Universidad de las Ciencias Informáticas

Facultad 1

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

Implementación del módulo Personal y Secretaría para el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria.

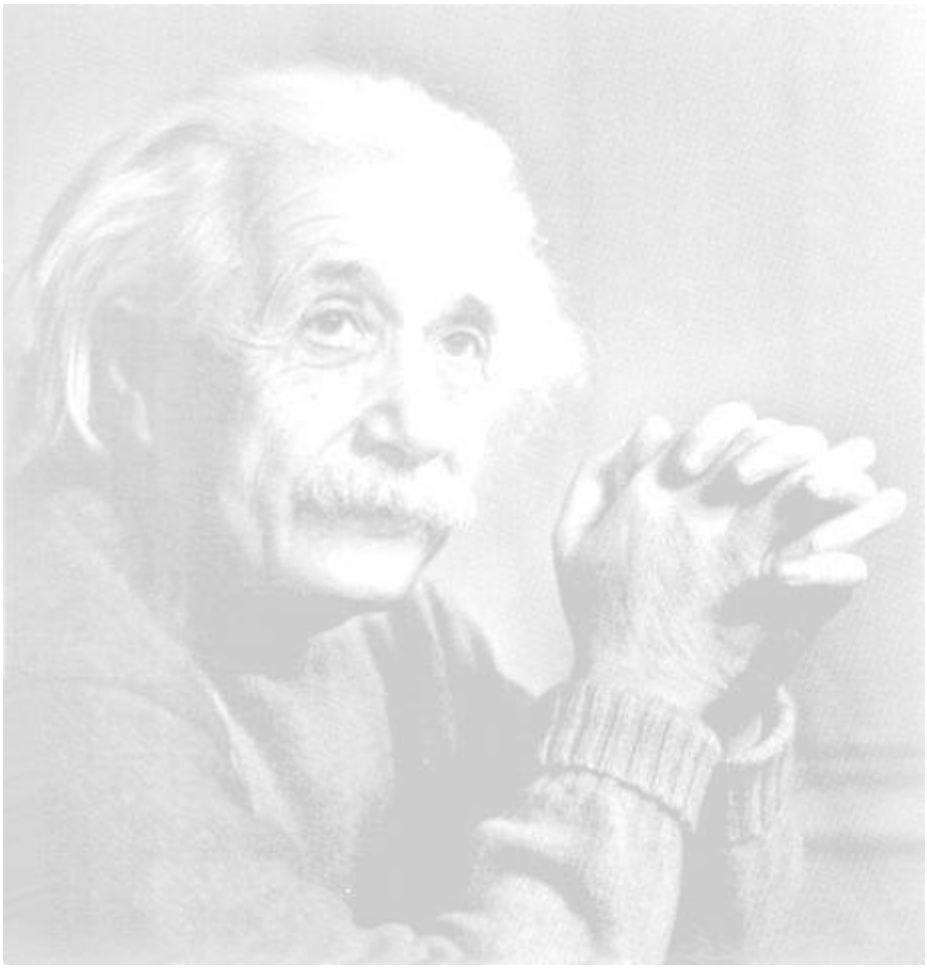
AUTORES: Eddy Felix González Pupo
Marcel Salgado Valdivia

TUTORES: Ing. Reiseer Mejías Ayala
Ing. Renier Jorge Téllez
Ing. Jorge Tamarit Cutiño

CO-TUTORES: Ing. Lianet Pineda de la Nuez
Ing. Lianet Liben Martinez

Ciudad de La Habana, 2010

"Año 52 de la Revolución"



"Cien veces todos los días me recuerdo a mí mismo que mi vida interior y exterior, depende de los trabajos de otros hombres, vivos y muertos, y que yo debo esforzarme a fin de dar en la misma medida en que he recibido."

Albert Einstein

Agradecimientos

Agradecemos a todas aquellas personas que nos han ayudado.

Dedicatoria

A la Revolución, nuestras familias, novias y al futbol.

Resumen

Hoy en día, con el creciente auge de la informatización en todas las esferas de la sociedad y el acelerado desarrollo de las Tecnologías de la Informática y las Comunicaciones, la mayoría de las empresas e instituciones necesitan informatizar las tareas que realizan. En la actualidad en la Universidad de las Ciencias Informáticas (UCI) no se cuenta con un sistema que responda en su totalidad a las exigencias que plantea la gestión universitaria, motivo por el cual es ineludible el desarrollo de una aplicación que agrupe y automatice todos los procesos que ocurren dentro de este centro de estudios de una manera coherente. Por tanto, el objetivo principal del presente trabajo se centra en la implementación del módulo Personal y Secretaría para el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria. El documento recoge un estudio sobre otros sistemas de gestión académica en el ámbito nacional e internacional, además se detallan las tecnologías, lenguajes y herramientas utilizadas, la descripción de la propuesta de solución, clases y algoritmos no triviales implementados y de la validación de la solución mediante las pruebas de aceptación.

Palabras Clave: gestión universitaria, gestión de personal, implementación

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Desde que el hombre primitivo se convirtió en *"homo sapiens"*, se inclinó por las estadísticas y las expresó en la forma de artes gráficas, creando una incipiente modalidad de cálculo. Guiado por la necesidad de contabilizar y organizar sus recursos, así como realizar la mayor cantidad de operaciones en el menor tiempo posible y automatizar tareas para humanizar el trabajo surge la computación, resultado evolutivo de ideas y realizaciones de muchas personas.

Este año se cumplen 52 años de la entrada de la primera computadora electrónica en Cuba y 40 años de la fabricación por primera vez de una computadora en el país, siendo de los primeros países de América en utilizar la computación para automatizar tareas.

En esta etapa de la evolución humana, donde los cambios sociales, económicos y tecnológicos se suceden a velocidades vertiginosas, nuestro país se da a la tarea de lograr la informatización de la sociedad. Nuestra economía ha tenido que adaptarse a la fluctuante economía mundial y hoy se propone convertir la industria de bienes y servicios, encabezada por la industria del software, que mueve anualmente miles de millones de dólares, en una de las principales fuentes de ingresos para el país.

Nuestro Comandante en Jefe uno de los principales estadistas del siglo XX, fue el creador intelectual de la UCI, como parque tecnológico y universidad de excelencia.

Desde sus inicios, la UCI se dio a la tarea de la informatización de los procesos que ocurren dentro de la universidad, siendo los relacionados con la gestión académica unos de los más importantes. Para ello fue desarrollado el sistema Akademos, aportando una serie de facilidades al personal docente y estudiantil a través de varios servicios que brinda de manera dinámica.

El actual sistema Akademos está desarrollado completamente en software propietario, por lo que no se adecua a las políticas de la universidad. Posee muy poca documentación, lo que dificulta su mantenimiento, soporte y reutilización. En cuanto a la gestión de los procesos de personal, no permite más de un plan de estudio. Cuando un estudiante se encuentra de licencia el sistema no permite el cambio de cohorte estudiantil ni la asignación de la nueva versión del plan de estudio adecuado a las asignaturas cursadas. El cambio de nivel solo puede realizarse después de finalizados los mundiales, de lo contrario se pierde la información estadística de los estudiantes que no llevaban cambio de nivel. No se gestionan los procesos de solicitud de bajas, traslados y licencia, ni su documentación asociada. No se gestionan los procesos de ratificación de matrícula, ni los procesos asociados a la gestión de los alumnos ayudantes. La gestión de algunos de sus procesos se lleva a cabo en sistemas diferentes como: Gestión de Traslados y Asset (sistema para la gestión de la información de los profesores).

En la actualidad el sistema no responde a las exigencias del complejo proceso de gestión académica de la universidad, por lo que surge la necesidad de llevar a cabo la informatización de estos procesos de manera diferente y se pretende crear una nueva versión del mismo donde se solucionen las limitaciones del anterior y se alcance una mayor eficiencia en los numerosos procesos involucrados.

En nuestra universidad la docencia de pregrado, las investigaciones y la extensión universitaria conjuntamente con la producción, la superación posgraduada, la residencia, los laboratorios, la biblioteca, la plataforma virtual de aprendizaje y la cooperación internacional desempeñan un papel primordial en la formación de los profesionales, todos estos procesos influyen directamente en el ingreso y la ubicación laboral de estos por lo que no se pueden ver como elementos aislados dentro de la gestión universitaria.

Debido a que estos procesos son vitales en el desarrollo de la universidad y que unos no pueden existir sin la presencia de otros es necesario que todos converjan en una misma solución, motivo por el cual es ineludible el desarrollo de un sistema de gestión universitaria que los agrupe de una manera coherente. Por lo que se propone la realización del Sistema de Gestión Universitaria como solución integral para la gestión de los procesos sustantivos de la universidad, que incluye la realización de varias líneas de desarrollo agrupadas en Pregrado, Posgrado, Producción, Investigación, Ingreso, Ubicación Laboral, Laboratorio, Residencia, Extensión Universitaria, Cooperación Internacional, Biblioteca y Teleformación. Todas estas líneas de desarrollo tienen en común una serie de procesos de gestión entre ellos los de gestión de personal.

Pregrado es la línea encargada del desarrollo de todos los procesos de gestión universitaria relacionados con la formación, tanto de los estudiantes del curso regular diurno como del curso para los trabajadores.

El presente trabajo abordará la implementación del módulo Personal y Secretaría para el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria.

Por consiguiente, el trabajo de diploma se propone resolver el siguiente **problema científico**: ¿Cómo construir el módulo Personal y Secretaría para el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria a partir del análisis y diseño existente?

El **objeto de estudio** está enmarcado en: los procesos de gestión de personal en los sistemas de gestión académica y el **campo de acción** en: los procesos de gestión de personal en el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria de la UCI.

Como solución al problema planteado anteriormente se define el siguiente **objetivo general**: implementar el módulo Personal y Secretaría para el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria.

Con la finalidad de dar cumplimiento al objetivo general trazado se proponen los siguientes **objetivos específicos**:

- Evaluar el estado del arte de las soluciones de software para la gestión académica en el ámbito nacional e internacional, así como de las tecnologías y herramientas utilizadas.
- Analizar el diseño propuesto por los analistas del proyecto.
- Realizar la implementación del módulo Personal y Secretaría.
- Validar la propuesta de solución.

Para el cumplimiento organizado y bien distribuido de los objetivos se establecen las siguientes **tareas investigativas**:

- Análisis del estado del arte de las soluciones de software para la gestión académica, tanto en el ámbito nacional como internacional.
- Estudio crítico y valorativo del lenguaje de programación PHP, del marco de trabajo CodeIgniter, del gestor de base de datos PostgreSQL y de las tecnologías a utilizar en la realización de la propuesta de solución.
- Estudio crítico del análisis propuesto por los analistas.
- Valoración del análisis propuesto por los analistas.
- Descripción de los algoritmos no triviales a implementar.
- Implementación de los procesos fundamentales definidos para la gestión de personal dentro del subsistema Gestión de Pregrado.
    - Procesos de prematrícula
    - Procesos de matrícula
    - Procesos de ratificación de matrícula
    - Procesos de contratación de profesores
- Descripción de las pruebas unitarias teniendo en cuenta:
    - Objetivo de la prueba
    - Alcance
    - Tipo de prueba y detalles de la misma.
- Ejecución de las pruebas de unidad.
- Evaluación de los resultados obtenidos con la ejecución de las pruebas unitarias.

Con el propósito de guiar la investigación se plantea la siguiente **hipótesis**: si se implementa el módulo Personal y Secretaría para el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria se posibilitará resolver las deficiencias de la antigua versión concernientes a la gestión de los procesos de personal.

Posibles resultados: módulo Personal y Secretaría para el subsistema Gestión de Pregrado del Sistema de Gestión Universitaria funcional con su documentación asociada. Llámese documentación asociada a la descripción detallada de las clases utilizadas, así como las funcionalidades de las mismas.

Con el módulo implementado se obtendría los siguientes **beneficios:**

- El sistema permitirá más de un plan de estudio.
- Integración de sistemas legados como Gestión de Traslados y Asset.
- Gestión de los procesos de ratificación de matrícula y los procesos asociados a la gestión de los alumnos ayudantes.
- La gestión de los movimientos y solicitudes establecidas dentro del subsistema Gestión de Pregrado como bajas, traslados, licencias, cambios de grupo y de facultad.
- Las estadísticas asociadas a la información existente de estudiantes y profesores se podrán realizar en cualquier momento sin que exista perdida de la información.
- Cuando un estudiante se encuentre de licencia el sistema permitirá el cambio de cohorte estudiantil y la asignación de la nueva versión del plan de estudio adecuado a las asignaturas cursadas.

La metodología utilizada como parte insustituible de la investigación científica está fundamentada en la aplicación de los métodos teóricos y empíricos que más se ajustan al objeto de estudio y al cumplimiento de los objetivos trazados. A continuación se presentan los métodos científicos de investigación utilizados:

Métodos teóricos

El método **analítico-sintético** permitió el análisis de los procesos de gestión de personal dentro del subsistema Gestión de Pregrado y comprender específicamente las características más relevantes de éstos, examinando los distintos documentos que están vinculados con este tema.

El método **histórico-lógico** se usó con el objetivo de estudiar todo lo referente a las soluciones de software de gestión académica en el ámbito nacional e internacional, para profundizar en el tema, sus características e importancia. Posibilitó un mejor análisis histórico de los procesos de gestión de personal.

Métodos empíricos

El método **observación** permitió investigar los procesos externamente sin tener que llegar a la esencia de los mismos, lo que ayuda al planteamiento del problema científico, además posibilitó conocer bien el proceso delimitado como objeto de estudio, lo cual ayuda a tener un conocimiento más detallado de lo que se quiere, lo que hace falta hacer y cómo hay que hacerlo.

El método **análisis de documentos** posibilitó realizar un estudio de los modelos y documentos manejados por el grupo de gestión de personal y ayudó a la determinación del estado del arte del objeto de investigación.

Fundamentación teórica

1.1 Introducción

En el capítulo se muestra el resultado de la investigación bibliográfica sobre los sistemas de gestión académica en el ámbito nacional e internacional, además se detallan las tecnologías, lenguajes y herramientas utilizadas para la implementación del módulo Personal y Secretaría, analizándose desde un punto de vista crítico sus características, ventajas y desventajas.

1.2 Los sistemas de gestión académica

Realizar los procesos docentes de forma manual puede resultar una tarea compleja, debido al gran volumen de información involucrado, las posibilidades de errores humanos, tardanzas en la realización de estos y pérdidas de materiales se maximizan. Con la incorporación de las nuevas tecnologías de la información se han automatizado estos procesos mediante el empleo de sistemas informáticos para la gestión académica, los cuales constituyen una poderosa herramienta de trabajo que permite hacer más eficientes los procesos docentes y minimizar los riesgos de su realización manual.

"Un sistema de gestión[1] es una estructura probada para la gestión y mejora continua de las políticas, procedimientos y procesos de la organización" [1]. Entre los más usados se encuentran:

- Sistemas de gestión académica.
- Sistemas de gestión de información.
- Sistemas de gestión documental.
- Sistemas de gestión de recursos humanos.
- Sistemas de gestión de proyectos.
- Sistemas de gestión integrales.

1.3 Los sistemas de gestión académica en el mundo

Ágora

Desarrollado por la empresa española Kherian Soft, es un producto de software estándar de gran calidad para la gestión de Centros Docentes y Academias de todo tipo. Es un sistema completo, altamente configurable y flexible pues gestiona sin limitaciones los datos de alumnos, profesores,

[1] Gestión: Del latín *gestio*, entre sus acepciones, acción de llevar a cabo, administración o dirección de un asunto, empresa o negocio, organización, utilización y aprovechamiento de los recursos.

asignaturas, clases, aulas, grupos, matrículas y clientes. Brinda diferentes opciones de presentación y estética, se adapta fácilmente a las necesidades de sus usuarios y se adecua a cualquier tipo de centro o formación, ya sea esta de tipo oficial o de carácter libre (academias de enseñanza general, de idiomas, informática o música), ya se trate de un pequeño centro o una gran empresa con una gestión centralizada de varias sucursales distantes geográficamente.

Ágora aporta soluciones en diversos sentidos, como son:

- La gestión centralizada de datos a través de la cual se gestionan todas las bases de datos de alumnos, profesores, aulas y clases; la generación automática de la documentación que permite emitir todo tipo de reportes, como recibos y facturas, además de proporcionar listados de soporte de todo tipo.
- Gestiona con gran facilidad las excepciones que trastocan la planificación: alumnos que sólo pueden venir a la mitad de las clases proyectadas de su grupo, o que no se adaptan a ningún grupo y toman clases de varios, clases que cambian de horario, profesores que están de baja y han de ser sustituidos, entre otras.
- Todos los datos manejados por la aplicación pueden filtrarse por cualquier criterio. Un sencillo asistente le permitirá construir la selección y guardarla para luego aplicarle el filtro que desee. Los datos pueden igualmente ordenarse por cualquier campo, del mismo modo, localizará inmediatamente un dato buscado con tan sólo introducir sus primeros caracteres.
- Con el control automatizado de docencia real, el programa calculará cada día la docencia que, en función de los horarios y las matrículas, teóricamente se habrá impartido, generando los registros correspondientes, permite al usuario verificarlos periódicamente para ajustar las incidencias. Es especialmente eficaz para calcular los pagos a profesores. Además, calculará de forma automática la disponibilidad horaria de profesores y aulas.
- El control automatizado de asistencia real permite un registro de esta, al cual el usuario puede añadir las incidencias producidas y tener sin ningún esfuerzo, controladas todas las ausencias.
- Ágora genera estadísticas, con presentación gráfica, sobre aquellos temas más relevantes como puedan ser la rentabilidad, progresión de matrícula y procedencia del alumnado, a las que podrá añadir sus propias estadísticas personalizadas.

Actualmente se encuentra disponible en 4 ediciones distintas que se adaptan a disímiles requerimientos o perfiles de sus usuarios: Edición Estándar, Edición Profesional, Edición Empresarial y Edición Corporativa.

Este sistema tiene como inconveniente que está desarrollado en software propietario y el costo de licencia para su uso, mantenimiento y soporte es elevado.

Búho

Búho está diseñado específicamente para resolver los problemas cotidianos de gestión académica de una institución educativa. Mediante el uso de tecnologías web y una arquitectura cliente servidor, permite integrar la información que se genera diariamente en una institución educativa a la web. Los datos docentes pueden ser fácilmente consultados mediante la Internet o el acceso a través de la Intranet de la institución.

Búho permite a su vez generar todos los reportes requeridos por las autoridades educativas tanto en formato digital como en formato impreso. Además, cuenta con una interfaz donde puede permitirse a los representantes de los estudiantes realizar preinscripciones directamente desde la página web de la institución, ahorrando así tiempo y esfuerzo en esta delicada labor cada inicio de año escolar.

El mismo consta de 3 módulos:

- El sistema, el cual se sustenta en una base de datos MS SQL Server y una arquitectura cliente-servidor.
- La intranet, la cual da soporte web para actividades internas de la organización como son: la creación de los planes de estudio y el ingreso de las evaluaciones por parte del personal docente de la institución.
- El internet, que permite el acceso al sistema por parte de los representantes de los alumnos, facilitando información académica y permitiendo la inscripción remota.

Búho está implementado en software propietario y el costo de su licencia es elevado, además los procesos de gestión académica que automatiza no responden a las exigencias de la universidad.

SIGA

El Sistema Integral de Gestión Académica (SIGA) es una solución informática utilizada por la universidad de Córdoba para la automatización de la gestión académica. Esta aplicación, desarrollada por el servicio de informática de la universidad, se encuentra en continua evolución, ampliando nuevas funcionalidades y adaptándose a las nuevas tecnologías.

