Universidad de las Ciencias Informáticas

Facultad 1

GENERADOR DINÁMICO DE REPORTES PARA LA GESTIÓN ACADÉMICA DE PREGRADO

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

Autores: Rodain Terrer Molina
Yaimy Alfonso Álvarez

Tutores: Ing. Alexander Rodríguez Mompié
Ing. Norges Sanchez Tumbarell
Ing. Lianet Liben Martínez

La Habana, Junio de 2012
"Año 54 de la Revolución"

Declaración de autoría

MINISTERIO DE LA EDUCACIÓN SUPERIOR

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Autorizamos a dicho centro para que haga el uso que estime pertinente con este trabajo.

Para que así pueda constar firmamos el presente documento a los ___ días del mes de _____ del año _____.

Rodain Terrer Molina

_____

Firma del autor

Yaimy Alfonso Álvarez

_____

Firma del autor

Ing. Alexander Rodríguez Mompié

_____

Firma del tutor

Ing. Norges Sánchez Tumbarell

_____

Firma del tutor

Ing. Lianet Liben Martínez

_____

Firma del tutor

Datos de Contacto

- **Autores**

Nombre y apellidos: Rodain Terrer Molina.
Fecha de nacimiento: 21 de Septiembre de 1988
Dirección: Calle 5ta A e/ I y J No.307, Reparto Jesús Lores, Imías, Guantánamo.
E-mail: rterrer@estudiantes.uci.cu

Nombre y apellidos: Yaimy Alfonso Álvarez.
Fecha de nacimiento: 2 de Enero de 1989
Dirección: Ave 93B e/ 106 y 108 No. 10 622, Reparto Enrique Hart, Alquizar, Artemisa.
E-mail: yaalvarez@estudiantes.uci.cu

- **Tutores.**

Ing. Alexander Rodríguez Mompié. Graduado con título de oro en la Universidad de las Ciencias Informáticas en la graduación del curso correspondiente a los años 2006-2007. Durante su trayectoria como profesor ha impartido clases de las asignaturas Base de Datos I y Base de Datos II. Pertenece al Centro de Informatización Universitaria (CENIA) donde se desempeña como jefe del departamento Gestión Universitaria y posee la categoría docente de Instructor. Correo electrónico: arodriguezm@uci.cu.

Ing. Norges Sánchez Tumbarell. Graduado en la Universidad de las Ciencias Informáticas en la graduación del curso correspondiente a los años 2006-2007. Durante su trayectoria como profesor ha impartido clases de las asignaturas Matemática III y Matemática IV. Pertenece al Centro de Informatización Universitaria (CENIA) donde se desempeña como líder del proyecto Pregrado y posee la categoría docente de Instructor. Correo electrónico: nsanchez@uci.cu.

Ing. Lianet Liben Martínez. Graduado en la Universidad de las Ciencias Informáticas en la graduación del curso correspondiente a los años 2006-2007. Pertenece al Centro de Informatización Universitaria (CENIA) donde se desempeña como analista del proyecto Pregrado y posee la categoría docente de Instructor. Correo electrónico: lliben@uci.cu.

Dedicatoria

A mis padres…

Porque me dieron la vida, porque creyeron en mí y me sacaron adelante, por haber estado ahí siempre que los necesité, cuidándome y dándome fortaleza para seguir. Los quiero con todo mi corazón y este trabajo que me llevó un año hacerlo es para ustedes, por ser el más chico de sus hijos aquí está lo que ustedes me brindaron, solamente les estoy devolviendo lo que ustedes me dieron en un principio.

A mi hermano Ruber...

Por ser el mejor amigo del mundo, por estar conmigo siempre y por apoyarme. También te adoro.

A ti Yesenia…

A pesar de que no estás aquí ahora en estos momentos conmigo, sé que tu alma si lo está y porque esto es parte de un sueño que una vez compartimos juntos. Nunca te olvidaré…

A todos aquellos seres angelicales que son parte de mi universo y que me alientan a vivir cada día.

A todos ustedes por ser las personas que más amo en el mundo, espero no defraudarlos y contar siempre con su valioso apoyo, sincero en incondicional.

Rodain

Dedico este trabajo a mis padres, por lo mucho que me quieren y por toda la confianza que han depositado en mí, este logro también es de ustedes, porque son mi fuente de inspiración. A mi hermano y mis abuelos por compartir tantos momentos buenos en mi vida.

A mis amigos y amigas de la Universidad; Marta, Noel, Wilver, Ale y a todos aquellos que me han ayudado aunque no los mencione aquí. A mi compañero de tesis por permitir la realización de este trabajo de diploma.

A mi novio, por no agotar su paciencia durante el tiempo que me tomó terminar esta tesis, por apoyarme en todas las decisiones que he tomado, por haber entrado en mi vida.

A todos quienes me brindaron su amistad e hicieron que mi vida universitaria sea una experiencia que no olvidaré en toda mi vida.

Yaimy

Resumen

El Sistema de Gestión Académica de Pregrado (SGAP) forma parte de una solución integral desarrollada en la Universidad de las Ciencias Informáticas (UCI) para la gestión de los procesos sustantivos de la universidad. Actualmente en el sistema no se realiza un proceso homogéneo para la gestión de los reportes. Esto se debe principalmente, a que no se cuenta con un mecanismo que permita a los usuarios obtener, de forma rápida, información sobre el proceso académico de los estudiantes. En la presente investigación se propone la concepción y desarrollo de un asistente para gestionar reportes dinámicos, integrado al SGAP que permita la obtención de información de forma eficiente y facilite la toma de decisiones en el proceso de gestión académica de la UCI. Para esto se hace un análisis de los conceptos asociados a la gestión académica y a la eficiencia en los productos informáticos destinados a la gestión de información. Se realiza un estudio de las principales tendencias actuales en cuanto a la gestión de reportes y de algunas soluciones informáticas existentes en diferentes ámbitos con el objetivo de identificar elementos significativos que pudieran guiar su construcción. En el desarrollo de la propuesta se utiliza PHP 5.3 como lenguaje de programación, GUUD 1.0 como marco de trabajo, PostgreSQL 8.4.1 como Sistema Gestor de Base de Datos, Apache 2.2.2 como servidor web, NetBeans 6.7.1 como Entorno de Desarrollo Integrado y Visual Paradigm 8.0 como herramienta de modelado, todas ellas integradas en un proceso de desarrollo de *software* con enfoque ágil basado en el nivel 2 de CMMI que permite adaptarse rápidamente a los cambios y a las nuevas necesidades de los clientes.

Palabras clave: gestión académica, gestión de reportes, módulo de reportes.

Abstract

The Undergraduate Academic Management System (UAMS) is part of a full solution developed in the Informatics Sciences University (ISU) for managing essential processes of the university. Currently the system management reports are not done evenly. This is mainly because there is no mechanism that allows users to obtain in a fast way for the users, information about the students' academic process. In this research proposes the conception and development of an assistant to manage dynamic reports, integrated to UAMS that allows obtaining information efficiently and facilitates decision-making in the academic management process of the ISU. For this, has been made an analysis of the concepts associated with the academic management and efficiency in software products for information management. Also was performed a study of the main trends in terms of report management and about some existing solutions in different areas in order to identify significant elements that could guide its construction. In the assistant's development is used PHP 5.3 as programming language, GUUD 1.0 as a framework, PostgreSQL 8.4.1 as Database Management System, Apache 2.2.2 as web server, NetBeans 6.7.1 as Integrated Development Environment and Visual Paradigm 8.0 as tool for modeling, all this tools are integrated into a software development process with agile approach based on the CMMI level 2 that allows adapt quickly to changes and new customer needs.

Keywords: *academic management, report management, reporting module.*

Observaciones introductorias

- **Glosario de términos**
  Se recomienda consultar la sección Glosario de Términos antes de comenzar a leer el texto principal de la investigación. Los términos aparecen ordenados alfabéticamente.
- **Notas al pie de página**
  Aparecen enumeradas al pie de página y proporcionan información adicional sin interrumpir la secuencia lógica del texto principal.
- **Voces extranjeras**
  Atendiendo a la última edición de la *Ortografía de la lengua española* (Real Academia Española 2010), las voces o expresiones de otras lenguas que se utilizan en el documento, aparecen marcadas gráficamente con letras cursivas.
- **Referencias bibliográficas**
  Desde el texto principal, se indican entre corchetes el marcador con el cual aparecen identificadas en la sección Referencias bibliográficas. En dicha sección se muestran los datos completos de cada obra:
  [PRE10] PRESSMAN, Roger S. *Ingeniería de Software: Un enfoque prático.* 7ma. Edición. New York: McGraw-Hill Companies, 2010 [fecha de consulta: 5 Enero 2012]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/Ediciones_del_Pressman/Pressman_7ma_edicion. ISBN: 9780073375977.
- **Formatos para lectura**
  El presente trabajo se ha optimizado para uso digital e impreso en papel *carta*. No obstante, se recomienda usar la versión digital cuando así proceda; puesto que es un archivo PDF, generado a partir de un documento ODT creado en OpenOffice.org Writer[1], con enlaces que facilitan la navegación.

[1] OpenOffice.org Writer es una herramienta para la edición o procesamiento de texto multiplataforma, libre y de código abierto que forma parte del conjunto de aplicaciones de la suite ofimática OpenOffice.org.

Índice general

Índice de tablas

Introducción

Actualmente el mundo está avanzando a la era tecnológica. En la última década las nuevas Tecnologías de la Información y las Comunicaciones (TIC) han ido evolucionando radicalmente e incorporándose al quehacer cotidiano de las personas, esto ha provocado un gran impacto en todas las esferas de la sociedad y la educación. Las TIC se presentan cada vez más como una necesidad en el contexto de la sociedad donde los rápidos cambios, el aumento de los conocimientos y la demanda de una educación de alto nivel constantemente actualizada, se convierten en una exigencia permanente. La relación entre las TIC y la educación tiene dos vertientes: por un lado, los ciudadanos se ven abocados a conocer y aprender sobre las mismas y por el otro, estas tecnologías pueden aplicarse al proceso educativo.

Cuba en su afán por alcanzar una educación de excelencia y ante esta situación, ha diseñado e iniciado la aplicación de un conjunto de estrategias que permiten convertir las TIC en un instrumento a disposición del proceso educativo de los educandos a todos los niveles.

Como parte de la implementación de dichas estrategias surge la Universidad de las Ciencias Informáticas (UCI) en el año 2002, con la misión de formar profesionales comprometidos con su Patria y altamente calificados, además de servir de soporte a la industria cubana de la informática [UCI11]. La UCI es una universidad que cuenta con un plan de estudio que vincula el estudio con el trabajo como modelo de formación. Para cumplir con este modelo de formación y para brindar soluciones informáticas a problemáticas reales de la sociedad y de la propia universidad se hace necesaria la creación de centros productivos que agrupen estudiantes y profesores con objetivos afines.

A partir de la complejidad de los procesos por los que transita un estudiante durante su estadía en la universidad y dada la necesidad de su informatización, se crea en la UCI el Centro de Informatización Universitaria (CENIA) y dentro de este el Sistema de Gestión Universitaria (SGU). El SGU está compuesto por los sistemas Ingreso, Pregrado, Residencia, Investigación, Producción, Cooperación, Laboratorios, Biblioteca, Extensión, Teleformación, Egreso y Postgrado.

El Sistema de Gestión Académica de Pregrado, se encarga de gestionar y mejorar los procesos relacionados con la formación del estudiante. El mismo cuenta en su estructura con los módulos *Diseño de Carrera*, *Personal y Secretaría*, *Tesis y Títulos*, *Control Docente* y *Estudiante*. Cada uno de estos módulos gestiona la información referente a un área específica de dichos procesos. En ocasiones se hace necesario obtener de forma rápida y de acuerdo con determinados criterios de búsqueda, información que es gestionada por diferentes módulos dentro del sistema. Actualmente en el sistema no se realiza un proceso homogéneo para la gestión de los reportes. Esto se debe principalmente a que no se cuenta con un mecanismo que permita a los usuarios obtener, de forma

rápida y centralizada, información sobre el proceso académico de los estudiantes. Por tal motivo, cuando un usuario solicita información de determinados estudiantes acerca del mencionado proceso, se crean entonces reportes estáticos[2] en los módulos encargados de manejar la información solicitada. Estos reportes, debido a su carácter estático, necesitan ser actualizados con cierta periodicidad, no permiten modificaciones en su estructura y además no siempre muestran los resultados de la misma forma ya que no existe una única persona en el equipo de desarrollo encargada de su construcción. Lo antes expuesto trae como consecuencia que exista un bajo nivel de accesibilidad a la información relacionada con el proceso académico dificultando en ocasiones la toma de decisiones.

Luego de haber analizado la situación problemática existente respecto al proceso de gestión de reportes se define como **problema a resolver:** ¿Cómo obtener información del Sistema de Gestión Académica de Pregrado para la toma de decisiones garantizando la confiabilidad y seguridad de los datos?

Constituye el **objeto de estudio** de la presente investigación los procesos de gestión académica. Enmarcando su **campo de acción** en el diseño y obtención de reportes dinámicos de la información gestionada en el proceso de gestión académica.

Por tanto, el **objetivo general** es desarrollar un asistente para gestionar reportes dinámicos, integrado al Sistema de Gestión Académica de Pregrado, que permita la obtención de información de forma eficiente y facilite la toma de decisiones en el proceso de gestión académica de la Universidad de las Ciencias Informáticas.

De donde se derivan los siguientes **objetivos específicos:**

- Realizar el análisis y diseño del módulo para la gestión de reportes del Sistema de Gestión Académica de Pregrado.
- Implementar el módulo para la gestión de reportes del Sistema de Gestión Académica de Pregrado.
- Validar, mediante la realización de pruebas, la propuesta de solución del problema.

Con el propósito de guiar la investigación se plantea la siguiente **idea a defender:** El desarrollo de un asistente para diseñar reportes dinámicos, integrado al Sistema de Gestión Académica de Pregrado, permitirá la obtención de información, siempre actualizada, de forma eficiente para facilitar la toma de decisiones en el proceso de gestión académica.

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

[2] En este contexto, los reportes estáticos son reportes predefinidos que muestran solo determinada información a petición del usuario.

1. Estudio de los procesos de gestión académica.
2. Estudio del estado de las soluciones de *software* destinadas a la gestión de reportes tanto en el ámbito nacional como en el internacional.
3. Caracterización del proceso de desarrollo de *software* a utilizar.
4. Caracterización de las herramientas y tecnologías a utilizar en el desarrollo de la propuesta de solución.
5. Análisis de técnicas de ingeniería de requerimientos.
6. Obtención de los artefactos generados según el proceso de desarrollo seleccionado para la realización de la propuesta de solución.
7. Definición de las pruebas a realizar a la solución y realización de las mismas.
8. Consulta de tendencias actuales para la implementación de aplicaciones web y sistemas de gestión de información.
9. Definición de la arquitectura de la información y de *software*.
10. Descripción de la arquitectura y el modelado de la base de datos.
11. Definición de los patrones de diseños más adecuados para la construcción del sistema propuesto.
12. Definición del estándar de codificación y explicación de las técnicas de programación a utilizar.
13. Implementación de las funcionalidades definidas por el analista para la gestión de reportes.
14. Validación de la propuesta de solución.

Los **métodos científicos** empleados en la investigación son:

Métodos teóricos:

- **Histórico-lógico:** Se utilizó para estudiar las formas de solución a problemas similares sobre la gestión de reportes en todo el mundo, permitiendo constatar teóricamente cómo ha evolucionado este fenómeno en el tiempo.
- **Analítico-sintético:** Se utilizó para el análisis de teorías y documentos, se extrajeron los elementos más importantes de cada uno de los aspectos esenciales de las herramientas y la literatura seleccionada para el tema a estudiar.

Métodos empíricos:

- **Entrevista:** Se utilizó en la investigación para precisar el problema a resolver, así como las necesidades existentes en el proceso de gestión de reportes en el Sistema de Gestión Académica de Pregrado.

Justificación de la investigación

Con el desarrollo de la presente investigación se obtendrá un módulo para la gestión de los reportes del Sistema de Gestión Académica de Pregrado, así como todos los artefactos que se generen como resultado de la misma. La propuesta de solución presentada se utilizará como herramienta para validar la migración de los datos al nuevo sistema de gestión académica. Además, permitirá mejorar cuantitativamente el control y estandarización de los reportes, efectuar un proceso homogéneo para la gestión de reportes y lograr una alta accesibilidad a la información manejada referente al proceso de formación de los estudiantes.

La presente investigación está estructurada en 3 capítulos los cuales contienen la siguiente información:

Capítulo 1 Fundamentación teórica: en este capítulo se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. Se hace una descripción de los conceptos fundamentales asociados al dominio del problema. Se realiza un estudio del estado de los sistemas de gestión de reportes en el ámbito internacional, nacional y en la universidad, así como de las tecnologías, metodologías y herramientas que se utilizarán en el desarrollo de la propuesta de solución.

Capítulo 2 Descripción de la propuesta de solución: en este capítulo se analiza la propuesta de solución, realizando una descripción de cómo debe funcionar y destacando sus características distintivas. Se especifican los usuarios que tendrán acceso a la misma y se identifican las funcionalidades que debe brindar a partir de la especificación de requerimientos. Además, se describe la arquitectura, se presenta el modelo de datos y la distribución física o diagrama de despliegue de dicha propuesta.

Capítulo 3 Construcción y validación de la propuesta de solución: en este capítulo se describe la implementación del módulo propuesto teniendo en cuenta las técnicas de programación, los estándares de codificación empleados y las clases u operaciones necesarias. Se presenta una estrategia de prueba que define la realización de pruebas unitarias, de integración y del sistema (rendimiento y resistencia) para validar que este módulo satisface todas las necesidades del cliente.

Además de conclusiones, recomendaciones, referencias bibliográficas, bibliografía consultada, glosario de términos y anexos.

Capítulo 1. Fundamentación teórica

1.1 Introducción capitular

En el presente capítulo se presentan los elementos teóricos que sirven de base a la investigación del problema planteado y se describen los conceptos fundamentales asociados al dominio del problema. Además, se estudian algunos sistemas de gestión de reportes en el mundo, el país y en la universidad así como las tecnologías, metodologías y herramientas que se utilizarán en el desarrollo de la propuesta de solución.

1.2 Conceptos fundamentales

Antes de comenzar con el desarrollo de la investigación y con el objetivo de lograr un mejor entendimiento por parte de los lectores, es conveniente precisar algunos conceptos básicos que serán utilizados a lo largo de la misma y que a continuación se relacionan.

1.2.1 Gestión Académica

Hacer referencia a la gestión en el marco institucional, amerita definirla en el contexto en el que ella se sitúa; "*la gestión incluye la acción y el efecto de administrar de manera tal que se realicen diligencias conducentes al logro apropiado de las respectivas finalidades de las instituciones*" [INI06].

Este planteamiento implica atribuir especificidad al término gestión en el escenario académico de la educación; en tal sentido, se denomina gestión académica al *conjunto de procesos mediante los cuales se administran los diferentes componentes y subcomponentes curriculares que apoyan la práctica pedagógica en el continuo que permite construir y modelar el perfil deseable del estudiante* [INI06].

En este orden de ideas, la gestión académica cubre un recorrido que involucra las acciones de atención al estudiante desde el ingreso hasta su desarrollo y egreso del sistema.

1.2.2 Reportes

Un reporte es aquel documento que se utilizará cuando se quiera informar o dar noticia acerca de una determinada cuestión. Aunque básicamente su objetivo será el de informar, los reportes también podrán incluir algunos elementos persuasivos, como pueden ser recomendaciones o sugerencias y también algunas conclusiones a través de las cuales se le indique al lector alguna acción o conducta a adoptar en el futuro.

En el ámbito de la informática, los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para

mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios, confiriéndole así una mayor utilidad a la información.

1.2.3 Eficiencia

La *eficiencia* en un concepto que posee diversas interpretaciones dependiendo del contexto en el que se utilice. De manera general la norma ISO 9000:2005 (Sistemas de gestión de la calidad. Fundamentos y vocabulario) define este término como: "*(…) relación entre el resultado alcanzado y los recursos utilizados*" [ISO9000]. Debido a que la presente investigación pretende mejorar un proceso de gestión de información mediante un producto *software* que garantice la eficiencia en dicho proceso, es de vital importancia dejar bien claro en qué consiste la eficiencia en un sistema informático y cuándo una información se considera eficiente.

Eficiencia en los sistemas informáticos

Según la ISO 25000:2005 (Calidad del producto *software*) la eficiencia se define como la capacidad del producto *software* para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas. O sea, la eficiencia involucra el comportamiento temporal (capacidad para proporcionar tiempos de respuesta y tiempos de procesos apropiados) y la utilización de recursos (capacidad para usar las cantidades y tipos de recursos adecuados cuando el *software* lleva a cabo su función) [ISO25000].

Información eficiente

Para que la información resulte eficiente, debe reunir una serie de requerimientos, de modo que la utilidad que proporciona, justifique el empleo de los recursos que se apliquen para producirla. Entre estos requerimientos se pueden mencionar:

- **Economía**. El costo asociado a la producción de una información no debe ser superior al beneficio esperado por su uso.
- **Oportunidad**. La información debe estar disponible en el momento en que se requiera.
- **Utilidad**. Toda salida de un sistema de información debe satisfacer una necesidad o requerimiento.
- **Claridad**. La información debe atender al nivel intelectual y técnico del destinatario.
- **Comparabilidad**. La información debe ser comparable en espacio, en tiempo y en alcance.
- **Confiabilidad**. La información debe ser lo suficientemente confiable, como para tomar decisiones basadas en ella.

1.2.4 Seguridad de la información

El estándar ISO/IEC 27001:2005 (Tecnología de la Información. Sistemas de gestión de seguridad de la información) plantea que la seguridad de la información es la protección de la información de un rango amplio de amenazas. Ello implica la preservación de la *confidencialidad*, *integridad* y *disponibilidad* de la misma. La confidencialidad se puede entender como la propiedad de que esa información esté disponible y no sea divulgada a personas, entidades o procesos no autorizados. Por su parte la integridad hace referencia a la propiedad de salvaguardar la información, es decir, que esta no pueda ser modificada por quien no esté autorizado. Mientras que la disponibilidad se define como la propiedad de estar disponible y utilizable cuando lo requiera una entidad autorizada [ISO27001].

1.2.5 Confiabilidad de la información

La confiabilidad de la información abarca tres aspectos fundamentales: la *seguridad*, la *veracidad* y la *consistencia* de la misma. El primer aspecto se explicó en el acápite anterior, por lo tanto, solo se definirán los dos restantes. La veracidad de la información hace referencia a la propiedad de estar conforme con la verdad, o sea, que la información sea auténtica, correcta y exacta, lo que implica que no debe contener errores. En cuanto a este aspecto, es válido acotar que en el contexto de esta investigación la veracidad de la información quiere decir que la información se muestra exactamente como está registrada en el sistema, o sea, excluye por completo errores cometidos por las personas a la hora de ser ingresada. Por su parte la consistencia en la información significa que esta deber permanecer estable en el tiempo, es decir, que no cambie continuamente [ARE09].

1.3 Tendencias en el uso de sistemas de gestión de reportes

Hoy en día la mayoría de las empresas suelen usar con frecuencia sistemas de gestión con el fin de gestionar sus recursos, actividades y orientarse hacia la obtención de buenos resultados. Cuando se implanta un sistema de este tipo y debido al gran cúmulo de información con que suelen trabajar, la implantación de un *sistema o módulo de reportes* se convierte en una necesidad primordial. Los *sistemas de reportes o informes*, como también se les conoce, facilitan la distribución de la información a todos los niveles dentro de la estructura organizacional, proporcionando a cada persona la información que necesita [PRO11].

Ventajas de tener un sistema o módulo de reportes

Además de los beneficios intrínsecos de disponer de información válida para tomar decisiones, la gran ventaja de los sistemas de reporte es la automatización de la generación de informes. Esto evita los siguientes problemas que traería consigo realizar esta tarea de forma manual:

- Asignación de recursos humanos a realizar tareas monótonas y repetitivas.
- Duplicidades o incongruencias en la información recibida.

Además de los costos fijos asociados al puesto de trabajo de la persona que realiza este tipo de tareas, que generalmente suele ser cualificada.

1.3.1 Estado de los sistemas de gestión de reportes en el mundo

- **Universitas XXI – Académico**

Universitas XXI – Académico es un módulo del sistema ERP[3] Universitas XXI, desarrollado por la Oficina de Cooperación Universitaria, multinacional de origen español, conformada por seis universidades y el grupo financiero Santander. Este módulo se encarga de automatizar todos los procesos que tienen relación con el alumnado y con la planificación y seguimiento de los recursos docentes, es decir, los procesos de gestión académica de la Universidad, estando adaptado al Espacio Europeo de Educación Superior (EEES) [OCU07].

Las funcionalidades de la aplicación abarcan desde la organización de las pruebas de acceso y preinscripción de los estudiantes a la tramitación de los títulos pasando por la matrícula, la calificación de las actas, el control del expediente, estadísticas, gestión de becas nacionales e internacionales, organización de prácticas en empresas, así como lo referente a la gestión económica de la actividad académica de la Universidad. La definición de los planes de estudios y la planificación de la docencia son los pilares básicos de la aplicación y garantizan la integración y coherencia de toda la información registrada.

En su estructura, este sistema cuenta con un *Generador de Reportes* que da la posibilidad de seleccionar entre diferentes formatos de reportes que han sido prediseñados sobre la base de información recolectada en varias instituciones educativas del país y en el Ministerio de Educación. Entre los reportes que permite obtener están:

- Informes relacionados con el ámbito académico.
- Asistencias.
- Matrículas.
- Cuadros de calificaciones.
- Informes de grado.
- Listados de profesores y estudiantes.
- Certificados disciplina.

