D>.
24. GRANDE, F. O. *Análisis, diseño e implantación de un sistema de servidores departamental basado en máquinas virtuales*. Edtion ed. Madrid: Leganés, 2007.
25. GUERRERO, J. M. S. Y. C. A. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. In Información tecnológica 2013, vol. vol. 24, nº 3.
26. HAMILTON, R. M. Y. K. Learning UML 2.0. O'Reilly, 2006. [en línea] Disponible en: <http://portal.unimap.edu.my:7778/portal/page/portal30/Lecturer%20Notes/KEJURUTERAAN_KOMPUTER/Semester%202%20Sidang%20Akademik%2020112012/EKT420%20Software%20Engineering/E-Books/UML/Learning%20UML%202.0%20%5B2006%5D.pdf>.
27. HERNÁNDEZ, J. B. L. Y. A. R. Revisión, Verificacción y Validación en un proceso de desarrollo de software enero-abril /2011 [Type of Work]. 2011, vol. Vol. XXXII/No. 1, no. ISSN:1815-5936, pp. p. 28-36. Disponible en:<http://rii.cujae.edu.cu/index.php/revistaind/article/view/342>.
28. HUMANOS. Servicio de Directorio con LDAP. Introducción. 2014. Disponible en:<https://humanos.uci.cu/2014/01/servicio-de-directorio-con-ldap-introduccion/>.

29. IEEE. Software Engineering Standars Commitee of the IEEE Computer Society. In.: The Institute of Electrical and Electronics Engineers 2000.
30. ISW, C. D. P. 2011. Conferencia #1: Continuación de la Disciplina Análisis y Diseño. En Ingenieria de software II. La Habana : Departamento de ISW. Universidad de las Ciencias Informáticas.
31. ISW, Colectivo. 2011. Arquitectura y Patrones de Diseño. Tema I: Culminación de la Fase de Elaboración. La Habana. Universidad de las Ciencias Informáticas.
32. LANTIGUA, G. F. G. Y. A. E. Desarrollo de una herramienta para la evaluación de la seguridad. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2013.
33. LARMAN, C. UML y Patrones. Edtion ed. España: Pearson Prentice Hall, 2003. ISBN 84-205-3438-2.
34. ORACLE. Introduction to Java Platform. 2013. [en línea]Disponible en:<https://www.oracle.com/java/technologies/java-ee.html>.
35. ORTIZ, R. A. La nueva universidad cubana. [en línea] Revista Pedagogía Universitaria. Editorial Universitaria 2008, vol. XIII No. 2 p. 1-24.
36. OSCAR, S. Visual Paradigm for Uml. International Book Market Service Limited, 2013.Disponible en:<https://images.visual-paradigm.com/docs/vp_user_guide/VP_Users_Guide.pdf>.
37. PRESSMAN, R. S. Ingeniería del Software, Un Enfoque Práctico. Edtion ed. España: McGraw-Hil, 2002. ISBN 8448132149.
38. RAMÍREZ, A. S. Procedimiento para la realización de ensayos de Aceptación y/o Piloto. 2014. [en línea] Disponible en:<http://www.laccei.org/LACCEI2014-Guayaquil/RefereedPapers/RP049.pdf>.
39. RICA, U. D. C. Sistema de Estudios de Postgrado. 2012. [en línea] Disponible en:<http://www.sep.ucr.ac.cr/>.
40. RIVERO, Y. G. GECMA: Aplicación para la gestión y apoyo al control del Proceso de Enseñanza Aprendizaje de la Matemática Discreta 1. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2015.
41. RUMBAUGH, I. J. G. B. Y. J. El proceso unificado de desarrollo de software. Edtion ed. Madrid: Pearson Education, 2000. ISBN 84-7829-036-2.
42. SÁNCHEZ, T. R. Metodología de desarrollo para la Actividad productiva de la UCI. 2015. Disponible en :<http://mejoras.prod.uci.cu>.
43. SOLÍS, C. A. M. Verificación y validación de software. [en línea] In.: Infosys, Universidad TecMilenio - Centro de Estudios en Línea, 2011.
44. SOMMERVILLE, I. Ingenería de Software Edtion ed. Madrid: Pearson Education, 2005. ISBN 84-7829-074-5.