Las funcionalidades soportadas por SIGA son muy amplias, abarcan desde la gestión del acceso a la universidad hasta la tramitación de los títulos pasando por la matrícula, actas de examen, expedientes académicos, estadísticas, informes, gestión de becas y convalidaciones.

Se estructura en los siguientes módulos:

- General: aparecen activados todos los módulos adquiridos por el cliente y en sombra los no adquiridos o desactivados por el supervisor. Instala automáticamente nuevos módulos, actualiza el sistema, la fecha y la hora, las estadísticas de utilización del programa y el *Copyright*[2].
- Definición de estudios: permite al usuario organizar sus estudios en varios planes, cada plan en varios cursos y cada curso con varias asignaturas. Los términos plan, curso, asignatura, grupo, turno, son personalizables por el usuario, es decir, puede llamarles, año, programa, módulo, provincia, área si lo considera, adecuándose la mayoría de los listados y menús a estos términos. Puede variar el idioma o colocar los caracteres que desee. A cada nivel académico se le pueden añadir características propias como un límite de alumnos o destacarlo en importancia por alguna característica.
- Opciones de alumnos: está dividido en cuatro sub módulos: Gestión de tutorías, Gestión de mensajerías, Gestión de asistencias y Gestión de calificaciones.
- Gestión de Calificaciones: posibilita al usuario definir cuántos exámenes querrá hacer y de qué tipo y así hacer medias de exámenes con condiciones. Se generan listados para los expedientes académicos, calificaciones de un alumno/curso, observaciones, diplomas, entre otros.
- Profesores.
- Inventario.
- Ingresos-Gastos.
- GENFOR (Generador de informes, estadísticas y gráficos).
- Generador de horarios.
- Generador de diplomas.
- Matrícula: permite la inscripción del estudiante en las asignaturas que desee y le estén permitidas, conforme a su expediente académico y a la normativa existente. Además, posibilita actualizar datos estadísticos, solicitar grupos de clase y renovar la tarjeta universitaria.
- Automatrícula: permite que los alumnos se matriculen fácilmente vía Internet sin necesidad de realizar las gestiones antes pertinentes, utilizando para ello las aulas de informática de cada centro o incluso desde el ordenador de sus casas, por supuesto, los alumnos que lo deseen podrán acudir en cualquier momento a la secretaría de su centro para matricularse.

Tiene como desventaja que está desarrollado en software propietario y es comercializado a un costo elevado, además no satisface las necesidades de la universidad.

[2] Es usado para indicar que una obra está sujeta al derecho de autor, el cual es un conjunto de normas y principios que regulan los derechos morales y patrimoniales que la ley concede a los autores.

ALBA

"Desarrollado por un grupo de estudiantes argentinos, es un proyecto de desarrollo de software libre para la realización de un Sistema Informático Abierto de Gestión Unificada para Unidades Educacionales que brinda una herramienta para mejorar el trabajo cotidiano en las escuelas.

La primera versión de ALBA fue pensada para realizar pruebas en escuelas primarias de la ciudad de Buenos Aires. En octubre del 2007 fue liberada la versión 1.0 final, incorporando mejoras en los módulos liberados en la versión anterior con el objetivo de beneficiar la funcionalidad de los mismos y tomando como punto de partida las sugerencias de los centros donde se estaba utilizando y de colaboradores que han sumado ideas y aportes al proyecto.

El software está dividido en varias secciones, Seguridad, Gestión escolar, Docentes, Calendario, Horario escolar, entre otras. Entre las funcionalidades que brinda se encuentran la gestión de los alumnos dentro del centro de estudios, posibilitando que sean dados de alta y baja, que sean realizados listados y búsquedas de alumnos dadas ciertas características, permite además la gestión de notas, asistencias y ciertos registros médicos de los alumnos. Garantiza la gestión de los profesores de los centros de estudios y de la confección de los horarios para cada uno de estos. Otra de las principales funcionalidades que ofrece es la gestión escolar, en esta se definen los años que componen un plan de estudio, así como la gestión de secciones en la que son agrupadas las materias. Posee secciones de informes o consultas donde se puede obtener información acerca de los estudiantes y docentes que componen el centro. Brinda una ayuda detallada del sistema a los usuarios del mismo, la cual puede ser consultada a través de la aplicación o de internet." [2]

El sistema es desarrollado en el marco de trabajo Symfony con PHP 5, utiliza como servidor de base de datos MySQL 4.1 y el sistema operativo Linux.

Presenta como desventaja que su arquitectura no se ajusta a la definida por la UCI para sus soluciones de software y no responde con garantías a la gestión de los complejos procesos docentes de la universidad.

1.4 Los sistemas de gestión académica en Cuba

SIGENU

Desarrollado por el equipo SIGENU-CUJAE, conformado por especialistas del Ministerio de Educación Superior (MES) y de la universidad José Antonio Echevarría (CUJAE). El proyecto SIGENU (Sistema de Gestión de la Nueva Universidad) surge en 2004 a solicitud de la dirección del MES y como respuesta a la necesidad de automatizar los procesos fundamentales de la gestión académica de un centro de educación superior en sus modalidades de estudio.

El sistema está compuesto por varios módulos que gestionan la información de un estudiante desde que se matricula hasta que se gradúa o causa baja definitiva.

Consta de dos aplicaciones: un sistema transaccional y otro para la toma de decisiones.

Entre las funciones del sistema de gestión se encuentra: la inscripción de un estudiante, registro de asignaturas a cursar, evaluaciones, control de bajas, emisión de reportes oficiales, entre otros. Este soporta el almacenamiento de información agregada y nominalizada de cualquier estudiante que pertenezca a una institución de educación superior cubana, aunque no utilice como sistema de gestión académica el proyecto SIGENU.

El proyecto se desarrolla con herramientas de software libre. Se encuentra en explotación sobre el sistema gestor de base de datos PostGreSQL, aunque su diseño e implementación es independiente del gestor de base de datos y el sistema operativo.

Consta de los siguientes módulos: Codificadores, Matrícula, Control de Estudiantes, Plan de Estudio, Evaluaciones y Reportes.

El módulo Matrícula se encarga de automatizar el proceso a través del cual los nuevos estudiantes pasarán a ser registrados en el sistema como estudiantes de la educación superior. Tiene como particularidad que cuando una persona arriba a una secretaría para ser matriculada, se registra inicialmente como estudiante de nuevo ingreso y así es considerada hasta que finalice todo el proceso de matrícula con la operación, cierre de matrícula. A partir de este momento, todos los nuevos ingresos pasan a ser estudiantes del sistema.

El módulo control de estudiantes permite buscar un estudiante registrado en el sistema, modificar los datos de un estudiante tanto personales como docentes, ubicar a un estudiante en un grupo o cambiarlo de grupo, realizar el pase de estudiantes a otros años de estudio y definir los que serán repitentes, así como dar baja a un estudiante del centro ya sea por licencia de matrícula, resolución o traslado.

Este sistema no responde con garantías a la gestión de los complejos procesos docentes de la universidad, se limita a hacer la gestión de los datos de los estudiantes sin permitir la gestión de ningún dato referente a los profesores, ni las relaciones de estos con los estudiantes y grupos académicos. Está diseñado a nivel de secretaría, por lo que otros usuarios que forman parte importante del ámbito académico como los profesores y los estudiantes no pueden interactuar con él. No brinda la oportunidad de que el usuario pueda definir cuáles son los estados que se usan en la universidad. Tampoco permite gestionar los documentos que avalen el cambio de estado (por el momento para Pregrado en la parte de estudiante se tienen definidos como estados: ingreso, prematrícula, matrícula, baja, baja académica, egresado, eliminado, repitente, licencia especial, licencia de matrícula, traslado) por lo que es un sistema poco flexible.

GESTACAD

GESTACAD es un sistema de gestión académica, desarrollado por el Departamento de Informática de la universidad de Matanzas. Permite actualizar y mantener la información sobre estudiantes y profesores de una universidad y obtener determinados resultados propios del trabajo de las áreas implicadas, aunque el grueso de las informaciones se obtiene mediante el acceso al sitio web de la universidad.

En la actualidad el sistema es capaz de realizar diversas acciones y brindar numerosos reportes, los cuales son fruto de los requisitos funcionales del mismo. Estos son la búsqueda de un alumno, el listado de estudiantes por grupo, reportes dinámicos de la información existente, reportes de notas por asignatura y grupo, tablas con los resultados docentes de un grupo en un semestre, reportes de los resultados académicos de un estudiante en toda su carrera, actas de exámenes de las diferentes asignaturas, registros de características de un grupo de estudiantes, así como dar baja a los estudiantes.

El sistema consta de los siguientes módulos:

- Administración: para la gestión de las tablas del sistema vía web así como agregar nuevas consultas al sitio oficial y establecer los distintos niveles de acceso a estas.
- Web para las secretarias docentes: permite la realización de acciones generales comunes en una secretaría docente, así como la obtención de reportes oficiales.
- Web para los Jefes de Departamentos Docentes: se incluyen acciones relativas como la asignación de la carga docente y el control sobre los profesores del departamento.
- Web para la gestión de la matrícula: posibilita gestionar los datos de los estudiantes.
- Web para los profesores: estos pueden llevar el control docente de sus estudiantes, así como de sus evaluaciones y obtener reportes relativos a su carga docente.
- Un sitio web con reportes en línea: con la utilidad del registro docente para los profesores además de la búsqueda de estudiantes, la cual devuelve algunos datos personales del estudiante y su ubicación según el horario docente detallando: aula, asignatura y tipo de clases que está recibiendo.

Vale la pena destacar que dicho sistema fue usado en la UCI en el curso 2002-2003, cuando la Universidad contaba con una matrícula de 2000 estudiantes y sólo un plan de estudio, en el curso 2003-2004 se dejó de usar.

Posee problemas con la navegabilidad, el ambiente de trabajo resulta en ocasiones restringido e inflexible, aspectos que recrudecen la búsqueda de información, además no brinda la posibilidad de contar con más de un plan de estudio y gestionar las evaluaciones de una asignatura. Debido a la

cantidad de estudiantes que presenta la UCI, su diversidad, la complejidad de la gestión de personal y los procesos docentes, GESTACAD no satisface sus necesidades.

1.5 Sistemas de gestión académica en la Universidad de las Ciencias Informáticas

En el primer año de la UCI fue usado el sistema GESTACAD, cuando la universidad contaba con una matrícula de 2000 estudiantes y sólo un plan de estudio. Este sistema brindó diversas funcionalidades como la gestión de la matrícula, la gestión docente y la realización de reportes en línea, pero no cubría todas las necesidades del centro en cuanto a la gestión académica.

Basado en las experiencias obtenidas con el uso de GESTACAD se desarrolló en la universidad UCIMAT. Este sistema permite la búsqueda de estudiantes por determinados criterios e incluye la matrícula y modificación de los datos de estos, realiza reportes generales de los datos de los estudiantes que están matriculados en el sistema, gestiona su información académica y la información de los profesores que dirigen el proceso docente educativo. Como desventajas, el sistema no brinda la posibilidad de reingreso y registro de traslado, fue desarrollado en software propietario, es poco flexible y no realiza la gestión de tesis ni de títulos.

Akademos se pone en funcionamiento en el curso 2004/2005, el mismo fue implementado por estudiantes y profesores de la universidad a partir de las experiencias obtenidas en la utilización de los sistemas de gestión académica anteriores. Tiene como objetivo garantizar la gestión automatizada de los elementos que intervienen en la labor académica del centro, fue desarrollado en Visual Studio .NET 2003 utilizando SQL Server 2000 como gestor de base de datos y se diseñó teniendo en cuenta los siguientes principios:

- El dinamismo del proceso de gestión académica constituye la principal fuente de riesgos para un sistema que intente automatizarlo.
- Un sistema que automatice la gestión académica debe lograr que todos los involucrados (directivos, personal de secretaría, profesores y estudiantes) tengan un papel activo en el proceso.
- El plan de estudios es la entidad fundamental del proceso de gestión académica y rige todos sus subprocesos (matrícula, control, planificación).

Consta de siete módulos, que abarcan la mayoría de los procesos involucrados en la gestión docente, los mismos interactúan entre sí para llevar a cabo cada una de las tareas que el sistema automatiza.

- Plan de Estudio: permite la creación del plan de estudio de la carrera definiendo los niveles divididos en momentos que los estudiantes deben vencer una vez concluidas las asignaturas, así como la definición de los diferentes perfiles y disciplinas en las que se agrupan las asignaturas.
- Estudiante: garantiza que el estudiante interactúe con el sistema para consultar el registro de sus evaluaciones.

- Expediente: almacena de forma digital un conjunto de información referente al desempeño académico y a otros aspectos del comportamiento de un estudiante durante su estancia en la universidad.
- Matrícula: gestiona los datos de los estudiantes que van a ingresar en la universidad, así como la gestión de sus movimientos en su paso por la misma. En los centros que aceptan cada año miles de estudiantes, la tarea de matricularlos puede convertirse en una labor titánica, si no se involucra a una gran cantidad de personas en la misma, el módulo Matrícula permite la descentralización del proceso de ingreso en un conjunto de personas tan grande como se decida y es el encargado de definir la estructura del centro de estudios, la cual se concibe como un conjunto de estructuras de diferentes tipos: facultades y grupos docentes, que se organizan jerárquicamente para agrupar por un criterio determinado, tanto a los estudiantes como a los demás implicados en la labor docente, sean profesores, personal de secretaría o directivos.
- Profesor: permite planificar la carga docente de los profesores, gestionando la asignación de un profesor a un departamento y grupos docentes.
- Reportes: posee diferentes herramientas para el diseño y publicación de nuevos reportes que involucren diferentes aspectos de los estudiantes.
- Registro: gestiona el control de las asistencias y evaluaciones aplicadas por un profesor a sus estudiantes

El sistema está desarrollado completamente en software propietario, por lo que no se adecua a las políticas de software tanto de la universidad como del país, posee escasa documentación, lo que dificulta su mantenimiento, soporte y reutilización, además en la actualidad no responde a los complejos procesos de gestión académica del centro.

Usuarios potenciales del módulo Reportes no tienen acceso al mismo, y no satisface la totalidad de las necesidades de los que si tienen acceso. Aunque posee asistentes para la confección de los reportes, el trabajo con ellos resulta complejo, y no aprovecha otras vías de difusión de la información como el correo electrónico.

Los procesos realizados en los módulos matrícula y profesor, en realidad conforman uno solo: la gestión de personal, por ende muchas de las funcionalidades que brindan por separado se encuentran repetidas, perdiendo eficiencia el sistema.

El módulo Plan de Estudio, es aplicable a un centro de estudios donde exista solo una carrera, realiza una configuración estática de los planes de estudio, lo que conlleva serios problemas en los módulos Registro y Reportes. No es flexible a cambios que se puedan realizar y maneja algunos conceptos y operaciones que no poseen funcionalidad alguna para la nueva forma de concebir los planes de estudio.

No gestiona los procesos de tesis, lo que deriva numerosos problemas en la conformación de los jurados, demoras en la asignación de los temas de tesis, entre otros. Tampoco garantiza la gestión de los procesos de postgrado, lo que conlleva a que no exista un medio de información para que el usuario se mantenga actualizado.

En el módulo Registro, las evaluaciones y asistencia de los estudiantes a los turnos de clases son sólo accesibles por los profesores y personal de secretaría, impidiendo la interacción en línea de los estudiantes con su información académica. Para asignarle un profesor a un grupo docente es necesario ir al módulo Profesor, lo que hace el trabajo de secretaría más engorroso al tener que buscar funcionalidades dispersas en el sistema; provocando así pérdida de tiempo. Solicitar una asignatura de adelanto es complejo, y no se gestionan dichas solicitudes en el sistema.

El módulo Profesor no permite la obtención de un grupo de reportes relativos a la planificación docente, que resultan de importancia significativa para los directivos de cualquier centro de estudios y carece de un buscador que simplifique el trabajo a la hora de realizar las asignaciones a los profesores.

1.6 Lenguaje de Programación PHP

“Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.” [3]

“PHP es un lenguaje creado por una gran comunidad de personas, fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf y denominado Personal Home Page Tools. Adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.” [4]

En el último año, el número de servidores que utilizan PHP se ha disparado, logrando situarse cerca de los 5 millones de sitios y 800.000 direcciones IP, lo que le ha convertido a PHP en una tecnología popular. Esto es debido, entre otras razones, a que PHP es el complemento ideal para que el tándem Linux-Apache sea compatible con la programación del lado del servidor de sitios web.

Ventajas:

- Fácil de aprender.
- Se caracteriza por ser un lenguaje rápido.
- Soporta en cierta medida la orientación a objeto.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.

- Capacidad de conexión con la mayoría de los manejadores de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- Capacidad de expandir su potencial utilizando módulos.
- Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones, además de una gran comunidad de desarrolladores.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables.

1.7 Sistema de gestión de bases de datos PostGreSQL

"Los sistemas de gestión de bases de datos o SGBD son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos[3], el usuario y las aplicaciones que la utilizan." [5]

"PostGreSQL es un sistema de gestión de bases de datos objeto-relacional[4] basado en el proyecto POSTGRES, de la universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por *Defense Advanced Research Projects Agency* (DARPA), el *Army Research Office* (ARO), el *National Science Foundation* (NSF), y ESL, Inc." [6]

PostGreSQL es una derivación libre de este proyecto, fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostGreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

A continuación se enumeran las principales características de este gestor de bases de datos:

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, entre otros. También permite la creación de tipos propios.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, entre otras.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.

[3] Base de datos: cualquier conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora

[4] Existen cuatro modelos principales de bases de datos: el modelo jerárquico, el modelo en red, el modelo relacional (el más extendido hoy día) y el modelo de bases de datos deductivas

- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a estos.

1.8 Marco de trabajo

"En el desarrollo de software, la palabra inglesa *framework* (marco de trabajo) es una estructura conceptual y tecnológica de soporte, definida normalmente con artefactos de software concretos, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio." [7]

CodeIgniter

"CodeIgniter es un marco de trabajo para PHP. Es adecuado para desarrollos que requieran mucho rendimiento, que ejecutan muchas versiones de PHP con diferentes configuraciones. Una de sus mayores ventajas es la documentación que se ofrece en internet. Su principal objetivo es ayudar a que los desarrolladores puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero.

También hay que destacar que CodeIgniter es más rápido que muchos otros entornos. Incluso en una discusión sobre entornos de desarrollo con PHP, Rasmus Lerdorf (el creador de PHP) expresó que le gustaba CodeIgniter "porque es rápido, ligero y parece poco un entorno". [8]

JQuery

El jQuery es un marco de trabajo de JavaScript para acceder a los objetos del DOM de un modo simplificado.

El autor de esta librería es John Resig que además trabaja para Mozilla Corporation.

"Es un producto con una gran aceptación por parte de los programadores y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del *framework*. Posee una gran comunidad de creadores de componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, votaciones, efectos diversos, entre otros". [9]

Ventajas:

- Ahorra muchas líneas de código.
- Hace transparente el soporte de la aplicación para los navegadores principales.

- Provee de un mecanismo para la captura de eventos.
- Provee un conjunto de funciones para animar el contenido de la página.
- Integra funcionalidades para trabajar con AJAX.
- Tiene licencia para uso en cualquier tipo de plataforma, personal o comercial.
- Mejora el tiempo de creación y depuración.

1.9 Tecnologías utilizadas

AJAX

AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML".

"AJAX no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes. "[10]

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. *"La capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor."* [11]

Ventajas:

- Utiliza tecnologías ya existentes.
- Soportada por la mayoría de los navegadores modernos.
- Interactividad, el usuario no tiene que esperar hasta que lleguen los datos del servidor.
- Portabilidad
- Mayor velocidad, debido a que no hay que retornar toda la página nuevamente.
- La página se asemeja a una aplicación de escritorio.