[3] Los sistemas de Planificación de Recursos Empresariales, o ERP (por sus siglas en inglés, *Enterprise Resource Planning*) son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa.

Adicionalmente cuenta con opciones para poder crear sus propios reportes dinámicos y exportarlos a Microsoft Excel, a Microsoft Word o en formato texto (.txt).

Especificaciones: El módulo de Universitas XXI – Académico está desarrollado en tecnología Oracle forms[4] 10gR2.

- **InfoMaker**

InfoMaker es un producto colombiano que permite el acceso, administración y reporte de datos para desarrolladores y usuarios finales. Además, permite la creación de reportes con alta calidad de presentación y poderosas consultas sin la complejidad de la programación. Complementa las aplicaciones y herramientas de desarrollo cliente/servidor, así como paquetes de productividad de oficina.

InfoMaker combina las tecnologías de acceso a información OLE 2.0[5] de Windows y ODBC[6] para una generación potente y flexible de reportes. Proporciona las facilidades para construir rápidamente formatos de entrada de datos, desempeña funciones de edición y mantenimiento de la base de datos. InfoMaker viene con una versión embebida de Sybase SQL Anywhere[7], una poderosa base de datos de uso personal, que le permite trabajar sin conectarse a la base de datos empresarial. Como rasgo distintivo de este sistema se tiene que los reportes se construyen dinámicamente y para ello parte de la relación entre un conjunto de tablas ***T***, entre las cuales se puede realizar operaciones de un conjunto ***O*** para obtener campos agrupados en un conjunto ***C***. Como es de suponer todo esto es transparente al usuario, el cual solo deberá seleccionar la información que desea y las características que debe cumplir dicha información para ser mostrada en el reporte [MTB09].

Características

Algunas de las principales características de este producto son:

- InfoMaker posee una interfaz fácil de utilizar.
- No se tiene que conocer el lenguaje de la base de datos para saber cómo acceder a ellos.
- Ofrece plantillas de diseño instantáneo de informes.

[4] Herramienta de Oracle cuyo principal objetivo es el desarrollo rápido de aplicaciones, sobre todo de gestión. Está diseñado solo para interactuar con una Base de Datos Oracle.

[5] *Object Linking and Embedding* (OLE) es un sistema de objeto distribuido y un protocolo desarrollado por Microsoft que permite a un editor encargar a otro la elaboración de parte de un documento y después volver a importarlo.

[6] *Open DataBase Connectivity* (ODBC) es un estándar de acceso a bases de datos desarrollado por SQL Access Group cuyo objetivo es hacer posible el acceso a cualquier dato desde cualquier aplicación independientemente del Sistema Gestor de Base de Datos.

[7] Sistema de Bases de Datos Relacionales (RDBMS) de alto rendimiento desarrollado por la compañía Sybase iAnywhere, una subsidiaria de Sybase.

1.3.2 Estado de los sistemas de gestión de reportes en Cuba

- **Sistema para la Gestión Académica (GESTACAD)**

Aplicación desarrollada por la Facultad de Informática de la Universidad de Matanzas "Camilo Cienfuegos". Brinda funcionalidades que permiten mantener una información actualizada de estudiantes y profesores del centro, así como la obtención de reportes. Presenta en su estructura cinco módulos que posibilitan un mejor control y gestión del proceso académico:

I. Módulo para las Secretarias Docentes para la gestión de estudiantes: permite la realización de acciones generales comunes en una Secretaría Docente así como la obtención de reportes oficiales.
II. Módulo para los Jefes de Departamentos: se incluyen acciones relativas como la asignación de la carga docente y el control sobre los profesores del Departamento.
III. Módulo para la Gestión de la Matrícula: contempla el registro de toda la información referente al proceso de la matrícula del estudiante.
IV. Módulo para los Profesores: permite llevar el control docente de los estudiantes que tiene asignado un profesor, el control de las evaluaciones así como reportes relativos a su carga docente.
V. **Módulo de Reportes en Línea:** Posibilita la creación de una gran variedad de reportes con los datos personales del estudiante, su ubicación según el horario docente detallando aula, asignatura y tipo de clases que está recibiendo, incluyendo si se le ha registrado la asistencia al turno de clases. Una característica esencial de este módulo es que permite incorporar dinamismo en los reportes. Esto significa que el sistema le da la posibilidad al usuario de seleccionar los campos de datos que desea obtener en el reporte, así como el título de este y las condiciones que debe cumplir la información a mostrar. Además de los ya mencionados reportes dinámicos, el reportador de GESTACAD cuenta con reportes predefinidos como:
    - Reporte de notas por asignatura y grupo: examen final, extraordinario y premio.
    - Tabla con los resultados docentes de un grupo en un semestre.
    - Reporte de los resultados académicos de un estudiante en toda su carrera: *hoja de rendimiento*.

Este sistema se utilizó en la UCI en el primer curso académico cuando la universidad contaba con una matrícula de 2000 estudiantes y solo un plan de estudio. Pero su uso no fue factible a partir del segundo curso académico al crecer la matrícula y presentar dos planes de estudio para una misma

carrera, unido a ello se necesitaba mantener un seguimiento en la trayectoria de las evaluaciones de los estudiantes.

1.3.3 Estado de los sistemas de gestión de reportes en la UCI

En la UCI debido al alto nivel de informatización con que cuentan los procesos, la cantidad de información que se gestiona de forma automatizada y el uso frecuente de sistemas de gestión ha propiciado que la mayoría de los proyectos productivos incorporen a sus productos informáticos mecanismos para gestionar reportes o informes. Además, la universidad cuenta con el Centro de Tecnologías de Gestión de Datos (DATEC) que tiene como objetivo proveer soluciones, productos y servicios relacionados con la gestión de datos.

Entre dichas soluciones se encuentra el Generador Dinámico de Reportes (GDR). El GDR es una aplicación web que tiene como objetivo generar reportes de forma rápida, interactiva y con una amplia gama de alternativas para los usuarios.

El Generador permite a los usuarios, entre otras opciones, agilizar la toma de decisiones y generar reportes en varios formatos y con gran variedad de opciones en su diseño, marcando una diferencia entre los reportes tradicionales y los dinámicos.

El principal inconveniente que tiene este sistema es que no está orientado del todo a usuarios finales con pocos conocimientos en Bases de Datos puesto que una parte de él basa su funcionamiento en la suposición de que el usuario posee conocimientos previos en esta área, o sea, cómo trabajar con vistas, tablas y cómo estructurar consultas. Por esta razón debe existir un intermediario entre el usuario final y los resultados que muestre el reporte. Este intermediario es el encargado de diseñar, a petición del usuario, el modelo del reporte (cómo se muestran los datos) y además la forma en que se obtendrá la información (consulta a la Base de Datos).

- **Sistema Automatizado para la Gestión Académica, Akademos**

Este sistema surge en el año 2003 en la UCI para apoyar la labor desarrollada en la secretaría docente de la UCI con el objetivo de desarrollar una aplicación genérica que sea adaptable a cualquier plan de estudio y adaptable a cualquier centro universitario. Su desarrollo se basó en tres principios fundamentales:

- 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 estudio es la entidad fundamental del proceso de gestión académica y rige todos sus subprocesos (matrícula, control y planificación).

Akademos está compuesto por los módulos:

I. Módulo Plan de Estudio: permite la creación de los planes de estudio, la configuración de las disciplinas y las asignaturas que debe vencer un estudiante a lo largo de la carrera.
II. Módulo Matrícula: permite la gestión de la matrícula administrativa y de los movimientos a que son sometidos los estudiantes en su paso por la universidad, así como la definición de los estados y sus transiciones.
III. Módulo Expediente: posibilita la gestión de los expedientes de los estudiantes que pueden almacenar documentos basados en plantillas, así como otros de libre formato o generados por el propio sistema.
IV. Módulo Profesor: gestiona la información referente a la asignación del profesor a las diferentes estructuras administrativas registradas en el sistema. Permite mantener un control de la carga del profesor y mantener un seguimiento de los mismos en cada facultad.
V. Módulo Estudiantes: permite que el estudiante pueda consultar sus evaluaciones en las distintas asignaturas vencidas, así como su desempeño en las asignaturas del curso o período docente en el que se encuentre.
VI. Módulo Registro del Profesor: contribuye al control del registro docente del estudiante permitiendo que los profesores registren las evaluaciones y la asistencia a turnos de clases.
VII. **Módulo Reporte:** provee al usuario de un conjunto de herramientas que posibilitan la configuración de reportes relacionados con los datos registrados de los estudiantes.

Características de los reportes en Akademos

Este sistema incorpora dinamismo a algunos reportes mientras que el resto, como los relacionados con la promoción de los estudiantes, se crean de forma estática. Dicho de otra manera, son reportes que han sido previamente diseñados. Esta característica representa un gran inconveniente puesto que un reporte que sea creado en un período específico dentro de un proceso tan dinámico como lo es la gestión académica, pierde valor una vez transcurrido dicho período. Por ejemplo, si alguien crea un reporte con los estudiantes de 4^{to} año suspensos en una asignatura determinada en el segundo semestre de un curso académico específico, en el próximo curso académico este reporte sería inservible. Además, si un usuario quisiera cambiar la información que muestra un reporte, o sea, modificarlo le resultaría imposible puesto que Akademos no brinda esa posibilidad.

1.3.4 Análisis crítico de los sistemas estudiados

Principales inconvenientes

En el estudio realizado del estado de los *sistemas de reportes* se destaca como principal inconveniente el hecho de que la mayoría de las herramientas más potentes para la gestión de reportes son tecnologías privativas o propietarias por lo que no están al alcance de todos. En otros casos se ajustan solo a las necesidades y características de las instituciones que las desarrollaron o no están orientadas a todo tipo de usuarios.

Aspectos positivos

Entre los principales aspectos que aporta el estudio anteriormente mencionado y que constituyen puntos de partida para la propuesta de solución se pueden mencionar tres ideas fundamentales:

1. *El carácter dinámico de los reportes.* Darle a usuario la posibilidad de configurar la información que desea obtener en su reporte.
2. *Reportes predefinidos.* El usuario puede seleccionar reportes prediseñados.
3. *Modo de construcción basado en conjuntos*. Para la construcción de la sentencia que obtendrá la información de la Base de Datos se tendrán conjuntos que agrupen las tablas, operaciones entre ellas y posibles campos.

1.5 Proceso de desarrollo del *software*

Un proceso de desarrollo de *software* es el conjunto de actividades necesarias para transformar los requerimientos de un usuario en un sistema *software*. No existe ningún proceso de desarrollo que sea de aplicabilidad universal. La realidad es que estos varían porque tienen lugar en diferentes contextos, desarrollan diferentes tipos de sistemas y se ajustan a diferentes tipos de restricciones del negocio (plazo, costos, calidad y fiabilidad). A pesar de la diversidad existente en cuanto a procesos de desarrollo del *software*, se pueden mencionar algunas actividades fundamentales que son comunes para todos ellos [SOM05], tales como:

1. *Especificación del software*. Se debe definir la funcionalidad del *software* y las restricciones en su operación.
2. *Diseño e implementación del software*. Se debe producir *software* que cumpla su especificación.
3. *Validación del software*. Se debe validar el *software* para asegurar que hace lo que el cliente desea.
4. *Evolución del software*. El *software* debe evolucionar para cubrir las necesidades cambiantes del cliente.

Debido a que no existe el proceso de *software* "ideal", en cada organización puede haber enfoques para mejorarlos.

1.6 Proceso de desarrollo con enfoque ágil basado en el nivel 2 de CMMI

Actualmente la industria del *software* se caracteriza por un gran dinamismo y variabilidad. Esto, unido a un mercado caracterizado por el rápido desarrollo de aplicaciones y la reducción de la vida de los productos, obliga a las organizaciones incrementar la productividad, a disminuir el tiempo de reacción y a adaptarse rápidamente a los cambios y a las nuevas necesidades de los clientes [BOE06]. Atendiendo a esta situación, en la presente investigación se definió emplear un proceso de desarrollo con enfoque ágil ya que, según el Manifiesto Ágil[8] [BEC01], esto permite:

- Valorar al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
- Desarrollar un *software* que funcione por encima de una completa documentación.
- Valorar la colaboración con el cliente por encima de la negociación contractual.
- Responder a los cambios más que seguir estrictamente un plan.

Para asegurar la mejora del proceso definido, el mismo estará basado en las metas y prácticas del nivel 2 de CMMI[9] (para mayor información consultar el **Anexo 1**) el cual asegura que el proceso sea gestionado además de ejecutarse, que se planifique, se revise y se evalúe para comprobar que cumple con los requerimientos [SEI11].

El proceso de desarrollo con enfoque ágil basado en el nivel 2 de CMMI tiene definido el siguiente ciclo de vida:

Estudio Preliminar: se realiza un estudio profundo de la organización cliente que posibilita obtener la información requerida para determinar el alcance del proyecto, así como la estimación del costo, tiempo y el esfuerzo.

Modelado de Negocio: se comprende el negocio de la entidad con el objetivo de que el *software* a desarrollar cumpla con lo que realmente quiere el cliente.

Requisitos: el objetivo fundamental es desarrollar el modelo del sistema, identificando los requerimientos funcionales y no funcionales con las descripciones correspondientes en cada caso.

Análisis y Diseño: se realiza el análisis y el modelado del sistema a partir de los requerimientos definidos previamente.

[8] Documento que resume la filosofía ágil.
[9] *Capability Maturity Model Integration* que traducido al español significa Modelo de Capacidad y Madurez Integrado.

Implementación: a partir de los artefactos obtenidos durante el análisis y diseño se procede a realizar la implementación del *software* en términos de componentes de implementación.
Pruebas Internas: se realizan las pruebas internas con el equipo del proyecto en cada una de las iteraciones o versiones finales próximas a ser liberadas, según lo defina el proyecto.
Pruebas de Liberación: pruebas realizadas por parte de la oficina o institución encargada de la calidad y de la certificación del proyecto a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
Despliegue: se realiza la entrega de la aplicación al cliente, así como la configuración y prueba en el ámbito del cliente. Las pruebas realizadas durante esta fase incluyen *pruebas de aceptación* y *pruebas piloto*. Se debe realizar además capacitaciones a los trabajadores del sistema.
Soporte: por un tiempo limitado el proyecto ofrecerá un servicio para resolver conflictos y problemas de usabilidad y rendimiento del *software* entregado al cliente, suministrándole actualizaciones y parches a errores.

1.7 Lenguajes y tecnologías

Para el desarrollo de la propuesta se emplearán las tecnologías y herramientas establecidas por el Grupo de Implantación, Tecnologías y Soporte del Centro de Informatización Universitaria, el cual establece utilizar como lenguaje de programación PHP 5.3, como marco de trabajo GUUD[10] 1.0, el sistema gestor de base de datos PostgreSQL 8.4.1, siguiendo un proceso de desarrollo de *software* con enfoque ágil basado en el nivel 2 de CMMI.

1.7.1 HTML 4.01

HTML 4.01 en la versión más utilizada actualmente de HTML, siglas de *HyperText Markup Language* («lenguaje de marcado de hipertexto»), desarrollado por el *World Wide Web Consortium*[11] (W3C), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido de las páginas 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 cierto punto, la apariencia de un documento, y puede incluir *scripts* (por ejemplo, escritos en JavaScript), los cuales pueden afectar el comportamiento de navegadores web y otros procesadores de HTML [GAL09].

Ventajas

[10] Gestión Universitaria de Universidad Digital.
[11] El *World Wide Web Consortium* es una comunidad internacional que desarrolla estándares Web.

HTML es sencillo, permite describir hipertexto, el texto es presentado de forma estructurada y agradable, no necesita de grandes conocimientos cuando se cuenta con un editor de páginas web, sus archivos son pequeños, es fácil de aprender y lo admiten todos los exploradores web [GAL09].

Desventajas

Es un lenguaje estático, que no permite la creación de páginas dinámicas. La interpretación de cada navegador puede ser diferente, guarda etiquetas que pueden convertirse en "basura" dificultando la corrección, y además estas etiquetas son muy limitadas [GAL09].

1.7.2 Lenguaje de programación PHP 5.3

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente, se ejecuta en un servidor web tomando el código en PHP como su entrada y creando páginas web como salida. A esto se añaden valores como el hecho de ser un proyecto de código abierto, gratuito y multiplataforma. PHP es libre por lo que se presenta como una alternativa de fácil acceso para todos y permite técnicas de Programación Orientada a Objetos. Además, posee una extensa biblioteca nativa de funciones, así como una amplia documentación oficial [COG05].

1.7.3 JavaScript 1.6

Lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la de los lenguajes Java y C. Es un lenguaje orientado a objetos, ya que dispone de herencia, la cual se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. JavaScript se ejecuta en el cliente al mismo tiempo que las sentencias van descargándose junto con el código HTML [SÁN06].

1.7.4 CSS 2

El nombre *hojas de estilo en cascada* viene del inglés *Cascading Style Sheets*, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores web. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación [VAN07].

Ventajas de usar hojas de estilo

Entre las ventajas de usar CSS o (u otro lenguaje de estilo) se pueden citar [SCH06]:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o, incluso, a elección del usuario.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (siempre y cuando no se utilice estilo en línea).

1.7.5 SQL

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés *Structured Query Language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar, de una forma sencilla, información de interés de una base de datos, así como también hacer cambios sobre ella [CHA09].

Características generales del SQL

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones en estos últimos. Es un lenguaje declarativo (que especifica qué es lo que se quiere y no cómo conseguirlo), que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, permite una alta productividad en codificación y la orientación a objetos.

1.7.6 Marcos de trabajo que soportarán el desarrollo *«Framework de desarrollo»*

La palabra inglesa *framework* define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de *software* un *framework* o marco de trabajo es una estructura conceptual y tecnológica de soporte definido, normalmente, con artefactos o módulos de *software* concretos. Sobre la base de esta estructura otro proyecto de *software* puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto [JOR06].

Nombre de la herramienta: GUUD 1.0

GUUD es un marco de trabajo propuesto por el equipo de arquitectura del CENIA. El mismo integra a su vez los marcos de trabajo CodeIgniter 1.7.3 y JQuery 1.3.2 en una sola infraestructura, razón por la cual posee las mismas características que estos. En esta integración se incluyen además un conjunto de novedades o mejoras y algunas modificaciones hechas específicamente al CodeIgniter que se explican en el acápite 1.7.6.1.

¿Qué es CodeIgniter?

CodeIgniter es un marco de trabajo para el desarrollo de aplicaciones web escritas en PHP. Este permite el desarrollo de proyectos mucho más rápidos que si se escribiera código desde cero. Provee una amplia colección de librerías para las tareas necesarias más comunes. CodeIgniter permite concentrarse en el desarrollo del proyecto en cuestión, minimizando la cantidad de código necesario para realizar las tareas. CodeIgniter usa el patrón de diseño arquitectónico Modelo-Vista-Controlador como paradigma de arquitectura de desarrollo, la cual separa en 3 capas distintas: la representación de datos, la interfaz de usuario y el controlador de eventos respectivamente [ELL11].

¿Qué es JQuery?

El jQuery es un nuevo tipo de biblioteca o marco de trabajo de JavaScript que permite acceder a los objetos del DOM[12] de un modo simplificado. JQuery ofrece funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio. El gran aporte de JQuery es que permite cambiar el contenido de la página web sin necesidad de recargarla, utilizando DOM y AJAX[13] de manera sencilla gracias a su sintaxis [JQU11].

1.7.6.1 Novedades que incorpora el GUUD

A continuación se muestra una relación de las principales mejoras y modificaciones que incorpora el GUUD en su infraestructura.

Del lado del cliente:

[12] El *Document Object Model* o Modelo de Objetos del Documento, es interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

[13] Acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de estas que trabajan conjuntamente (XHTML o HTML, CSS, DOM, XMLHttpRequest y XML).

1. Se implementó un plugin para JQuery 1.3.2 que permite el manejo de espacios de nombre e internacionalización.
2. Se implementaron *widgets*[14] para utilizarlos de interfaz de algunos de los que ya posee jquery-ui como por ejemplo el *date*, el *tab* (ambos son interfaces de los widgets de mismo nombre de jquery-ui) y el *popup* (interfaz del *dialog* de jquery-ui). Además de los ya mencionados se implementaron otros nuevos entre los que se encuentran: *attach*, *menu*, *message*, *tooltip*, *form* (se construyó con la unión de los *plugins*[15] *form* de JQuery el cual se utiliza para el envío de formularios AJAX y el *validate* utilizado para validar formularios), *grid* (utiliza como plugin el *jqgrid*), *multiselect* (para hacer selecciones múltiples), *navbar* (para la creación de barras de navegación), *tree* (para la creación de árboles) y el *graph* (utiliza la librería Highchart).
3. Se implementaron funciones comunes para todo el sistema (contenidas en los archivos *core.js* y *common.js*) entre las que se destacan: *loadIn*, *getDataJson*, *fromJson*, *toJson*, *createSelect*, *isArray*, *isFunction* y *site_url*.

Del lado del servidor (incorporadas a CodeIgniter):

1. Se agregó el manejo de excepciones y mensajes.
2. Se implementó el IOC (*inversion of control*) para la interacción entre módulos.
3. Se añadió la característica de la modularidad o sea que una aplicación pueda dividirse en módulos.
4. Se añadieron, modificaron y extendieron los *helpers* o *asistentes*[16] entre los que se encuentran:
   - Añadidos: *template* (brinda la posibilidad de usar plantillas, característica que no posee CodeIgniter. Para esto se añadió también la librería *template*), *assets* (utilizado para la integración en las vistas de javascript, css, imágenes y el template), *grid*, *json*.
   - Modificados: *form*, *array* y *security*.
5. Se añadieron los *plugins export_pi* (permite exportar a los formatos: pdf, csv y xls) e *import_pi* (permite importar desde archivos en formatos csv o xls).

[14] Pequeñas aplicaciones o programas, usualmente presentados en archivos o ficheros pequeños, cuyo principal objetivo es dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.
[15] Un *plugin* es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.
[16] Los *helpers* o asistentes, como su nombre lo indica, son una colección de funciones de una categoría particular que ayudan en la realización de determinadas tareas.

1.7.7 Sistema Gestor de Base de Datos (SGBD)

Los sistemas de Gestión de Bases de Datos (en inglés *Database Management System*, abreviado DBMS), son un tipo de *software* dedicados a servir de interfaz entre la base de datos[17], el usuario y las aplicaciones que la utilizan [FER10].

Nombre de la herramienta: PostgreSQL 8.4.1

Licencia del producto: Se distribuye bajo la licencia de postgresql, un liberal de licencia de código abierto, similar a las licencias BSD[18] o MIT.

PostgreSQL es un Sistema Gestor de Bases de Datos Objeto-Relacional (ORDBMS de sus siglas en inglés), basado en el proyecto Postgres, de la Universidad de Berkeley. Es una derivación libre de este proyecto, y utiliza el lenguaje SQL92/SQL99. Debido a la licencia libre, PostgreSQL puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita para cualquier propósito, sea comercial privado, o académico [ADR11].

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. Entre sus principales características están:

- Soporta distintos tipos de datos.
- Incorpora funciones de diversa índole, incluyendo algunas para el manejo de fechas.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a estos.

1.7.8 Servidor web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales síncronas[19] o asíncronas[20] con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la

[17] Cualquier conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora.
[18] La licencia BSD es la licencia de *software* otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Es una licencia de *software* libre permisiva, es decir, flexible respecto a la distribución de modo que un *software* bajo esta licencia puede ser redistribuido como *software* libre o *software* propietario.
[19] Que tiene un intervalo de tiempo constante entre cada evento.
[20] Que no tiene un intervalo de tiempo constante entre cada evento.

transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente, para estas comunicaciones se utiliza el protocolo HTTP perteneciente a la capa de aplicación del modelo de Interconexión de Sistemas Abiertos[21] (OSI). El término también se emplea para referirse al ordenador que ejecuta el programa [SER11].

Nombre de la herramienta: Apache 2.2.2 o superior.

Apache es el servidor web más utilizado en el mundo[22]. Su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. La licencia Apache es una descendiente de la licencia BSD [KAB09].

Características de Apache (tomadas de [KAB09])

- Corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código abierto.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados cuando sea necesario.
- Trabaja con gran cantidad de lenguajes entre los cuales se encuentra PHP.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.

1.7.9 Lenguaje de modelado

Lenguaje Unificado de Modelado (tomado de [LAR99])

El Lenguaje Unificado de Modelado o UML (siglas de *Unified Modeling Language*) es un lenguaje para especificar, construir, visualizar y documentar los artefactos (información que se utiliza o produce mediante un proceso de *software*). Este lenguaje de modelado no es una guía para realizar el análisis y diseño orientado a objeto, es decir, no es un proceso, es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos informáticos, de arquitectura o de cualquier otra rama.

[21] El modelo OSI (*Open System Interconnection*) es el modelo de red descriptivo creado por la Organización Internacional para la Estandarización (ISO) y se utiliza como referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

[22] Según Netcraf, compañía que publica periódicamente las estadísticas de los servidores más utilizados, hasta el 2 Mayo del 2012 aproximadamente el 64.20% de los servidores web estaban montados sobre Apache (para mayor información consultar http://news.netcraft.com/archives/category/web-server-survey).

1.8 Herramientas

1.8.1 Entornos de Desarrollo Integrado

Nombre de la herramienta: NetBeans 6.7.1 o superior.

Licencia del producto: Doble licencia; Common Development and Distribution License (CDDL) y GNU General Public License version 2 with Classpath exception (GPL2).

NetBeans es un proyecto exitoso de código abierto con una comunidad en constante crecimiento. Sun MicroSystems[23] fundó el proyecto NetBeans en junio 2000 y continúa siendo su patrocinador principal. Hoy en día hay disponibles dos productos: el Entorno de Desarrollo Integrado NetBeans y la Plataforma NetBeans.

NetBeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además, un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso [ORA11].

1.8.2 PgAdmin III

Nombre de la herramienta: PgAdmin III 1.10.5 o superior.

Licencia del producto: BSD

PgAdmin III 1.10.5 es una aplicación gráfica para administrar el gestor de bases de datos PostgreSQL, siendo la más completa y popular de código abierto. Es capaz de gestionar versiones a partir de PostgreSQL 7.3 ejecutándose en cualquier plataforma. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita su administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar *scripts* programados. La conexión al servidor puede hacerse mediante conexión TCP/IP y puede encriptarse mediante SSL (acrónimo de *Secure Sockets Layer* - Protocolo de Capa de Conexión Segura) para mayor seguridad [PGA11].

1.8.3 Ingeniería del *Software* asistida por computadoras «CASE»

Nombre de la herramienta: Visual Paradigm 8.0 o superior.

Visual Paradigm es una herramienta de modelado profesional que hace uso del Lenguaje Unificado de Modelado (de sus siglas en inglés UML). Una característica esencial de Visual Paradigm es que

[23] Sun MicroSystems fue una empresa informática recientemente (2009) adquirida por Oracle Corporation, anteriormente parte de Silicon Valley, fabricante de semiconductores y software.

soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción y despliegue. Visual Paradigm ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación [VSP11].

1.8.4 Evolus Pencil

Nombre de la herramienta: Evolus Pencil 1.2.1 o superior.

Es una herramienta libre y de código abierto para crear diagramas y prototipos de interfaz gráfica de usuario que todos puedan usar. Evolus Pencil es la evolución de Pencil y con ella se pueden crear, de formar fácil, ventanas de prototipos arrastrando los diferentes elementos ya sea como extensión de Firefox o como aplicación estándar para Windows o Linux [EVO11].

Características Principales:

- Construcción de prototipos.
- Exportación a formatos HTML, PNG, documento Word y PDF.
- Multiplataforma: Puede ser instalado tanto en Windows como Linux. Además, puede agregarse como complemento para el navegador Mozilla Firefox.
- Tipo de funcionamiento: arrastrar y soltar.

1.8.5 Apache JMeter

Nombre de la herramienta: jmeter 2.6 o superior.

Licencia del producto: Apache License 2.0

Es una aplicación de escritorio (100 % Java) de código abierto desarrollada por *Apache Jakarta* que es un proyecto de la *Apache Software Foundation.* Básicamente, fue diseñada para realizar pruebas de carga y medir el rendimiento de sistemas. Esta herramienta se puede utilizar para probar el rendimiento de los recursos estáticos y dinámicos de los sistemas (archivos, *servlets*, *scripts* de Perl, objetos Java, bases de datos y consultas, servidores FTP) y para simular la carga de un servidor, red u objeto con el fin de poner a prueba su resistencia (conocidas como pruebas de estrés) [APA12].

1.9 Conclusiones parciales

En este capítulo se ha realizado un estudio de los principales conceptos que guiarán esta investigación. A partir del estudio de sistemas homólogos se pudieron definir las tendencias en la UCI, en Cuba y en el mundo en cuanto al desarrollo de sistemas de gestión de reportes pudiendo demostrar

que la mayoría de las soluciones existentes no son aplicables al campo de acción por las siguientes causas:

1. Son tecnologías propietarias.
2. Se ajustan a las necesidades y características de las instituciones que las desarrollaron.
3. No están orientadas a todo tipo de usuarios.

De igual forma se analizaron las herramientas y tecnologías establecidas por el CENIA, profundizando así los conocimientos necesarios para el desarrollo de la propuesta de solución al problema planteado, lo cual conlleva a utilizar a PHP 5.3 como lenguaje de programación, GUUD 1.0 como marco de trabajo, PostgreSQL 8.4.1 como Sistema Gestor de Base de Datos, Apache 2.2.2 como servidor web, NetBeans 6.7.1 como Entorno de Desarrollo Integrado, Evolus Pencil 1.2.1 para el diseño de prototipos de interfaz, Visual Paradigm 8.0 como herramienta de modelado y PgAdmin III 1.10.5 como herramienta gráfica para la administración del PostgreSQL.

Capítulo 2. Descripción y análisis de la propuesta de solución

2.1 Introducción capitular

En este capítulo se presenta la propuesta de solución al problema planteado en el diseño teórico de la investigación. Para ello se desarrolla el modelado del dominio y se describen los conceptos fundamentales asociados a este. Se especifican además los usuarios que tendrán acceso a la misma, los requerimientos funcionales y no funcionales y se muestran las técnicas utilizadas para su obtención. Por último, se realiza la descripción de la arquitectura, presentando luego el modelo de datos y la distribución física o diagrama de despliegue de dicha propuesta.

2.2 Modelo de dominio

Debido a que la propuesta que se presentará más adelante formará parte de otro sistema, el Sistema de Gestión Académica de Pregrado (SGAP), y a que básicamente su función será la de obtener y filtrar información gestionada por otros módulos de dicho sistema, no existe un negocio bien definido. Lo anterior, unido a que este sistema está igualmente en desarrollo, ha propiciado que se haga un modelado de dominio en lugar del negocio. El objetivo del modelado del dominio es capturar, comprender y describir las clases más importantes dentro del contexto del sistema. El Modelo del Dominio, junto al Diccionario de Clases del Dominio, ayuda a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común. En la figura 1 se muestra el Diagrama de Clases del Dominio.

Por su parte, el Diccionario de Clases del Dominio describe textualmente las clases identificadas durante el modelado del dominio del problema. Este diccionario sirve como glosario de términos y se muestra a continuación:

- **Usuario:** representa a todas las personas autenticadas en el sistema y que acceden al módulo con el fin de crear determinado tipo de contenido.
- **Contenido:** elemento que puede ser gestionado por los usuarios en el sistema.
- **Reporte:** tipo de contenido que representa un informe que organiza y exhibe la información necesitada por el usuario. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar.
- **Carpeta:** tipo de contenido que representa un espacio donde el usuario puede almacenar contenidos siguiendo una estructura jerárquica.
- **Listado:** tipo de reporte que permite obtener un listado con información nominal de personas que cumplen con determinadas especificidades.

- **Cuantitativo:** tipo de reporte para realizar conteos de personas por tantos conceptos[24] como el usuario desee.
- **Cruzado:** tipo de reporte que establece la relación existente entre dos conceptos.
- **Promoción:** tipo de reporte que brinda información referente a los resultados académicos-docentes.
- **XSL:** hojas de estilo que realizan la transformación del reporte utilizando una o varias reglas de plantilla, dichas reglas son interpretadas por un procesador de XSL[25] el cual realiza las transformaciones necesarias colocando el resultado en un archivo de salida.

2.2.1 Diagrama de clases del dominio

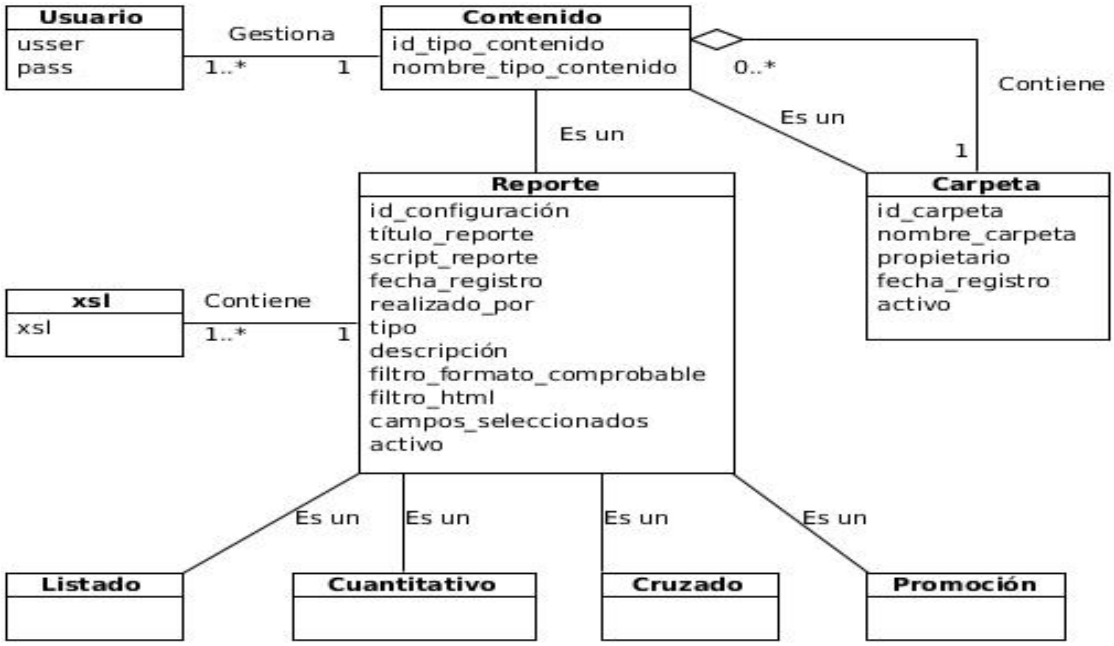


Figura 1. Diagrama de clases del dominio.

2.2.2 Descripción del modelo de dominio

En sentido general, al acceder al módulo el usuario puede gestionar dos tipos de contenido: reportes o directorios. Los reportes pueden ser de tipo *listado*, *cuantitativo*, *cruzado* o de *promoción* y tienen asociado uno o varios XSL quienes son los encargados de transformarlos en diferentes formatos de salida como hojas de cálculo (xls) o formato de documento portátil (pdf). Por su parte, una carpeta o directorio puede contener ambos tipos de contenidos.

[24] En este contexto se refiere campos en la Base de Datos.

[25] Siglas de Extensible Stylesheet Language, expresión inglesa traducible como "lenguaje extensible de hojas de estilo". Es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio.

2.3 Propuesta de solución

Debido a que el SGAP es un sistema con el que interactúan un gran número de usuarios y a la diversidad de estos, se hace necesario un módulo que permita hacer reportes de manera fácil e intuitiva y que garantice la seguridad y confiabilidad de la información manejada por estos.

Por esta razón se propone un módulo para la gestión de reportes basado en tecnología web, que pueda ser integrado a dicho sistema y que cuyas principales funcionalidades son: gestionar reportes y gestionar directorios.

Los reportes a gestionar se clasifican en varios tipos dependiendo de las necesidades del usuario. Además, dicho módulo permitirá elaborar gráficas a partir de los datos obtenidos de los reportes generados por el usuario, así como exportar dichos datos a diferentes formatos como pdf y xls.

Este módulo estaría integrado al SGAP que a su vez forma parte del SGU. Esta integración permite el uso de un conjunto de facilidades como la reutilización de varios componentes implementados por otros subsistemas y la eliminación de redundancias en el código ya que la arquitectura del sistema tiene concebidos mecanismos para que cualquier módulo de cualquier subsistema pueda acceder a funcionalidades y componentes dentro del SGU. Precisamente la solución que se propone utiliza elementos necesarios de otros módulos del SGAP y del núcleo del SGU, de ahí que no sea necesario implementar un gran número de funcionalidades como las relacionadas con la autenticación del usuario y la gestión de roles. Para mayor información remitirse al acápite 2.3.3.

En sentido general, el módulo que se propone debe permitir a los usuarios abstraerse de conocimientos relacionados con los gestores de bases de datos y proporcionar un 100 % de confiabilidad de la información contenida en los reportes, ya que, en ocasiones esta puede ser sensible para la gestión académica. De igual forma, debe garantizar que toda acción realizada sea ejecutada en tiempos inferiores a 5 segundos.

Características de los reportes

Básicamente, en el módulo que se propone todos los reportes tendrán carácter dinámico, es decir, la información que mostrará un reporte será personalizada por el usuario y además esta se actualizará por sí sola cuando así lo requiera. Esto significa que, a diferencia de Akademos, si alguien crea un reporte con los estudiantes suspensos en el 2do semestre de 4to año de un curso académico determinado, en el próximo curso no se necesitará volver a crear el mismo reporte. Además, cada reporte que se cree es editable.

2.3.1 Usuarios que tendrán acceso al módulo

Dado que el módulo que se propone permitirá obtener información, a veces sensible, relacionada con la formación de los estudiantes, solo será visible para el rol *"Pregrado_Reporte"*. A continuación se definen los usuarios o grupos de usuarios asociados a este rol:

- Directivos de las facultades (Decano, Vicedecanos, Directores de Centros de Desarrollo, Subdirectores de Formación, Jefes de Departamentos y Secretarias Docentes).
- Directivos de la universidad (Vicerrectoría, Secretaría General).
- Otros directivos (definidos por Secretaría General o Vicerrectoría de Formación).

Esto se logrará mediante la integración con el módulo Seguridad del SGU que es horizontal para todos los sistemas que lo conforman, incluyendo el SGAP. Para mayor información consultar al acápite 2.3.3.

2.3.2 Flujo de acciones en el sistema

Como primera acción en el sistema el usuario deberá autenticarse. Según los permisos que tenga, podrá entonces acceder a los diferentes módulos. Si una vez autenticado, el usuario tiene defino el acceso al módulo de reportes, entonces puede acceder a este desde la portada principal. Después de esta acción el usuario podrá:

- Gestionar directorios (crear, renombrar, cortar, copiar, pegar y eliminar).
- Gestionar reportes (crear, modificar, cortar, copiar, pegar y eliminar).

Para ello es necesario tener en cuenta las siguientes consideraciones o restricciones iniciales:

1. Para poder crear un tipo de contenido (reportes o directorios) el usuario debe estar autenticado.
2. Al crear un reporte o un directorio el usuario se convierte en su propietario.
3. Un reporte solo puede ser creado dentro de un directorio.
4. Si al crear un reporte, el usuario no define un nombre para este, entonces se creará como un reporte temporal.
5. Los reportes temporales solo se pueden consultar en el momento de su creación puesto que como su nombre lo indica no persistirán en el tiempo.
6. Un usuario puede consultar cualquier reporte aunque no sea propietario de este.
7. Los reportes y directorios solo pueden ser movidos, modificados o eliminados por su propietario.
8. Un directorio no puede ser copiado o movido dentro de sí mismo.

2.3.3 Integración de la propuesta de solución al SGU

El SGU es un sistema que está compuesto a su vez por los subsistemas Ingreso, Pregrado (SGAP), Residencia, Investigación, Producción, Cooperación, Laboratorios, Biblioteca, Extensión, Teleformación, Egreso y Postgrado que agrupan las diferentes áreas de procesos de la UCI. Este

sistema cuenta con módulos que son horizontales[26] para todos los subsistemas entre los que se encuentran: *Seguridad*, *Configuración* y *Trazas*. A continuación se describe cada uno de estos módulos y su relación con la solución que se propone.

Seguridad: permite la autenticación de los usuarios y gestiona los permisos de estos sobre las funcionalidades del sistema. Para acceder al módulo Reportes del SGAP, el usuario debe estar autenticado y además debe tener permisos sobre las funcionalidades del mismo.

Configuración: gestiona la información necesaria para realizar las configuraciones del sistema, tanto globales, como las que son usadas por más de un subsistema. Permite además, realizar acciones de exportación, importación y desinstalación e instalación de subsistemas o módulos. La propuesta de solución puede ser configurada en este módulo.

Trazas: gestiona todo lo referente a las incidencias de un usuario sobre el sistema, registrando el usuario, la acción realizada y el momento en que se ejecutó. Las acciones de los usuarios sobre la solución que se propone serán almacenas por este módulo.

Por su parte el SGAP está conformado por los módulos: *Carrera*, *Personal y Secretaría*, *Tesis y Títulos*, *Control Docente* y *Estudiante*. En *Carrera* se gestiona el proceso de organización de las carreras definidas por el MES[27], acorde a las necesidades de la entidad (planes de estudio, homologación de plan de estudio), además de realizar ajustes personalizados a los estudiantes. *Personal y Secretaría* gestiona todo el personal vinculado con los procesos de pregrado. También posibilita la realización de movimientos de estudiantes (traslados, bajas, licencias), así como acciones de secretaría (promoción de estudiantes y registros de datos docentes para los profesores). Por su parte *Control Docente* gestiona entre otras cosas los grupos docentes y todo el registro de las evaluaciones docentes y asistencia de los estudiantes. *Tesis y Títulos* es el módulo encargado de gestionar todo el proceso de desarrollo de una tesis de pregrado (asignación, seguimiento y control) y el otorgamiento de los títulos. Por último, el módulo *Estudiantes* permite a los estudiantes dar seguimiento a sus resultados en el proceso docente. Todos estos módulos tienen una estrecha relación con el módulo que se propone puesto que como se mencionaba en la introducción de la presente investigación, básicamente la función del mismo será la de construir reportes utilizando información que es provista por estos. Para lograr esto, la solución que se propone se comunica con todos los módulos anteriormente descritos mediante el IOC (Control de Inversiones) que es un mecanismo que permite que las funcionalidades de un módulo puedan ser llamadas en otros módulos.

[26] En este contexto significa que sus funcionalidades son aplicables a todos.
[27] Ministerio de la Educación Superior.

2.3.4 Requerimientos obtenidos

Una vez conocido todo esto, es preciso analizar: ¿Qué debe hacer o qué condiciones debe cumplir el sistema para que se cumplan los objetivos trazados?, para ello se definen los requerimientos del módulo.

¿Qué es un requerimiento?

Son varias las definiciones del término *requerimiento* emitidas por algunos de los autores más reconocidos en cuanto a temas relacionados con la Ingeniería de *Software*. Por ejemplo, Ian Sommerville[28] plantea:

"*Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema. El término requerimiento no se utiliza de una forma constante en la industria de software. En algunos casos, un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de* este *(requerimientos del usuario o cliente). En el otro extremo, es una definición detallada y formal de una función del sistema (requerimientos del sistema o producto)*" [SOM05].

Por otra parte, la IEEE[29] (1999) define un requerimiento como: "*Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente"* [I3E99].

Mientras que para Ralph R. Young este término hace referencia a: "C*apacidad, característica o factor de calidad de un sistema mediante el cual se pretende cumplir con restricciones operativas o ciertas necesidades. Aportando valor y utilidad a un cliente o usuario en el marco de solución de un problema"* [ROW01].

Evidentemente, todas las definiciones emitidas por estos autores poseen varios puntos de convergencia, pero, en el contexto de esta investigación se asume la definición dada por Ian Sommerville.

A menudo, los requerimientos de sistemas de *software* se clasifican en funcionales y no funcionales.

2.3.4.1 Definición de las técnicas de obtención de requerimientos

La extracción u obtención de requerimientos es una de las cuatro actividades que define la Ingeniería de Requerimientos (IR). Ayuda a reconocer la importancia que tiene para el desarrollo de un proyecto

[28] Profesor de la Universidad de St. Andrews en Escocia, presidente de la División de Informática del Instituto de Ingeniería Eléctrica del Reino Unido y autor del principal libro de referencia de la Ingeniería de *Software*.

[29] La IEEE (*Institute of Electrical and Electronics Engineers,* en español Instituto de Ingenieros Eléctricos y Electrónicos) es una asociación técnico-profesional mundial dedicada, entre otras cosas, a la estandarización.

de *software*, realizar una especificación y administración adecuada de los requerimientos de los clientes o usuarios [PRE10]. Existen varias técnicas propuestas por la IR para obtener requerimientos de *software*, en su mayoría basadas en que la aceptación del sistema dependerá de cuán bien este satisfaga las necesidades del cliente. A continuación se describen las técnicas utilizadas.

- **Entrevistas**: se realizaron conversatorios entre analistas y clientes con el objetivo de entender el dominio del problema y sus necesidades. Se basaron en un formato de preguntas y respuestas, buscando obtener las opiniones de los clientes entrevistados sobre qué esperaban del sistema. Estos conversatorios se realizaron tanto de forma personal como en grupo (en el **Anexo 2** se muestra el modelo de entrevista utilizado).
- **Prototipos:** un prototipo es una versión inicial de un sistema de *software* que se utiliza para demostrar los conceptos, probar las opciones de diseño y de forma general enterarse más acerca del problema y sus posibles soluciones. Se mostraron una serie de prototipos al cliente, quienes proporcionaron los requerimientos adicionales. Posteriormente, se realizó este proceso de forma iterativa y haciendo cambios a la aplicación hasta que la misma pudo cumplir con las necesidades del cliente.
- **Sistemas existentes:** se analizaron distintos sistemas ya desarrollados y que están relacionados con la gestión de reportes. De estos se analizaron las interfaces de usuario, observando el tipo de información que se maneja y cómo es manejada, así como las distintas salidas que estos producen. A partir de este análisis se encontraron algunas funcionalidades e interfaces que sirvieron de base para la propuesta de solución.

2.3.4.2 Requerimientos del cliente

- Permitir la creación y manipulación de reportes y directorios.
- Visualizar información nominal de los estudiantes.
- Realizar conteos de estudiantes de acuerdo a determinados criterios.
- Visualizar el avance docente de los estudiantes.
- Consultar y exportar reportes previamente creados.

2.3.4.3 Requerimientos del producto

- La interfaz de usuario se implementará con HTML 4.01 y JQuery 1.3.2.
- Para la creación de reportes y directorios, el reportador contará con un asistente el cual guiará todo el proceso y lo simplificará tanto como sea posible. Este asistente puede mostrar varias etapas en dependencia de las características y complejidad del elemento que se está creando.

2.3.4.4 Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe realizar, es decir, define qué es lo que el sistema debe hacer [SOM05]. Teniendo en cuenta el planteamiento anterior y de acuerdo con los objetivos planteados en la presente investigación, el sistema debe ser capaz de:

RFR1- Crear reporte de tipo listado.
RFR2- Modificar reporte de tipo listado.
RFR3- Ver detalles de reporte de tipo listado.
RFR4- Mostar reporte de tipo listado.
RFR5- Crear reporte de tipo cuantitativo.
RFR6- Modificar reporte de tipo cuantitativo.
RFR7- Ver detalles de reporte de tipo cuantitativo.
RFR8- Mostar reporte de tipo cuantitativo.
RFR9- Crear reporte de tipo cruzado.
RFR10- Modificar reporte de tipo cruzado.
RFR11- Ver detalles de reporte de tipo cruzado.
RFR12- Mostrar reporte de tipo cruzado.
RFR13- Crear reporte de tipo promoción.
RFR14- Modificar reporte de tipo promoción.
RFR15- Ver detalles de reporte de tipo promoción.
RFR16- Mostrar reporte de tipo promoción.
RFR17- Crear carpeta.
RFR18- Modificar carpeta.
RFR19-Ver detalles de carpeta.

2.3.4.5 Especificación de requerimientos funcionales

Con el objetivo de presentar los requerimientos funcionales de manera consistente y entendible se realizó la especificación de los mismos. Donde se definieron secciones o escenarios para describir detalladamente cada requerimiento, las acciones que se deben llevar a cabo por el usuario para cumplir con esa funcionalidad en el sistema, así como la prioridad del mismo otorgada por el cliente y la complejidad del desarrollo de dicha funcionalidad. Para ver dichas especificaciones consultar el **Anexo 5**.

2.3.4.6 Plan de iteración

El Plan de iteración tiene como entrada las especificaciones de requerimientos descritas anteriormente. En el mismo se establecen las iteraciones necesarias para desarrollar el producto y se definen qué requerimientos se deben implementar en cada una de estas. El objetivo de cada iteración es obtener una versión funcional del sistema que, aunque no cuente con todos los requerimientos definidos, constituye un resultado de valor para el cliente. A continuación se muestra dicho plan.

Tabla 1. Descripción del Plan de iteración.

Iteración	Descripción	Requerimientos a implementar	Duración total
Iteración 1	En esta iteración se van a implementar los	RFR1, RFR2, RFR4, RFR5 , RFR6, RFR8, RFR9, RFR10,	01/09/2011 – 01/12/2011

	requerimientos que tienen una prioridad alta para el cliente.	RFR12, RFR13, RFR14, RFR16	(3 meses)
Iteración 2	En esta iteración se van a implementar los requerimientos que tienen una prioridad media para el cliente.	RFR17, RFR18	05/01/2012 – 20/01/2012 (15 días)
Iteración 3	En esta iteración se van a implementar los requerimientos que tienen una prioridad baja para el cliente.	RFR3, RFR7, RFR11, RFR15, RFR19	25/01/2012 – 25/02/2012 (1 mes)

2.3.4.7 Requerimientos no funcionales

Los requerimientos no funcionales, como su nombre sugiere, son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a propiedades emergentes de este como fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento [SOM05].

Los requerimientos no funcionales no solo se refieren al sistema *software* a desarrollar, en algunos casos pueden restringir el proceso que se debe utilizar para desarrollar el sistema. Ejemplos de estos son la especificación de los estándares de calidad que se deben utilizar en el proceso, una especificación que el diseño debe producir con una herramienta CASE[30] particular y una descripción del proceso a seguir. Estos surgen de las necesidades del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas de *software* o *hardware*, o a factores externos como regulaciones de seguridad o legislaciones sobre privacidad. Para ver el listado de los requerimientos no funcionales de la propuesta de solución consultar el **Anexo 4**.

2.4 Descripción de la arquitectura

Actualmente en la literatura es posible encontrar numerosas definiciones del término arquitectura, cada una con planteamientos diversos.

Entre las definiciones más reconocidas se encuentra la de Paul Clements: *"La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema"* [REY06].

[30] Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Computadora) son diversas aplicaciones informáticas que ayudan a realizar un grupo de tareas del clico de vida del proceso de desarrollo de *software*.

De manera similar Bass en [BAS03] la define de la siguiente forma: *"La Arquitectura de Software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos."*

Por otra parte, la definición que más reconocida internacionalmente por muchos arquitectos tributa a la industria y pertenece a la IEEE 1471, la cual cita así: *"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución"* [REY06].