45. SUAZA, K. V. Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software. In.: Universidad Nacional de Colombia 2013.
46. VEGAS, N. J. A. M. M. Y. S. Técnicas de evaluación de software. 2005. [citado 5 de marzo 2015]. Disponible en:<http://eva.uci.cu/file.php/257/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_3/Comun/Tecnicas_de_evaluacion_de_software_Jurisco-Moreno.pdf.>. [en línea]
47. VELÁZQUEZ, G. B. M. Y. O. J. A. Módulo Complejo Residencial v2.0 para el Sistema de Gestión. Trabajo de Diploma Universidad de las Ciencias Informáticas, 2013.
48. WALLS, C. Spring in Action. Edtion ed. Estados Unidos: Manning Publications Co., 2008. ISBN 9781935182351.
49. WEITZENFELD, A. Ingeniería de software orientada a objetos con UML. Edtion ed.: Thomson, 2005. ISBN 9789706861900.
50. Servicios de directorio, LDAP. Disponible en:< www.profesordeinformatica.com/descargas/capitulo6-ldap.pdf>.
51. Patrón de Diseño DAO. 2011. Disponible en:<http://www.genbetadev.com/java-j2ee/spring-framework-el-patrn-dao>.
52. PrimeFaces User Guide 5.3. 2015. Disponible en:<https://speakerdeck.com/player/057b8d7962d2466f9f24cd45f559b8da?slide=432>.

A

Aplicación web: son aquellas aplicaciones informáticas que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una red interna mediante un navegador web.

Artefacto: término general para cualquier tipo de información creada, producida, o utilizada por los trabajadores en el desarrollo del sistema.

B

Base de datos: un sistema informático a modo de almacén. En este almacén se guardan grandes volúmenes de información. Permite automatizar el acceso a la información y poder acceder a esta de forma rápida y fácil, además de poder efectuar cambios de manera más eficiente.

C

CASE: programas que se utilizan para ayudar a las actividades del proceso de software.

CamelCase: Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello. Se podría traducir como Mayúsculas/Minúsculas Camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula y tiene su origen en la disposición de los tipos móviles en casilleros o cajas.

Clase: molde para describir un objeto que agrupa características (atributos) y comportamientos (métodos).

F

Framework: es una estructura conceptual y tecnológica de soporte definido, generalmente, con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Los frameworks son herramientas de desarrollo que facilitan infraestructuras que ahorran horas de trabajo.

G

GECMA: Gestión de la enseñanza y el conocimiento de la Matemática.

H

HTML: *Hypertext Markup Language*. Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986. Es un lenguaje de marcas. Los lenguajes de marcas

no son equivalentes a los lenguajes de programación, aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que se pueden controlar desde cualquier editor ASCII.

HTTP: *HyperText Transfer Protocol.* Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas Web.

I

IEEE: Por sus siglas en inglés *Institute of Electrical and Electronics Engineers*, en español Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

L

Lenguaje de Consulta Estructurado: lenguaje de base de datos normalizado, especializado en la descripción y la utilización de BD relacionales.

Sigla: SQL.

Lenguaje de programación: es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

P

Paquete del diseño: colección de clases, relaciones, realizaciones de casos de uso y diagramas que están de alguna forma relacionados.

S

Software: Programas de sistema, utilerías o aplicaciones expresadas en un lenguaje de máquina.

Subsistemas: forma de organizar los artefactos del modelo de diseño en piezas más manejables. Puede contener clases del diseño, realizaciones de casos de uso, interfaces y otros subsistemas. Pueden proporcionar interfaces que representan la funcionalidad que exportan en términos de operaciones. Tiene una semántica, que lo hace algo más que una simple agrupación de elementos, el subsistema tiene “personalidad”, responsabilidades (interfaces) bien definidas, provee servicios que son empleados por otros subsistemas o sistemas en sí.