Desventajas:

- Se pierde el concepto de volver a la página anterior.

- Si se guarda en favoritos no necesariamente al visitar nuevamente el sitio se ubique donde nos encontrábamos al grabarla.
- La existencia de páginas con AJAX y otras sin esta tecnología hace que el usuario se desoriente.
- Problemas con navegadores antiguos que no implementan esta tecnología.
- No funciona si el usuario tiene desactivado el JavaScript en su navegador.
- Requiere programadores que conozcan todas las tecnologías que intervienen en AJAX.
- Dependiendo de la carga del servidor se puede experimentar tiempos tardíos de respuesta que desconciertan al visitante.

HTML

"El origen de HTML se remonta a 1980, cuando el físico Tim Berners-Lee, trabajador del CERN (*Consejo Europeo para la Investigación Nuclear*)[5] propuso un nuevo sistema de *"hipertexto"* para compartir documentos.

En el ámbito de la informática, el *"hipertexto"* permitía que los usuarios accedieran a la información relacionada con los documentos electrónicos que estaban visualizando. De cierta manera, los primitivos sistemas de *"hipertexto"* podrían asimilarse a los enlaces de las páginas web actuales.

HTML (*Lenguaje de Marcas de Hipertexto*), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento." [12]

XML

"El XML es el estándar de *Extensible Markup Language*, proviene de un lenguaje que inventó IBM en los años 70. El lenguaje se llama GML (*General Markup Language*) y surgió por la necesidad que tenían en la empresa de almacenar grandes cantidades de información de temas diversos

XML no es más que un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. Es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

En teoría HTML es un subconjunto de XML especializado en presentación de documentos para la web, mientras que XML es un subconjunto de SGML especializado en la gestión de información para la web. En la práctica XML contiene a HTML aunque no en su totalidad. La definición de HTML

[5] El CERN tiene 20 estados miembros y el 50% de los físicos de partículas de todo el mundo. Su sede se encuentra en la ciudad suiza de Meyrin. En las afueras del principal edificio de esta institución se encuentra una tarja donde se lee ¨Aquí nació internet¨.

contenido totalmente dentro de XML y por lo tanto que cumple en su totalidad la especificación SGML es XHTML (*Extensible, Hypertext Markup Language*)." [13]

1.10 Entorno de Desarrollo Integrado NetBeans 6.8

"Un entorno de desarrollo integrado o IDE, es un programa informático compuesto por un conjunto de herramientas de programación.

Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes." [14]

NetBeans comenzó como un proyecto estudiantil en República Checa (originalmente llamado Xelfi).

El NetBeans IDE es un ambiente libre de desarrollo integrado para desarrolladores de software. El mismo ofrece todas las herramientas necesarias para crear escritorios profesionales, web y aplicaciones móviles con el lenguaje Java, JavaFX, C / C + + y lenguajes dinámicos como PHP, Java Script, Groovy y Ruby. El NetBeans IDE es de fácil instalación y uso, se ejecuta en Windows, Linux, Mac OS X y Solaris (TM).

Su nueva versión NetBeans IDE 6.8 brinda soporte completo para Java EE 6 y ofrece PHP mejorado, soporte para JavaFX y C/C ++, además ofrece otras nuevas características y mejoras que incluyen:

- Soporte PHP ampliado: expande el soporte de los lenguajes dinámicos con apoyo para PHP 5.3 y el esquema de Symfony acelera el desarrollo de aplicaciones web PHP.
- Una integración más ajustada con Project Kenai.
- Mejora de C / C + + Profiling: perfila y sintoniza aplicaciones C / C + + con el nuevo indicador Microstate Accounting, supervisor de uso I/O.
- JavaFX ™: código de finalización mejorado, sugerencias y navegación para JavaFX en el editor NetBeans.

"Con esta nueva versión de NetBeans, Sun continúa con su compromiso de ofrecer herramientas de desarrollo *Open Source*[6] (Código abierto), Java EE 6 y *GlassFish* v3 permiten a los desarrolladores crear aplicaciones empresariales con mayor facilidad y menos códigos, acelerando de manera significativa la velocidad de desarrollo y despliegue de aplicaciones." [15]

"Mientras que otras estructuras IDE de *Open Source* han crecido requiriendo una personalización significativa para el mercado durante las dos últimas décadas, NetBeans 6.8 ofrece una plataforma completamente capaz, moderna, ligera y flexible para el futuro de IDEs integrados. " [16]

[6] Es el término con el que se conoce al software distribuido y desarrollado libremente. Fue utilizado por primera vez en 1998 por algunos usuarios de la comunidad de software libre.

1.11 Conclusiones

En el capítulo fueron analizados algunos de los sistemas de gestión académica desarrollados en el mundo, en Cuba y en nuestra universidad.

Como ventajas sobresalen: la mayoría de estas soluciones permiten la gestión centralizada de todas las bases de datos de alumnos, profesores, aulas y clases; personalizar los roles de usuarios estableciendo los niveles de acceso al sistema; la generación automática de la documentación y gestionar con gran facilidad las excepciones que trastocan la planificación de un centro de estudios.

Como desventajas se identifican las siguientes: de los sistemas encontrados en el mundo, la mayor parte está desarrollada en software propietario lo que va en contra de las políticas del país y se necesita licencia para su uso, mantenimiento y actualización, de los implementados en la universidad y en el resto del país, se puede señalar, que en la actualidad, debido a lo complejo de la gestión de personal y de los procesos docentes, no satisfacen las necesidades del centro.

Se evidencia el uso de las tecnologías web como principal pilar para lograr un software que automatice la gestión académica, donde todos los involucrados tengan un papel activo en el proceso, garantizando muy bajos costos y gran seguridad en la transmisión de datos.

Se realizó un análisis de las tendencias y tecnologías actuales y se demostró la factibilidad del uso de herramientas libres como PHP y los marcos de trabajo CodeIgniter y JQuery, como gestor de bases de datos, PostGreSQL y como IDE, NetBeans 6.8.

Descripción y análisis de la solución propuesta

2.1 Introducción

En el capítulo se aborda: la descripción de la solución propuesta, la valoración crítica del diseño propuesto por los analistas, y de posibles implementaciones, componentes o módulos que son reutilizados, así como la descripción de los algoritmos no triviales empleados e información detallada de las clases implementadas.

2.2 Procesos de gestión de personal en el subsistema Gestión de Pregrado

El Sistema de Gestión Universitaria incluye la realización de varias líneas de desarrollo, todas estas tienen en común una serie de procesos de gestión, como son los de gestión de personal. Dentro de estos, los de Pregrado son los más importantes por estar encargados de la gestión de estudiantes y profesores, principales elementos dentro de un centro universitario.

Prematrícula: en la UCI el proceso de prematrícula se realiza antes de la incorporación de los estudiantes a la universidad, permitiendo tener un registro previo de los que se incorporan a la misma, obtener reportes y realizar algunas actividades que agilicen el proceso de matrícula. Para realizar la prematrícula se importa el listado de estudiantes en Ingreso (este listado puede estar en una base de datos Access, o en un fichero Excel o XML) con la información de estos para la base de datos del sistema, con este listado se realiza la distribución por estructura, se crea un Id-expediente y el Portafolio Digital del estudiante en el Sistema de Gestión Documental Alfresco, posteriormente se confeccionan las tirillas con la ubicación del grupo y la residencia. Cuando el estudiante conoce su ubicación procede a sacarse las fotos y se realiza la asociación de la foto con su Id-expediente, creándose la Hoja de Prematrícula. El estudiante en ese momento queda en estado prematriculado.

Matrícula: la matrícula es el proceso que permite registrar oficialmente a un estudiante en un centro de estudios. Una vez que el estudiante este prematriculado debe dirigirse a realizar la matrícula y entregar una fotocopia de la certificación de estudios terminados, la baja del servicio y la certificación de notas, estos documentos son escaneados y archivados en el Portafolio Digital del estudiante. Después de recogida toda la información se genera la Hoja de Matrícula la cual es impresa y se archiva en el Portafolio Digital de este y en su expediente físico, de esta forma, queda finalmente matriculado. Si llega después del plazo fijado para la matrícula, debe dirigirse a la secretaría general. Si sus datos están en la base de datos y el estudiante está en estado de ingreso, la secretaria le

informa a que facultad y grupo pertenece, esta procede a realizar la matrícula una vez que el colegial se haya tirado la foto y se presente en la secretaría. Si el estudiante no está en estado de ingreso (no está en la base de datos), la secretaria general lo incluye en la base de datos, lo ubica en una facultad y en un grupo y procede a hacer la matrícula una vez que este se haya tirado la foto. Pasados seis meses aquel estudiante que no realizó la matrícula y no presentó las causas que le impidieron incorporarse a la universidad es dado de baja del ingreso y pierde la plaza, si el estudiante presenta las causas que impidieron su incorporación se le guarda la plaza para el próximo curso para que realice el proceso de matrícula.

Contratación de profesores fijos o adjuntos a una facultad: estos se presentan en la Dirección de Capital Humano del centro universitario a hacer su contrato laboral. Se registran sus datos en la Ficha de Contrato y se crea el Portafolio Digital del Trabajador Docente. Los decanos de cada una de las facultades reciben el listado de los profesores que fueron ubicados en dicha facultad, y se procede a realizarles una entrevista, luego el decano en conjunto con el vicedecano docente y los jefes de departamento ubican al profesor en un departamento, posteriormente se le asignan las responsabilidades que el mismo debe asumir tanto en una asignatura como en un grupo administrativo.

Gestión de la ubicación de personas por estructura: siendo estructura el local donde se ubica una persona para que ejerza su función, dígase departamentos, facultades, oficinas, grupos administrativos, la gestión de una persona a una estructura se puede hacer de forma dinámica o de forma manual.

Gestión de movimientos: constituyen movimientos todos aquellos procesos que impliquen un cambio de estado, de estructura o de responsabilidad/cargo de una persona o un grupo de personas. Los movimientos pueden ser: **internos** y **externos**. Se llamarán movimientos internos a los cambios de estructuras (cambiar de grupo, cambiar de facultad), se llamarán movimientos externos a los cambios que generan cambios de estados: bajas, traslados (de la universidad para otro Centro de Enseñanza Superior (CES) y viceversa), reingresos, licencias.

Traslados entre grupos: en el caso de los traslados para otro grupo el estudiante debe presentarse en la secretaría docente de su facultad y solicitar el cambio de grupo, la secretaria docente se lo hace saber al decano de la facultad que es quien autoriza el cambio y si el decano autoriza el cambio la secretaria docente cambia al estudiante de grupo.

Traslados para otra facultad: en el caso de que el traslado sea de facultad, el estudiante debe presentarse a la secretaría docente de su facultad y solicitar el traslado de facultad, presentando para ello una carta donde expone los motivos por los cuales quiere cambiar de facultad, la secretaria le entrega la carta al decano de la facultad para que apruebe el traslado, de ser aprobado el traslado y

haber sido aceptado por el decano de la facultad para la cual se va a trasladar, la secretaria docente de la facultad a la cual pertenece informa a la secretaria general y ella es la encargada de realizar el cambio de facultad, emitiendo para ello una resolución del decano de la facultad a la cual pertenecía el estudiante autorizando el cambio.

Traslado externo (para otra universidad): en el caso de que el traslado sea para otra universidad, el estudiante se presenta en la secretaría docente de su facultad y lo solicita, presentando una carta donde expone los motivos, la secretaria le entrega al decano de la facultad la carta y este es el encargado de aprobarlo, para esto deben de comprobar una serie de datos del estudiante como son las notas, su comportamiento y las causas por las cuales solicita el traslado. Si la facultad lo acepta, la secretaria docente se lo hace saber a la secretaría general y esta lo presenta en la rectoría para que sea aprobado; en caso positivo, el centro le presenta a la otra universidad una copia del expediente del estudiante y estos atendiendo a una serie de factores como la capacidad de la facultad para donde se va a realizar el traslado y la trayectoria lo aceptan o no; en el caso de que se acepte el traslado, la secretaria general le da baja y emite una resolución rectoral, esta se archiva en el expediente del estudiante, el cual se envía para la otra universidad.

Bajas: en el caso de las bajas, el estudiante se presenta en la secretaría docente de su facultad y la solicita, presentando para ello una carta donde expone sus motivos. La secretaria le entrega la carta al decano de la facultad que es el autorizado a aprobarla, si esta es aprobada, la secretaria docente informa a la secretaria general, y es esta quien le da la baja al estudiante, llenando para ello un documento en el que se plasman todos los datos de este y los motivos de la baja. Esta puede ser voluntaria, por insuficiencia docente, sanción disciplinaria, entre otras.

Ratificación de matrícula: se les realiza a los estudiantes continuantes o de licencia que se incorporan a las actividades docentes. Estos deben presentarse en la secretaría docente de su facultad a ratificar matrícula en el período que el centro universitario estime pertinente. Deberán presentar su carnet de identidad para que la secretaria docente de su facultad llene el modelo de ratificación de matrícula, en el cual se recogen los datos básicos como nombre y apellidos, dirección particular y carnet de identidad. Este proceso es de utilidad para rectificar los datos de los estudiantes y corregir algún error que exista.

Gestión de la documentación: este proceso gestiona toda la documentación asociada al Portafolio Digital de Estudiantes y Trabajadores Docentes, dando la posibilidad de insertar, visualizar y modificar un documento.

2.3 Descripción de la solución propuesta

Registrar estudiante (Prematrícula manual)

Para registrar a un estudiante primeramente se busca para comprobar que no se encuentre registrado en el sistema, si está registrado se verifican su estado y sus datos, en caso contrario se muestra un mensaje y la vista de la última configuración, si se está conforme con la configuración se carga la vista de la plantilla Prematrícula del Sistema de Gestión Documental, mostrándose en forma de formulario donde se pueden insertar los datos del estudiante, una vez insertados los datos, el sistema debe generar un id-expediente. La plantilla es guardada en el Sistema de Gestión Documental, y los datos del estudiante en la base de datos del sistema, el colegial se une a la lista de estudiantes en estado ingreso sin ubicación. Finalizado el proceso se muestra un mensaje de confirmación. Si no se estuviera conforme, se presiona **Cancelar** y se cambia la configuración.

Si el estudiante se encuentra en estado Aplazado y se desea activar, se presiona el botón **Registrar** mostrándose la vista de configuración para ver si se está conforme con esta, en caso positivo se presiona el botón **Aceptar**, se carga la plantilla y se procede a registrar al estudiante, en caso contrario se presiona el botón **Cancelar** mostrándose la vista de configuración. El sistema gestionará las trazas de: Fecha en que se realizó la acción, y usuario que realizó la acción.

Fig. 1 Buscar prematrícula manual

Fig. 2 Formulario Hoja de Prematrícula Curso Regular Diurno

Realizar matrícula

La matrícula debe realizarse después que el estudiante se haya tirado la foto, este se presenta al matriculador que es la persona encargada de registrar sus datos en la aplicación, el cual busca al estudiante que desee matricular mediante los criterios de búsqueda (nombre, apellido, carnet de identidad(CI), sexo, provincia, municipio), mostrándose un listado con la(s) persona(s) que coincida con ese criterio, el que tendrá(foto, nombre y apellidos, estado, y un vínculo para cargar la Hoja de Matrícula.

Al seleccionar el vínculo para cargar la Hoja de Matrícula, se muestra una nueva ventana con los datos a llenar, aparecen algunos datos llenos provenientes de la Prematrícula (todos los que coincidan en las dos hojas), estos datos se rectifican y se agregan los que falten en la plantilla. Cuando se complete la plantilla se guardan los datos, y se muestra un mensaje de confirmación, la información se guarda en la base de datos del sistema y se conforma un documento que se registra en el Sistema de Gestión Documental. El sistema gestionará las trazas de: Fecha en que se realizó la acción, y usuario que realizó la acción.

Fig. 3 Formulario Hoja de Matrícula Curso Regular Diurno

Registrar profesor (manual)

Para registrar a un profesor en el sistema se debe seleccionar la opción **Registrar**, mostrándose un listado de las personas registradas en el sistema, el listado se mostraría en blanco se cargaría según el resultado de la búsqueda que se realiza.

Si se selecciona trabajador docente se mostrará una plantilla (Ficha de Contrato) para introducir los datos correspondientes, una vez introducidos, se presiona el botón **Aceptar/Cancelar**. Si presiona el botón **Aceptar** y los datos se han introducido correctamente se mostrará un mensaje de confirmación.

Una vez introducido los datos se cargaría la persona registrada en la vista del listado mostrándose (Nombre y apellidos/ocupación/área a la que pertenece) y la columna opciones donde se tendría un icono de Ficha de Contrato (vista de impresión), la opción de editar datos (esta mostraría la misma Ficha de Contrato de forma que se puedan modificar los datos de las personas) y otras opciones como: registrar asignatura y departamento. El sistema gestionará las trazas de: Fecha en que se realizó la acción, y usuario que realizó la acción.

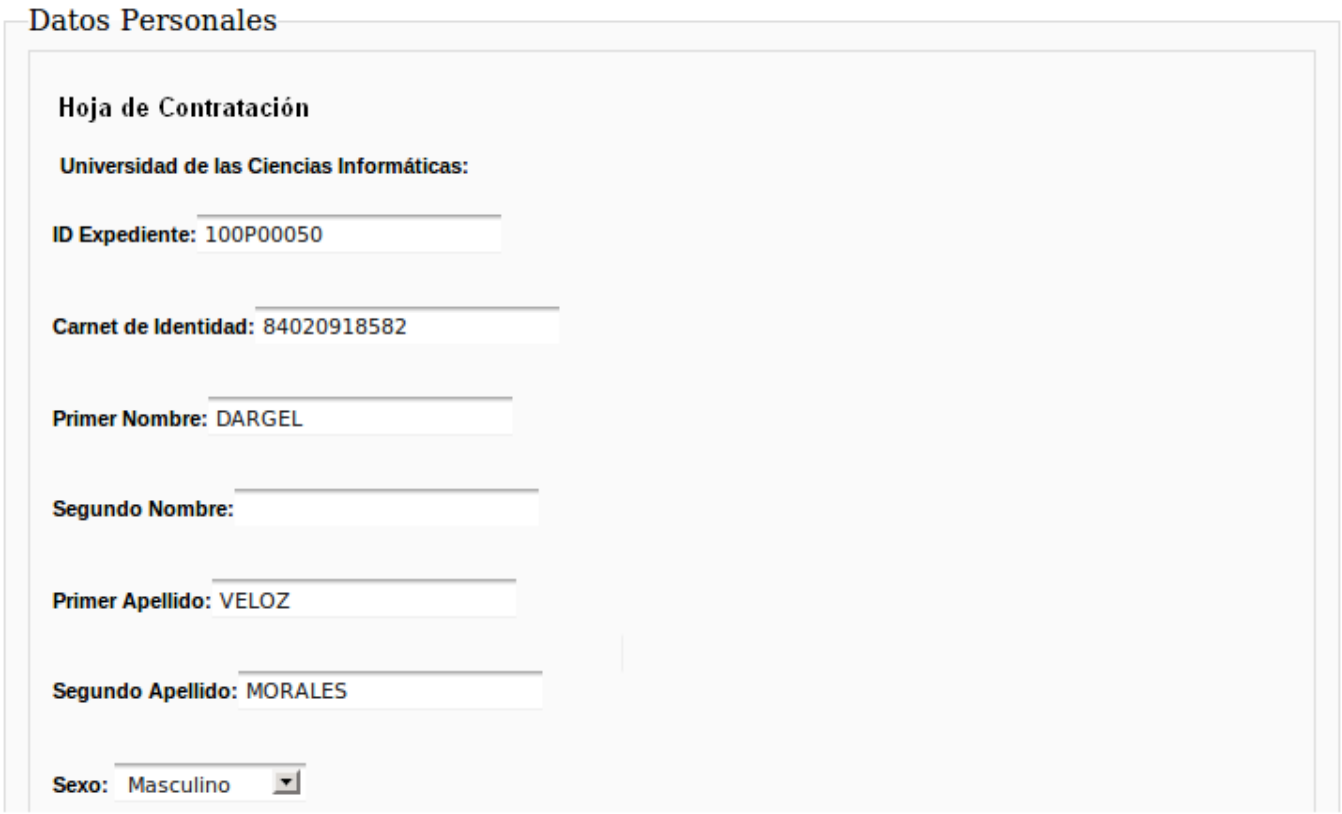


Fig. 4 Formulario Hoja de Contratación de trabajador docente

Ubicar estudiantes en estructuras dinámicamente

Para ubicar estudiantes en estructuras dinámicamente, se debe seleccionar del listado de estudiantes registrados los estudiantes que se les desee asignar una estructura (grupo administrativo) y se presiona el botón **Siguiente**, mostrándose una vista donde se deberán seleccionar primero los datos para la ubicación dinámica, luego los criterios de ubicación y las estructuras donde se ubicarán a los estudiantes. Además, se pueden especificar otros datos como: por cada facultad la cantidad total de estudiantes, cantidad de grupos por facultad y cantidad de estudiantes por grupos. Una vez seleccionado los datos se presiona el botón **Siguiente**, mostrándose una nueva vista con la cantidad de estudiantes y grupos creados por facultad y estudiantes por cada grupo. Si se realiza la acción correctamente se muestra un mensaje de confirmación. El sistema gestionará las trazas de: Fecha en que se realizó la acción, y usuario que realizó la acción.