A pesar de la abundante cantidad de definiciones que existen de Arquitectura de *Software*, la gran mayoría coincide en que esta se refiere a la estructura a grandes rasgos del sistema (alto nivel de abstracción), estructura consistente en componentes y relaciones entre ellos. Por lo descrito anteriormente, la presente investigación define Arquitectura de *Software* como la estructura y organización de los componentes de *software* de un sistema informático, cómo y en qué condiciones y configuraciones se van a comunicar dichos elementos.

Esta definición destaca el papel de los "componentes de *software*" en cualquier representación arquitectónica. De manera formal, la Especificación Unificada del Lenguaje de Modelado emitida por el *Object Management Group*[31] define en [OMG11] a un componente como *"una parte modular desplegable y reemplazable de un sistema que encapsula implementación y expone un conjunto de interfaces".*

2.4.1 Niveles de abstracción de la arquitectura

Los niveles de abstracción que componen la Arquitectura de *Software*, de lo general a lo particular, están representados por: estilos arquitectónicos, patrones arquitectónicos y patrones de diseño. Existe una diferencia entre ellos que debe marcarse a fin de evitar las grandes confusiones que inevitablemente, concluyen en el mal entendimiento y en los resultados poco satisfactorios. En este sentido, puede decirse que:

- El **Estilo Arquitectónico** es el encargado de:
  - Describir la estructura general de un sistema, independientemente de otros estilos.
  - Definir los componentes del sistema, su relación e interactividad.
- El **Patrón Arquitectónico** es el nivel en el cual la arquitectura de *software*:
  - Define la estructura básica de un sistema, pudiendo estar relacionado con otros patrones.

[31] Es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos.

- Representa una plantilla de construcción que provee un conjunto de subsistemas aportando las normas para su organización.
- El **Patrón de Diseño** es el tercer nivel de abstracción de la arquitectura de *software*, cuya finalidad es la de precisar en detalle los subsistemas y componentes de la aplicación.

2.4.2 Estilos y patrones arquitectónicos

Como se mencionó en el acápite 2.2, la propuesta de solución está concebida como un módulo del SGAP. Por esta razón se ajustará a la arquitectura definida para dicho sistema, la cual propone como estilo arquitectónico una Arquitectura Cliente-Servidor y como patrón arquitectónico el Modelo-Vista-Controlador (MVC), que a su vez es el patrón de arquitectura base que utiliza el marco de trabajo GUUD, estructura de soporte en la cual se está desarrollando este sistema.

2.4.2.1 Arquitectura Cliente-Servidor

El modelo cliente-servidor es una arquitectura en donde el sistema se organiza como un conjunto servicios y servidores asociados, más unos clientes que acceden y usan dichos servicios. Sus principales componentes son [SOM05]:

- Un conjunto de servidores que ofrecen servicios a otros subsistemas. Ejemplos de servidores son: servidores de impresoras que ofrecen servicios de impresión, servidores de ficheros que ofrecen servicios de gestión de ficheros y servidores de compilación que ofrecen servicios de compilación de lenguajes de programación.
- Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores. Estos son normalmente subsistemas en sí mismos. Puede haber varias instancias de un programa cliente ejecutándose concurrentemente.
- Una red que permite a los clientes acceder a estos servicios. Esto no es estrictamente necesario ya que los clientes y los servidores podrían ejecutarse en una única máquina. En la práctica, sin embargo, la mayoría de los sistemas cliente-servidor se implementan como sistemas distribuidos[32].

Los clientes pueden conocer los nombres de los servidores disponibles y los servicios que estos proporcionan. Sin embargo, los servidores no necesitan conocer la identidad de los clientes o cuántos clientes tienen. Los clientes acceden a los servicios proporcionados por un servidor a través de llamadas a procedimientos remotos usando un protocolo de petición-respuesta tal como el protocolo

[32] Colección de computadoras separadas físicamente y conectadas entre sí por una red de comunicaciones distribuida que se comunican y coordinan sus acciones para conseguir un objetivo común.

HTTP usado en la WWW[33]. Básicamente, un cliente realiza una petición a un servidor y espera hasta que recibe una respuesta.

La ventaja más importante de este modelo es que es una arquitectura distribuida. Es fácil añadir un nuevo servidor e integrarlo con el resto del sistema o actualizar los servidores de forma transparente sin afectar al resto del sistema.

2.4.2.2 Modelo-Vista-Controlador (MVC)

Modelo-Vista-Controlador (MVC) es un patrón que ayuda a darle cierta estructura lógica a las aplicaciones porque separa la lógica de aplicación de la presentación, manteniendo una clara separación de la lógica de negocios, presentación y acceso a datos. Permite flexibilidad y facilidad a la hora de hacer futuros cambios [SOM05]. De manera general su funcionamiento se puede representar como se muestra en la figura 2.

En este patrón se identifican tres componentes fundamentales que se relacionan entre sí: el modelo, el controlador y la vista.

Según Sommerville en [SOM05] el **modelo** es el responsable de:

- Acceder a la capa de almacenamiento de datos.
- Define las reglas de negocio (la funcionalidad del sistema).

Por su parte el **controlador** es quién:

- Recibe los eventos de entrada.
- Contiene reglas de gestión de eventos, del tipo "Si evento *X*, entonces acción *Y*". Estas acciones pueden suponer peticiones al modelo o a las vistas.

Mientras que las **vistas** son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Mantener el aspecto de la página que se mostrará al usuario.

[33] La WWW (acrónimo de World Wide Web o Red informática mundial) es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet.

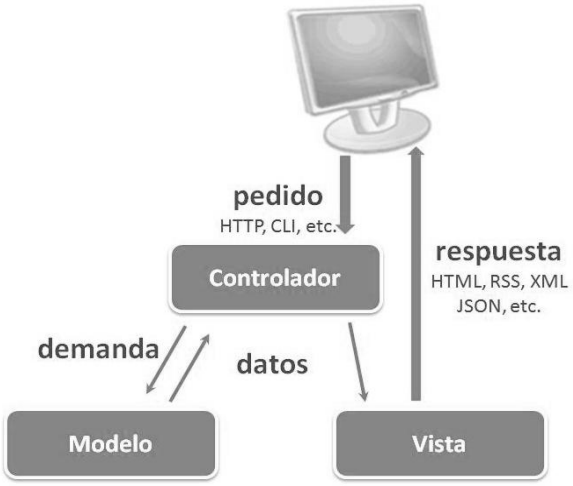


Figura 2. Patrón Modelo-Vista-Controlador.

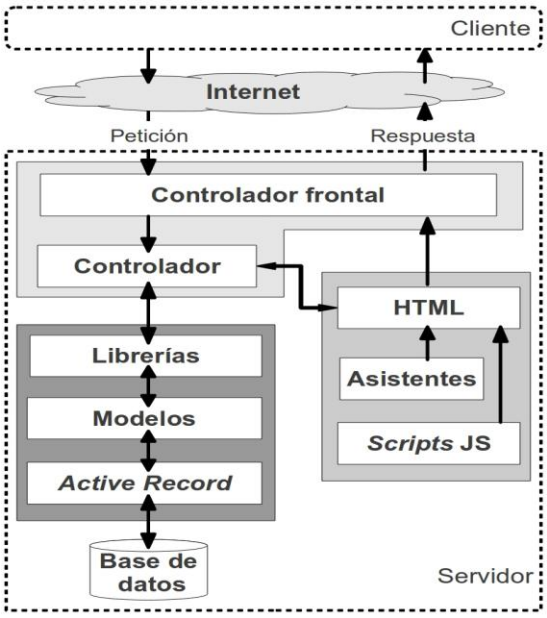


Figura 3. Implementación que realiza GUUD del MVC.

Implementación del MVC que realiza GUUD

Debido a la flexibilidad del MVC, no todos los sistemas que emplean este patrón lo hacen de la misma forma. GUUD implementa el MVC de forma que el desarrollo de aplicaciones sea rápido y sencillo. La figura 3 ilustra la particularidad del MVC implementado por el GUUD y que a continuación se describe.

Este marco de trabajo cuenta con un controlador frontal que inicializa los recursos básicos necesarios para correr el CodeIgniter (parte estructural del GUUD). Una vez realizada una petición HTTP por un cliente, el controlador frontal se encarga de analizar la URI[34] y a partir de esta determina cuál controlador de aplicación (controlador de un determinado módulo) debe ser cargado para atender la petición realizada. Cada controlador de aplicación tiene asociada una o varias librerías responsables de procesar los datos e implementar la lógica del negocio inherente a las acciones relacionadas con dicho controlador. De manera similar cada librería tiene asociado uno o varios modelos encargados del acceso a los datos.

Cuando un controlador de aplicación es cargado, este examina la petición para determinar si solo debe cargar una vista determinada o si es necesario interactuar con la base de datos. En este último caso el controlador de aplicación envía los datos recibidos a la o las librerías. Estas a su vez cargan los modelos necesarios para obtener, registrar o actualizar en la base de datos la información solicitada o

[34] Un *Uniform Resource Identifier* o URI (en español, Identificador Uniforme de Recurso) es una cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etcétera). Su principal diferencia con el *Uniform Resource Identifier* o URL (en español, Localizador Uniforme de Recurso) es que permite especificar que parte del recurso se solicita (segmentos).

enviada. Para realizar esta tarea los modelos hacen uso de la clase *Active Record*. Esta clase permite consultar la base de datos con mínima codificación y abstrayéndose del gestor que se use. La sintaxis de la consulta es generada por cada adaptador de base de datos. Cuando los datos son obtenidos, se retornan al controlador de aplicación en un proceso inverso al descrito anteriormente y como se muestra en la figura 3**¡Error! No se encuentra el origen de la referencia.**. Posteriormente, el controlador carga estos datos a archivos escritos en HTML los cuales pueden incluir llamadas a archivos escritos en JavaScript para manejar dinámicamente su contenido o hacer uso de asistentes (*helpers*) para la creación de forma simplificada de código HTML. Finalmente, el resultado obtenido de todo este proceso es enviado al navegador web como respuesta a la petición inicial.

Desarrollar una aplicación siguiendo este patrón arquitectónico tiene muchas **ventajas**:

a) El componente del modelo se desarrolla de manera independiente del componente vista.
b) Las modificaciones a las vistas no afectan los otros módulos de la aplicación.
c) Si se desea hacer modificaciones en el modelo como agregar datos o métodos, solo debe modificarse el modelo y las interfaces del mismo, sin afectar la aplicación en su totalidad.

Adicionalmente permite un bajo acoplamiento entre los componentes de modelo, vista y controlador.

2.4.3 Patrones de diseño

El término “patrón" posee disímiles definiciones en la literatura. Por solo citar algunas, se pueden mencionar las siguientes:

“Un patrón proporciona una solución probada a un problema común, documentada en un formato coherente (…)” [ERL09].

“En general, un patrón es un binomio problema-solución en un contexto dado. Un patrón no solo documenta *cómo* una solución resuelve un problema, sino también *por qué* se resuelve, es decir, el fundamento de esta solución en particular (…)” [ZDU05].

Desde el punto de vista de los autores del presente trabajo de diploma, un patrón describe detalladamente una solución probada a un problema recurrente, definiendo el contexto en el cual es aplicable dicha solución y sus consecuencias.

2.4.3.1 Patrones GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns*, Patrones Generales de *Software* para Asignar Responsabilidades) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones [ERL09]. Para el diseño del módulo se tuvieron en cuenta los 5 patrones GRASP: *Experto*, *Creador*, *Controlador*, *Alta cohesión* y *Bajo*

acoplamiento, ya que el marco de trabajo usado busca un máximo rendimiento y flexibilidad en sus soluciones y pone en práctica estos patrones para lograr un sistema reusable y flexible.

Descripción de los patrones empleados

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 en qué clase colocar las funcionalidades de acuerdo con la información que estas necesitan.

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 estar en mayor medida relacionada con la clase. En SGU 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 las 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.

2.4.3.2 Patrones GOF

Los patrones GoF (*Gang of Four*), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de estas y la formación de estructuras de mayor complejidad. Se clasifican en 3 grandes categorías de según sus propósitos: *de creación o creacionales, estructurales* y *de comportamiento* [ERL09].

Los *patrones de creación* abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas.

- **Instancia única** (Singleton): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Un ejemplo de este patrón en la propuesta de solución lo constituye la utilización del IOC. El IOC es una clase única que puede ser accedida desde cualquier módulo para interactuar con cualquier otro módulo.

Los *patrones de comportamiento* estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con dimensión temporal.

- **Mediador** (Mediator): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Las librerías ejemplifican claramente la utilización de este patrón, ya que funcionan como mediadoras entre las clases controladoras y las modelos.
- **Observador** (Observer): Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. La aplicación de este patrón se evidencia en el uso, por parte de las controladoras, de una instancia u objeto de la clase Loader. Esta clase es responsable de cargar los elementos del marco de trabajo (dígase librerías, modelos o asistentes). Cuando uno de estos elementos cambia, entonces el objeto de la clase Loader se encarga de actualizar todas las clases que contienen una instancia del elemento que ha cambiado.

2.4.3.3 Patrones de Bases de Datos (tomados de [BLA10])

Estos patrones le permiten al usuario crear una base de datos más fortalecida ya que definen una guía que especifica cómo debe ser la base de datos.

Llaves subrogadas

Este patrón es muy utilizado pues se decide generar una llave primara única para cada entidad en vez de usar un atributo identificador en el contexto dado. Normalmente, se usa enteros en columnas *identity* o GUID (*Global Unique Identifier*), que está demostrado que no se repiten o que pueden hacerlo con una probabilidad extremadamente baja. Esto permite que las tablas sean más fáciles de consultar por el identificador dado que se conoce el mismo tipo de dato de todos en cada tabla. Mediante este patrón, fueron generados los identificadores de todas las tablas pertenecientes al modelo de datos que se muestra más adelante.

Árbol estructurado

Un árbol es un conjunto de nodos conectados en la estructura de hijo a padre. Un nodo puede tener cero o más nodos hijos pero solo un padre, con excepción del nodo raíz y no existen ciclos, por lo que un camino solo conecta a dos nodos. El patrón o modelo árbol estructurado es usado cuando se necesita diferenciar los nodos hojas (*leaf*), de aquellos que generan una nueva rama (*branch*), porque

ambos tipos de nodos tienen diferentes atributos, relaciones y/o semántica. No pueden existir ciclos, es decir, un hijo no puede ser su propio padre. La generalización tiene cubrimiento total y exclusivo, cada elemento de la entidad *Node*, debe tener su correspondiente elemento en la entidad *Leaf* o en la entidad *Branch* (ver figura 4). El modelo de datos propuesto en el acápite 2.5 basa su estructura fundamentalmente en este patrón.

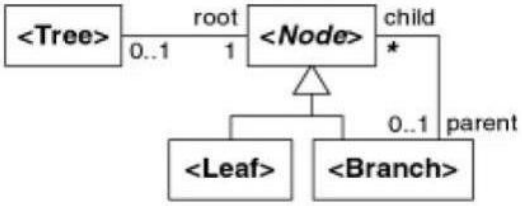


Figura 4. Patrón árbol estructurado (Tomada de [BLA10]).

2.4.4 Arquitectura de la información

La Arquitectura de la Información es la disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactivos y no interactivos. Es un proceso iterativo, transversal, que se da a lo largo de todo el diseño del sitio y en cada una de sus fases, para asegurarse de que los objetivos de su producción y del desarrollo de la interfaz se cumplen de manera efectiva [WOD09].

La solución que se propone muestra el cumplimiento de algunos principios heurísticos como:

- Relación entre el sistema y el contexto del usuario: se emplean etiquetas familiares para los usuarios.
- Uso y control por el usuario: el usuario puede controlar las actividades que se realizan en el sistema. Los objetos, acciones y opciones del mismo están a la vista del usuario.
- Consistencia y estándares: los nombres y de las secciones son consistentes a lo largo de todo el sistema y no existe duplicación en las etiquetas para evitar la desorientación del usuario.

2.4.4.1 Organización de la información

Los reportes son un tipo de contenido que, por razones de organización, estarán ubicados dentro de directorios, como se mencionó en el acápite 2.2.2, estos últimos también pueden contener a su vez otros directorios. Por esta razón toda la información referente a los reportes estará organizada en un árbol de directorios (ver **Anexo 6**) similar al de cualquier Sistema Operativo para que de esta forma el modo de almacenamiento de los reportes le resulte familiar al usuario.

2.5 Modelo de datos

Como ya se había mencionado anteriormente el uso del patrón Modelo-Vista-Controlador permite que se abstraiga la lógica de negocio del almacenamiento de datos. La capa correspondiente al modelo contiene toda la lógica de acceso, dejando a la base de datos como simple almacén de datos sin ninguna lógica, solo algunos disparadores para manejar parte de la seguridad de la misma. Debido a que la base de datos del SGAP está estructurada por esquemas, el modelo de datos de la propuesta de solución cuenta con un total de 5 tablas persistentes agrupadas en el esquema *sq_reportes*. A continuación se muestra el modelo físico de dicho esquema.

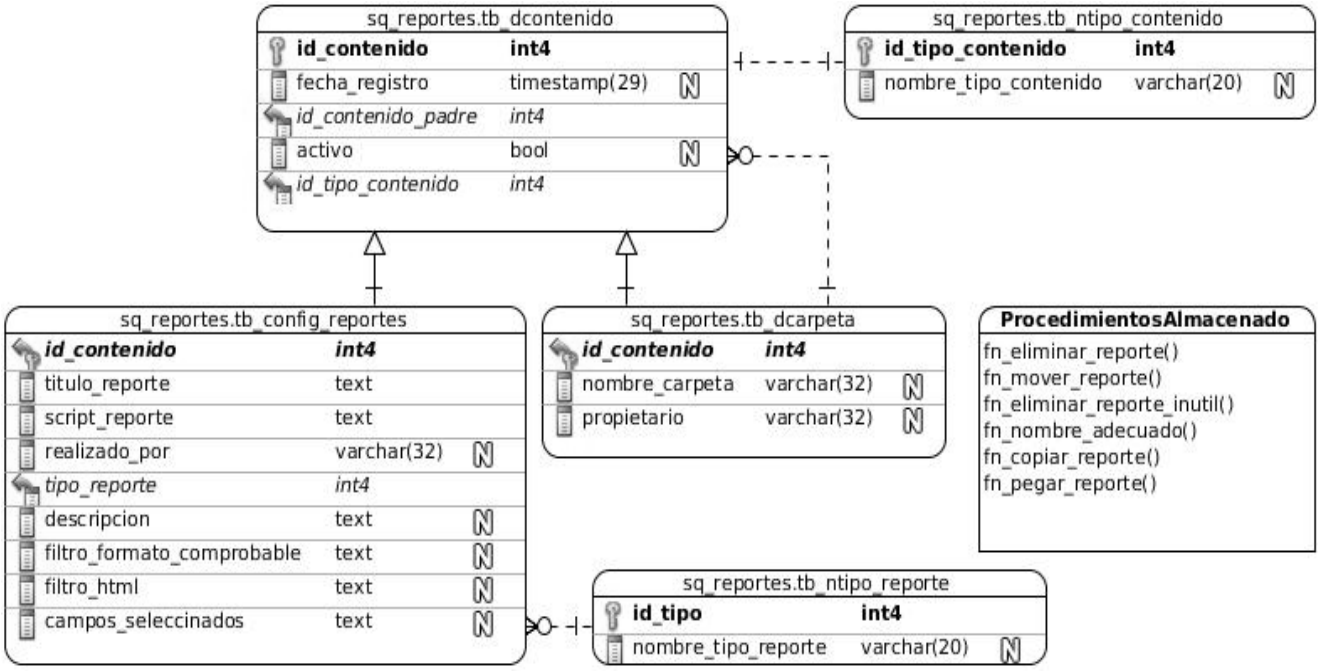


Figura 5. Diagrama de entidad-relación (modelo físico) de la propuesta de solución.

2.6 Modelo de despliegue

Este modelo da una medida de la tecnología necesaria para el correcto funcionamiento del módulo a desarrollar, propone la distribución física de los elementos que lo conforman ya que representa cómo estarán distribuidos y cómo se satisfacen los requerimientos no funcionales de *hardware* y *software*.

Figura 6. Diagrama de distribución física del módulo de reportes.

Descripción de elementos e interfaces de comunicación

- **PC Cliente:** Su función es acceder al módulo e interactuar con el mismo según sus necesidades. Al estar la aplicación desarrollada sobre la web, la máquina cliente necesita disponer de muy pocas prestaciones puesto a que solo necesita un navegador web para poder acceder al sistema y realizar las operaciones necesarias.
- **Servidor aplicaciones:** En este nodo es donde descansa la capa de presentación del sistema, la cual es accedida por las máquinas clientes a través de un navegador web. Contiene además, toda la funcionalidad del sistema.
- **Servidor de Base de Datos:** Es el encargado de almacenar toda la información generada del sistema.
- **<<HTTPS>>:** El Protocolo Seguro de Transferencia de Hipertexto es un protocolo de red basado en HTTP por lo que está orientado a transacciones, sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente y un servidor. La principal diferencia entre ellos es que este está destinado a la transferencia segura de datos de hipertexto, en otras palabras, es la versión segura de HTTP.
- **<<TCP/IP>>:** La familia de protocolos de Internet es un conjunto de protocolos de red en la que se basa Internet y que permite la transmisión de datos entre redes de computadoras. Denominada en muchas ocasiones conjunto de protocolos TCP/IP en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (*TCP*) y Protocolo de Internet (*IP*), que fueron los dos primeros en definirse, y que son los más utilizados de la familia. El TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes Sistemas Operativos sobre redes de área local (LAN) y área extensa (WAN).

2.7 Conclusiones parciales

En este punto quedan establecidos de forma clara la conceptualización, las funcionalidades, las pautas del diseño y la arquitectura en general, la distribución física y el modelo de datos de un módulo que solucionaría el problema científico de la presente investigación, evidenciando la solidez del mismo y la efectividad en el proceso de investigación efectuado. Todo esto tributa a que al concluir este capítulo estén creadas las condiciones necesarias para el desarrollo y validación de la primera versión del módulo Reportes del SGAP.

Capítulo 3. Construcción y validación de la propuesta de solución

3.1 Introducción capitular

Un producto listo para ser entregado, requiere el completo desarrollo y validación de las funcionalidades previamente definidas. En el presente capítulo se describe la implementación de la solución propuesta en el capítulo anterior teniendo en cuenta las técnicas de programación, los estándares de codificación empleados y las clases u operaciones necesarias. Se presenta también, una estrategia de prueba que define la realización de pruebas *unitarias*, de *integración* y del *sistema* para validar que dicha propuesta satisface todas las necesidades del cliente y cumple con todas sus especificaciones.

3.2 Técnicas de programación

Las técnicas de programación constituyen una parte fundamental en el proceso de desarrollo e Ingeniería del *Software* dentro del ámbito informático. Cada técnica tiene sus propias características y distintos métodos para darle solución a determinados problemas, es de gran importancia aprender a implementarlas a la hora de realizar cualquier proyecto de desarrollo de *software*.

3.2.1 Programación modular

La programación modular es una técnica donde los procedimientos con una funcionalidad común son agrupados en diferentes módulos. Cada módulo contendrá sus propios datos, lo que permitirá que cada cual maneje un estado interno que será modificado por las llamadas a procedimientos de estos.
Existe un estado para cada módulo y estos existen aunque sea una vez en todo el programa. Por lo que un programa ya no consistirá en una sección, sino que estará dividido en varias secciones más pequeñas que interactúan a través de llamadas a procedimientos y que integran el programa en su totalidad.
El programa principal coordina las llamadas a procedimientos en módulos separados y pasa los datos apropiados en forma de parámetros.

3.2.2 Programación Orientada a objetos (POO)

La POO es un paradigma o técnica de programación que utiliza la abstracción para crear modelos basados en el mundo real, los *objetos*. Un objeto es un ente real o abstracto con ciertas características y funcionalidades que lo definen. Las propiedades y funcionalidades de un objeto se especifican en su *clase*, la cual vendría a ser el molde de cada instancia particular del este. Con esta técnica un

programa puede ser descompuesto en una serie de objetos que interactúan entre sí para lograr un propósito común [BON03].

La POO está basada en cuatro aspectos: definición de tipos de datos abstractos, herencia, encapsulamiento y polimorfismo.

En la POO encapsular significa que se reúne y controla todo el grupo resultante en un conjunto y no de forma individual. La abstracción es un término externo al objeto, que controla la forma en que es visto por los demás. La herencia se define como una jerarquía de clases derivadas y la relación entre estas, donde se comparte la estructura y el comportamiento de una o más clases consideradas como clases padres. El polimorfismo constituye la definición de múltiples clases con funcionalidades diferentes, pero con métodos o propiedades denominados de forma idéntica [BON03].

3.3 Estándares o convenciones de codificación

¿Qué son las convenciones o estándares de codificación?

Las convenciones o estándares de codificación son un conjunto de directrices que especifican cómo debe escribirse el código fuente. Algunos ejemplos de directrices son aquellas que pautan el nombrado de variables, clases y/o paquetes; la correcta identación del código, cómo escribir los bucles y estructuras de control o incluso qué información incluir en los comentarios.

3.3.1 Convención de nomenclatura

Variables locales:

- Los nombres de las variables se rigen por la nomenclatura *CamelCase*[35], específicamente por el tipo *lowerCamelCase* que es cuando la primera letra de cada palabra se escribe con mayúscula a excepción de la primera palabra que se escribe con minúscula. Ejemplo:

```
$variable;
$variableNombreCompuesto;
```

- Los nombres de algunas variables locales, como los iteradores o los contadores, pueden especificarse en minúscula y de forma abreviada, siempre que su contexto sea específicamente local y su lectura sea intuitiva.

```
Ejemplos: $cont, $i, $j.
```

[35] 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. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.

- Al hacer asignaciones, debe existir un espacio a ambos lados del signo igual (=), esto funciona tanto para asignar un valor fijo, de otra variable o del resultado de una función.