U

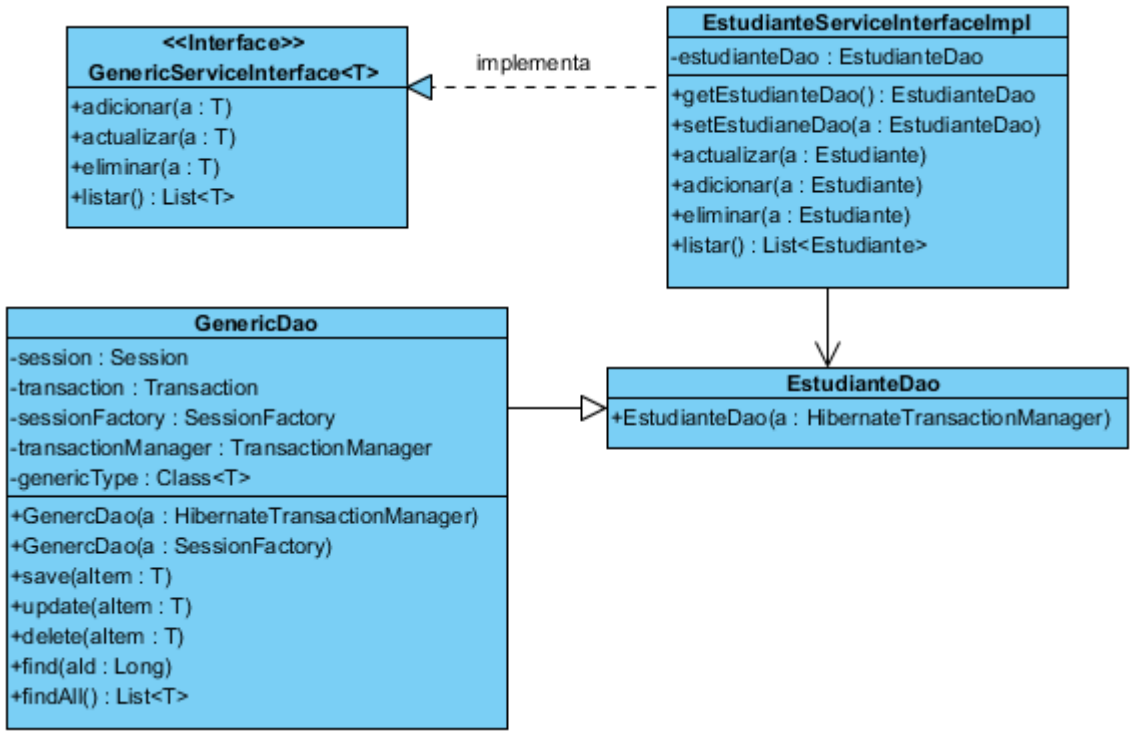
URL: Localizador de recursos uniforme (*URL*, del inglés *Uniform Resource Locator*).

X

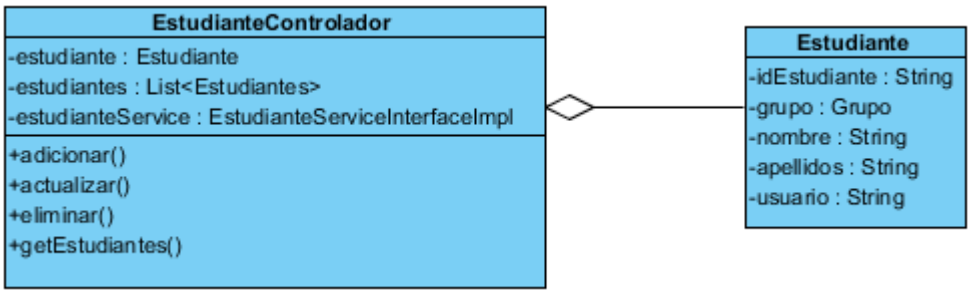
XHTML: Por sus siglas en inglés *eXtensible HyperText Markup Language*. XHTML es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico.

ANEXOS

Anexo 1 Ejemplo de patrón DAO



Anexo 2 Ejemplo de patrón Creador



Anexo 3 Ejemplo de patrón Bajo acoplamiento

Anexo 4 Criterios de evaluación de la métrica TOC

Atributo de calidad	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Anexo 5 Criterios de evaluación de la métrica RC

Atributo de calidad	Categoría	Criterio
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de implementación Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Anexo 6 Cuestionario aplicado a profesores que imparten Matemática Discreta

1. ¿Conoce usted la aplicación web GECMA versión 1?

   _____ Sí _____ No

2. ¿Ha trabajado con ella?

   _____ Sí _____ No

3. ¿Considera que satisfizo todas sus expectativas?

   _____ Sí _____ No

4. ¿Qué dificultades encontró al probar sus módulos de trabajo? Explique brevemente.

5. ¿Le gustaría que se desarrollara una nueva versión?

   _____ Sí _____ No

6. ¿Qué sugiere incluir en la nueva versión? Explique brevemente.