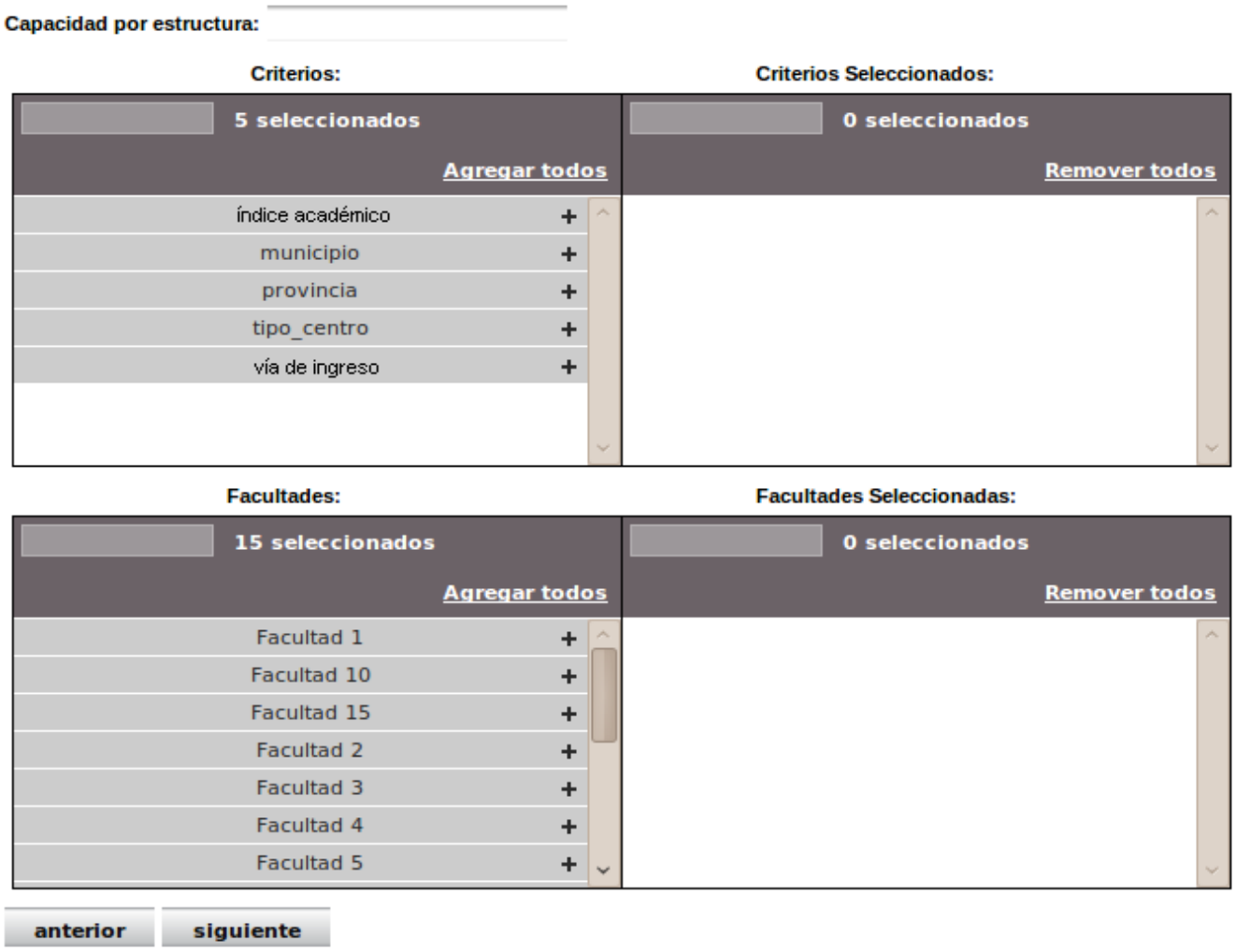


Fig. 5 Criterios y estructuras de ubicación.

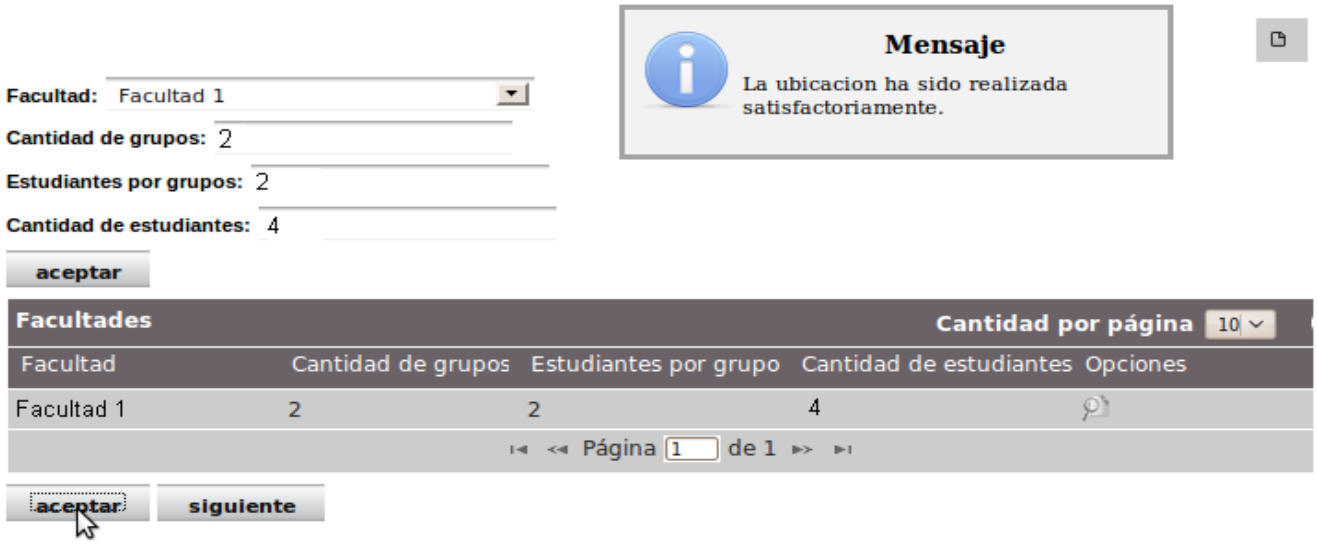
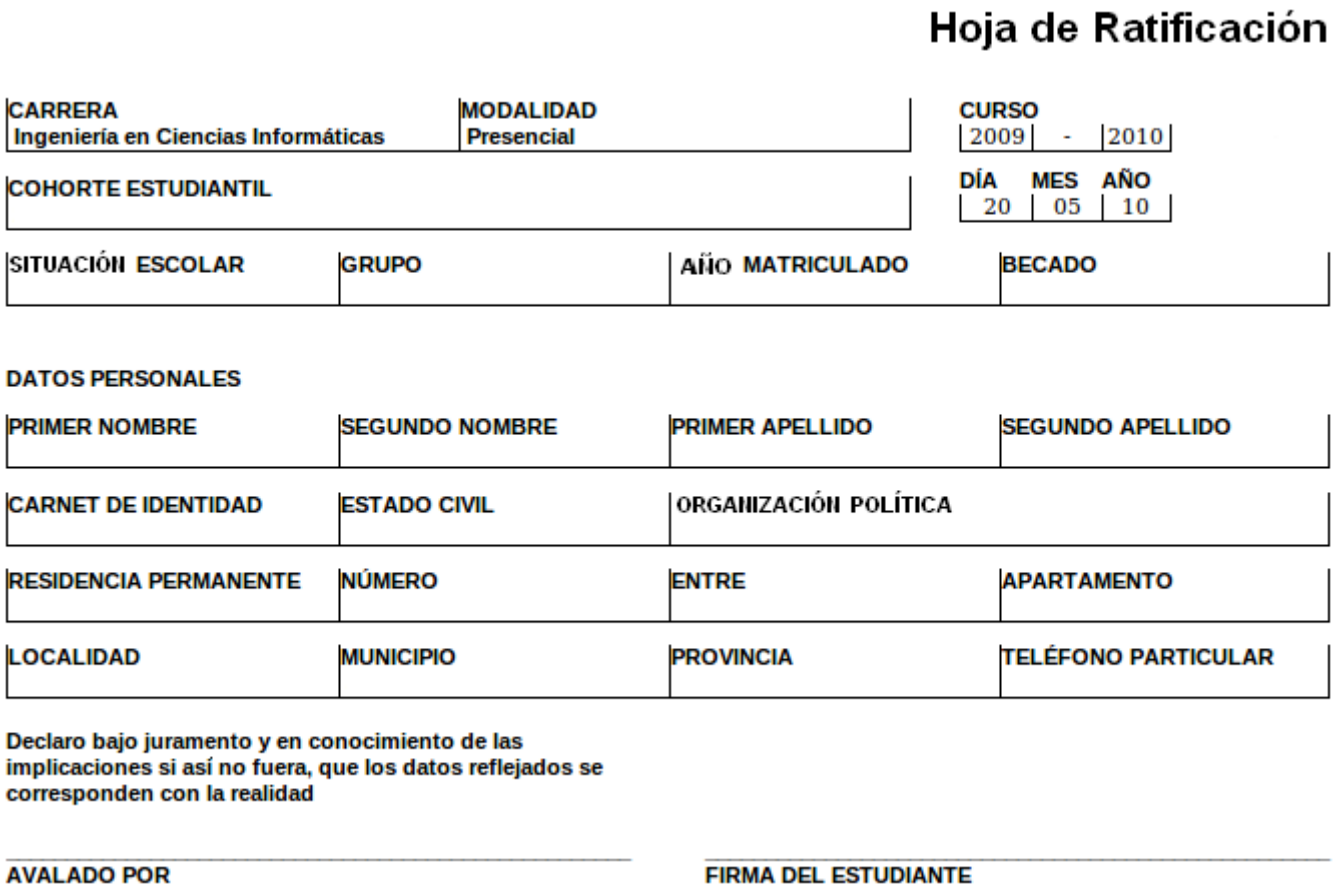


Fig. 6 Especificaciones de ubicación

Ratificar matrícula

Para ratificar matrícula se mostrará una vista que permite realizar una búsqueda por criterios para localizar al estudiante que se desea ratificar, se mostrará un listado con los estudiantes que coincidan con los criterios de búsqueda, se selecciona al estudiante que desee y se levanta la vista de la Hoja de Ratificación. En la plantilla se mostrarán algunos datos del estudiante, una vez registrados se procede a guardar la información mostrándose un mensaje de confirmación. El sistema gestionará las trazas de: Fecha en que se realizó la acción, y usuario que realizó la acción.



Hoja de Ratificación

CARRERA	MODALIDAD	CURSO
Ingeniería en Ciencias Informáticas	Presencial	2009 - 2010

COHORTE ESTUDIANTIL	DÍA	MES	AÑO
	20	05	10

SITUACIÓN ESCOLAR	GRUPO	AÑO MATRICULADO	BECADO

DATOS PERSONALES

PRIMER NOMBRE	SEGUNDO NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO

CARNET DE IDENTIDAD	ESTADO CIVIL	ORGANIZACIÓN POLÍTICA

RESIDENCIA PERMANENTE	NÚMERO	ENTRE	APARTAMENTO

LOCALIDAD	MUNICIPIO	PROVINCIA	TELÉFONO PARTICULAR

Declaro bajo juramento y en conocimiento de las implicaciones si así no fuera, que los datos reflejados se corresponden con la realidad

AVALADO POR

FIRMA DEL ESTUDIANTE

Fig. 7 Vista de impresión Hoja de Ratificación

Gestionar nomenclador Tipo de movimiento

Los Tipos de movimientos se muestran en un listado, en la columna Opciones se brinda la posibilidad de **Modificar**, **Ver detalles**, **Activar/Desactivar** y la opción de **Asociar estados.**

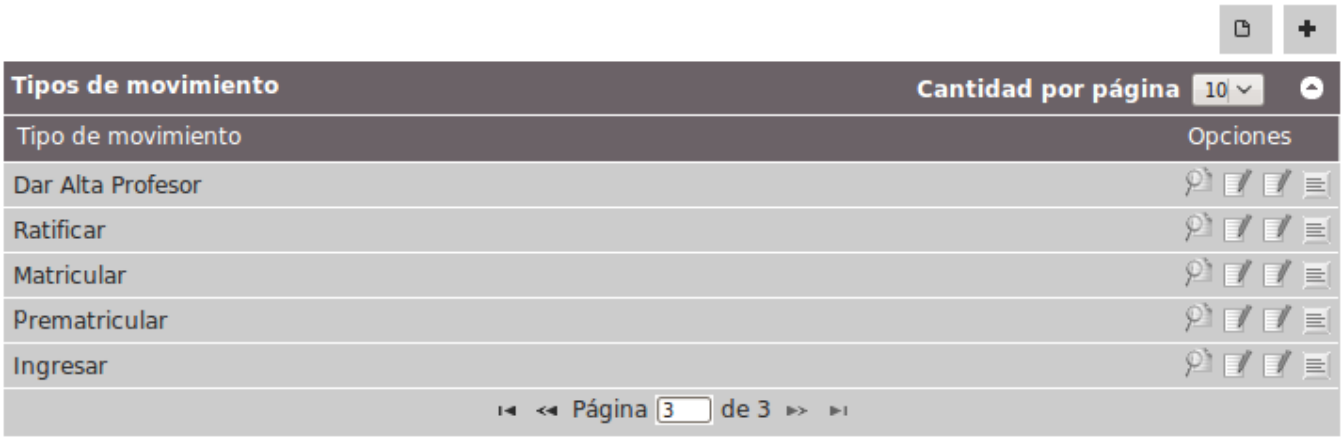


Fig. 8 Listar Tipos de movimiento

Para crear un Tipo de movimiento se debe insertar el nombre, la descripción, tipo de persona asociada, flujo asociado, si es solicitud y seleccionar la opción **Activar** (si desea que el nomenclador se cree de forma activa). Se guardan los datos insertados, mostrándose un mensaje de confirmación.

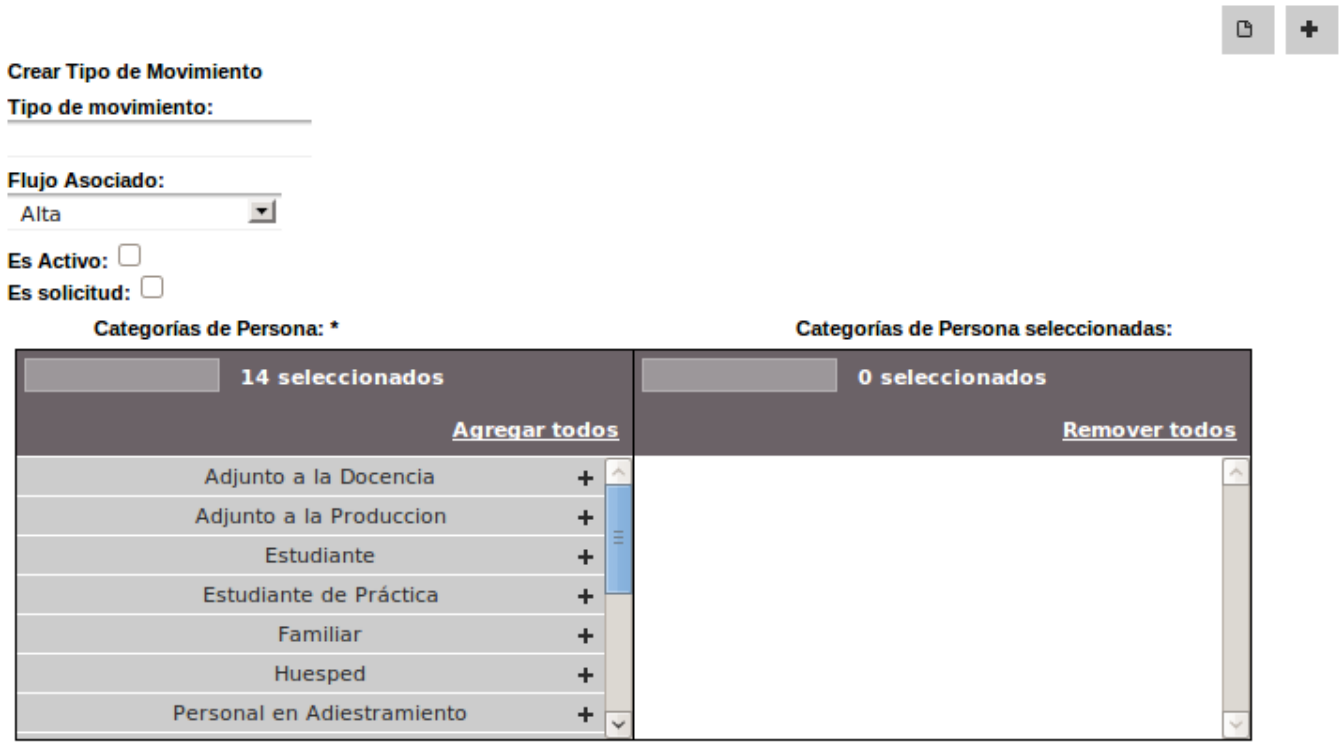


Fig. 9 Crear Tipo de movimiento

Para modificar un Tipo de movimiento se selecciona el movimiento que se desea modificar del listado anterior, mostrándose los datos que se pueden modificar de manera editable. Se modifican los datos que se desean y se procede a guardar los cambios, mostrándose el listado actualizado de todos los movimientos.

Para ver detalles de un Tipo de movimiento se selecciona el movimiento que se desea ver presionando el ícono lupa.

Para activar o desactivar un Tipo de movimiento se selecciona el movimiento que se desea y se activa o se desactiva según el caso presionando el botón correspondiente.

Para asociar los estados se selecciona en la columna Opciones **Asociar**, mostrándose una vista para relacionar los estados iníciales y finales o editar los estados definidos y activar o desactivar según el caso.

El sistema gestionará las trazas de: Fecha en que se realizó la acción.