Variables globales (constantes): los nombres de variables globales deben ser siempre en mayúsculas, separando las palabras con guiones bajos ("_"). Ejemplo:

```
define($CONSTANTE, valor);
define($CONSTANTE_COMPUESTA, valor);
```

Clases:

- El nombre debe ser descriptivo, evitando abreviaciones y siempre comienzan con mayúscula. En caso de nombres compuestos, se usará el caracter subrayado "_" para separar las palabras. Ejemplo:

```
Class Clase {
      // Bloque de instrucciones
}
```

```
Class Clase_nombre_compuesto {
      // Bloque de instrucciones
}
```

- La llave de inicio de la clase en la línea siguiente, identada correctamente.

Funciones:

- El nombre debe ser lo más descriptivo posible, evitando el uso de abreviaciones y empleando la convención *lowerCamelCase*.
- Los parámetros o argumentos son separados por espacios luego de la coma que los separa y aquellos con valores por defecto deben ser colocados al final de la lista.
- Siempre intentar retornar un valor significativo.
- La llave de inicio de la función se coloca en la línea siguiente, identada correctamente.

```
function funcion($param1, param2 = array())
{
      // Bloque de instrucciones
}
function funcionNombreCompuesto($param1, param2 = array())
{
      // Bloque de instrucciones
}
```

Ficheros: todo siempre en minúscula, 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 agregándose al final del nombre el sufijo *_mdl*.
- *Librerías:* con el mismo nombre de la clase que representa agregándose al final del nombre el sufijo *_lib*.
- *Controladoras:* con el mismo nombre de la clase que representa.

3.3.2 Etiquetas de bloque PHP

Siempre utilizar **<?php** y **?>** para iniciar y terminar un bloque de código PHP, no las variantes **<?** y **?>** o **<%** y **%>**. Esto asegura compatibilidad entre diversas configuraciones de equipos.

3.2.3 Identación, llaves de apertura y cierre, tamaño de líneas

- Se identará con 4 espacios, sin tabulador, para que cualquier editor de texto reconozca correctamente la identación. Por otro lado, si bien existen editores que realizan corte automático de línea, es recomendable hacerlo en forma manual. Por esta razón la longitud de las líneas será de aproximadamente 75-80 caracteres.
- El uso de llaves de apertura y cierre será siempre en una nueva línea.

3.3.4 Estructuras de control

- Incluye **if**, **for**, **while**, **switch**, **foreach**. Deben tener un espacio entre la palabra clave y el paréntesis de apertura, para diferenciarlos de las llamadas a funciones. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales.
- Si las condiciones son muy largas que sobrepasan el tamaño de la línea, estas se dividen en varias líneas. En el mejor de los casos cuando la condición es muy extensa, se puede dividir esta en variables y compararlas dentro de la estructura de control. Ejemplo:

```
if ($condicion)
{
      // Bloque de instrucciones
}
else
{
      // Bloque de instrucciones
}
```

3.3.5 Documentación

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

```
/**
* Breve descripción de la clase
* @package nombre del paquete o módulo al que pertenece
* @subpackage nombre del subpaquete (si es el caso)
* @category categoría de la clase (controladora, librería, modelo)
* @author nombre y apellidos del autor (se puede agregar el e-mail)
*/
```

De la misma forma las funciones de cada clase deberán ser documentadas como se muestra a continuación.

```
/**
* Breve descripción de la función
* @param tipo de dato y nombre del parámetro (por cada uno)
* @return tipo de dato que retorna
* @author nombre y apellidos del autor (se puede agregar el e-mail)
*/
```

3.3.6 Comentarios

Se aconseja el uso de comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos. Los comentarios pueden ser con fin documental o bien como "ayuda-memoria". Para ello se recomienda utilizar los estilos de C (/* */) y C++ (//).

3.4 Descripción de las clases y operaciones necesarias

Como la característica principal del módulo que se propone es que tiene que estar orientado a usuarios finales, es de vital importancia que el mismo permita a los usuarios abstraerse de conocimientos relacionados con los gestores de bases de datos. Para lograr esto fue necesario implementar la librería *"reporte_config_lib.php"*, la cual, como su nombre lo indica, básicamente establece la configuración para el acceso a los datos. En este caso en la configuración se definen un conjunto de tablas ***T*** de donde se obtendrán los datos y la forma en que se accede a estas, las posibles operaciones entre los elementos de *T* agrupadas en un conjunto ***O***, así como las dependencias ***D*** que existe entre estos. Por último, un conjunto de campos ***C*** disponibles para el usuario.

En el **Anexo 7** se describe detalladamente esta y las restantes clases implementadas.

3.5 Descripción de los algoritmos no triviales utilizados

3.5.1 Validación del filtro

En la propuesta de solución uno de los pasos para la generación de un reporte es la construcción de un filtro que se encargará de obtener solo la información que cumple con determinados criterios de búsqueda. A pesar de que este filtro se construye en un lenguaje muy parecido al lenguaje natural, debe cumplir con determinadas reglas. En su estructura, el filtro puede contener algunos componentes como paréntesis, operadores lógicos[36] y de comparación, campos y valores de estos, los cuales pueden introducir algunos errores (por ejemplo: paréntesis desbalanceados). Por esta razón, se hace necesario validar que el filtro esté correctamente construido. Para esto se utilizó un mecanismo que emplean todos los compiladores[37] para determinar si una sentencia es válida según las reglas de un determinado lenguaje, las *gramáticas*. De manera general una gramática es un ente o modelo matemático que permite especificar un lenguaje, es decir, es el conjunto de *reglas* capaces de generar todas las posibilidades combinatorias de ese lenguaje, y solo las de este lenguaje, ya sea un lenguaje formal o un lenguaje natural [AHO07]. A partir de dicha gramática se implementó la clase *"interprete_filtro"* que se encarga de analizar sintácticamente el filtro en la medida en que este se va construyendo. Analizar sintácticamente el filtro, significa, examinar la secuencia de los elementos que lo componen para determinar si el orden de esa secuencia es correcto de acuerdo con ciertas reglas estructurales (en este caso, las establecidas por la gramática). En el **Anexo 10** se muestra la gramática que define la sintaxis[38] del filtro y una representación de la clase anteriormente mencionada utilizando pseudocódigo[39].

3.5.2 Algoritmo para la construcción de la consulta SQL

Entradas

- Campos seleccionados por el usuario (representan los datos que mostrará el reporte).
- Filtros (representan las condiciones que debe cumplir la información para ser mostrada en el reporte).

Pasos

1. Obtener la configuración del acceso a datos a partir de la librería *reporte_config_lib.php*.

[36] Los operadores lógicos proporcionan un resultado a partir de que se cumpla o no una cierta condición.
[37] Programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje máquina).
[38] Forma correcta en que deben estar dispuestos los elementos que componen una instrucción.
[39] Lenguaje algorítmico que permite representar las construcciones básicas de los lenguajes de programación, pero a su vez, manteniéndose próximo al lenguaje natural.

2. Obtener, a partir de los campos seleccionados por el usuario y por los filtros, las tablas que estarán implicadas en el *script*. Esto es, para cada campo seleccionado y para cada campo de los filtros obtener la tabla a la que hace referencia en la configuración.
3. Obtener, a partir de la configuración, las operaciones a realizar necesarias para relacionar todas las tablas que están implicadas.
4. Retornar la cadena compuesta por: "SELECT" + campos seleccionados + "FROM" + tablas implicadas junto con las operaciones que las relacionan + "WHERE" + filtros.

Salidas

- Se obtiene una cadena con el *script* SQL que será el encargado de obtener de la base de datos la información que el usuario desea.

3.6 Aspectos relacionados con la seguridad del módulo propuesto

La seguridad es uno de los aspectos fundamentales a tener en cuenta a la hora de desarrollar cualquier *software*. Una aplicación con un bajo nivel de seguridad se vuelve vulnerable a los ataques de individuos malintencionados. Por esta razón, en este acápite se describen elementos que contribuyen a elevar el nivel de seguridad del módulo propuesto.

Privilegios

Como se mencionó en el capítulo anterior, los tipos de contenidos que se gestionan en el módulo (reportes y directorios) solo pueden ser movidos, modificados o eliminados por su propietario, es decir, por la persona que los creó. De esta forma se garantiza la integridad de la información.

Protección contra ataques XSS (*Cross-site scripting*)

El *Cross-site scripting* es un tipo de ataque típico de las aplicaciones web que permite a un atacante inyectar código HTML y/o javascript en páginas web vistas por el usuario con el objetivo de conseguir algún provecho. Este ataque se basa en la explotación de las vulnerabilidades que poseen algunos sistemas en la validación de los campos de un formulario [SCA11]. El módulo propuesto cuenta con los siguientes mecanismos para garantizar la protección contra ataques XSS:

- Validación de formularios en el cliente. Todos los formularios antes de ser enviados ejecutan primero una función que valida todos sus campos. Dicha función define un conjunto de reglas que se deben cumplir para cada campo y que determinan si la información contenida en estos es válida. Estas reglas pueden restringir los caracteres que son admitidos, si el campo es requerido o no, la longitud mínima y máxima del valor del campo, así como la longitud mínima y máxima de cada palabra (solo en el caso de los campos de texto). En caso de que un campo contenga algún

error el formulario no es enviado, se indica al usuario qué campo es incorrecto y cuál regla ha sido violada.

- Validación de datos del lado del servidor. Todos los datos enviados al servidor por un formulario son validados para asegurarse que conforman el tipo correcto, el largo correcto y que no contienen etiquetas HTML o PHP.
- Filtro XSS. CodeIgniter (parte estructural del GUUD) cuenta con una clase (*Input Class*) que provee un filtro para la prevención de ataques XSS. Esta clase busca técnicas comunes usadas para activar código javascript u otros tipos de códigos maliciosos. El filtro puede ser corrido automáticamente para filtrar los datos *POST* y *COOKIE* que se encuentren o a través de la llamada a la función *xss_clean()* de dicha clase.

Protección contra inyecciones SQL

Una inyección SQL es una técnica de ataque que se vale de la vulnerabilidad de un sistema informático en el nivel de validación de las entradas para realizar consultas a una base de datos. Con esta técnica el atacante puede crear o alterar comandos SQL para exponer datos ocultos, eludir condiciones o restricciones importantes, o ejecutar modificaciones de la base de datos [SCA11]. Para proteger al módulo propuesto de este tipo de ataque se emplearon los siguientes mecanismos:

- Todas las entradas de los usuarios son validadas en el cliente y en el servidor con el objetivo de comprobar si estas poseen la longitud y el tipo esperados. La validación en el cliente verifica además los caracteres permitidos para cada entrada.
- No se realizan conexiones a la base de datos como súper usuario o propietario de esta, sino que se utilizan usuarios con privilegios limitados.
- Cuando ocurre un error de base de datos no es mostrada la respuesta del gestor porque esta suele ser tan específica que revela información sobre la estructura de la base de datos. En lugar de ello, se muestran mensajes predefinidos en el sistema.
- Para la construcción de las consultas SQL se emplea la clase *Active Record* de CodeIgniter. Esta clase posee un grupo de funciones que permiten la creación de consultas o *querys* de manera simple y segura. Estas funciones escapan automáticamente los variables que almacenan los valores entrados por los usuarios, es decir, reemplazan los caracteres especiales en SQL por su equivalente textual, de forma tal que SQL interprete todo el contenido de la variable como si fuera texto.

3.7 Validación de la solución propuesta

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 [SOM05]. 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. A continuación se presenta la validación de la solución propuesta.

3.7.1 Criterios de validación de requerimientos

El proceso de desarrollo con enfoque ágil basado en el nivel 2 de CMMI propone criterios que permiten validar los requerimientos del cliente y del producto, donde se responde a varias interrogantes que validan si el requerimiento se aprueba o no.

3.7.1.1 Interrogantes para validar los requerimientos del cliente

- ¿El proveedor del requerimiento es un proveedor válido?
- ¿El requerimiento está identificado como único?
- ¿El requerimiento es modificable?
- ¿El requerimiento no es ambiguo?
- ¿El requerimiento está completo?
- ¿El requerimiento es congruente con otros requerimientos relacionados?
- ¿El requerimiento puede ser implementado?
- ¿El requerimiento puede ser probado?
- ¿El resultado de la evaluación de impacto es positivo?
- ¿El requerimiento está correcto?
- ¿El requerimiento es traceable?

3.7.1.2 Interrogantes para validar los requerimientos del producto

- ¿Están identificados los elementos de entrada?
- ¿Están identificados los elementos de salida?
- ¿El requerimiento es dado por el superior determinado en el organigrama del proyecto?
- ¿El requerimiento no es ambiguo?
- ¿Es técnicamente factible?
- ¿Puede ser verificado?
- ¿Está correcto?
- ¿El resultado de la evaluación de impacto es positivo?
- ¿El requerimiento es traceable?

Con la aplicación de los criterios para validar requerimientos del cliente y del producto quedaron aprobados los requerimientos en su totalidad.

3.7.2 Técnicas de validación de requerimientos

En [SOM05] Sommerville define un grupo de técnicas que permiten que el proceso de validación tenga una mejor calidad, de ellas se utilizaron las que a continuación se relacionan.

- **Revisión de requerimientos:** se realizaron reuniones entre varios miembros del proyecto fundamentalmente analistas donde se revisaron los documentos de especificación de requerimientos y se detectaron un número de inconsistencias como anomalías, omisiones u otros tipos de errores, a las que se les dio solución.
- **Construcción de prototipos:** se mostró un modelo ejecutable del sistema a los clientes para que los mismos interactuaran con este y determinaran si cumplía o no con sus necesidades.
- **Generación de casos de prueba:** se realizaron los diseños de casos de pruebas para cada uno de los requerimientos especificados, permitiendo verificar que todos se pudieran probar, e identificar dependiendo de la complejidad del diseño de caso de prueba, requerimientos que deberían ser reconsiderados y cuáles pueden ser los más difíciles de implementar.

3.7.3 Resultado de la revisión de los requerimientos

Durante la revisión de requerimientos se realizaron verificaciones en el documento “Especificaciones de requerimientos”, este proceso comprendió las:

- Verificaciones de validez: los requerimientos deben cumplir con las necesidades del cliente. Luego de realizar algunos análisis y razonamientos surgieron funciones adicionales y cambios en las que ya estaban identificadas.
- Verificaciones de consistencia: los requerimientos no deben contradecirse es las especificaciones escritas, no debe haber restricciones o descripciones que estén opuestas a las reglas definidas.
- Verificaciones de completitud: los requerimientos deben incluir todas las funcionalidades propuestas por el cliente, satisfacer de manera general todas las necesidades acordadas.
- Verificabilidad: para evitar posibles discusiones entre los miembros del equipo del proyecto y el cliente, se revisó que los requerimientos estuvieran descritos de manera que puedan ser verificables, o sea, que se puedan diseñar casos de pruebas orientadas a estos y que demuestren que el sistema a entregar responde a las necesidades del cliente.

Al concluir el proceso de revisión se detectaron algunas inconsistencias que fueron erradicadas de inmediato. Entre las más comunes se pueden citar:

- Se interpretaron de forma incorrecta algunas de las funcionalidades y características solicitadas por el cliente.

- Descripciones poco detalladas de algunos de los requerimientos.
- Errores ortográficos.

3.7.4 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 módulo.

Las pruebas no pueden demostrar que el *software* está libre de defectos o que se comportará en todo momento como está especificado. Como dijo elocuentemente Edsger Dijkstra[40], una de las primeras figuras líderes en el desarrollo de la Ingeniería de *Software*, «las pruebas solo pueden demostrar las presencia de errores, no su ausencia» [SOM05].

3.7.5 Buenas prácticas

A continuación se presentan algunas características de una buena prueba [PRE10]:

- Una buena prueba ha de tener una alta probabilidad de encontrar un fallo. Para alcanzar este objetivo el responsable de la prueba debe entender el *software* e intentar desarrollar una imagen mental de cómo podría fallar.
- Una buena prueba debe centrarse en dos objetivos: *probar si el software no hace lo que debe hacer* y *probar si el software hace lo que no debe hacer.*
- Una buena prueba no debe ser redundante. El tiempo y los recursos son limitados, así que *todas las pruebas deberían tener un propósito diferente*.
- Una buena prueba debería ser la “mejor de la cosecha”. Esto es, se debería emplear la prueba que tenga la *más alta probabilidad de descubrir una clase entera de errores*.
- Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja, pero si se quieren combinar varias pruebas a la vez se pueden enmascarar errores, por lo que en general, *cada prueba debería realizarse separadamente*.

3.7.6 Técnicas de prueba

Las técnicas de prueba o de evaluación dinámica, como también se le conocen, proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas [PRE10]. Estas técnicas se agrupan en:

[40] Edsger Wybe Dijkstra (11 de mayo de 1930 - 6 de agosto de 2002) fue un científico de la computación de origen neerlandés. Entre sus contribuciones a la informática está el algoritmo de caminos mínimos; también conocido como Algoritmo de Dijkstra. También se le debe la autoría de la expresión "Crisis del software", aparecida en su libro *The Humble Programmer* y usada ampliamente en la famosa reunión de la OTAN de 1968 sobre desarrollo del software. Recibió el Premio Turing en 1972.

- *Técnicas de caja blanca o estructurales*, que se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- *Técnicas de caja negra o funcionales*, que realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

La figura 7 representa gráficamente la filosofía de las pruebas de caja blanca y caja negra. Como se puede observar las pruebas de caja blanca necesitan conocer los detalles procedimentales del código, mientras que las de caja negra únicamente necesitan saber el objetivo o funcionalidad que el código ha de proporcionar.

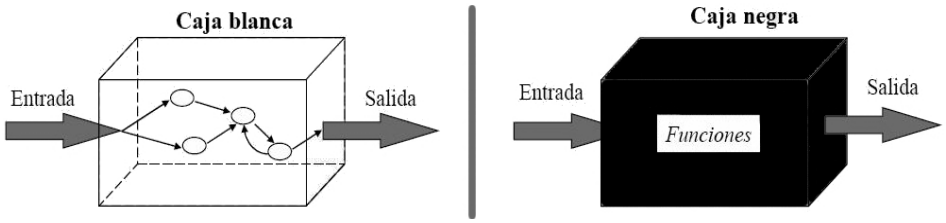


Figura 7. Filosofía de las pruebas de caja blanca y caja negra.

En el proceso de validación de la solución propuesta en la presente investigación se emplearán ambas técnicas de prueba.

3.7.6.1 Pruebas de caja blanca o estructural

A este tipo de técnica se le conoce también como *prueba de caja transparente o de cristal*. 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. Como es de suponer, es poco práctico realizar una prueba exhaustiva de todos los caminos de un programa. Por ello existen ciertas técnicas para el diseño de este tipo de pruebas que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Entre las más usadas se encuentran las siguientes:

- 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 se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe y representa un límite inferior para el número de casos de pruebas que se deben realizar para asegurar que se ejecuta cada camino del programa.

En el ámbito de la presente investigación la técnica empleada para generar los casos de pruebas es la *Prueba del camino básico*. En el **Anexo 8** se muestra el procedimiento que se debe realizar con esta técnica.

3.7.6.2 Pruebas de caja negra

Estas pruebas, también conocidas como *pruebas de comportamiento*, se llevan a cabo sobre la interfaz del *software* y permiten obtener un conjunto de condiciones de entrada que ejercitan completamente todos los requerimientos funcionales del programa [PRE10]. El término "caja negra" se debe a que el comportamiento del componente que se prueba solo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas.

Objetivos: revelar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaz, rendimiento y errores de inicialización y terminación.

Alcance: se centra principalmente en los requerimientos funcionales del *software* para verificar el comportamiento de la unidad observable externamente y la calidad funcional.

Descripción: estas pruebas se llevan 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 prueba de caja negra intenta encontrar errores de las siguientes categorías [PRE10]:

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

Al igual que ocurría con las técnicas de caja blanca, para confeccionar los casos de prueba de caja negra existen distintos criterios. Algunos de ellos son: *Particiones de equivalencia, Análisis de valores límites, Métodos basados en grafos, Pruebas de comparación y Análisis causa-efecto*. En el contexto

de este trabajo de diploma se utilizó el primer criterio (para mayor información sobre el tema consultar el **Anexo 9**).

3.7.7 Estrategia de prueba

La estrategia de prueba del *software* describe el enfoque y los objetivos generales de las actividades de prueba. Esta debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código fuente se han implementado correctamente, así como pruebas de alto nivel que validen las principales funciones del sistema frente a los requerimientos del cliente [PRE10]. La estrategia que se ha de seguir a la hora de evaluar dinámicamente un *software* debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el *software* en su conjunto.

Para los efectos de esta investigación se define una estrategia basada en las técnicas y métodos propuestos por Roger S. Pressman en [PRE10] en la cual se propone realizar:

1. Pruebas unitarias.
2. Pruebas de integración.
3. Pruebas del sistema (rendimiento y resistencia).

A continuación se procede a la descripción y realización de cada una de estas pruebas.

3.8 Pruebas unitarias

La prueba de unidad es la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del *software* de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado. En estas pruebas cada módulo será probado por separado y lo hará, generalmente, la persona que lo creó. En general, un módulo se entiende como un componente *software* que cumple las siguientes características:

- Debe ser un bloque básico de construcción de programas.
- Debe implementar una función independiente simple.
- Podrá ser probado al 100 % por separado.
- No deberá tener más de 500 líneas de código.

Para probar de la manera más completa posible se han de aplicar tanto pruebas de caja blanca como de caja negra. En el caso particular de las pruebas de caja blanca y con el objetivo de verificar el resultado real de la prueba para cada uno de los caminos, se empleó un mecanismo que posee CodeIgniter (parte estructural del marco de trabajo, GUUD en este caso) para la automatización de pruebas unitarias y que se describe a continuación.

Pruebas unitarias en CodeIgniter

CodeIgniter posee una librería o clase especializada en la ejecución de pruebas estructurales. Aunque es bastante sencilla, pues cuenta con una sola función de evaluación y dos funciones de resultados, permite determinar con certeza si un código específico produce el tipo de dato y resultado esperado. Para correr una prueba utilizando dicha librería es necesario suministrar el código a probar y un resultado esperado de la siguiente forma:

```
$this->unit->run(código, resultado esperado, 'nombre de prueba');
```

Donde **"código"** es el segmento de código que se desea probar, **"resultado esperado"** es lo que se espera que devuelva la evaluación del código (puede ser un tipo de dato o un valor literal) y **"nombre de prueba"** es un nombre opcional que se le puede dar a la prueba. Los tipos de datos posibles son: "*is_string*", "*is_bool*", "*is_true*", "*is_false*", "*is_int*", "*is_numeric*", "*is_float*", "*is_double*", "*is_array*", "*is_null*".

3.8.1 Casos de pruebas de caja blanca

En el **Anexo 13.1** se describen detalladamente todos los casos de pruebas de caja blanca realizados a la propuesta de solución y cuyos resultados fueron satisfactorios en su totalidad.

3.8.2 Casos de pruebas de caja negra

Para la realización de las pruebas de caja negra se generaron los artefactos *"Diseño de casos de pruebas basado en requerimientos".* Por cada requerimiento funcional del sistema se generó un documento en donde se recogen todos los datos necesarios para probar la interfaz. En el **Anexo 13.2** se muestra como ejemplo el caso de prueba para el RFR1 Crear reporte de tipo listado. Para ver los diseños de casos de pruebas en su totalidad, consultar el expediente del proyecto Pregrado. La realización de estas pruebas permitió detectar algunas no conformidades la cuales fueron erradicadas inmediatamente. A continuación se muestra la relación de no conformidades por iteración.

Iteración	Cantidad de no conformidades	Asociadas a
1era	15	Errores de interfaz, de validación y de rendimiento.
2da	8	Errores de interfaz, funciones incorrectas o ausentes.
3ra	0	

3.9 Pruebas de integración

Aun cuando los módulos de un programa funcionen bien por separado es necesario probarlos conjuntamente ya que un módulo puede tener un efecto adverso o inadvertido sobre otro. La prueba de integración es una técnica sistemática para construir la estructura de un programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción [PRE10]. Existen dos tipos de integración: *no incremental* e *incremental*. En el primer caso se combinan todos los módulos y se prueba el programa en su conjunto, como es lógico pensar el resultado puede ser caótico con un gran número de fallos y la consiguiente dificultad para identificar el módulo que los provocó. Por su parte, la integración incremental es la antítesis del enfoque no incremental. El programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir. Por esta razón se escogió el enfoque incremental para la realización de las pruebas de integración de la propuesta de solución. Estas pruebas aparecen relacionadas en el **Anexo 12**. Al finalizar las pruebas de integración no fueron detectados errores asociados a la interacción del módulo propuesto con los restantes módulos del SGAP.

3.10 Pruebas del sistema

Este tipo de prueba está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del mismo (*hardware* u otro *software*) y que realizan las funciones adecuadas [PRE10].

3.10.1 Prueba de rendimiento

La *prueba de rendimiento* está diseñada para probar el rendimiento del *software* en tiempo de ejecución dentro del contexto de un sistema integrado. Las pruebas de rendimiento, a menudo, van emparejadas con las pruebas de resistencia y, frecuentemente, requieren instrumentación tanto de *software* como de *hardware*. Es decir, muchas veces es necesario medir la utilización de recursos (por ejemplo, ciclos de procesador) de un modo exacto [PRE10].

3.10.2 Prueba de resistencia

Las *pruebas de resistencia* o *de estrés* como también se le suelen llamar, están diseñadas para enfrentar a los programas con situaciones anormales. Este tipo de pruebas ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales hasta que el sistema falla. Según Pressman, entre sus principales funciones están [PRE10]:

1. Prueba el comportamiento de fallo de ejecución del sistema.
2. Manifestar defectos que normalmente no serían descubiertos.