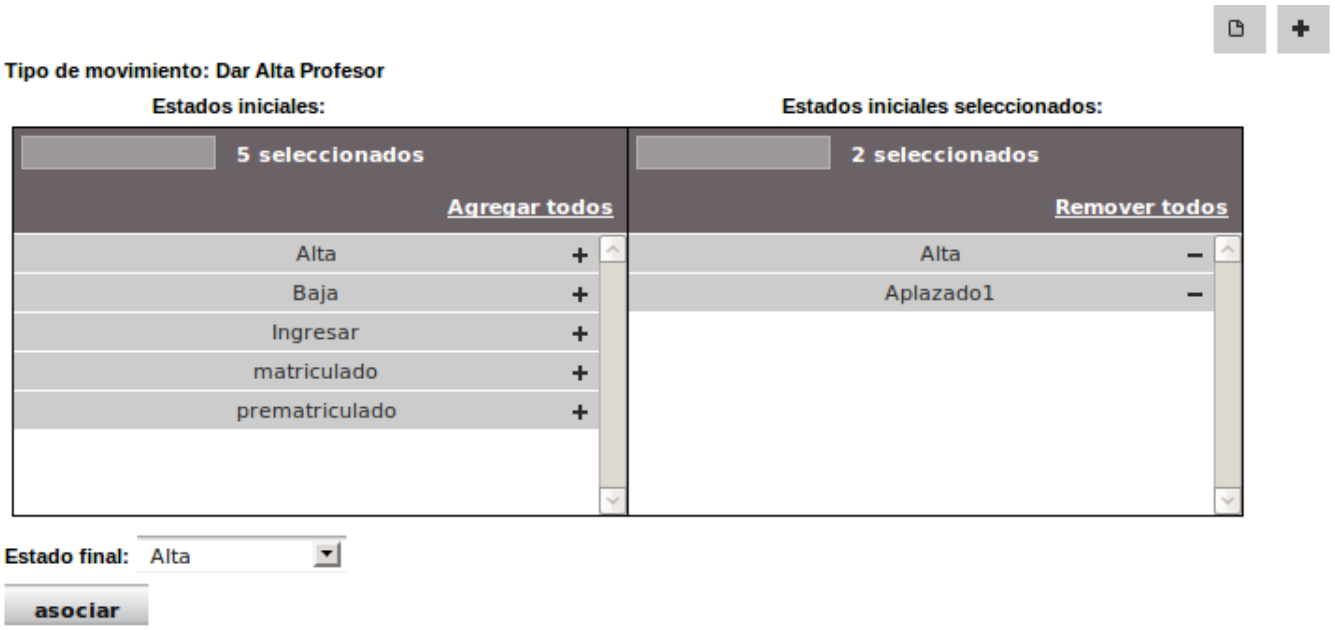


Fig. 10 Asociar estados

2.4 Valoración crítica del diseño propuesto

Haciendo uso de la metodología Scrum-XP los analistas y diseñadores del sistema realizaron una propuesta de solución que satisface en su totalidad las necesidades del cliente. Se generaron diferentes artefactos, el más importante de ellos, para el programador es la Historia de Usuario (HU) que le es entregada para la implementación.

Las HU cuentan con una breve descripción de alguna funcionalidad en específico que debe cumplir el sistema y las interfaces correspondientes para la solución de las mismas. Emplean terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

2.5 Arquitectura cliente/servidor

"Los patrones arquitectónicos expresan una organización estructural para un sistema de software, proveen un conjunto de subsistemas predefinidos e incluyen reglas y lineamientos para conectarlos."[18]

La arquitectura cliente/servidor consiste en varios clientes distribuidos en diferentes nodos, conectados en red a uno o varios nodos servidores, donde el servidor puede atender a varios clientes a la vez. En los nodos clientes se encuentra la presentación de usuario y en los nodos servidores la lógica del negocio. La arquitectura cliente/servidor es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema.

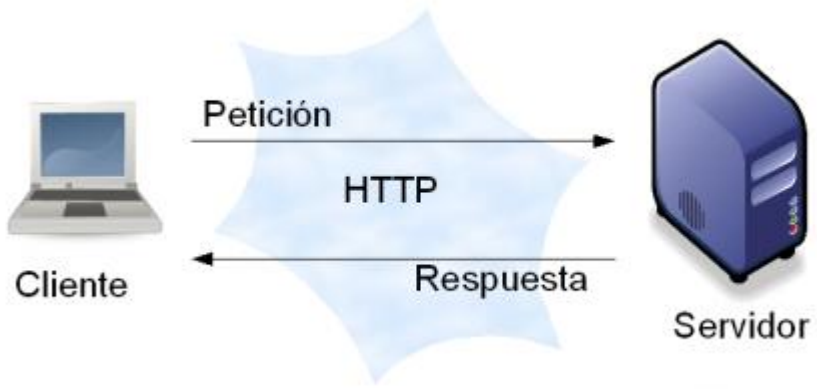


Fig. 11 Arquitectura cliente/servidor

2.6 Patrón Modelo Vista Controlador, Vista Arquitectónica

Buscando la flexibilidad, reusabilidad y escalabilidad del módulo Personal y Secretaría, se utiliza una arquitectura de software distribuida, cliente/servidor en tres capas, lo cual simplifica la comprensión y la organización del sistema, reduciendo las dependencias de forma que las capas más bajas no sean conscientes de ningún detalle o interfaz de las superiores. Esta arquitectura permitirá la interoperabilidad en entornos distribuidos con un nivel de abstracción superior, además de lograr una interfaz de usuario más flexible permitiendo que la aplicación sea más simple y escalable.

El marco de trabajo utilizado está basado en el patrón Modelo Vista Controlador el cual separa la lógica de la aplicación de la presentación quedando la Vista Arquitectónica de la siguiente forma:

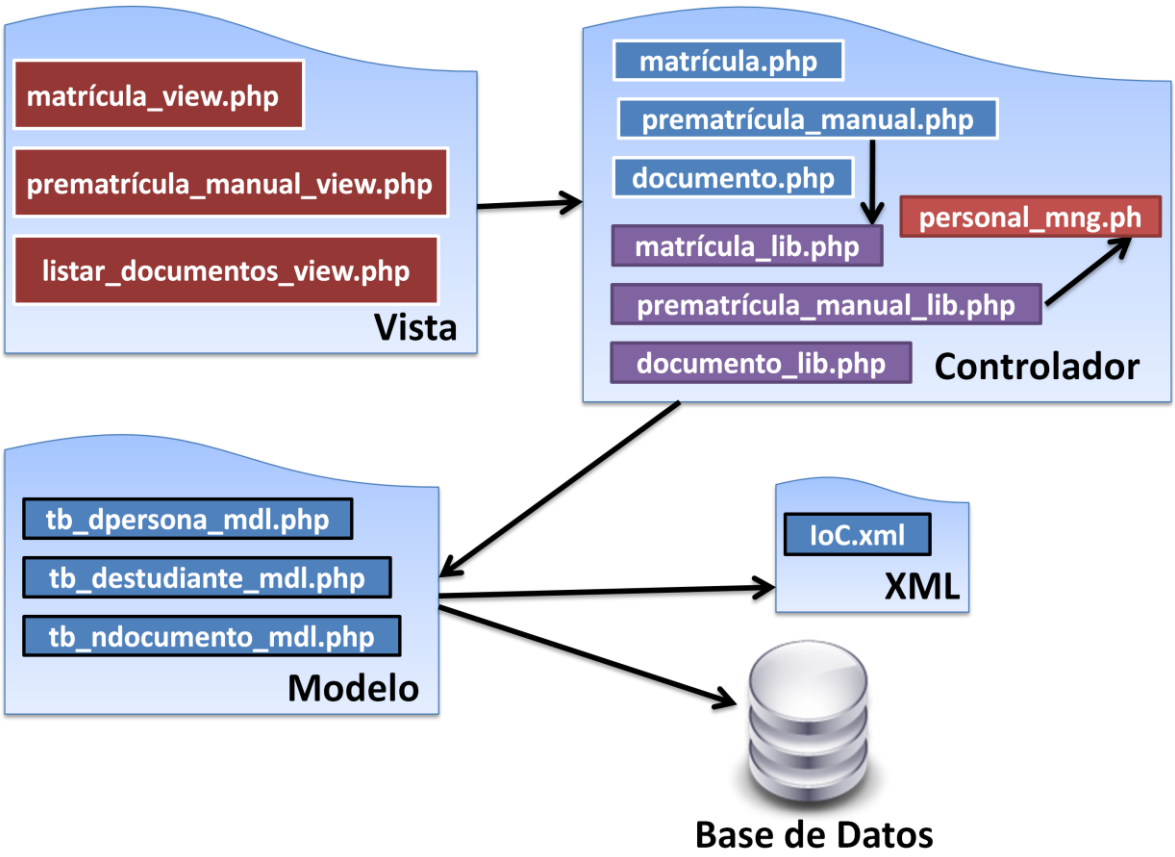


Fig. 12 Vista Arquitectónica

El **Modelo** representa la estructura de datos. Típicamente sus clases de modelo contendrán funciones que lo ayudarán a recuperar, insertar y actualizar información en la base de datos.

La **Vista** es la información que es presentada al usuario, normalmente, será una página web, pero en CodeIgniter, una vista también puede ser un fragmento de una página como un encabezado o un píe de página.

El **Controlador** sirve como un intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para procesar la petición HTTP y generar una página web.

El **IoC**[7] (*Inversion of Control*) permite la comunicación entre los demás módulos y subsistemas.

Las **librerías** gestionan los datos entre la controladora y la *manager*. Implementan las reglas del negocio. Por cada controladora se implementa una librería.

La ***manager*** engloba un conjunto de modelos que se usan en el mismo proceso. Es instanciada en la librería.

[7] Es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones.

2.7 Patrones de diseño «GRASP» «GOF»

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error. En el diseño del Sistema de Gestión Universitaria se tuvieron en cuenta los patrones de asignación de responsabilidades (GRASP) y los patrones GOF (*Gang-Of-Four*) que se verán a continuación.

Experto: el patrón experto en información define el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Con la utilización de este patrón se definió donde colocar en cada clase las funcionalidades que necesitan de esa información.

Creador: el patrón creador permite identificar quién debe ser el responsable de la instanciación de nuevos objetos o clases. Este patrón se utilizó para identificar qué clase A debe crear elementos de una clase B, apoyándose en que la clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B.

Alta Cohesión: este patrón define que la información que almacena una clase debe de ser coherente y está mayor medida relacionada con la clase. En el Sistema de Gestión Universitaria es necesario controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas, por esto, las clases que fueron identificadas con una gran cantidad de funcionalidades se dividieron en otras clases de manera que se repartiera equitativamente el peso de la complejidad, manteniendo además la coherencia de las clases.

Bajo Acoplamiento: Este patrón se utilizó con la idea de tener las clases lo menos ligadas entre sí posibles. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Controlador: el patrón controlador se utilizó para que sirviera como intermediario entre cada una de las capas, de forma tal que se garantice la comunicación entre los eventos externos del sistema en la capa de presentación y los componentes de la capa de negocio.

Patrón Solitario (*Singleton*): este patrón se utilizó para garantizar que una clase sólo tenga una única instancia y proporcionar un punto de acceso global a ella. Reduce el espacio de nombres, es

una mejora sobre las variables globales, además de que es usado debido a la necesidad de trabajar con el mismo objeto en distintos momentos.

Patrón Fachada (*Facade*): el patrón de diseño fachada se empleó para proveer de una interfaz unificada y sencilla, que haga de intermediaria entre el cliente y la interfaz o grupo de interfaces más complejas, permitiendo así una mayor flexibilidad en el desarrollo del sistema. Permite reducir la complejidad y minimizar las dependencias.

Patrón Observador (Observer): el patrón Observador define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes de forma automática. El objetivo de utilizar este patrón es desacoplar la clase de los objetos clientes del objeto, aumentando la modularidad del lenguaje.

2.8 Integración con otros módulos y subsistemas del Sistema de Gestión Universitaria.

En la actualidad casi ninguna aplicación se concibe como un sistema aislado, y el módulo Personal y Secretaría no es la excepción. Se contempla su integración al Sistema de Gestión Universitaria como plataforma única para la gestión de los procesos de personal en el subsistema Gestión de Pregrado. La comunicación se realizará a través del IoC.

Módulos

Diseño de Carrera: es el módulo encargado de permitir la configuración de una serie de elementos que van a conformar la carrera: los períodos lectivos, años académicos, curso académico, modalidades de estudio, disciplinas, asignaturas, entre otros. El módulo Personal y Secretaría se comunica con el módulo Diseño de Carrera para poder asignarle una carrera y un plan de estudio a un estudiante, esto se realiza durante el proceso de prematrícula. La comunicación se ejecuta haciendo uso del IoC, utilizando el método obtenerCarrera y obtenerPlanEstudio.

Registro y Control Docente: es el módulo encargado de crear los registros de asistencia y evaluaciones para un estudiante, además de los grupos docentes. El módulo Personal y Secretaría se vincula con este módulo cuando es necesario saber si un estudiante está listo para realizar el cambio de período lectivo y/o año académico durante los procesos de cierre de período y cierre de año. La comunicación se ejecuta haciendo uso del IoC, utilizando el método evaluaciónEstudiante.

Subsistemas

Núcleo del Sistema de Gestión Universitaria

Seguridad: módulo que garantiza la seguridad de todo el Sistema de Gestión Universitaria, permite autenticar usuarios, gestionar roles, usuarios, grupos de usuarios, dominios y modos de acceso.

Brinda un grupo de políticas de accesibilidad a las diferentes funcionalidades del Sistema de Gestión Universitaria en dependencia del nivel de autorización que presente un usuario determinado.

Traza: módulo que permite el control de incidencias dentro del Sistema de Gestión Universitaria. El sistema define cinco tipos de incidencias o trazas estas son: URL, acción, consultas, interacción y excepción. Para cada una de estas se definen atributos comunes a registrar como son: cuándo ocurrieron (fecha y hora), quién las realizó (usuario), dónde la realizó (IP desde el cual se accede a la aplicación) y la URL, estas son registradas desde diferentes lugares del sistema. El módulo Personal y Secretaría presenta definido en todas sus clases entidades los campos de fecha de creación y modificación de la entidad.

Estructura y Composición: módulo que garantiza la creación de todas las estructuras de la universidad, en el cual se crean los grupos administrativos cuando se realiza el proceso de prematrícula, además estos módulos interactúan cuando se lleva a cabo el proceso de movimiento con el cambio de período lectivo y de año académico. La comunicación se ejecuta haciendo uso del IoC, utilizando el método crearGrupoAdministrativo, obtenerPeríodoLectivo, obtenerAñoAcadémico.

Configuración: módulo encargado de la configuración del Sistema de Gestión Universitaria. Agrupa un conjunto de elementos comunes entre los demás módulos del sistema. Permite gestionar módulos o subsistemas, entidades, tipos de entidades, aéreas político-administrativas, activar/desactivar funcionalidad, configurar la conexión a la base de datos y al servidor de Gestión Documental Alfresco. Interactúa con el módulo Personal y Secretaría durante los procesos de prematrícula, matrícula y ratificación de matricula al utilizar las áreas político-administrativas y los tipos de áreas político-administrativas. La comunicación se ejecuta haciendo uso del IoC, utilizando el método obtenerÁreasPolíticoAdministrativas y obtenerTiposÁreasPolíticoAdministrativas.

Sistema de Gestión Documental

Sistema con el cual la comunicación se realiza en el proceso de prematrícula para la creación del Portafolio Digital del estudiante y del profesor, para guardar la Hoja de Matrícula, Prematrícula y de Ratificación de Matrícula, la certificación de estudios terminados, la baja del servicio y la certificación de notas de los estudiantes, de los profesores guarda la Ficha de Contrato. Cuando se realizan los diferentes movimientos tanto internos como externos que generan documentación entre ellos como bajas y traslados, la misma se archiva en el Portafolio Digital del estudiante. La comunicación se realiza haciendo uso de una librería que se le incluyó al marco de trabajo para la comunicación con el Sistema de Gestión Documental Alfresco.

FotoStore

Aplicación de escritorio que permite asociarle una foto a un estudiante y adicionarla al servidor de fotos de la universidad. Sistema con el cual la comunicación se realiza en el proceso de matrícula,

esta se ejecuta haciendo uso del servicio web identificación v5, utilizando el método obtenerFotoDadoIDExpediente.

2.9 Descripción de los algoritmos no triviales utilizados

Algoritmo de ubicación

Entradas

- Estudiante
- Facultades en la que se desea ubicar.
- Criterios (estos criterios se ordenan según la prioridad que tengan en orden alfabético).
  - Los criterios pueden ser: provincia, municipio, vía de ingreso, sexo, raza, índice académico(estos criterios se gestionan como nomencladores)
- Cantidad de estudiantes por grupo
- Cantidad de grupos por facultad

Pasos

1- Obtener los estudiantes ordenados por criterios (id_expediante).

Se van ordenando los estudiantes por los criterios seleccionados, estos criterios tienen una prioridad definida por el usuario, y además se ordenan alfabéticamente.

2- Calcular cantidad de grupo

Cantidad de grupos = Cantidad total de estudiantes / cantidad de estudiantes por grupo

Luego de este cálculo se obtiene la cantidad de grupos que van a existir.

3- Repartir los estudiantes por grupo (se realiza uno a uno hasta que todos los estudiantes tengan un grupo).

4- Repartir los grupos con estudiantes por facultad**.** (Estilo baraja)

Aquí se tendrá en cuenta la cantidad de grupos que se definió para cada facultad, en caso de que no se haya definido la cantidad de grupos en algunas facultades se irá realizando una distribución equitativa hasta que se terminen los grupos.

5- Hay que tener en cuenta para realizar el algoritmo algunos datos adicionales que se puedan introducir

- Para una facultad determinada se pueden definir algunos datos específicos que serían reglas adicionales que se deben tener en cuenta como: cantidad de grupos que se definan en la facultad, cantidad de estudiantes por grupo y cantidad total de estudiantes que se le asignará a la facultad.

Salidas

Se obtiene una propuesta de ubicación, quedando conformados los grupos administrativos.

En caso de no quedar conforme con la propuesta de ubicación se realiza nuevamente la distribución.

Reglas

- La prioridad de los criterios está dada según el orden en que el usuario los seleccione

2.10 Descripción de las nuevas clases u operaciones necesarias

Clases controladoras (ver Anexo 12)

Clases entidad (ver Anexo 13)

2.11 Estándares de codificación

"Un estándar es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad."[19]

Los estándares de codificación consisten en estilos de codificación a la hora de escribir el código. Estos se definen teniendo en cuenta el estilo personal del programador, las características propias del lenguaje de programación, los recursos que se utilizarán y el tipo de programa que se debe implementar.

En cada grupo de desarrollo se definen cuáles serán los aspectos a estandarizar y qué estilos se aplicarán a cada uno de ellos. El cumplimiento de estándares hace que todo el código lleve el sello personal del programador y en caso de ser varios los programadores se busca que el código parezca que ha sido implementado por la misma persona. De esta manera, se consigue mayor legibilidad y facilidad de mantenimiento. Los estándares deben responder además a acciones prácticas que acomoden al programador. A continuación se definirán los estándares usados para la actividad de implementación correspondiente a este trabajo.

Identación, llaves de apertura, cierre y tamaño de las líneas.

Usar una identación sin tabulaciones, con un equivalente a 4 espacios. El uso de las llaves "{ }" será en una nueva línea. Para mantener la legibilidad del código, la longitud de las líneas debe ser aproximadamente de 75-80 caracteres.

Ejemplo:

```
1 ....$a = $b;
2
3 ....function ejemplo()
4 ....{
5     ....//BI
6 ....}
```

Fig. 13 Identación, llaves de apertura y cierre

Conversión de nomenclatura

Las variables se rigen por la nomenclatura *CamelCase*[8], siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Ejemplo:

```
1 ....$variable
2 ....$variableNombreCompuesto
```

Fig. 14 Conversión de nomenclatura, variables

Las constantes son escritas en mayúsculas, con caracteres de subrayado "_" para separar palabras en caso de nombres compuestos.

Ejemplo:

```
1 ....define(CONSTANTE,valor);
2 ....define(CONSTANTE_COMPUESTO,valor);
```

Fig. 15 Conversión de nomenclatura, constantes

Los nombres de las clases siempre comienzan con mayúscula, en caso de nombre compuesto las palabras se separan con el carácter subrayado "_" y el resto en minúscula.

Ejemplo:

```
1 ....class Clase
2 ....{
3     ....//BI
4 ....}
5
6 ....class Clase_nombre_compuesto
7 ....{
8     ....//BI
9 ....}
```

Fig. 16 Conversión de nomenclatura, clases

Las funciones se rigen por la nomenclatura *CamelCase*, siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma.

[8] *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.

Ejemplo:

```
1 ....function funcion($parametro1,.$parametro2)
2 ....{
3     ....//BI
4 ....}
5
6 ....function funcionNombreCompuesto($parametro1,.$parametro2)
7 ....{
8     ....//BI
9 ....}
```

Fig. 17 Conversión de nomenclatura, funciones

Los ficheros siempre se escriben con minúscula y en caso de nombres compuestos se usa el carácter subrayado "_".

Vistas: intuitivo y relacionado con el formulario y/o vista que representa.

Modelos: con el mismo nombre de la clase que representa, que contiene en el nombre el sufijo _mdl o _base en caso de ser modelos o bases.

Librerías: con el mismo nombre de la clase que representa, que contiene en el nombre el sufijo _lib.

Controladoras: con el mismo nombre de la clase que representa.

***Manager*:** con el mismo nombre de la clase que representa que contiene en el nombre el sufijo _mng.

Estructuras de control

Se incluye un espacio entre las estructuras de control (if, for, foreach, while, switch) y los paréntesis. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos

Ejemplo:

```
1  ....if.(condicion)
2  ....{
3       ....//BI
4  ....}
5  ....elseif(condicion)
6  ....{
7       ....//BI
8  ....}
9  ....else
10 ....{
11      ....//BI
12 ....}
13
14 ....switch.(valor)
15 ....{
16      ....case valor1:
17          ....//BI para valor1
18      ....break;
19      ....case valor2:
20          ....//BI para valor2
21      ....break;
22      ....default:
23          ....//BI por defecto
```

Fig. 18 Estructuras de control

Si las condiciones son muy largas que sobrepasan el tamaño de la línea, éstas se dividen en varias líneas.

Ejemplo:

```
1 ....if.(condicion1
2 ....|| condicion2)
3 ....||(condicion3
4 ....&& condicion4))
5 ....{
6     ....//BI
7 ....}
```

Fig. 19 Estructuras de control, condiciones

En el mejor de los casos cuando la condición es muy extensa, se puede dividir en variables y compararlas dentro de la estructura de control.

Ejemplo:

```
1 ....$variableCondicion1 = condicion1 || condicion2;
2 ....$variableCondicion2 = condicion3 && condicion4;
3
4 ....if.($variableCondicion1 || $variableCondicion2)
5 ....{
6     ....//BI
7 ....}
```

Fig. 20 Estructuras de control, condición extensa

Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Para esto se debe cumplir con el siguiente bloque al principio de cada clase.

Clase:

```
1 /**
2 *Breve descripción de la clase
3 *
4 *PHP versión #
5 *
6 *@category Categoría de la clase implementada "Libreria,
7 * Controladora, Modelo"
8 *@package  Nombre del paquete o módulo al que pertence
9 *@author  Nombre y Apellidos del autor y  correo electrónico
10*/
```

Fig. 21 Documentación, clase

Funciones:

```
1 /**
2 *Breve descripción de la función
3 *
4 *@param tipo y nombre del paremetro(por cada parametro que
5 * recibe la función)
6 *@return  tipo que retorna
7 *@author  Nombre y Apellidos del autor y  correo electrónico
8 */
```

Fig. 22 Documentación, funciones

Buenas prácticas

Los valores booleanos y nulos siempre se escriben con mayúscula, usando una línea en blanco antes de las estructuras de control y definición de las funciones.

```
1 ....$variableBooleana = FALSE;
2 ....$variableNula = NULL;
3 ....
4 ....if(condicion)
5 ....{
6     ....//BI
7 ....}
```

Fig. 23 Buenas prácticas

2.12 Estándares de diseño

Para el diseño del módulo Personal y Secretaría se siguieron las pautas y el mismo diseño del Sistema de Gestión Universitaria trayendo como principal ventaja la uniformidad en la estructura de todas las páginas web, definiéndose un mapa de navegación y una taxonomía general para todos los módulos y subsistemas.

Diagramación general

Vista de Gestión de Procesos

1- área de subprocesos horizontales.
2- área del nombre de la aplicación.
3- área del buscador.
4- área de líneas de procesos.
5- área de módulos.
6- área de nombre de usuario.
7- área de traza.
8- área de línea de progreso.
9- área de contexto.
10- área de menú de módulos.

11- área de pie de página.

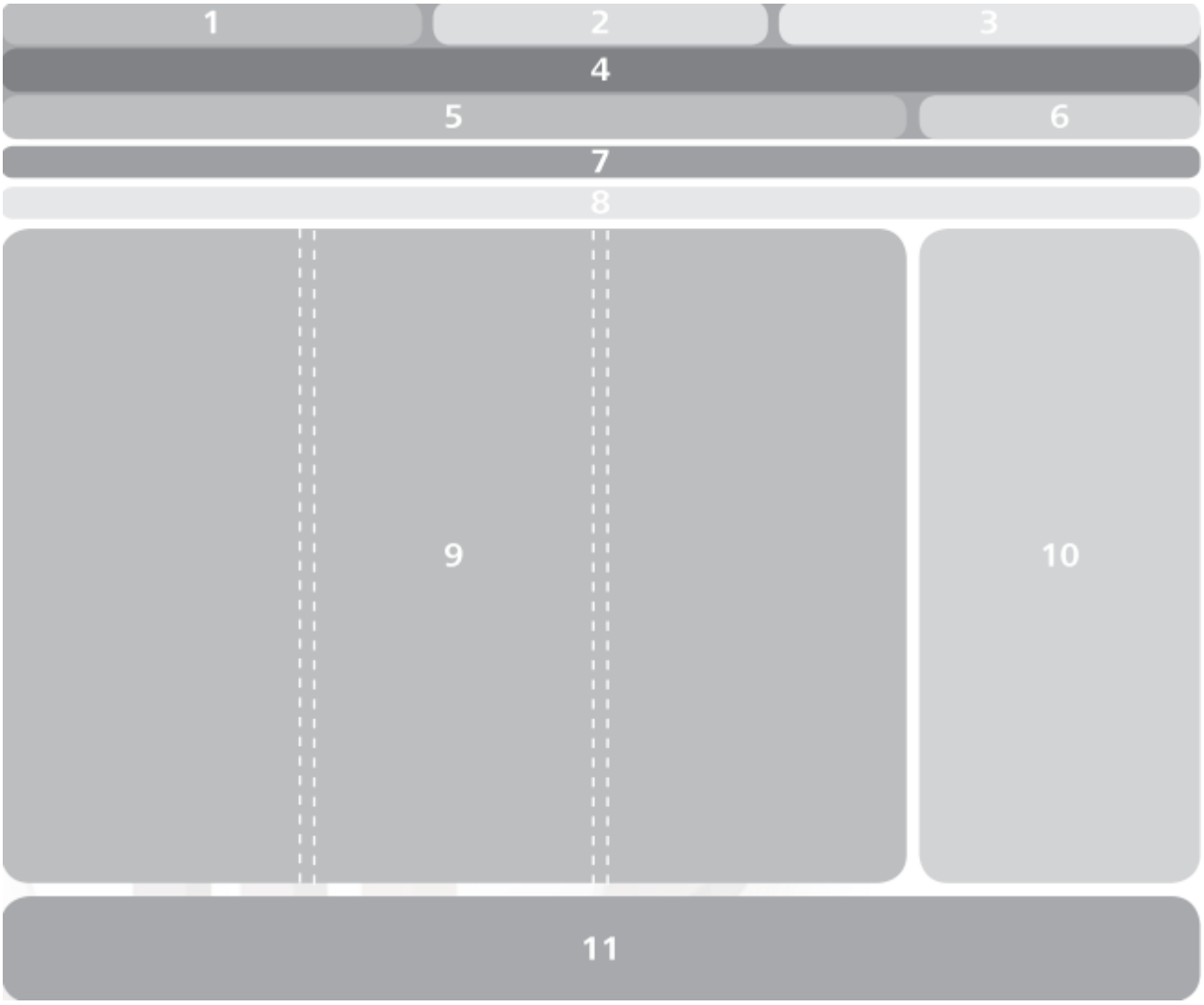


Fig. 24 Diagramación general, Vista de Gestión de Procesos

Dimensiones

Vista de Gestión de Procesos

*Todas las dimensiones están expresadas en píxeles (px).

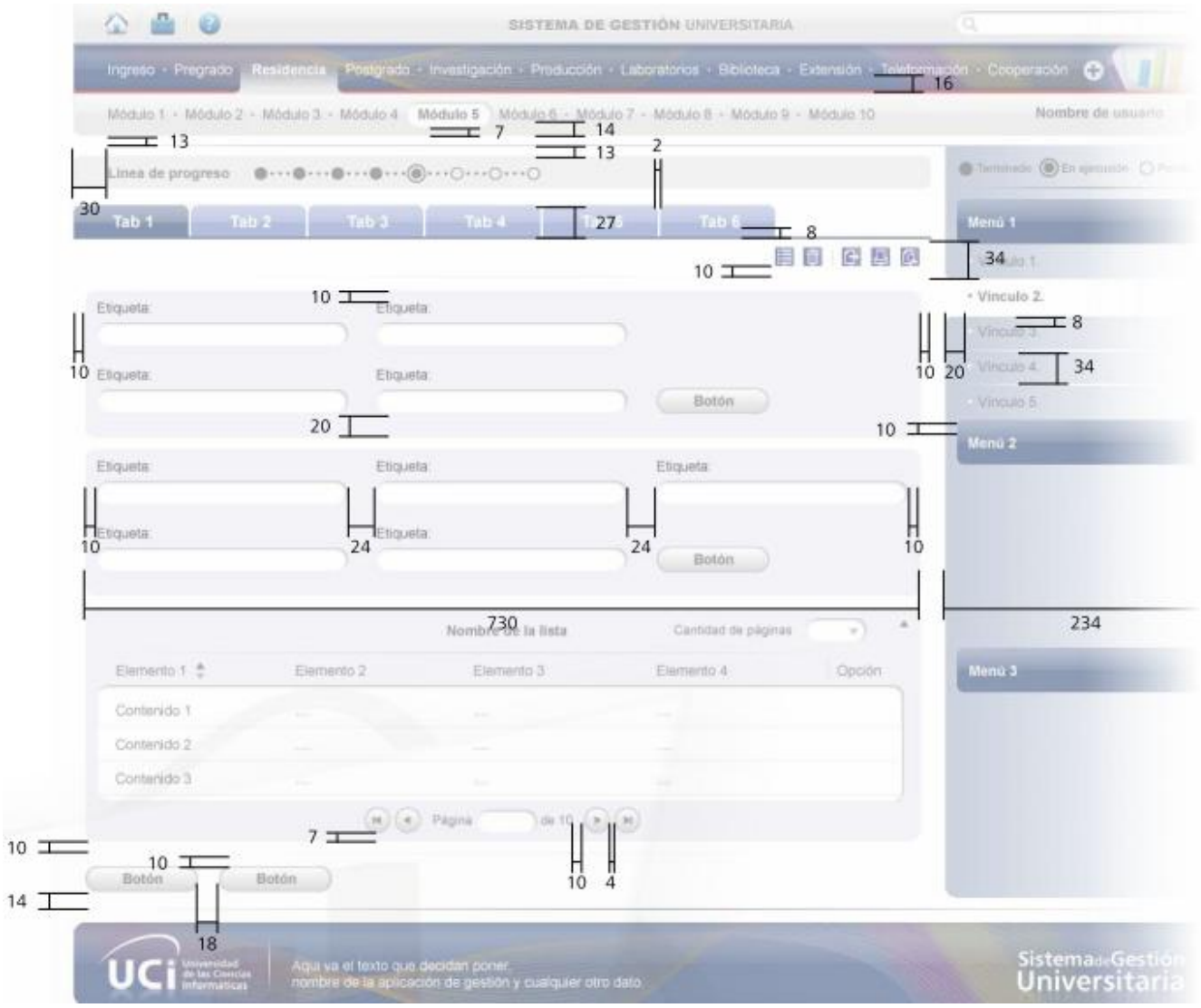


Fig. 25 Dimensiones, Vista de Gestión de Procesos

Pauta cromática

1- #CCCCCC
2- #233865
3- #FDFDFD
4- #FFFFFF
5- #3C3C3C
6- #555555
7- #6F6F6F
8- #E8E8F2

Fig. 26 Pauta cromática

Pauta tipográfica

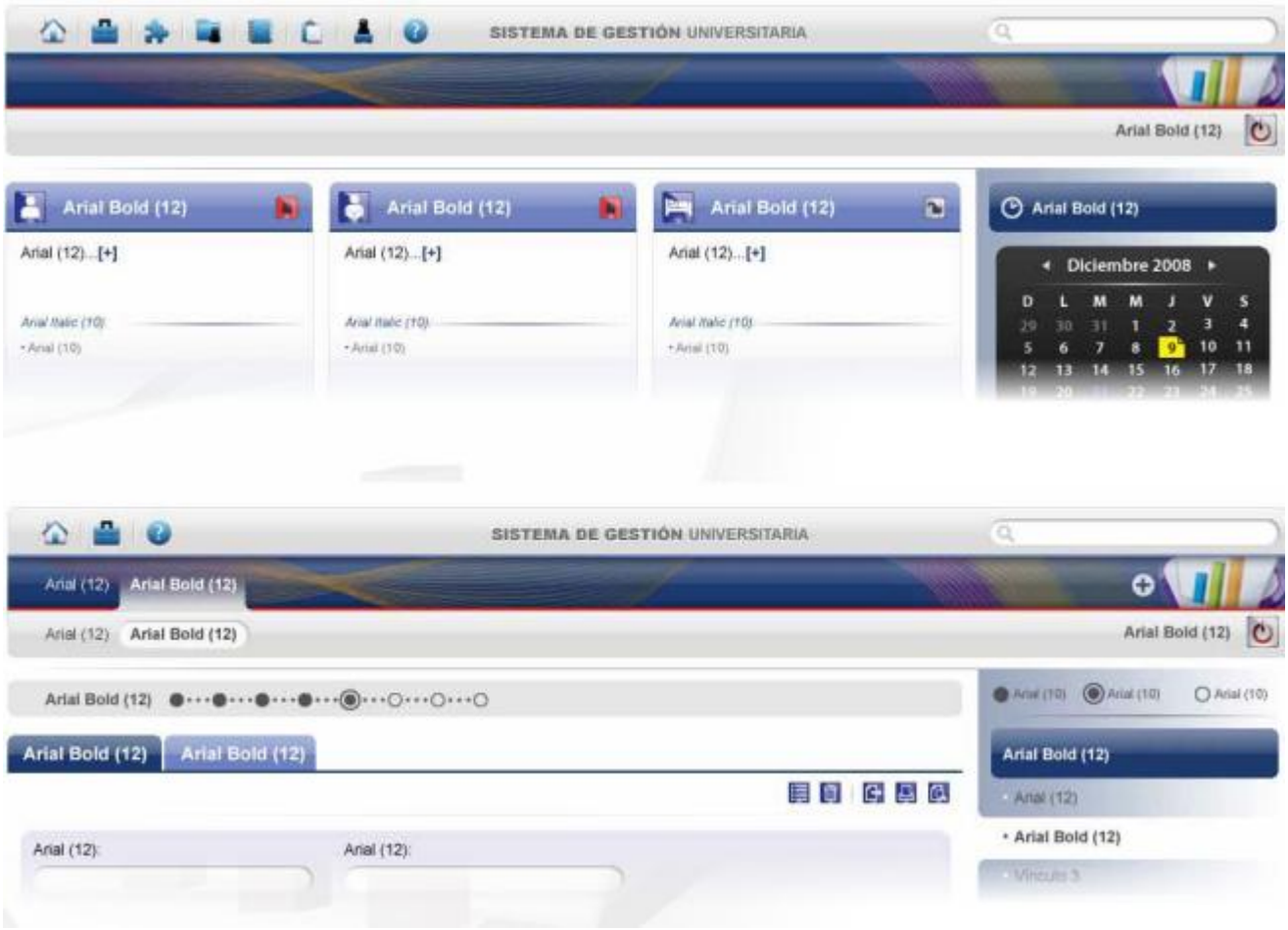


Fig. 27 Pauta tipográfica

2.13 Pautas para la implementación de las HU

El desarrollo en la metodología utilizada por el sistema de Gestión Universitaria está guiado por las HU.

Para la implementación de estas se definen algunos patrones a seguir:

- Regirse por el estándar de código definido para la implementación del proyecto.
- Todas las interfaces interactúan mediante AJAX con el servidor.
- Las acciones se dividen generalmente en dos métodos en las controladoras:
  - Para obtener los datos necesarios en caso de que lo necesite y mostrar la vista. El nombre usado para estos métodos se escribe en infinitivo, ejemplo: crear, listar.
  - Para obtener y validar los datos recibidos desde la vista, interactúa con la librería asociada y crea el mensaje que se envía a la vista. El nombre usado es el mismo infinitivo usado en el otro método con el o los elementos afectados en la acción ejemplo: crearGrupo,

modificarGrupo, asociarTrabajadorGrupo, en el caso de que no sea necesario mostrar una vista, éste se escribe en infinitivo ejemplo: eliminar, activar.

- En los métodos implementados en las clases modelo, las consultas no se hacen con código SQL directamente, se realiza utilizando el *active record*[9] de CodeIgniter.

Ejemplo:

```
1 ....$this->db->select('sq_esquema.tb_ddatos.id_datos');
3 ....$this->db->where ($key, $value);
4 ....$this->db->get ('sq_esquema.tb_ddatos')->result();
```

Fig. 28 Active record

- 5. Para la creación de las interfaces se utilizarán las funciones del asistente de formulario (*helper form*[10]) de CodeIgniter para garantizar homogeneidad en el html, ejemplo: *form_open*[11], *form_dropdown*[12], *form_input*[13].
- 6. Los mensajes de error en la vista se lanzan con un *throw Exception* ("mensaje").
- 7. No se utiliza el *.load* de jQuery pues el envío de los datos los hace por *get*[14] y para garantizar la seguridad se deben enviar por post[15] siempre, para esto hay que utilizar $.AJAX.
- 8. En los métodos de las controladoras que serán encuestados mediante AJAX, se utiliza *echo*[16] y no *die*[17] para mostrar los datos o la vista cargada, esto garantiza el buen funcionamiento de la programación orientada a objetos.
- 9. Todas las implementaciones se realizan en español.

2.14 Modelo de despliegue

El modelo de despliegue describe la distribución física del sistema, muestra la distribución de los componentes de software entre los distintos nodos de cómputo. Permite comprender la correspondencia entre la arquitectura de software y la de hardware.