Debido a que para la realización de las pruebas del sistema se necesitan hacer mediciones exactas, se utilizará la herramienta **JMeter 2.6** descrita en al acápite 1.8.5. A continuación se muestran los resultados obtenidos con la misma y que fueron extraídos de las imágenes relacionadas en el **Anexo 11**.

Tabla 2. Resultados de las pruebas del sistema.

Aplicado a	Usuarios (Muestras)	Tiempos de ejecución (ms)					Rendimiento	
		Mín.	Máx.	Media	Mediana	%Error	Pet/seg	Kb/seg
Registrar reporte (independientemente del tipo)	1	157	157	157	157	0.00	6.4	4.0
	10	199	599	458	515	0.00	5.4	3.4
	100	396	4392	1625	1665	0.00	5.2	3.3
	1000	424	8769	3535	3920	10.00	6.4	5.0
Registrar Carpeta	1	141	141	141	141	0.00	7.1	4.2
	10	128	588	372	252	0.00	7.1	4.2
	100	290	2077	1045	828	0.00	4.2	2.5
	1000	289	5435	2699	2596	7.00	4.3	2.6
Detalles reportes (independientemente del tipo)	1	178	178	178	178	0.00	5.6	57.1
	10	255	520	395	390	0.00	5.5	56.1
	100	260	2058	1429	1651	0.00	4.3	43.7
	1000	197	5561	3770	4095	10.00	4.6	42.4
Detalles carpeta	1	138	138	138	138	0.00	7.2	71.9
	10	319	562	454	436	0.00	4.7	46.8
	100	476	7012	4392	4691	0.00	6.2	61.8
	1000	447	9571	6883	8319	0.00	9.0	89.1
Mostrar reportes (independientemente del tipo)	1	93	93	93	93	0.00	10.8	36.5
	10	294	493	397	396	0.00	4.9	16.7
	100	226	1682	1070	1360	0.00	4.8	16.3
	1000	265	4538	3322	3871	0.00	5.3	17.9

Los valores que se reflejan en la tabla anterior se pueden interpretar de la siguiente forma:
La columna *Mínimo* (*Mín.*) indica el mínimo de tiempo de ejecución invertido para una petición con *n* (columna *Muestras*) usuarios haciendo peticiones de manera concurrente. De manera similar se puede interpretar el resultado de la columna *Máximo* (*Máx.*) con la diferencia de que esta última muestra el mayor tiempo de ejecución para una petición. Como su nombre lo indica, la *Media* representa el tiempo

de ejecución promedio de una petición con *n* usuarios. En el caso de la *Mediana,* su valor significa que el 50% de las peticiones realizadas por *n* usuarios tardaron menos del valor reflejado. El *Error* indica la relación entre el total de peticiones y el número de peticiones que originaron errores. Por último, el *Rendimiento (Pet/seg)* hace referencia al número de peticiones que el servidor puede procesar en un segundo mientras que el *Rendimiento (Kb./seg)* hace referencia a la cantidad de datos que el servidor puede procesar en un segundo.

Resultados de las pruebas del sistema

De manera general en la Tabla 2 se puede observar que las peticiones son ejecutadas en tiempos relativamente rápidos (inferiores a 5 segundos en la mayoría de los casos). En el caso de la ocurrencia de errores no sucede de igual forma ya que cuando existen 1000 usuarios haciendo peticiones concurrentemente, el sistema pierde la capacidad de procesar todas las peticiones y empiezan a aparecer algunos errores. Al analizar los porcientos de errores se puede observar que en los casos más críticos (cuando 1000 usuarios tratan de registrar u obtener los detalles de un reporte) solo ocurre algún tipo de error en el 10 % del total de las peticiones hechas. Esto significa que de las 1000 peticiones hechas, 900 se ejecutan satisfactoriamente. Si se tiene en cuenta que actualmente en la universidad existen solo 278 personas que cumplen con las especificaciones para poder tener acceso al módulo (consultar acápite 2.3.1), se puede asumir entonces que este resultado sigue siendo bueno, ya que 900 es 3.2 veces mayor que 278, lo que indica que el sistema es capaz de soportar 3 veces más carga que en el peor de los casos con el que puede enfrentarse. Como es lógico pensar, en la medida que el número de usuarios conectados sea mayor que 1000, mayor será la probabilidad de ocurrencia de errores. Por tanto, se puede considerar que el límite permisible de procesamiento de peticiones concurrentes es de 1000.

3.11 Conclusiones parciales

Tras el desarrollo y validación de la propuesta de solución, se ha obtenido un módulo capaz de gestionar de forma rápida, intuitiva y amigable para los usuarios, los reportes generados en el SGAP. Las pruebas efectuadas permitieron demostrar que dicha solución cumple con todas las especificaciones dadas por el cliente y además, que el límite de procesamiento de peticiones concurrentes permisible es muy superior (3 veces mayor) al que se enfrentará en el peor de los casos. Todo esto evidencia la fiabilidad y eficiencia de este módulo.

Conclusiones generales

Una vez realizada la fundamentación teórica que sustentó la presente investigación, definidas las características del módulo Reportes del SGAP y efectuado su desarrollo y validación, se obtuvieron resultados que permiten a los autores de la misma arribar a las siguientes conclusiones:

- La evaluación del estado de las soluciones de *software* para la gestión de reportes tanto en la UCI como nacional e internacionalmente permitió identificar que la mayoría de estas no son aplicables al campo de acción por las siguientes causas:
  a. Son tecnologías propietarias o privativas por lo que no están acordes a las políticas de la universidad y del país.
  b. Se ajustan a las necesidades y características de las instituciones que las desarrollaron o para las que fueron desarrolladas.
  c. No están orientadas a todo tipo de usuarios.
- Con la implementación del módulo Reportes quedaron estandarizados los reportes del SGAP relacionados con el proceso de formación de los estudiantes en la universidad y se dotó a dicho sistema con una herramienta para validar la migración de los datos del anterior Sistema Automatizado para la Gestión Académica, Akademos.
- El módulo implementado, gracias a su carácter dinámico, permite que la información de los reportes se actualice por sí sola cuando así lo requiera.
- La realización de pruebas al módulo permitió garantizar que los requerimientos fueron cumplidos, demostrar su capacidad para proporcionar tiempos de respuesta haciendo un uso adecuado de los recursos del sistema y establecer el límite de su capacidad de procesamiento de peticiones concurrentes.
- La propuesta de solución fue avalada por parte del Centro de Informatización Universitaria.

Recomendaciones

1. Extender el uso de la solución propuesta a los demás subsistemas del SGU.
2. Desarrollar futuras versiones del módulo Reportes del SGAP que incorporen posibles mejoras como:
   - Permitir a los usuarios imprimir los reportes que cree en el sistema.
   - Inclusión de un Diseñador de modelos que permita a los usuarios diseñar la forma en que el sistema mostrará un reporte determinado creado por él, no siendo este un paso obligatorio en el proceso de creación de los reportes.
   - Permitir la asignación de permisos de lectura y escritura a nivel de campos. Esto significa que de todos los datos que muestra un reporte, el propietario del mismo pueda decidir cuáles resultados serán visibles o podrán ser modificados por otros usuarios y por quién.
4. Impartir capacitaciones previas a la puesta en marcha de la solución propuesta sobre el funcionamiento de la misma.
5. Consultar el manual de usuario o la ayuda en línea que brinda el módulo para aclarar cualquier duda que pueda surgir.

Referencias bibliográficas

[ADR11] ADRI MARTIN, Sergio. *PostgreSQL* [en línea]. Madrid: Editorial Academia Española, 2011 [fecha de consulta: 20 Octubre 2011]. Disponible en: http://books.google.com.cu/books?id=NmB_tQAACAAJ&dq=postgreSql&hl=es&sa=X&ei=97e2T9PSIsKjgwfF_e1H&ved=0CFMQ6AEwBA ISBN: 9783846575277

[AHO07] V. AHO, Alfred, S. LAM, Monica [et al.]. *Compilers: principles, techniques & tools* [en línea]. 2da Edición. California: Pearson/Addison Wesley, 2007 [fecha de consulta: 5 Enero 2012]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=6624&subdir=/Compilers_-_Principles_Techniques_and_Tools_Alfred_Aho_ ISBN: 9780321486813

[APA12] Apache Software Foundation. *Apache JMeter - Apache JMeter™* [en línea]. 2012 [fecha de consulta: 5 Enero 2012.] Disponible en: http://jmeter.apache.org/.

[ARE09] AREITIO, Gloria y Ana. Información, Informática e Internet: del ordenador personal a la empresa [en línea]. [s.l.]: Editorial Visión Libros, 2009 [fecha de consulta: 18 Octubre 2011]. Disponible en: http://books.google.com.cu/books?id=mnFTzjdoczIC&printsec=frontcover&dq=Informaci%C3%B3n,+Inform%C3%A1tica+e+Internet:+del+ordenador+personal+a+la+empresa&hl=es&sa=X&ei=sXqwT9CUB8qy6QH1o7SWCQ&ved=0CDMQ6AEwAA#v=onepage&q=Informaci%C3%B3n%2C%20Inform%C3%A1tica%20e%20Internet%3A%20del%20ordenador%20personal%20a%20la%20empresa&f=false ISBN: 9788498865257

[BAC00] BACHMANN, F. [et al.]. 2000. *Technical concepts of component-based software engineering.* [s.l.] : Software Engineering Institute, Carnegie Mellon University, 2000. Volume II, 2nd edition.

[BAS03] BASS, L., CLEMENTS, P. and KAZMAN, R. 2003. *Software Architecture in Practic,* 2da Edición. Addison - Wesley, 2003.

[BEC01] BECK, K., BEEDLE, M. [et al.]. *Manifesto for Agile Software Development* [en línea]. 2001 [fecha de consulta: 4 Noviembre 2011]. Disponible en: http://agilemanifesto.org/principles.html

[BLA10] BLAHA, Michael**.** *Patterns of Data Modeling. Emerging Directions in Database Systems and Applications*. Edición ilustrada. [s.l.]: Taylor & Francis Group, 2010 [fecha de consulta: 3 Enero 2012]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=11576&subdir=/Diseno_de_BD/Patrones_de_diseno_ ISBN: 9781439819890

[BOE06] BOEHM, B. *A View of 20th and 21st Century Software Engineering*. En: *Proceedings of 28 International Conference on Software Engineering*. ACM: Shangai, 2006

[BON03] BONANATA, Maximiliano. *Programación y Algoritmos* [en línea]. Edición ilustrada. Buenos Aires: MP Ediciones, 2003 [fecha de consulta: 5 Enero 2012]. Disponible en: http://sunshine.prod.uci.cu/search/programaci%C3%B3n%20orientada%20a%20objetos. ISBN: 9789875261563

[BRO98] BROWN, A., WALLNAU, W. and C. *The current state of CBSE.* [s.l.] : IEEE Software, 1998. 15(5):37-46.

[CHA09] CHARTE OJEDA, Francisco. *SQL* [en línea]. [s.l.]: Anaya Multimedia, 2009 [fecha de consulta: 27 Octubre 2011]. Disponible en: http://books.google.com.cu/books?id=TZJ5QgAACAAJ&dq=sql&hl=es&sa=X&ei=xvSzT8ScLImJ6AGzt7yuCQ&ved=0CDgQ6AEwAQ. ISBN: 8441526087

[CHR03] CHRISSIS, M. B., KONRAD, M. and SHRUM, S. 2003**.** *CMMI : Guidelines for Process Integration and Product Improvement.* Addison-Wesley Professional.

[COG05] COGGESHALL, John. *La biblia de PHP 5* [en línea]. [s.l.]: Anaya Multimedia, 2005 [fecha de consulta: 19 Octubre 2011] Disponible en: http://books.google.com.cu/books?id=D3jkPAAACAAJ&dq=php+5&hl=es&sa=X&ei=x7KzT62BGITWtgfGhvjxCA&ved=0CEQQ6AEwAjgK. ISBN: 8441518459

[ELL11] Ellislab Inc. *CodeIgniter* [en línea]. 2011 [fecha de consulta: 19 Octubre 2011] Disponible en: http://www.codeigniter.com.

[ERL09] ERL, Thomas. *Design Patterns*. *Upper Saddle River*, New Jersey: Prentice-Hall, 2009 [fecha de consulta: 2 Enero 2012].

[EVO11] Evolus. *Pencil Project* [en línea]. 2011 [fecha de consulta: 6 Noviembre 2011]. Disponible en: http://pencil.evolus.vn/en-US/Home.aspx

[FER10] FERNÁNDEZ ALARCÓN, Vincenc**.** *Desarrollo de sistemas de información: Una metodología basada en el modelado* [en línea]. [s.l.]: Upc Edicions, 2010 [fecha de consulta: 19 Octubre 2011]. Disponible en: http://books.google.com.cu/books?id=pTTQ735ac1EC&pg=PA115&dq=Sistemas+de+gesti%C3%B3n+de+base+de+datos&hl=es&sa=X&ei=9a2zT8DLGoa6twf01YHiCA&ved=0CGAQ6AEwBw#v=onepage&q=Sistemas%20de%20gesti%C3%B3n%20de%20base%20de%20datos&f=false. ISBN: 9788483018620.

[GAL09] GALEANO GIL, Germán, SÁNCHEZ ALONSO, José Carlos y DÍAZ MÁRQUEZ, Pablo. *HTML: manual imprescindible* [en línea]. [s.l.]: Anaya Multimedia, 2009 [fecha de consulta: 26 Octubre 2011]. Disponible en: http://books.google.com.cu/books?id=-

WuJPgAACAAJ&dq=HTML&hl=es&sa=X&ei=8LqzT7u8LojBtgf4kIWeAw&ved=0CFkQ6AEwBzgK. ISBN: 9788441525030

[I3E99] Institute of Electrical and Electronics Engineers. *IEEE standards software engineering: Resource and technique standards* [en línea]. Ed. Institute of Electrical and Electronics Engineers, 1999 [fecha de consulta: 6 Noviembre 2011]. Disponible en: http://books.google.com.cu/books?id=IvoYAQAAIAAJ&q=IEEE+standards+software+engineering&dq=IEEE+standards+software+engineering&hl=es&sa=X&ei=_EBnT42LC8qKgwe75sWfAg&ved=0CDAQ6AEwAA. ISBN: 0738115622

[INI06] INCIARTE, Alicia y MARCANO, Noraida. Gestión académico-administrativa en la educación básica. *Revista Venezolana de Gerencia* [en línea]. *A*bril-junio 2006, vol. 11, no. 34. [fecha de consulta: 25 septiembre 2011]. Disponible en: http://redalyc.uaemex.mx/src/inicio/ArtPdfRed.jsp?iCve=29003405. ISSN: 1315-9984

[ISO25000] International Organization for Standardization. ISO/IEC 25000 Of. 2005: *Software Product Quality Requeriments and Evaluation.* Suiza, 2005. 40p.

[ISO27001] International Organization for Standardization. ISO/IEC 27001 Of 2005: *Information technology - Security techniques - Information security management systems – Requirements*. Suiza, 2005. 41p.

[ISO9000] International Organization for Standardization. ISO 9000 Of. 2005: *Quality management systems – Fundamentals y vocabulary*. Suiza, 2005. 42p.

[JOR06] SÁNCHEZ, Jordi. *¿Qué es un framework?* [en línea]. 29 Septiembre 2006 [fecha de consulta: 18 Octubre 2011]. Disponible en: http://jordisan.net/blog/2006/que-es-un-framework

[JQU11] The JQuery Foundation. *JQuery* [en línea]. 2011. [fecha de consulta: 20 Octubre 2011] Disponible en: http://jquery.com/.

[KAB09] J. KABIR, Mohammed. *Servidor Apache 2: la biblia* [en línea]. [s.l.]: Anaya Multimedia, 2009 [fecha de consulta: 27 Octubre 2011]. Disponible en: http://sunshine.prod.uci.cu/search/apache

[KIC06] REYNOSO, Carlos Billy, KICILLOF, Nicolás**.** *MSDN Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. [En línea] 2006. [fecha de consulta: 7 Noviembre 2011]. Disponible en: http://www.microsoft.co.ke/spanish/msdn/arquitectura/roadmap_arq/style.aspx

[LAR99] LARMAN, Craig. *UML y Patrones* [en línea]. México D.F.: Prentice Hall, 1999 [fecha de consulta: 5 Octubre 2011]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/UML_y_Patrones. ISBN: 9701702611

[MTB09] MTBASE. *Informaker: software para la generación de reportes empresariales y gestión de Bases de Datos* [en línea]. [2009]. [fecha de consulta: 22 Octubre 2011]. Disponible en:

http://www.mtbase.com/productos/desarrollo/infomaker.

[OCU07] Oficina de Cooperación Universitaria (OCU). *Universitas XXI – Académico* [en línea], 28 de noviembre 2007. [fecha de consulta: 22 Octubre 2011]. Disponible en: http://www.ocu.es/portal/page/portal/inicio/sofware_gestion_universitaria/sistema_gestion_academica.

[OMG11] Object Management Group (OMG). *Unified Modeling Specification,* version 1.4 [fecha de consulta: 7 Noviembre 2011]. Disponible en: http://www.omg.org/uml.

[ORA11] Oracle Corporation. *NetBeans* [en línea]. 2011 [fecha de consulta: 4 Noviembre 2011] Disponible en: http://netbeans.org/index_es.html.

[PGA11] Pgadmin. *Introduction* [en línea]. 2011 [fecha de consulta: 6 Noviembre 2011]. Disponible en: http://www.pgadmin.org/

[PRE10] PRESSMAN, Roger S. *Ingeniería de Software: Un enfoque prático.* 7ma. Edición. New York: McGraw-Hill Companies, 2010 [fecha de consulta: 5 Enero 2012]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/Ediciones_del_Pressman/Pressman_7ma_edicion. ISBN: 9780073375977.

[PRO11] Proyectos Gestión Conocimiento. *Sistemas de reporting* [en línea]. [Octubre 2011] [fecha de consulta: 22 Octubre 2011]. Disponible en: http://www.pgconocimiento.com/Servicios/sistemas-de-reporting.html

[REY06] REYNOSO, Carlos Billy. *Introducción a la Arquitectura de Software* [en línea]. Buenos Aires: [s.n.], 2006 [fecha de consulta: 7 Noviembre 2011]. Disponible en: http://sunshine.prod.uci.cu/search/Lenguaje%20UMLL

[ROW01] ROWLAND YOUNG, Ralph. *Effective requirements practices* [en línea]. Edición Ilustrada. [s.l.]: Addison-Wesley, 2001 [fecha de consulta: 6 Noviembre 2011]. Disponible en: http://books.google.com.cu/books?id=cYt5QgAACAAJ&dq=effective+requirements+practices&hl=es&sa=X&ei=kEJnT4GPLo_gggeJopxZ&ved=0CDAQ6AEwAA. ISBN: 0201709120

[SÁN06] SÁNCHEZ MAZA, Miguel Ángel**.** *Javascript* [en línea]. [s.l.]: Editorial Innovación y Cualificación, 2006 [fecha de consulta: 20 Octubre 2011]. Disponible en: http://books.google.com.cu/books?id=3x09sewjaHIC&printsec=frontcover&dq=javascript&hl=es&ei=wpC5TsrED62DsgLs9ZSjCA&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCwQ6AEwAA#v=onepage&q&f=false. ISBN: 8495733188.

[SCA11] SCAMBRAY, Joel, LIU, Vincent and SIMA, Caleb. *Hacking exposed web appilications: Web applications security secrets and solutions*. 3rd Edition. New York: McGraw-Hill, 2011 [fecha de

consulta: 20 Enero 2012]. Disponible en: http://sunshine.prod.uci.cu/search/Cross-site%20request%20forgery. ISBN: 9780071740425

[SCH06] SCHMITT, Christopher, TRAMMELL, Mark [et al.]. *CSS* [en línea]. [s.l.]: Grupo Anaya Comercial, 2006. Disponible en: http://books.google.com.cu/books?id=zl0TPWJikkIC&printsec=frontcover&hl=es&source=gbs_atb#v=onepage&q&f=false. ISBN: 8441519544

[SEI11] Software Engineering Institute. *What is CMMI?* [en línea]. 2011 [fecha de consulta: 4 Noviembre 2011]. Disponible en: http://www.sei.cmu.edu/cmmi/index.cfm .

[SER11] Servidor Web. [fecha de consulta: 27 Octubre 2011]. Disponible en: http://es.wikipedia.org/wiki/Servidor_web.

[SOM05] SOMMERVILLE, Ian. *Ingeniería del software* [en línea]. 7ma Edición [s.l.]: Pearson Educación, 2005 [fecha de consulta: 2 Enero 2012]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8501&subdir=/Ediciones_del_Sommerville/Sommerville_7ma_edicion. ISBN: 8478290745

[SZY02] SZYPERSKI, Clemens. *Component software: Beyond object-oriented programming.* [s.l.] : Addison-Wesley Pub Co, 2002. 2da edición.

[UCI11] Universidad de las Ciencias Informáticas. *Misión de la UCI* [en línea]. 2011 [fecha de consulta: 20 Septiembre 2011.] Disponible en: http://www.uci.cu.

[VAN07] VAN LANCKER, Luc. *CSS 1 y CSS 2.1: Hojas de estilo para enriquecer el código HTML* [en línea]. Barcelona: Ediciones ENI, 2007 [fecha de consulta: 22 Octubre 2011]. Disponible en: http://books.google.com.cu/books?id=xBrgWWa31pcC&printsec=frontcover&dq=css&hl=es&sa=X&ei=n7WzT_SCDIiXtweouY3NCA&ved=0CGYQ6AEwCQ#v=onepage&q=css&f=false. ISBN: 9782746035836

[VSP11] *Visual Paradigm for UML* [en línea]. 2011 [fecha de consulta: 6 Noviembre 2011]. Disponible en: http://www.visual-paradigm.com/product/vpuml/

[WOD09] WODTKE, Christina and GOVELLA, Austin**.** *Information Architecture: Blueprints for the Web*. 2da Edición. [s.l.]: New Riders, 2009 [fecha de consulta: 5 Enero 2012]. Disponible en: http://books.google.com.cu/books?id=Tp40QFGCU2sC&dq=arquitectura%20de%20la%20informaci%C3%B3n&hl=es&source=gbs_similarbook. ISBN: 9780321600806.

[ZDU05] ZDUN, Uwe and AVGERIOU, Paris. *Modeling Architectural Patterns Using Architectural Primitives* [en línea]. En: *ACM OOPSLA* Conference (20th: 2005: San Diego, California). New York: ACM SIGPLAN, 2005 [fecha de consulta: 3 Enero 2012] Disponible en: http://dl.acm.org/citation.cfm?id=1094822. ISSN: 1-59593-031-0

Bibliografía consultada

HERNÁNDEZ LEÓN, Rolando A. y COELLO GONZÁLEZ, Sayda. *El proceso de investigación científica*. La Habana: Editorial Universitaria, 2011 [fecha de consulta: 4 Octubre 2011]. 110p. ISBN: 9789591913073.

HERNÁNDEZ SAMPIERI, Roberto, FERNÁNDEZ COLLADO, Carlos y BAPTISTA LUCIO, Pilar. Metodología de la investigación. 4ta Edición. México D.F.: McGraw-Hill, 2006 [fecha de consulta: 4 Octubre 2011]. 882p. ISBN: 9701036322.

JACOBSON, Ivar, BOOCH, Grady y RUMBAUGH. *El Proceso Unificado de Desarrollo de Software* [en línea]. Madrid: Pearson Education S.A, 2000 [fecha de consulta: 5 Octubre 2011]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/El_Proceso_Unificado_de_Desarrollo. ISBN: 8478290362

PRESSMAN, Roger S. *Ingeniería de Software: Un enfoque prático.* 5ta Edición. New York: McGraw-Hill Companies, 2010 [fecha de consulta: 5 Octubre 2012]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/Ediciones_del_Pressman/Pressman_5ta_edicion.

Glosario de términos

A

Aplicación o Sistema Informático: Programas con los cuales el usuario final interactúa a través de una interfaz y realizan tareas útiles para este.

Áreas de proceso: Aquellas actividades que facilitan el camino de la mejora. En cada una de estas áreas se define qué hay que hacer pero no cómo hay que hacerlo.

C

Cliente Servidor: Modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

Componente: Parte física y reemplazable de un sistema que se ajusta a, y proporciona la realización de, un conjunto de interfaces.

CSS: Hojas en estilo de cascada aplicables a documentos HTML que contiene diferentes estilos y son fáciles de cambiar y diseñar.

Capacidad de un proceso: Atributo de los procesos. El nivel de capacidad de un proceso indica si solo se ejecuta, o si también se planifica se encuentra organizativa y formalmente definido, se mide y se mejora de forma sistemática.

D

Deficiencia: Es toda pérdida o anormalidad de una estructura o función psicológica, fisiológica o anatómica.

Dependencia: Relación semántica entre dos elementos, en la cual un cambio en uno puede afectar al otro.

DHTML *(Dynamic HTML):* Idioma de hipertexto dinámico, compuesto de opciones agregados en los documentos HTML.

DOM *(Document Object Model):* Una especificación W3C para interfaces de programa de aplicación para el acceso al contenido de los documentos HTML y XML.

Dominio: Área de conocimiento o actividad caracterizada por un conjunto de conceptos y terminología comprendidos por los practicantes de ese dominio.

F

Framework: Una estructura para elaboración de proyectos donde se parte de un esqueleto de apoyo utilizado como base para algo que se está construyendo.

FTP *(File Transfer Protocol):* Protocolo de transferencia de archivos.

H

HTTP *(Hypertext Transfer Protocol)*: Protocolo de transmisión del hipertexto.

I

Informática: Disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales.

Informatizar: Proceso de aplicar sistemas o equipos informáticos al tratamiento de la información.

Internet: Red de computadoras alrededor de todo el mundo que comparten información unas con otras por medio de páginas o sitios.

Interoperabilidad: Condición necesaria para que los usuarios (humanos o mecánicos) tengan un acceso completo a la información disponible. Entre las iniciativas recientes más destacadas para dotar a la Web de interoperabilidad se encuentran los servicios Web y la Web semántica.

M

Madurez de un proceso: Atributo de las organizaciones que desarrollan o mantienen los sistemas de *software*. En la medida que estas llevan a cabo su trabajo siguiendo procesos, y en la que estos se encuentran homogéneamente implantados, definidos con mayor o menor rigor; conocidos y ejecutados por todos los equipos de la empresa; y medidos y mejorados de forma constante, las organizaciones serán más o menos "maduras".

Anexos

Anexo No.1 Modelo de Capacidad y Madurez Integrado (CMMI)

El modelo CMMI (Capability Maturity Model Integration), fue desarrollado por el SEI (*Software Engineering Institute*), centro de investigación y desarrollo, patrocinado por el departamento de defensa de los Estados Unidos, y gestionado por la universidad de Carnegie-Mellon.

Según la definición que hace SEI [SEI11], CMMI es un modelo para la mejora de procesos, que proporciona a las organizaciones, los elementos esenciales para procesos eficaces. CMMI ayuda a integrar funciones organizativas, tradicionalmente separadas, establece objetivos y prioridades en la mejora de procesos, proporciona una orientación para la calidad de procesos, y un punto de referencia para la evaluación de los procesos actuales.

El modelo CMMI define dos tipos de representaciones, una por etapas o escalonada y otra continua (ver figura 8); ambas tienen el mismo contenido pero diferente estructura. La representación por etapas se centra en un conjunto de áreas de procesos clave, que son identificadas dentro de ciertos niveles de madurez (1-5). Según este modelo, la organización no puede alcanzar el siguiente nivel de madurez hasta que no haya alcanzado el nivel previo. La representación continua ofrece mayor flexibilidad a las organizaciones permitiéndoles la selección de los procesos más relevantes sobre los que se realizarán las mejoras en función de sus objetivos de negocio y/o riesgos.

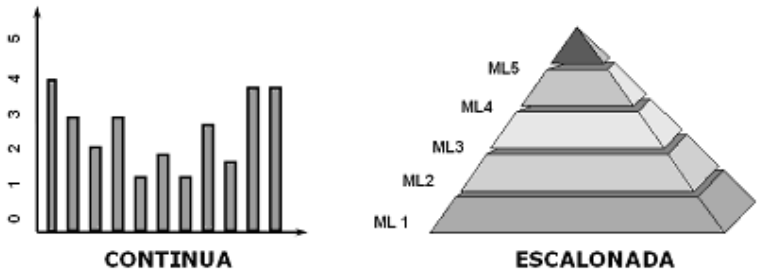


Figura 8. Representación continua y escalonada de CMMI.

Madurez de un proceso

La madurez de un proceso *software* es el grado en el cual un proceso específico es efectivo, definido, gestionado, medido y controlado. La madurez supone un potencial en crecimiento en cuanto a capacidad e indica la riqueza de los procesos de una organización y la consistencia con la cual estos son aplicados en los proyectos [CHR03].

La *representación por etapas (staged)* hace especial énfasis en el grado de madurez de los procesos, de forma que cada área de proceso se asocia a uno de los 5 niveles de madurez, que sirven como punto de referencia para conocer el grado de madurez total que posee una organización. Una

organización alcanza un nivel de madurez determinado cuando ha puesto en práctica todas y cada una de las áreas de proceso aplicables a ese nivel y a los niveles inferiores. Los cinco niveles se describen en la siguiente tabla:

Tabla 3. Niveles de madurez de los procesos.

Nivel	El proceso es…	Descripción
1	Inicial	El proceso *software* está caracterizado como "ad hoc", y en ocasiones puede ser incomprensible. Algunos procesos están definidos y el éxito depende de los esfuerzos a nivel de individuo.
2	Gestionado	Los procesos de gestión de proyectos están definidos de una manera básica para realizar el seguimiento de los costes, fechas y funcionalidad. El rigor en la definición de los procesos es el justo para poder repetir éxitos previos en proyectos de similares características.
3	Definido	El proceso de *software* para las actividades de gestión e ingeniería está documentado, estandarizado e integrado en el proceso estándar dentro de la organización. Todos los proyectos utilizan una versión estándar del proceso *software* aprobado por la organización y adaptado a las necesidades del proyecto para desarrollo y mantenimiento de *software*.
4	Gestionado Cuantitativamente	Se recogen de forma detallada medidas de los procesos *software* y la calidad de los productos. Los procesos y productos *software* son entendidos cuantitativamente y controlados.
5	Optimizado	La mejora continua de procesos se basa en los resultados cuantitativos de la aplicación de innovaciones y tecnologías en los procesos ya establecidos.

Estos cinco niveles reflejan el hecho de que el CMMI es un modelo para la mejora de la capacidad de las organizaciones de *software*. Las prioridades en el modelo no están dirigidas hacia proyectos individuales sino a procesos que aporten valor a la organización en su conjunto.

Capacidad de un proceso

La capacidad de un proceso *software* describe el rango de resultados esperados que se pueden obtener mediante la implementación del proceso *software*. La capacidad de un proceso *software* en una organización proporciona un medio para predecir los resultados más probables que se pueden esperar en proyectos que tengan similares características [CHR03].

La *representación continua (continuous)* hace especial énfasis en la capacidad de ciertas áreas para realizar adecuadamente sus actividades. En la representación continua, los niveles de madurez no

existen como tales. En cambio, los niveles de capacidad se designan para cada área de proceso, proporcionando un orden recomendado para acercarse a la mejora dentro de cada área de proceso. Una representación continua favorece la flexibilidad en el orden hacia el cual se dirigen las mejoras. En esta representación las áreas de proceso se pueden agrupar en las cuatro categorías generales: Nivel 0 Incompleto, Nivel 1 Ejecutado, Nivel 2 Gestionado, Nivel 3 Definido, Nivel 4 Gestionado Cuantitativamente y Nivel 5 Optimizando.

Anexo No.2 Modelo de entrevista

Modelo de entrevista para la obtención de requerimientos
Nombre(s) y apellidos del entrevistado:
Cargo que ocupa:
Centro al que pertenece:
Preguntas
1. ¿Cómo se desarrolla actualmente la gestión de reportes en el Sistema de Gestión Académica de Pregrado? 2. ¿Por qué se hace necesario realizar un módulo para la gestión de reportes? 3. ¿Para qué se utilizará y qué beneficios traerá este módulo? 4. El nuevo módulo, ¿estará integrado o no a algún sistema u otro módulo en específico? 5. ¿Qué funcionalidades debe brindar el módulo de reportes? 6. ¿Qué personas tendrán acceso al módulo de reportes? 7. ¿Existe algún sistema dentro del SGU que posea un mecanismo para gestionar reportes? 8. ¿Cómo estarán organizados los reportes en el sistema? 9. ¿Cómo deben mostrados los datos contenidos en los reportes? 10. Además de mostrar la información en forma de listados, ¿los reportes deberán mostrar los datos de otra forma? 11. ¿Qué formatos de salida deberán ser manejados por los reportes?
Resultados obtenidos:

Anexo No.3 Relación de requerimientos funcionales por complejidad

Tabla 4. Relación de requerimientos funcionales por complejidad.

Prioridad	Cantidad
Alta	12

Media	2
Baja	5
Total	19

Anexo No.4 Requerimientos no funcionales de la propuesta de solución

No.	Usabilidad
RNF 1	El sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo por parte de los usuarios, los cuales deben poder acceder de manera rápida y efectiva a la información solicitada. Debe, además, ser una interfaz de manejo cómodo donde la curva de aprendizaje para los usuarios sea lo menos inclinada posible y que posibilite en estos una rápida adaptación.
RNF 2	El sistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vistas a una mayor comprensión por su parte sobre la herramienta de trabajo.
RNF 3	Diferenciar las interfaces y opciones para los usuarios que accedan al sistema según los diferentes roles que estos tengan dentro del mismo.
RNF 4	El sistema debe presentar una serie de menús tanto laterales como en barras horizontales de íconos que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.
	Seguridad
RNF 5	La seguridad de la base de datos está a nivel de roles, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos, trayendo consigo la protección de la información.
RNF 6	Políticas de seguridad por usuarios y roles: El sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.
RNF 7	Garantizar que la información sea editada por quien tiene derecho a editarla.
RNF 8	El sistema debe brindar protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
RNF 9	El sistema deberá solicitar una verificación sobre acciones irreversibles (eliminaciones).
RNF 10	Registro sistemático de incidencias: El sistema debe ser capaz de registrar el accionar del usuario, así como permitir auditorías y exámenes de las trazas tanto en tiempo real como en históricos.
	Confiabilidad
RNF 11	El sistema estará disponible las 24 horas del día y los siete días de la semana.
	Eficiencia
RNF 12	El tiempo de respuesta del sistema debe ser como máximo de 5 segundos.
RNF 13	El sistema debe soportar una conexión simultánea de más de 1000 usuarios.
	Restricciones del diseño
RNF 14	El lenguaje de programación deberá ser PHP 5.3 o superior.

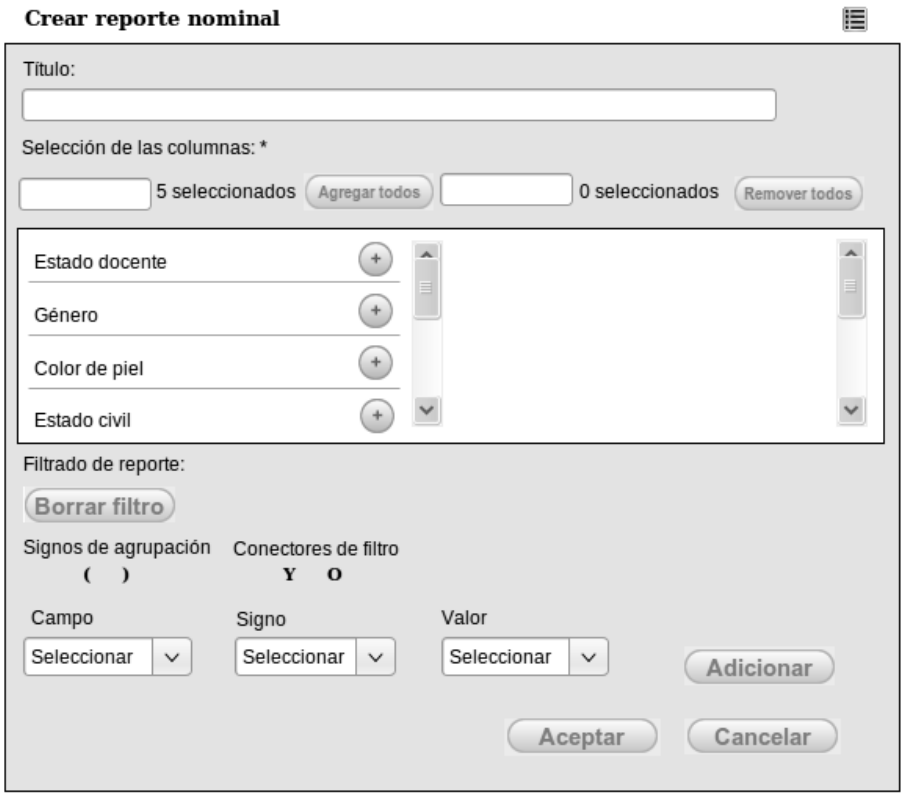
RNF 15	Como Entorno de Desarrollo Integrado se empleará NetBeans 6.7.1 o una versión superior.
RNF 16	Como servidor Web se explotará Apache 2.2.2.
RNF 17	El sistema gestor de bases de datos deberá ser PostgreSQL 8.4.1.
RNF 18	El diseño de la base de datos se realizará con Visual Paradigm 8.0 o una versión superior
RNF 19	El sistema operativo a utilizar en el entorno de desarrollo deberá ser: GNU Linux.
	Legales, de Derecho de Autor y otros
RNF 20	El sistema debe ser sometido a un análisis legal por parte de los abogados y personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su; así mismo se debe proceder a una evaluación y certificación por parte del cliente del producto.
	Hardware
RNF 21	Para el desarrollo se requiere de una PC Intel Pentium 4 o superior, CPU 3GHZ o superior, 512 MB RAM o superior, 160 GB HDD o superior.
RNF 22	Para la explotación del cliente se requiere de una PC Pentium 3 o superior, CPU 133 MHZ o superior, 256 RAM mínimo 512 RAM recomendada o superior.
RNF 23	Para la explotación del servidor se requiere de un CPU Dual Core 2.0 GHZ o superior, memoria RAM de 4 GB (recomendado 6 GB), 250 GB HDD.
	Interfaces de hardware
RNF 24	La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo de conexión segura TCP/IP.
RNF 25	La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTPS.
	Estándares aplicables
RNF 26	Remitirse al documento de arquitectura: CENIA_PRE_ADASP-v1.0 (en el mismo se especifica los requerimientos de estándares aplicables).
	Soporte
RNF 27	El sistema contará con un grupo de soporte y asesoría al cliente.
RNF 28	El sistema brinda como apoyo una Ayuda contextual en la cual se refleja detalladamente la explicación de cada una de las pantallas con sus respectivas funcionalidades.
RNF 29	El sistema deberá presentar un manual de usuario, permitiendo con ello un correcto uso de sus funcionalidades y brindarle al usuario una mayor experiencia del trabajo con el mismo.
RNF 30	Se precisa que la documentación del sistema esté actualizada en todos los aspectos, fases de trabajo y ciclos de desarrollo del mismo, permitiendo con ello un respaldo tanto ingenieril como legal del desarrollo de dicho sistema.
	Portabilidad
RNF 31	El sistema será multiplataforma.

Anexo No.5 Especificaciones de requerimientos funcionales

Tabla 5. CENIA_PRE_ESR Crear reporte de tipo listado.

Nº	*Nombre*	*Descripción*	*Prioridad Complejidad para cliente*

RFR1	Crear reporte de tipo listado	Para crear el reporte se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. El sistema recoge los datos: título, el usuario selecciona los datos que desea tenga el listado en el campo Selección de las columnas, y adiciona los filtros que desee mediante los campos: campo, signo y valor. Una vez creado el reporte se actualiza el listado y se muestra un mensaje de información: ¨El elemento ha sido creado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.	Alta	Alta
	Prototipo			



	Campos	Tipos de Datos	Reglas o Restricciones
	Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
	Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Campo	varchar	▪ Campo de selección.
	Signo	varchar	▪ Campo de selección.
	Valor	varchar	▪ Campo de selección.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se crea satisfactoriamente el reporte el sistema mostrará un mensaje de información "El elemento ha sido creado satisfactoriamente". 3. En caso de que exista un reporte con el mismo nombre el sistema creará el nueva reporte y como nombre le pondrá el definido por el usuario y adicionalmente un número que representa la cantidad de reportes con el mismo nombre. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: "Ha ocurrido un error cuando intentaba crear el reporte" mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá	

		escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aun desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.

Tabla 6. CENIA_PRE_ESR Modificar reporte de tipo listado.

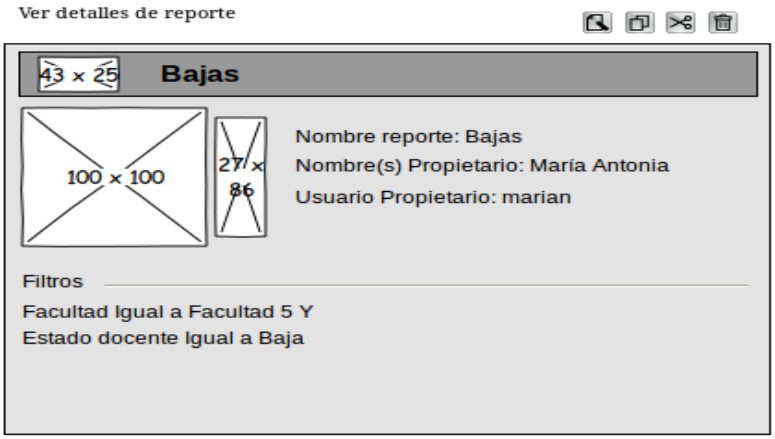
Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para client*
RFR2	Modificar reporte de tipo listado	Para modificar el reporte se selecciona el reporte deseado en el árbol del sistema y selecciona la opción Modificar en la barra de íconos flotantes o clic derecho sobre el reporte y selecciona la opción Modificar. El sistema muestra los datos registrados de reporte. El usuario puede modificar cualquier entrada, una vez modificados los datos se guardan los cambios y se muestra un mensaje de confirmación: ¨El elemento ha sido modificado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.	Alta	Alta
		Prototipo		

	Campos	Tipos de Datos	Reglas o Restricciones
	Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
	Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Campo	varchar	▪ Campo de selección.
	Signo	varchar	▪ Campo de selección.
	Valor	varchar	▪ Campo de selección.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se crea satisfactoriamente el reporte el sistema mostrará un mensaje de información "El elemento ha sido creado satisfactoriamente". 3. En caso de que exista un reporte con el mismo nombre el sistema creará el nueva reporte y como nombre le pondrá el definido por el usuario y adicionalmente un número que representa la cantidad de reportes con el mismo nombre. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: "Ha ocurrido un error cuando	

		intentaba modificar el reporte" mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación "Perderá la información que no ha sido guardada. ¿Aun desea cancelar?". 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio "Campo requerido". 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: "Entre al menos 3 caracteres" y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.

Tabla 7. CENIA_PRE_ESR Ver detalles de reporte de tipo listado.

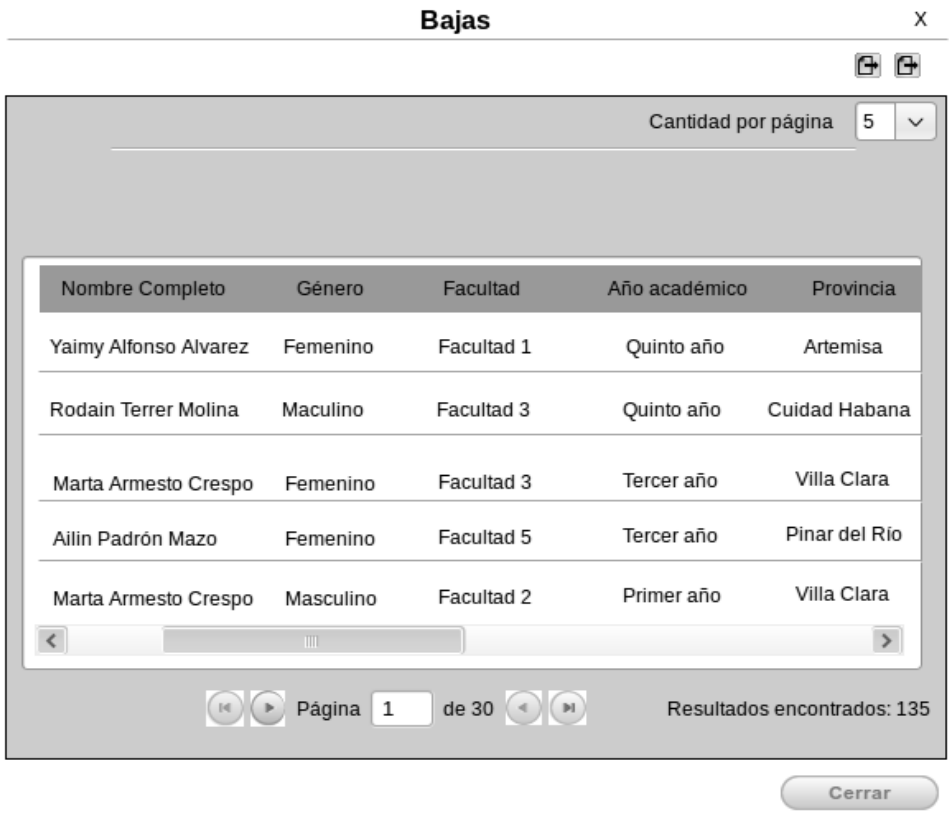
Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR3	Ver detalles de reporte de tipo listado.	Para ver detalles de un reporte se selecciona o se le pasa por encima en el árbol del sistema al reporte deseado. Se muestran los datos registrados del reporte: Nombre del reporte, Nombre(s) propietario, Usuario de propietario y los Filtros que posee dicho reporte. En el área de íconos flotantes se muestran las opciones: modificar, copiar, cortar y eliminar.	Baja	Baja
	Prototipo			



	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Nombre reporte	Varchar	Solo lectura
	Nombre(s) propietario	Varchar	Solo lectura
	Usuario propietario	Varchar	Solo lectura
	Filtros	Varchar	Solo lectura
	Nombre reporte	Varchar	Solo lectura
	Observaciones		

Tabla 8. CENIA_PRE_ESR Mostrar reporte de tipo listado.

Nº	***Nombre***	***Descripción***	***Complejidad***	***Prioridad para cliente***
RFR4	Mostrar reporte de tipo listado.	Para mostrar el reporte se selecciona en el árbol del sistema el reporte deseado dando clic sobre él o clic derecho y selecciona la opción Abrir. Se muestran todos los datos del reporte de tipo listado seleccionado. En el área de íconos internos se muestran las opciones: Exportar en documento pdf y Exportar en documento excel.	Alta	Alta
	Prototipo			



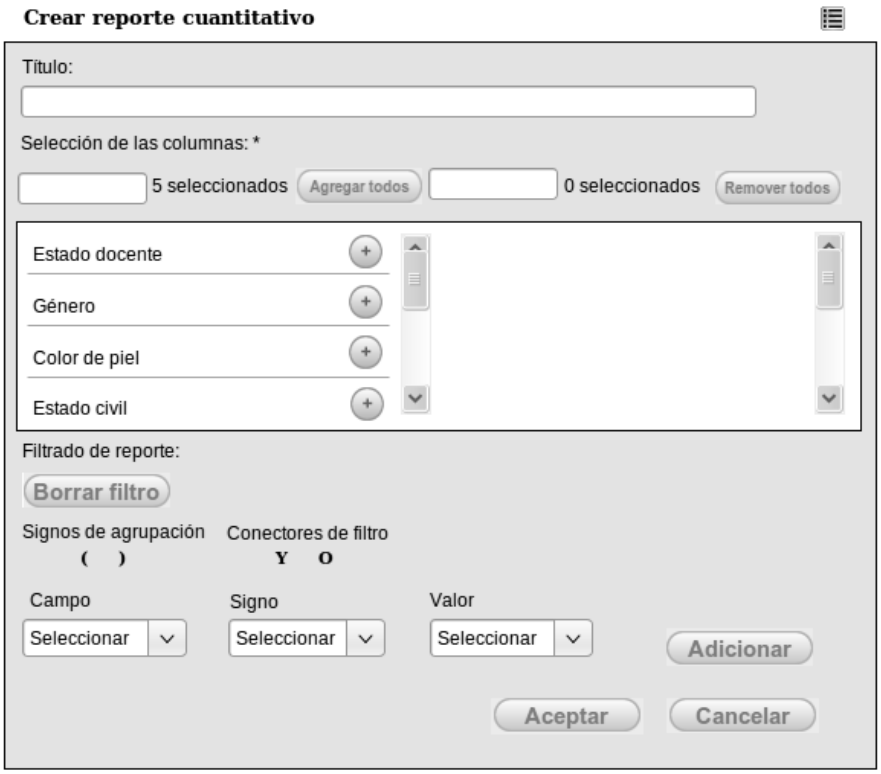
	Campos	Tipos de Datos	Reglas o Restricciones
		No procede	No procede
	Observaciones	1. El límite máximo de datos del reporte que se ven por página es de 20. 2. El límite mínimo de datos del reporte que se ven por página es de 5. 3. La cantidad de datos del reporte predeterminada es 5.	

Tabla 9. CENIA_PRE_ESR Crear reporte de tipo cuantitativo.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR5	Crear reporte de tipo cuantitativo.	Para crear un reporte de tipo cuantitativo se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. El sistema recoge los datos: título, el usuario selecciona las columnas que desea tenga el	Alta	Alta

		listado en el campo Selección de las columnas, y adiciona los filtros que desee mediante los campos: campo, signo y valor. Una vez creado el reporte se actualiza el listado y se muestra un mensaje de información: ¨El elemento ha sido creado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.		

Prototipo



	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
	Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Campo	varchar	▪ Campo de selección.

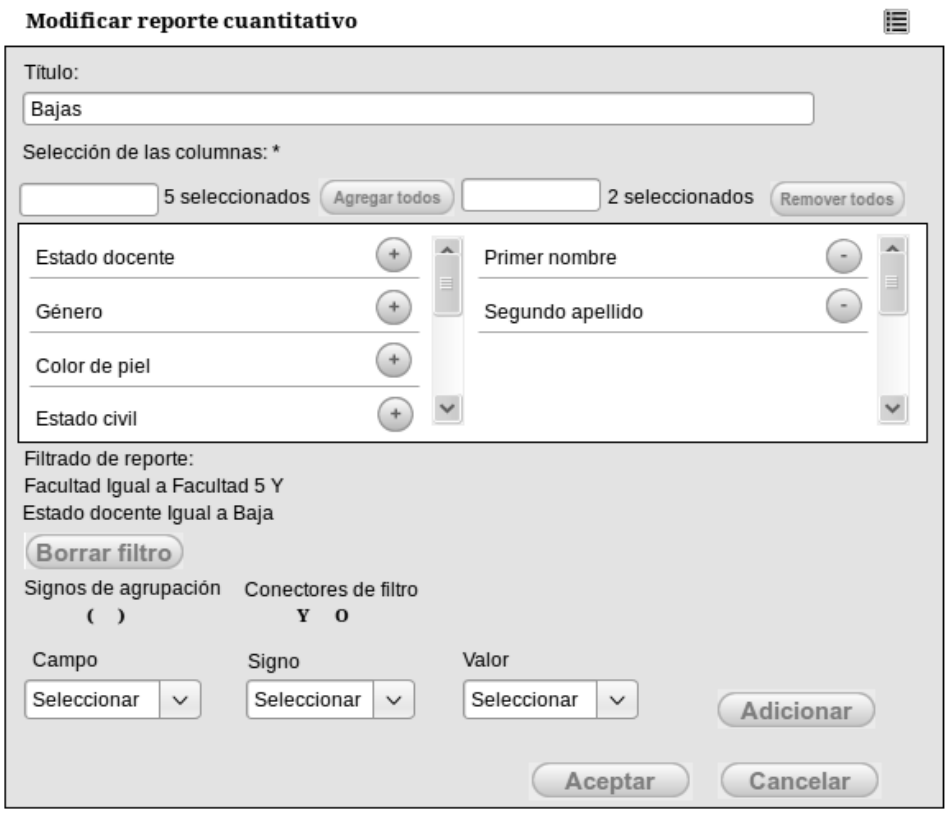
	Signo	varchar	▪ Campo de selección.
	Valor	varchar	▪ Campo de selección.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se crea satisfactoriamente el reporte el sistema mostrará un mensaje de información “El elemento ha sido creado satisfactoriamente”. 3. En caso de que exista un reporte con el mismo nombre el sistema creará el nueva reporte y como nombre le pondrá el definido por el usuario y adicionalmente un número que representa la cantidad de reportes con el mismo nombre. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba crear el reporte” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aun desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.	