[9] CodeIgniter usa una versión modificada del patrón de base de datos Active Record. Este patrón permite obtener, insertar y actualizar información en la base de datos con mínima codificación.
[10] El archivo asistente de formulario contiene funciones que ayudan en el trabajo con estos.
[11] Es una función que crea una etiqueta de formulario con la URL base construida desde su archivo de configuración.
[12] Es una función que permite crear un campo de menú desplegable estándar.
[13] Es una función que permite generar un campo de texto de entrada estándar.
[14] El método GET se utiliza para recuperar información identificada, también se puede utilizar para pasar una pequeña cantidad de información al servidor en forma de pares atributo-valor.
[15] El método POST se refiere normalmente a la invocación de procesos que generan datos que serán devueltos como respuesta a la petición.
[16] Es un comando para la impresión de texto en pantalla. Es utilizado en las terminales de los sistemas operativos y en ciertos lenguajes de programación como PHP.
[17] Es una función que envía como salida un mensaje y finaliza el script actual.

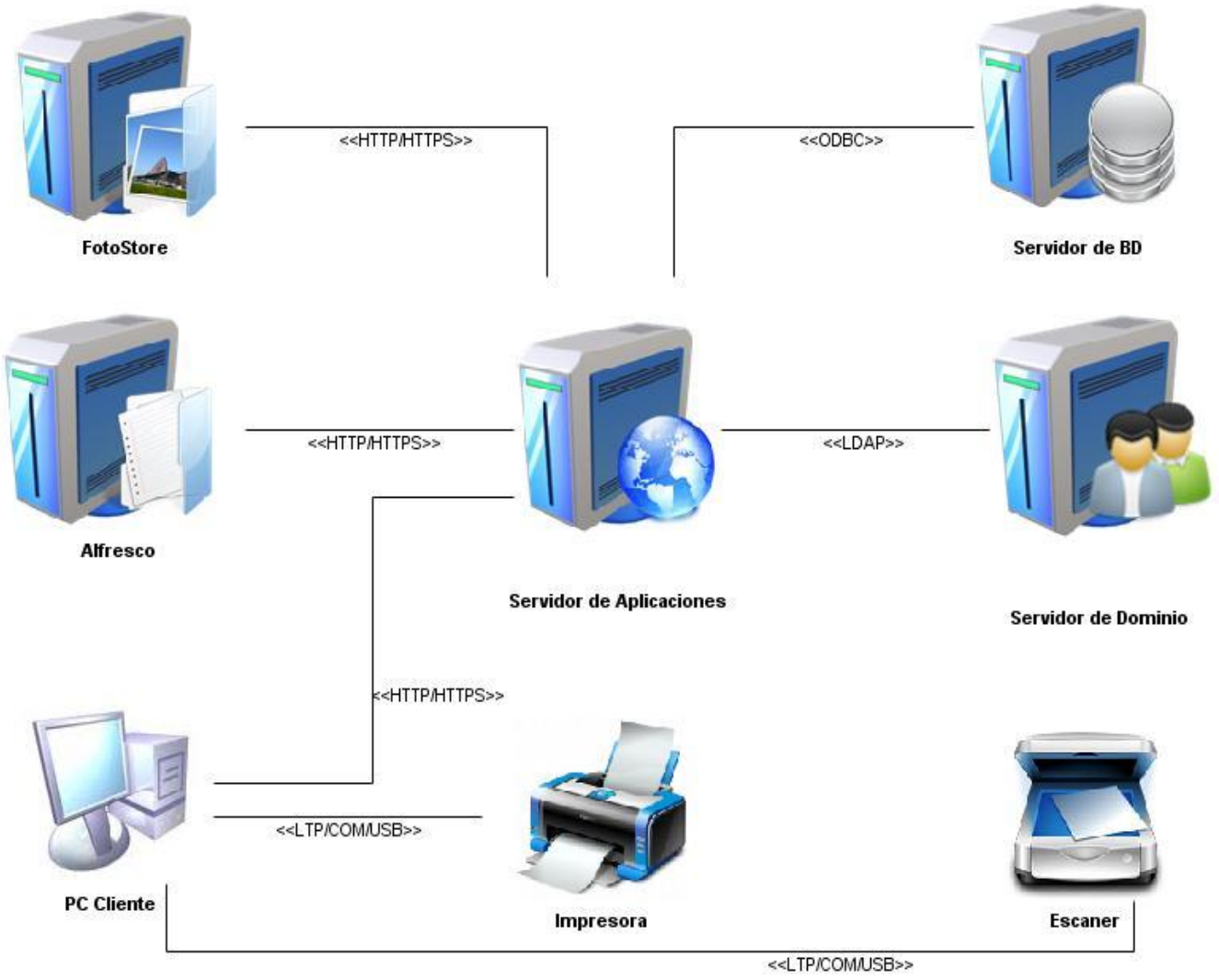


Fig. 29 Modelo de despliegue

2.15 Conclusiones

En este capítulo se brinda una breve descripción las funcionalidades implementadas en el módulo Personal y Secretaría, se critica al diseño propuesto por los analistas del sistema, se analizan las posibles implementaciones, componentes o módulos que son reutilizados y se describen los algoritmos no triviales empleados. Al finalizar queda implementado el sistema y descritas todas las clases utilizadas, cumpliendo con los objetivos específicos y gran parte de las tareas propuestas para la elaboración de este, dándole una respuesta satisfactoria al problema científico planteado.

Validación de la solución propuesta

3.1 Introducción

El desarrollo de software a raíz de la incapacidad humana de lograr una total perfección en las tareas con un cierto nivel de complejidad, debe de estar acompañado de una actividad que garantice su calidad. De esta forma, las pruebas de software son un elemento crítico para la garantía de la eficacia del mismo y representan una revisión final de las especificaciones del diseño y la codificación.

En el capítulo se presenta la validación de la solución propuesta a través de las diferentes pruebas de unidad y se describen los casos de prueba.

3.2 Proceso de pruebas

Las pruebas son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos específicos. Los resultados son observados, registrados y se realiza una evaluación del sistema o componente. Las pruebas verifican los resultados de la implementación del sistema.

En la unión de las metodologías Scrum-XP se anima a los desarrolladores a probar constantemente tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de este en el sistema y su detección.

La metodología divide a las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación. El objetivo es tener una forma de que el cliente conozca cuando una HU está lista. Los casos de prueba se deben escribir antes de comenzar la implementación, pero siempre que sea necesario se puede incluir uno nuevo. No existe restricción para su cantidad, se deben escribir casos de prueba hasta que quede claro el objetivo de la historia de usuario y se verifique que cumpla con todos los requerimientos.

3.3 Pruebas unitarias

Desarrolladas por los programadores a pequeña escala, las pruebas unitarias consisten en comprobaciones manuales o automatizadas realizadas para verificar que el código correspondiente a un módulo funciona de acuerdo con los requisitos del sistema. Las ventajas de este tipo de prueba son grandes, pues facilitan que ante un error el código sea cambiado por el programador para mejorar su

estructura. Permiten seguir realizando pruebas sobre los cambios para asegurarse de que no se han introducido nuevos errores y por consiguiente, reducen los efectos secundarios de estos.

3.4 Pruebas de Caja negra

La prueba de Caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software y permiten obtener un conjunto de condiciones de entrada que ejercitan completamente todos los requisitos funcionales del programa.

Objetivos: el objetivo de realizar este tipo de prueba al sistema es el de revelar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaz, rendimiento y errores de inicialización y terminación.

Alcance: el proceso de pruebas de Caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar el comportamiento de la unidad observable externamente y la calidad funcional.

Descripción: se llevan a cabo sobre la interfaz del software y son completamente indiferentes al comportamiento interno y a la estructura del programa. Los casos de prueba de Caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

La prueba de Caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.

3.5 Pruebas de Caja blanca

Estas pruebas se realizan sobre las funciones internas de un módulo en concreto. El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa.

En la siguiente tabla se representan las pruebas de Caja blanca:

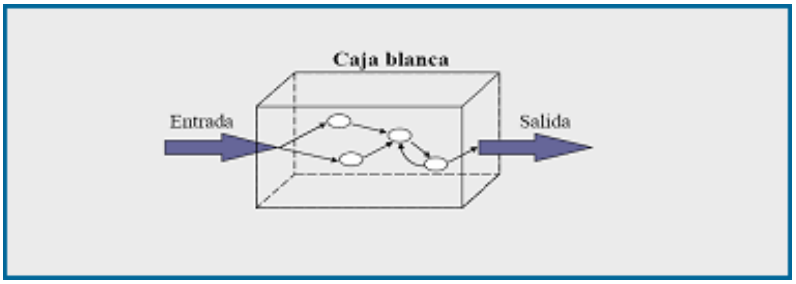


Fig. 30 Representación de Pruebas de Caja blanca

Entre algunas de las técnicas de prueba de Caja blanca podemos citar:

Prueba de Condición: es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

Prueba de Flujo de Datos: se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de Bucles: es una técnica de prueba de Caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Prueba del Camino Básico: "esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática." [19]

Los pasos que se siguen para aplicar esta técnica son:

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Un camino es aquel para el cual puede efectuarse una traza a través del código desde el comienzo del programa hasta el final del mismo. Hay que tener en cuenta que dos caminos son diferentes si se ejecutan distintas sentencias o las mismas pero en diferente orden.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones.

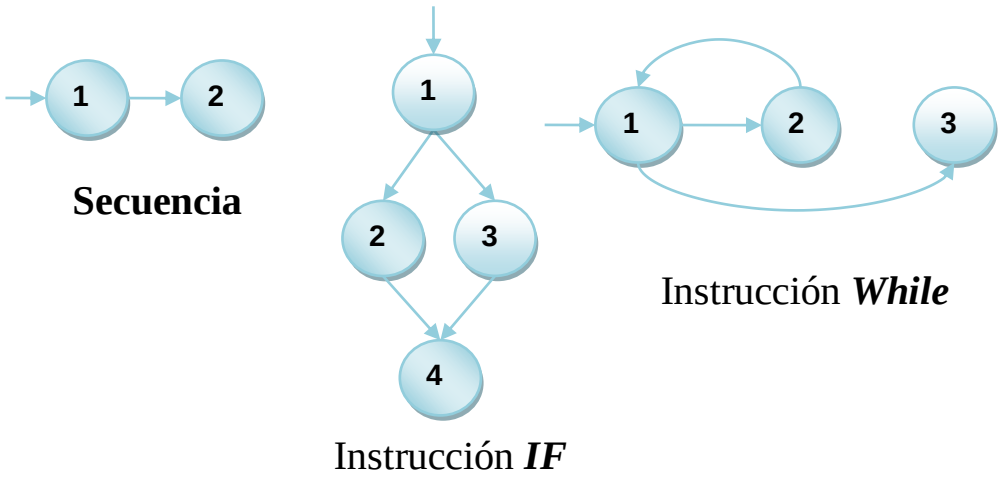


Fig. 31 Notación de grafos de flujo para las instrucciones: Secuenciales, *If* y *While*

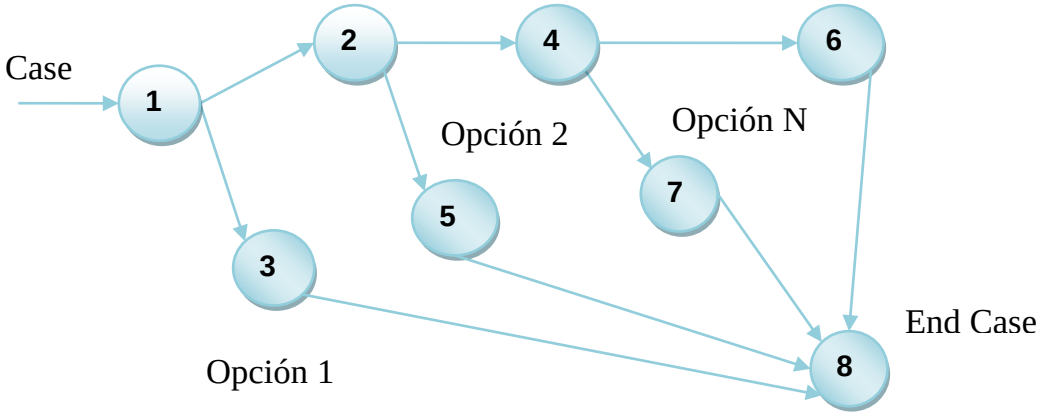


Fig. 32 Notación de grafos de flujos para la instrucción Case.

3.6 Casos de prueba de Caja negra

Escenario: ¨Crear Categoría de persona ¨

Condiciones de ejecución: el usuario debe haber sido autenticado en el sistema.

Sólo tiene acceso a realizar esta acción el administrador del sistema.

Flujo central: opción del menú “Categoría de persona” / Área de íconos flotantes “Crear”.

Tabla 1 Descripción del Caso de prueba Crear Categoría de persona

Clases válidas	Resultado esperado	Resultado de la prueba	Observaciones

Introducción correcta de Categoría de persona. “Categoría de persona: Estudiante” “Letra de identificación: E” “Descripción: Prueba” “Activo: True”	1. Tras la introducción de una Categoría de persona, si el proceso ha sido correcto, en la base de datos se registra la información de la nueva Categoría de persona: Estudiante. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	Satisfactorio	
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción incorrecta de Categoría de persona. “Categoría de persona: Pro·$%&/()=?123358” “Letra de identificación: 12” “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que la información de la Categoría de persona no es correcta. 2. La Categoría de persona no se registra en la base de datos.	Satisfactorio	
Introducción de Categoría de persona ya insertada.	1. Se muestra un mensaje indicando que la Categoría de persona ya ha sido registrada. 2. La Categoría de persona no se registra en la base de datos.	Satisfactorio	
Introducción de campos obligatorios de una Categoría de persona sin llenar. “Categoría de persona:* ”. “Letra de identificación:* ”. “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que no se han introducido todos los campos obligatorios. 2. La Categoría de persona no se registra en la base de datos.	Satisfactorio	
Evaluación de la prueba:	Satisfactoria		

Escenario: ¨Crear Especialidad militar ¨

Condiciones de ejecución: el usuario debe haber sido autenticado en el sistema. Sólo tienen acceso a realizar esta acción el administrador del sistema y la secretaria general.

Flujo central: opción del menú "Especialidades militares" / Área de iconos flotantes "Crear Especialidad militar"

Tabla 2 Descripción del Caso de prueba Crear Especialidad militar

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción correcta de Especialidad militar. "Especialidad militar: Piloto" "Descripción: Prueba" "Activo: True"	1. Tras la introducción de una Especialidad militar si el proceso ha sido correcto, en la base de datos se registra la información de la nueva Especialidad militar. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	Satisfactorio	
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción incorrecta de Especialidad militar. "Especialidad militar: $%&/()==??14587_9 " "Descripción: Prueba" "Activo: True"	1. Se muestra un mensaje indicando que la información de la Especialidad militar no es correcta. 2. La Especialidad militar no se registra en la base de datos.	Satisfactorio	
Introducción de Especialidad militar ya insertada.	1. Se muestra un mensaje indicando que la Especialidad militar ya ha sido registrada. 2. La Especialidad militar no se registra en la base de datos.	Satisfactorio	
Omitir campos obligatorios de Especialidad militar. "Especialidad militar:* "	1. Se muestra un mensaje indicando que no se han introducido todos los campos obligatorios.	Satisfactorio	

“Descripción: Prueba” “Activo: True”	2. La Especialidad militar no se registra en la base de datos.		
Evaluación de la prueba:	Satisfactoria		

Escenario: ¨Crear Atributo¨

Condiciones de ejecución: el usuario debe haber sido autenticado en el sistema. Sólo tienen acceso a realizar esta acción el administrador del sistema y la secretaria general.

Flujo central: opción del menú “Atributo” / Área de iconos flotantes “Crear”

Tabla 3 Descripción del Caso de prueba Crear Atributo

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción correcta de Atributo. “ Atributo: Ocupación” “Descripción: Prueba” “Activo: True”	1. Tras la introducción de un Atributo si el proceso ha sido correcto, en la base de datos se registra la información del nuevo Atributo: Ocupación. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	Satisfactorio	
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción incorrecta de Atributo. “Atributo: $%&)?179 “ “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que la información del Atributo no es correcta. 2. El Atributo no se registra en la base de datos.	Satisfactorio	
Introducción de Atributo ya insertado.	1. Se muestra un mensaje indicando que el Atributo ya ha sido registrado. 2. El Atributo no se registra en la base de datos.	Satisfactorio	

Omitir campos obligatorios de Atributo. “Atributo:* ”. “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que no se han introducido todos los campos obligatorios. 2. El Atributo no se registra en la base de datos.	Satisfactorio	
Evaluación de la prueba:	Satisfactoria		

Escenario: ¨Modificar Categoría de persona ¨

Condiciones de ejecución: el usuario debe haber sido autenticado en el sistema. Sólo tiene acceso a realizar esta acción el administrador del sistema.

Flujo central: opción del menú “Categoría de persona” / Área de iconos internos “Modificar”

Tabla 4 Descripción del Caso de prueba Modificar Categoría de persona

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción correcta de Categoría de persona. “Categoría de persona: Familia por Familiar” “Descripción: Prueba” “Activo: True”	1. Tras la introducción de la nueva Categoría de persona, cambio de Familia por Familiar, si el proceso ha sido correcto, en la base de datos cambia Familia por Familiar. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	Satisfactorio	
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción incorrecta de Categoría de persona. “Categoría de persona: Pro·$%&/()==??123358 “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que la información de la Categoría de persona no es correcta. 2. La Categoría de persona no se registra en la base de datos.	Satisfactorio	
Introducción de Categoría de persona ya insertada.	1. Se muestra un mensaje indicando que la Categoría de	Satisfactorio	

	persona ya ha sido registrada. 2. La Categoría de persona no se registra en la base de datos.		
Borrar campos obligatorios de una Categoría de persona. “Categoría de persona:* ” “Letra de identificación: * ” “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que no se han introducido todos los campos obligatorios. 2. La Categoría de persona no se registra en la base de datos.	Satisfactorio	
Evaluación de la prueba:	Satisfactoria		

Escenario: ¨Modificar Especialidad militar ¨

Condiciones de ejecución: el usuario debe haber sido autenticado en el sistema. Sólo tienen acceso a realizar esta acción el administrador del sistema y la secretaria general.

Flujo central: opción del menú “Especialidades militares” / Área de iconos internos “Modificar”.

Tabla 5 Descripción del Caso de prueba Modificar Especialidad militar

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción correcta de Especialidad militar “Especialidad militar: Piloto por Artillero” “Descripción: Prueba” “Activo: True”	1. Tras la introducción de la nueva Especialidad militar, Piloto por Artillero, si el proceso ha sido correcto, en la base de datos cambia Piloto por Artillero. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	Satisfactorio	
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción incorrecta de Especialidad militar. “Especialidad militar: PABC·$%&/()==??12369“ “Descripción: Prueba”	1. Se muestra un mensaje indicando que la información de la Especialidad militar no es correcta. 2. La Especialidad militar no se registra en la base de	Satisfactorio	

"Activo: True"	datos.		
Introducción de Especialidad militar ya insertada.	1. Se muestra un mensaje indicando que la Especialidad militar ya ha sido registrada. 2. La Especialidad militar no se registra en la base de datos.	Satisfactorio	
Borrar campos obligatorios de Especialidad militar "Especialidad militar:* ". "Descripción: Prueba" "Activo: True"	1. Se muestra un mensaje indicando que no se han introducido todos los campos obligatorios. 2. La Especialidad militar no se registra en la base de datos.	Satisfactorio	
Evaluación de la prueba:	Satisfactoria		

Escenario: ¨Modificar Atributo ¨

Condiciones de ejecución: el usuario debe haber sido autenticado en el sistema. Sólo tienen acceso a realizar esta acción el administrador del sistema y la secretaria general.

Flujo central: opción del menú "Atributo" / Área de iconos internos "Modificar"

Tabla 6 Descripción del Caso de prueba Modificar Atributo

Clases válidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción correcta de Atributo " Atributo: Ciudadano por Ciudadanía" "Descripción: Prueba" "Activo: True"	1. Tras la introducción del nuevo Atributo, cambio de Ciudadano por Ciudadanía, si el proceso ha sido correcto, en la base de datos cambia Ciudadano por Ciudadanía. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	Satisfactorio	
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**

Introducción incorrecta de Atributo. “Atributo: ciudadano por Pro·$%&/ ()=?12358“ “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que la información del Atributo no es correcta. 2. El Atributo no se registra en la base de datos.	Satisfactorio	
Introducción de Atributos ya insertados.	1. Se muestra un mensaje indicando que el Atributo ya ha sido registrado. 2. El Atributo no se registra en la base de datos.	Satisfactorio	
Borrar campos obligatorios de Atributo “Atributo:* ”. “Descripción: Prueba” “Activo: True”	1. Se muestra un mensaje indicando que no se han introducido todos los campos obligatorios. 2. El Atributo no se registra en la base de datos.	Satisfactorio	
Evaluación de la prueba:	Satisfactoria		

Escenario: ¨ Realizar matrícula¨

Condiciones de ejecución: el usuario debe haber sido autenticado en el sistema. Sólo tienen acceso a realizar esta acción el administrador del sistema y las personas que sean designadas como matriculadores.

Flujo central: opción del menú “Matrícula”

Tabla 7 Descripción del Caso de prueba Realizar matrícula

Clases válidas	Resultado esperado	Resultado de la prueba	Observaciones
Introducción correcta de datos de estudiante. “ Carnet de Identidad: 86090921885” “Número de serie del CI: 8702351” “Primer nombre: Eddy” “Segundo nombre: Felix” “Primer apellido:	1. Tras la introducción de los datos si el proceso ha sido correcto, en la base de datos se registra al estudiante como matriculado. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	Satisfactorio	

González” “Segundo apellido: Pupo”			
Clases inválidas	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Introducción incorrecta de datos de estudiante. “ Carnet de Identidad: ¡’+/¿?k” “Número de serie del CI: a!ªÇÇ’¿fk” “Primer nombre: 34()=¿” “Segundo nombre: 45%?” “Primer apellido: $%%43” “Segundo apellido: %$09”	1. Se muestra un mensaje indicando que los datos del estudiante no son correctos. 2. El estudiante no se registra en la base de datos.	Satisfactorio	
Introducción de estudiante ya insertado.	1. Se muestra un mensaje indicando que el estudiante ya ha sido registrado. 2. El estudiante no se registra en la base de datos	Satisfactorio	
Omitir campos obligatorios. “ Carnet de Identidad: ” “Número de serie del CI: ” “Primer nombre: ” “Segundo nombre: ” “Primer apellido: ” “Segundo apellido: ”	1. Se muestra un mensaje indicando que no se han introducido todos los campos obligatorios. 2. El estudiante no se registra en la base de datos.	Satisfactorio	
Evaluación de la prueba:	Satisfactoria		

3.7 Casos de pruebas de Caja blanca

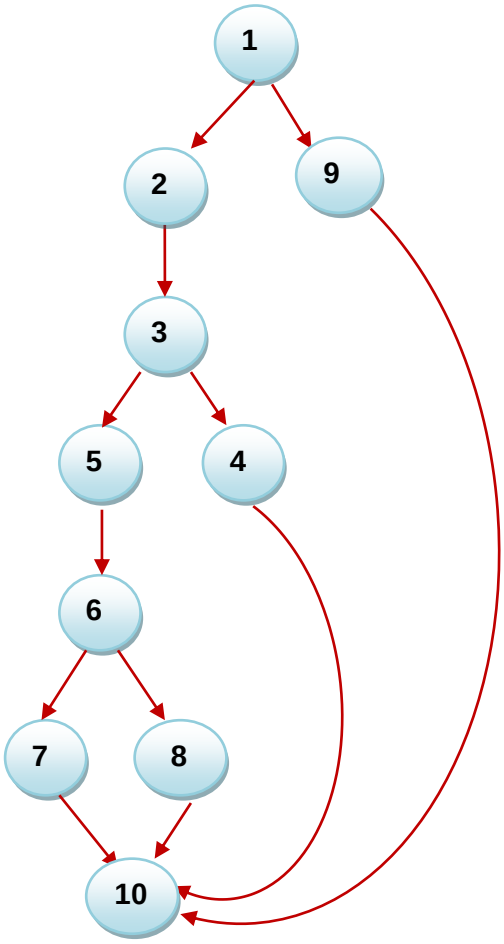
A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método crearDocumento ().

```
public function crearDocumento()
{
    if ($this->input->is_post_back(array('nombre_documento','id_documento_alfresco','persona','movimiento'))) 1
    {
        $post_vars = $this->input->all_post(true); 2
        if($this->documento_lib->validarDocumento($post_vars['nombre_documento']) == true ) 3
            {
                $response['succes'] = false; 4
                $response['message'] = 'El elemento ya se encuentra'; 4

            }
            else 5
            {
                if($this->documento_lib->registrarDocumento($post_vars)!=false) 6
                    {
                        $response['succes'] = true; 7
                        $response['message'] = 'El elemento ha sido creado satifactoriamente'; 7
                    }
                    else 8
                    {
                        $response['succes'] = false; 8
                        $response['message'] = 'Ocurrió un error durante la operaci&oacute;n'; 8
                    }
            }

    }
    else 9
    {
        $response['succes'] = false; 9
        $response['message'] = 'Uno o varios elementos se han introducido de forma incorrecta'; 9
    }
    die( json_encode($response) ) ; 10
}
```

Fig. 33 Representación del algoritmo crearDocumento ()



Fórmulas para calcular complejidad ciclomática:

V (G) = (A – N) + 2

V (G) = (12 - 10) + 2

V (G) = 4

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

Se puede usar también:

V (G) = P + 1

V (G) = 3 + 1

V (G) = 4

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = R

V (G) = 4

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede plantear que la complejidad ciclomática del código es de 4, lo que significa que existen a lo sumo cuatro posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.
Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo.
En la siguiente tabla se muestran los caminos básicos.

Tabla 8 Caminos básicos

Número	Camino básico
1	1,9,10
2	1,2,3,4,10
3	1,2,3,5,6,7,10
4	1,2,3,5,6,8,10

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.
Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:

- **Descripción**: se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución**: se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** se muestran los parámetros que entran al procedimiento
- **Resultados Esperados:** se expone el resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico 1:
Camino 1: [1–9–10]
Descripción: los datos de entrada serán atributos del documento a crear
Condición de ejecución: $id_documento_alfresco = null.
Entrada: $nombre_documento = “Hoja de Matrícula”, $persona = “estudiante”, $movimiento = “matrícula”, $id_documento_alfresco = null
Resultados esperados: se muestra un mensaje indicando que “Uno o varios elementos se han introducido de forma incorrecta”.

Caso de prueba para el camino básico 2:

Camino 2: [1–2–3–4–10]

Descripción: los datos de entrada serán atributos del documento a crear donde el $nombre_documento ya se encuentra en la base de datos.

Condición de ejecución: $nombre_documento = “Hoja de Matrícula”.

Entrada: $persona = “estudiante”, $movimiento = “matrícula”, $id_documento_alfresco = 15, $nombre_documento = “Hoja de Matrícula”

Resultados esperados: se muestra un mensaje indicando que “El documento ya se encuentra”.

Caso de prueba para el camino básico 3:

Camino 2: [1–2–3–5–6–7–10]

Descripción: Los datos de entrada serán atributos del documento a crear.

Condición de ejecución: Los valores de los atributos son válidos.

Entrada: $persona = “estudiante”, $movimiento = “matrícula”, $id_documento_alfresco = 15, $nombre_documento = “Hoja de Prematrícula”.

Resultados esperados: se muestra un mensaje indicando que “El elemento ha sido creado satisfactoriamente”.

Caso de prueba para el camino básico 4:

Camino 2: [1–2–3–5–6–8–10]

Descripción: Los datos de entrada serán atributos del documento a crear. Donde los datos de los atributos serán valores válidos.

Condición de ejecución: Fallo de conexión con la base de datos

Entrada: $persona = “estudiante”, $movimiento = “matrícula”, $id_documento_alfresco = 15, $nombre_documento = “Hoja de Prematrícula”.

Resultados esperados: se muestra un mensaje indicando que “Ocurrió un error durante la operación”.

3.8 Conclusiones

En el proceso de desarrollo de software las pruebas desempeñan un papel primordial ya que proporcionan al cliente conformidad y seguridad ante el producto final. En el capítulo se abordaron los métodos de pruebas usados durante el desarrollo de la solución y se describieron algunas de las pruebas de aceptación a las que fue sometida la aplicación, con la finalidad de asegurar que esta cumpla con los requerimientos funcionales. En todos los casos, las mismas devolvieron resultados satisfactorios, asegurando la fiabilidad y eficiencia del módulo Personal y Secretaría.

Conclusiones

Una vez finalizada la investigación se concluye que:

- La evaluación del estado del arte de las soluciones de software para la gestión académica en el ámbito nacional e internacional permitió identificar que, de los implementados en el mundo la mayor parte está desarrollada en software propietario y se necesita licencia para su uso, mantenimiento y actualización. De los implementados en la UCI y en el resto del país se puede señalar, que en la actualidad debido a lo complejo de la gestión de personal y de los procesos docentes, no satisfacen las necesidades del centro. Concluyendo que ninguno presenta las características ideales para ser utilizado en la universidad.
- El estudio de las herramientas usadas para la elaboración de la solución propuesta permitió conocer las ventajas y desventajas del uso de herramientas libres como PHP y los marcos de trabajo CodeIgniter y JQuery, como gestor de bases de datos, PostGreSQL y como IDE NetBeans 6.8, demostrando la factibilidad de su uso para la implementación de la propuesta de solución.
- El análisis de la modelación del módulo Personal y Secretaría permitió un mejor entendimiento de los procesos realizados durante la gestión de personal en la universidad, facilitando en gran medida la implementación de una aplicación funcional que gestionará estos procesos dentro del subsistema Gestión de Pregrado.
- La implementación del módulo Personal y Secretaría. Permitió resolver las deficiencias de la antigua versión e implementar nuevas funcionalidades, obteniendo así un sistema funcional capaz de gestionar los procesos de personal en el subsistema Gestión de Pregrado.
- La realización de las pruebas unitarias al sistema permitió garantizar que los requerimientos fueron cumplidos y que el sistema es estable.

Recomendaciones

Una vez vencidos los objetivos de esta investigación y tomando en consideración las experiencias obtenidas a lo largo de su desarrollo, se recomienda el Centro de Informatización Universitaria:

- Continuar trabajando en el desarrollo del sistema para culminar la implementación de los procesos de:
    - Movimientos
    - Cierres
    - Solicitud de ajuste y homologación
    - Registro de profesores
- Continuar el estudio de los procesos de gestión de personal en vista a añadir nuevas funcionalidades.

Referencias bibliográficas

1. INSTITUTION, T. B. S. ¿Qué son los sistemas de gestión? [En línea] 2008. [Citado el: 21 de enero] http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion

2. ¿Qué es el proyecto ALBA? [En línea] 2007. [Citado el: 21 de enero] http://www.proyectoalba.com.ar

3. SAAVEDRA, J. Lenguajes de Programación [En línea] 2008 [Citado el: 16 de enero de 2009] http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/

4. ÁLVAREZ, M. Breve historia de PHP. [En línea] 2007. [Citado el: 16 de enero de 2009] http://www.desarrolloweb.com

5. NETPECOS. PostGreSQL vs. MySQL [En línea] 2008 [Citado el: 17 de enero de 2009] http://www.netpecos.org/docs/mysql_postgres/x15.html

6. Introducción a PostgreSQL [En línea] 2008 [Citado el: 17 de enero de 2009] http://www.postregresql.org

7. Visión práctica de los framework [En línea] 2008 [Citado el: 17 de enero de 2009] http://soaagenda.com

8. ÁNGEL, M. Manual de CodeIgniter, [En línea] 2008 [Citado el: 16 de enero de 2010] http://www.desarrolloweb.com/articulos/codeigniter.html

9. Álvarez, Miguel. Introducción a jQuery, [En línea] 2008 [Citado el: 23 de enero de 2009] http://www.desarrolloweb.com

10. Garrett, Jesse. AJAX: A New Approach to Web Applications [Citado el: 23 de enero de 2010] http://adaptivepath.com/aboutus/jjg.php

11. Eguíluz, Javier. Introducción a AJAX, [En línea] 2009 [Citado el: 26 de enero de 2010] http://librosweb.es/ajax/index.html

12. GARCÍA, M. Ventajas del HTML, [En línea] 2009 [Citado el: 26 de enero de 2010] http://www.ugr.es/pages/investigacion/

13. Introducción a XML, 2008 [En línea] 2009 [Citado el: 26 de enero de 2010] http://sharekan.con.ar

14. WIKIPEDIA. Entorno de desarrollo integrado, [En línea] 2009 [Citado el: 28 de enero de 2010] http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

15. Parkinson, Jim. Vicepresidente de Productos y Programas de Desarrollo de Sun Microsystems. "SUN MICROSYSTEMS LANZA NETBEANS IDE 6.8" [Citado el: 23 de enero de 2009.] http://es.sun.com/sunnews/press/2009/20091214.jsp

16. Carlson, Derek. Vicepresidente de Desarrollo de Herramientas de Microchip Technology. "SUN MICROSYSTEMS LANZA NETBEANS IDE 6.8" [Citado el: 23 de enero de 2009.] http://es.sun.com/sunnews/press/2009/20091214.jsp

17. Real Academia Española [Citado el: 28 de enero de 2010] http://www.rae.es/rae.html

18. Garrett, Jon. 2010. Patrones Arquitectónicos [Citado el: 28 de enero de 2010] http://www.desarrolloweb.com.es

19. Márquez A, Yenni. 2008. Procedimiento general de pruebas de Caja Blanca aplicando la técnica del Camino Básico. [Documento] Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

Bibliografía

ACOSTA, J. H. B. B. Y. E. Q. Gestión de Riesgos para el proyecto de Gestión Académica Akademos V2.0. La Habana, UCI, 2009. 96. p.

ACRALYS FERRIOL ORTIZ, G. S. M. Análisis y Diseño del Módulo Registro y Control Docente para Akademos v2.0. La Habana, UCI, 2008. 212. p.

ÁNGEL, M. Manual de CodeIgniter, 2008. [2010] Disponible en: http://www.desarrolloweb.com/articulos/codeigniter.html

ALBA, E. D. T. P. Manual de Usuario, 2005. [2009]. Disponible en: http://www.proyectoalba.com.ar ¿Qué es el proyecto ALBA? , 2005. [2009]. Disponible en: http://www.proyectoalba.com.ar

ÁLVAREZ, M. Breve historia de PHP, 2007. [2009]. Disponible en: http://www.desarrolloweb.com Introducción a jQuery, 2010. [2010]. Disponible en: http://www.desarrolloweb.com

CENIA, Estándar de código SGU. La Habana, UCI, 2009.

CENIA, Manual de Directrices de experiencia de usuario. La Habana, UCI, 2010.

CENIA, Informe final. La Habana, UCI, 2009.

COMUNICACIONES, M. D. L. I. Y. L. SIGENU, Sistema de Gestión de la Nueva Universidad, 2007. [2010]. Disponible en: http://www.informaticahabana.cu/

CÓRDOBA, U. D. Sistema Integral de Gestión Académica, 2008. [2010]. Disponible en: https://www.gestion.uco.es/gestion/aplicaciones/siga

COSTA, G. I. Análisis de la calidad del proceso de desarrollo de software en el proyecto Akademos. UCI, La Habana, 2007. 90. p.

DARIEL ENRIQUE TAMAYO PALMA, Y. A. P. Análisis y Diseño de los procesos de Gestión de Carreras para el sistema Akademos v2.0. La Habana, UCI, 2009. 233. p.

EGUÍLUZ, J. Introducción a AJAX, 2008. [2010]. Disponible en: http://librosweb.es/ajax/index.html

ELISABEL PÉREZ URBAY, N. M. H. Análisis y Diseño del Módulo Estudiantes para Akademos. La Habana, UCI, 2007. 140. p.

FRANK BENAVIDES DALMENDRAY, D. C. R. Estudio de alternativas para la migración del Sistema Automatizado para la Gestión Académica de la Universidad de las Ciencias Informáticas – “Akademos” a Software Libre. La Habana, UCI, 2007. 97. p.

FOWLER, M. Inversion of Control, 2010 [2010]. Disponible en: http://martinfowler.com /InversionOfControl.html

GARCÍA, M. Ventajas del HTML, 2009. [2009]. Disponible en: http://www.ugr.es/pages/investigacion/

GARRETT, J. Ajax: A New Approach to Web Applications, 2005. [2009]. Disponible en: http://adaptivepath.com/aboutus/jjg.php

HERNÁNDEZ, A. S. Arquitectura para AKADEMOS v2.0. La Habana, UCI, 2008. 103. p.

INALVYS HERNÁNDEZ MANSO, A. P. P. Análisis y diseño del módulo Profesor del Sistema de Gestión Académica Akademos. La Habana, UCI, 2008. 108. p.

INSTITUTION, T. B. S. ¿Qué son los sistemas de gestión?, 2008. [2009]. Disponible en: http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion

Introducción a XML, 2008. [2009]. Disponible en: http://sharekan.con.ar

LÁZARO JULIO PEDROSO RAMOS, R. L. R. D. Gestión de comunicaciones para el proyecto Akademos. La Habana, UCI, 2009. 118. p.

LIDESOFT. Búho - Sistema de Gestión Académica, 2008. [2009]. Disponible en: http://www.lidesoft.com/productos/software-educativo.html

MARILIENNYS GLORIA MESQUIDA PEÑA, E. B. U. Propuesta de diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0. La Habana, UCI, 2009. 105. p.

MESEGUER, P. La nueva gestión académica, 2009. [2009]. Disponible en: http://www.microsoft.com/spain/enterprise/perspectivas/numero_7

MIREYDIS ORELLANA LEÓN, Y. A. B. Análisis y Diseño de los Procesos de Gestión de Personal para Akademos v2.0. La Habana, UCI, 2009. p.

NETPECOS. PostGreSQL vs. MySQL, 2008. [2010]. Disponible en: http://www.netpecos.org/docs/mysql_postgres/x15.html

OLIVIA RODRÍGUEZ ABRIL, D. S. A. Análisis y diseño del Módulo Reportes del Sistema de Gestión Académica Akademos. La Habana, UCI, 2007. 118. p.

ORLANDO BARRIEL VICTORIAL, G. S. M. Seguridad en AKADEMOS 2.0. La Habana, UCI, 2008. 84.

RODRÍGUEZ, C. M. Las esferas socioculturales del Software Libre, 2010. [2010]. Disponible en: http://www.derecho-internet.org/node/293

ROSELL PUPO POLANCO, Y. G. P. Implementación del componente réplica de base de datos para Akademos v2.0. La Habana, UCI, 2009. 130. p.

SAAVEDRA, J. Lenguajes de Programación, 2008. [2010]. Disponible en: http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/

SALGADO, B. SUN MICROSYSTEMS lanza NETBEANS IDE 6.8, 2009. [2009]. Disponible en: http://es.sun.com/sunnews/press/2009/20091214.jsp

VALDIVIA, Y. C. Las competencias, punto de partida para el rol de líder en el proyecto Akademos. La Habana, UCI, 2007. 105. p.

WIKIPEDIA. Entorno de desarrollo integrado, 2010. [2010]. Disponible en: http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

YANIRIS ROIG MOREJÓN, R. L. B. Análisis y Diseño del módulo Gestión de Tesis. La Habana, UCI, 2008. 111. p.

YOSVALDY FERNÁNDEZ FERNÁNDEZ, O. J. M. R. Análisis y diseño del módulo de Postgrado en Akademos v2.0. La Habana, UCI, 2008. 98. p.