Tabla 10. CENIA_PRE_ESR Modificar reporte de tipo cuantitativo.

Nº	***Nombre***	***Descripción***	***Complejidad***	***Prioridad para cliente***
RFR6	Modificar reporte de tipo cuantitativo.	Para modificar un reporte de tipo cuantitativo se selecciona el reporte deseado en el árbol del sistema y selecciona la opción Modificar en la barra de íconos flotantes o clic derecho sobre el reporte y selecciona la opción Modificar. El sistema muestra los datos registrados del reporte. El usuario puede modificar cualquier entrada, una vez modificados los datos se guardan los cambios	Alta	Alta

		y se muestra un mensaje de confirmación: ¨El elemento ha sido modificado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.		

Prototipo



	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
	Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Campo	varchar	▪ Campo de selección.
	Signo	varchar	▪ Campo de selección.
	Valor	varchar	▪ Campo de selección.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se modifica satisfactoriamente el reporte el sistema mostrará un	

		mensaje de información “El elemento ha sido creado satisfactoriamente”. 3. En caso de que exista un reporte con el mismo nombre el sistema creará el nueva reporte y como nombre le pondrá el definido por el usuario y adicionalmente un número que representa la cantidad de reportes con el mismo nombre. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba crear el reporte” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aun desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.

Tabla 11. CENIA_PRE_ESR Mostrar reporte de tipo cuantitativo

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR8	Mostrar reporte de tipo cuantitativo.	Para mostrar un reporte de tipo cuantitativo se selecciona en el árbol del sistema el reporte deseado dando clic sobre él o clic derecho y selecciona la opción Abrir. Se muestran todos los datos del reporte de tipo cuantitativo seleccionado. En el área de íconos internos se muestran las opciones: Exportar en documento pdf y Exportar en documento excel.	Alta	Alta
	Prototipo			

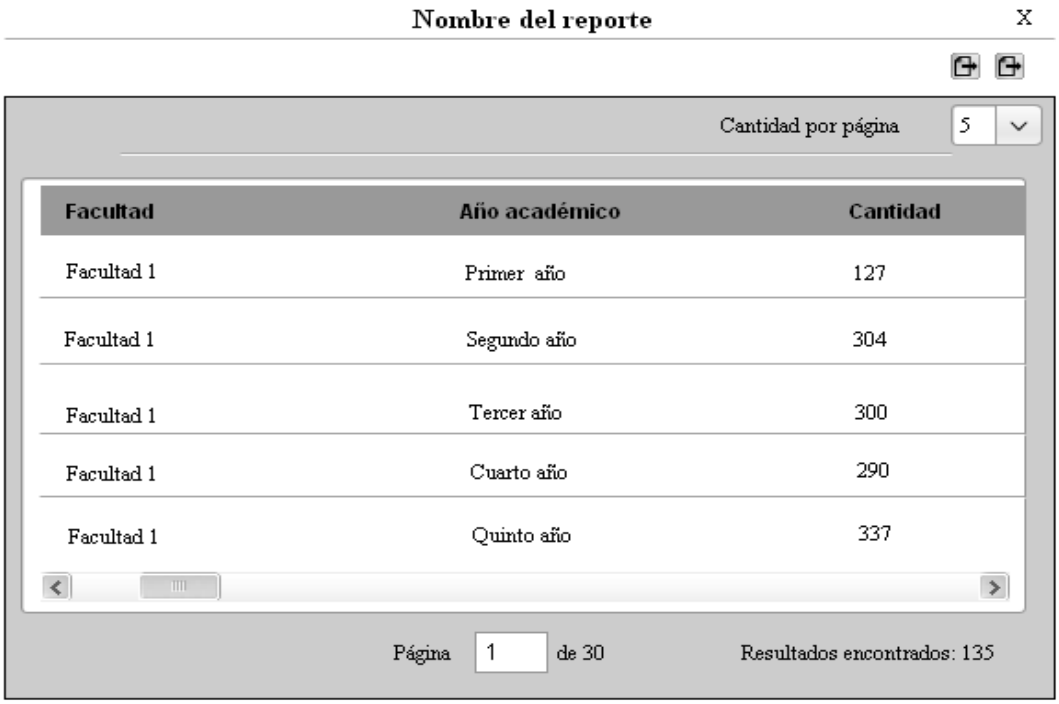
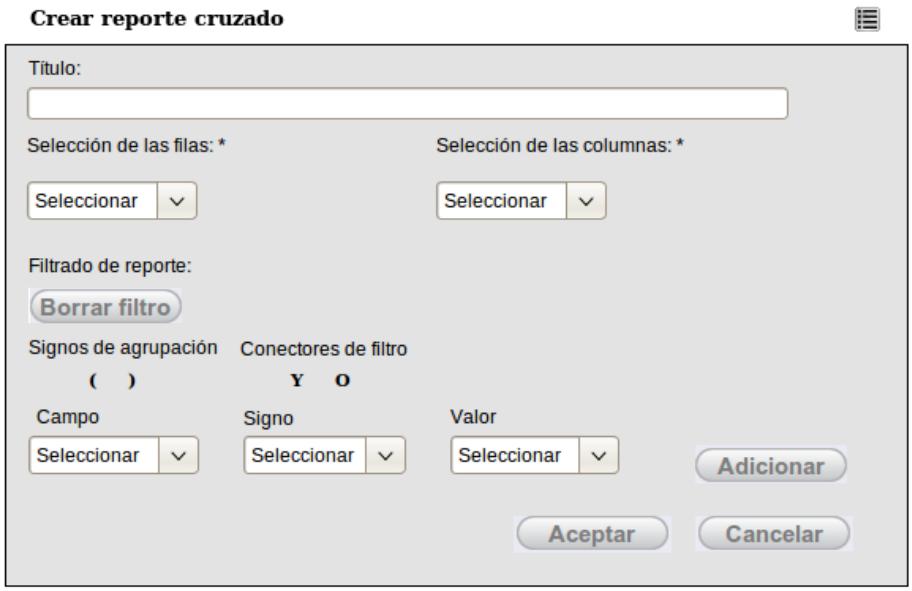
			
	Campos	***Tipos de Datos***	***Reglas o Restricciones***
		No procede	No procede
	Observaciones	1. El límite máximo de datos del reporte que se ven por página es de 20. 2. El límite mínimo de datos del reporte que se ven por página es de 5. 3. La cantidad de datos del reporte predeterminada es 5.	

Tabla 12. CENIA_PRE_ESR Crear reporte de tipo cruzado.

Nº	***Nombre***	***Descripción***	***Complejidad***	***Prioridad para cliente***
RFR9	Crear reporte de tipo cruzado.	Para crear un reporte de tipo cruzado se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. El sistema recoge los datos: título, el usuario selecciona las filas y columnas que desea tenga el listado, y adiciona los filtros que desee mediante los campos: campo, signo y valor.	Alta	Alta

		Una vez creado el reporte se actualiza el listado y se muestra un mensaje de información: ¨El elemento ha sido creado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.		

Prototipo

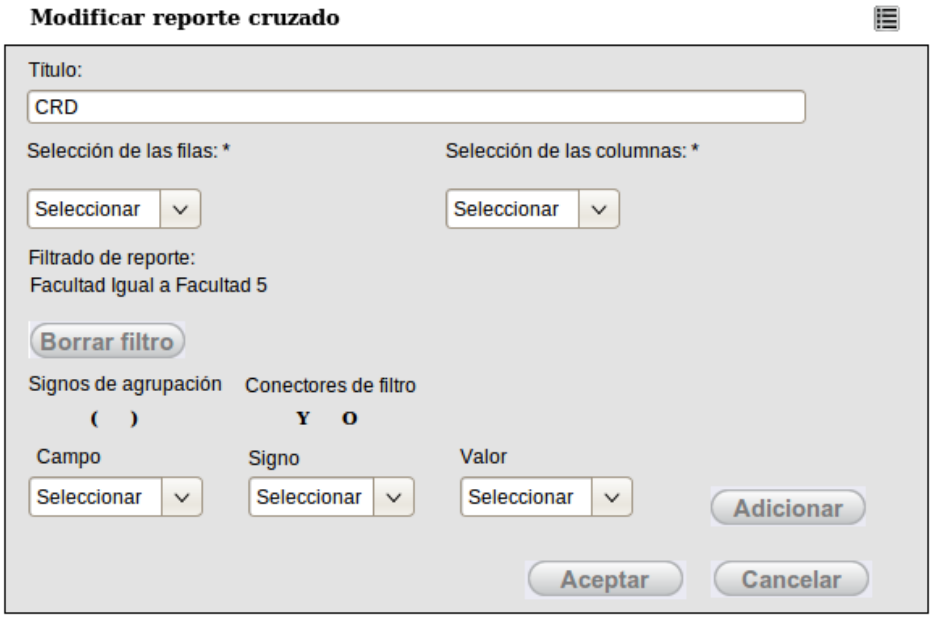


Campos	***Tipos de Datos***	***Reglas o Restricciones***
Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
Selección de las filas	varchar	▪ Obligatorio. ▪ Campo de selección.
Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
Campo	varchar	▪ Campo de selección.
Signo	varchar	▪ Campo de selección.
Valor	varchar	▪ Campo de selección.
Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se crea satisfactoriamente el reporte de tipo cruzado el sistema mostrará un mensaje de información “El elemento ha sido creado satisfactoriamente”.	

		3. En caso que el reporte de tipo cruzado exista se muestra un mensaje de error: “El elemento ya existe”. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba crear un reporte de tipo cruzado” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aun desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.

Tabla 13. CENIA_PRE_ESR Modificar reporte de tipo cruzado.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR10	Modificar reporte de tipo cruzado.	Para modificar un reporte de tipo cruzado se selecciona el reporte deseado en el árbol del sistema y selecciona la opción Modificar en la barra de íconos flotantes o clic derecho sobre el reporte y selecciona la opción Modificar. El sistema muestra los datos registrados del reporte. El usuario puede modificar cualquier entrada, una vez modificados los datos se guardan los cambios y se muestra un mensaje de confirmación: ¨E elemento ha sido modificado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.	Alta	Alta
		Prototipo		

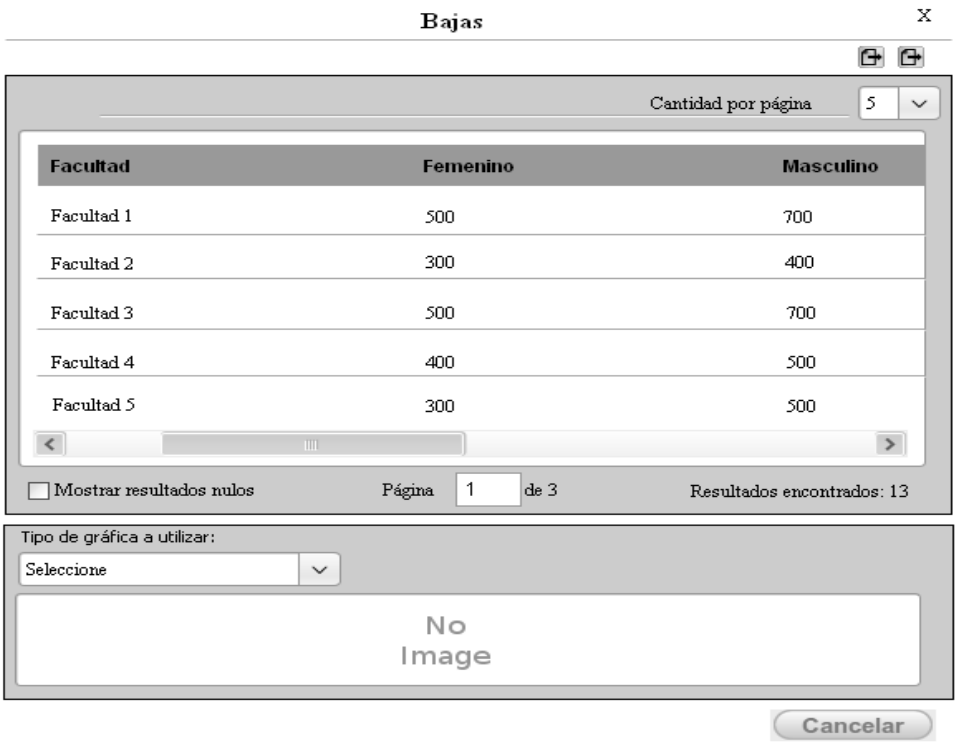


	Campos	Tipos de Datos	Reglas o Restricciones
	Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
	Selección de las filas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Campo	varchar	▪ Campo de selección.
	Signo	varchar	▪ Campo de selección.
	Valor	varchar	▪ Campo de selección.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se modifica satisfactoriamente el reporte de tipo cruzado el sistema mostrará un mensaje de información “El elemento ha sido modificado satisfactoriamente”. 3. En caso que el reporte de tipo cruzado exista se muestra un mensaje de error: “El elemento ya existe”. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba modificar el reporte” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación	

		"Perderá la información que no ha sido guardada. ¿Aun desea cancelar?". 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio "Campo requerido". 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: "Entre al menos 3 caracteres" y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.

Tabla 14. CENIA_PRE_ESR Mostrar reporte de tipo cruzado.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR12	Mostrar reporte de tipo cruzado.	Para mostrar un reporte de tipo cruzado se selecciona en el árbol del sistema el reporte deseado dando clic sobre él o clic derecho y selecciona la opción Abrir. Se muestran todos los datos del reporte de tipo cuantitativo seleccionado. En el área de íconos internos se muestran las opciones: Exportar en documento pdf y Exportar en documento excel. Además, se muestra la opción de seleccionar un tipo de grafica para mostrar los datos.	Alta	Alta
	Prototipo			



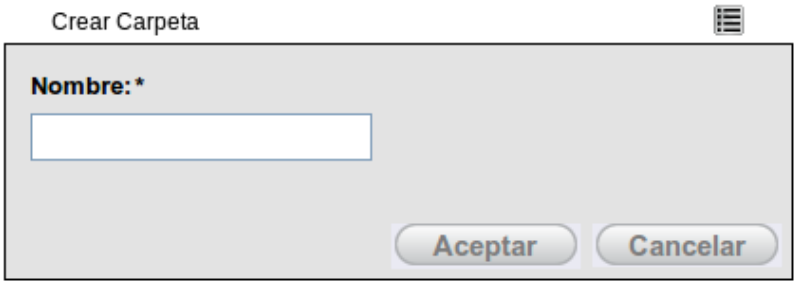
	Campos	*Tipos de Datos*	*Reglas o Restricciones*
		No procede	No procede
	Observaciones	1. El límite máximo de datos del reporte que se ven por página es de 20. 2. El límite mínimo de datos del reporte que se ven por página es de 5. 3. La cantidad de datos del reporte predeterminada es 5.	

Tabla 15. CENIA_PRE_ESR Crear carpeta.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR13	Crear carpeta.	Para crear una carpeta se selecciona la opción Crear en la barra de íconos flotantes o selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear la nueva carpeta y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nueva carpeta. El sistema recoge los datos: nombre. Una vez creada la carpeta se actualiza el listado y se muestra un mensaje de información: ¨El elemento	Alta	Alta

		ha sido creado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.		

Prototipo



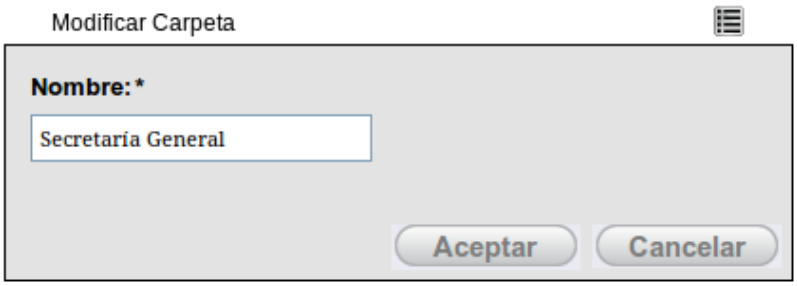
	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Nombre	Varchar	• Obligatorio • Admite entre 3 y 30 caracteres alfanuméricos y el caracter especial _.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se crea satisfactoriamente la carpeta el sistema mostrará un mensaje de información “El elemento ha sido creado satisfactoriamente”. 3. En caso de que exista una carpeta con el mismo nombre el sistema creará la nueva carpeta y como nombre le pondrá el definido por el usuario y adicionalmente un número que representa la cantidad de carpetas con el mismo nombre. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba crear el directorio” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aun desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo nombre introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo.	

		10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.

Tabla 16. CENIA_PRE_ESR Modificar carpeta.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR14	Modificar carpeta.	Para modificar una carpeta se selecciona e reporte deseado en el árbol del sistema y selecciona la opción Renombrar carpeta en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Modificar. El usuario modifica los datos deseados y presiona Aceptar. El sistema actualiza el listado y se muestra un mensaje de información: ¨El elemento ha sido modificado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.	Alta	Alta

Prototipo



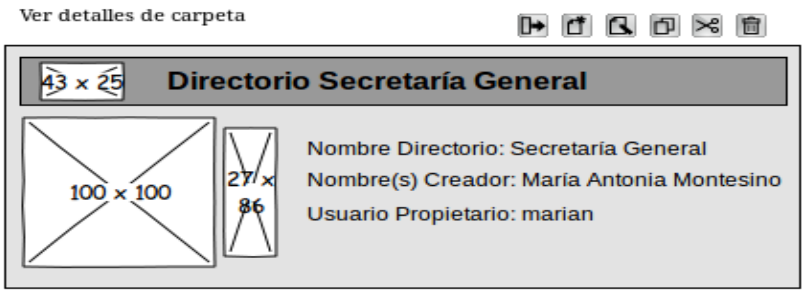
Campos	*Tipos de Datos*	*Reglas o Restricciones*
Nombre	Varchar	• Obligatorio • Admite entre 3 y 30 caracteres alfanuméricos y el caracter especial _.

	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se modifica satisfactoriamente la carpeta el sistema mostrará un mensaje de información “El elemento ha sido creado satisfactoriamente”.

		3. En caso de que exista una carpeta con el mismo nombre el sistema creará la nueva carpeta y como nombre le pondrá el definido por el usuario y adicionalmente un número que representa la cantidad de carpetas con el mismo nombre. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba modificar el directorio” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aun desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo nombre introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.

Tabla 17. CENIA_PRE_ESR Ver detalles de carpeta.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR15	Ver detalles de carpeta.	Para ver detalles de una carpeta se selecciona o se le pasa por encima en el árbol del sistema a la carpeta deseada. Se muestran los datos registrados de la carpeta: Nombre directorio, Nombre(s) creador y Usuario propietario. En el área de íconos flotantes se muestran las opciones: crear carpeta, crear reporte, renombrar carpeta, copiar, cortar y eliminar.	Baja	Baja
	Prototipo			



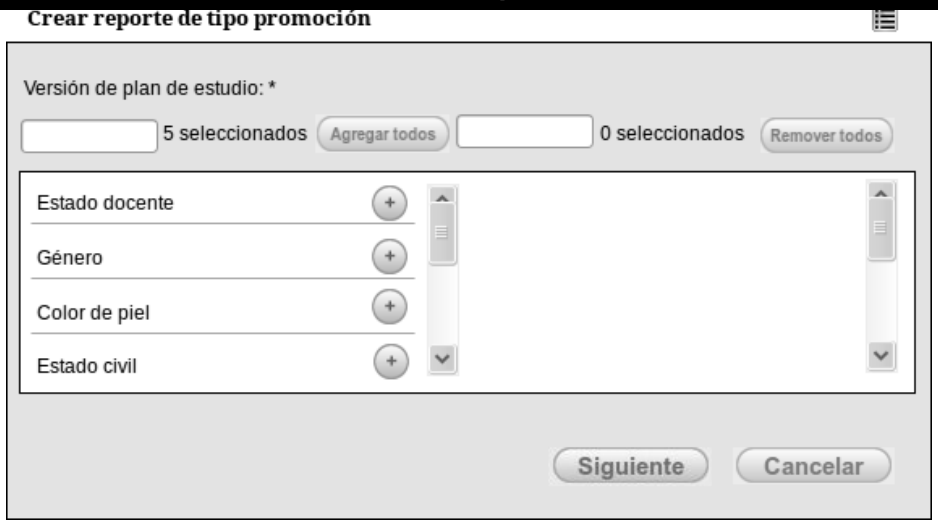
	Campos	Tipos de Datos	Reglas o Restricciones
	Nombre Directorio	Varchar	• Solo lectura
	Nombre(s) Creador	Varchar	• Solo lectura
	Usuario Propietario	Varchar	▪ Solo lectura
	Observaciones		

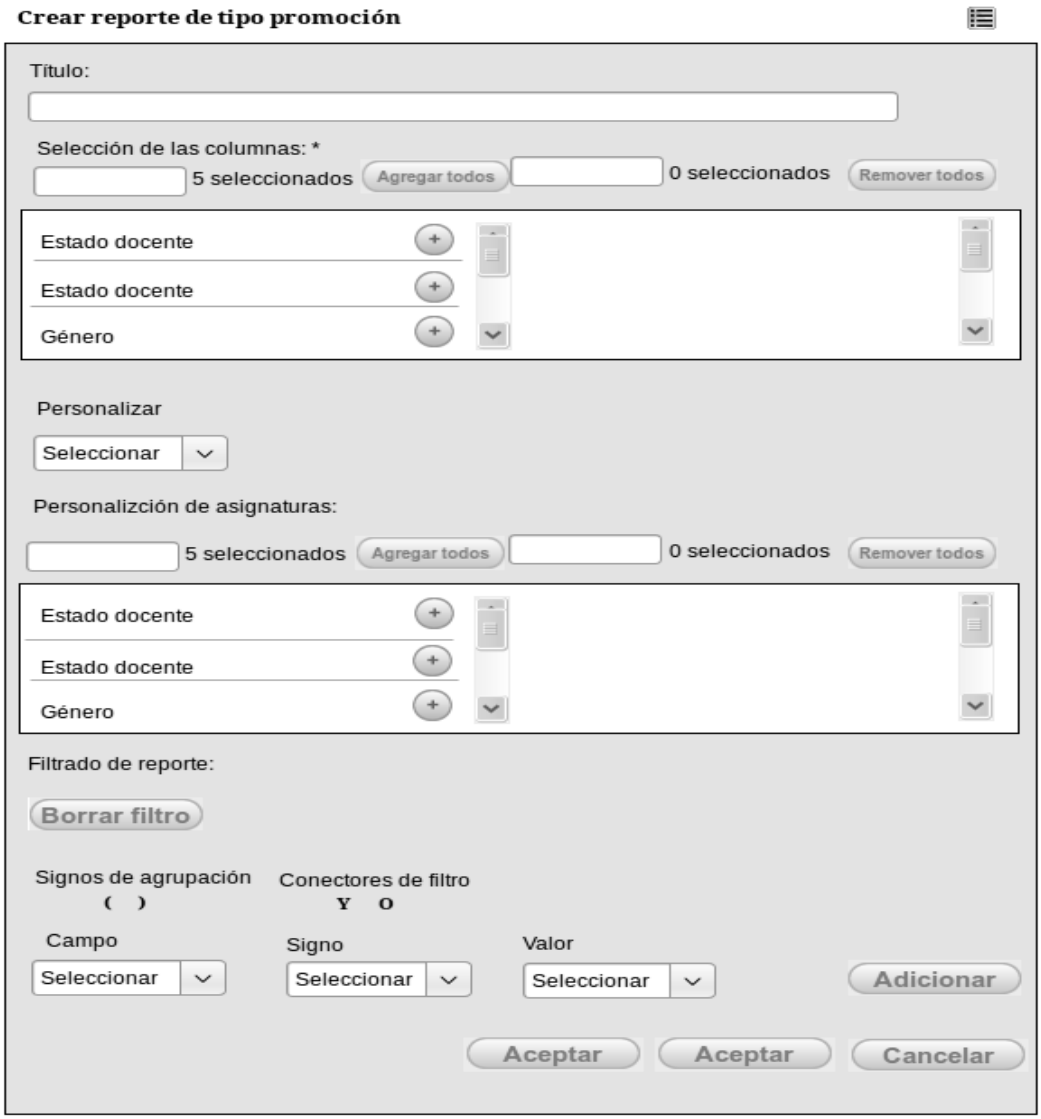
Tabla 18. CENIA_PRE_ESR Crear reporte de tipo promoción.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR16	Crear reporte de tipo promoción	Para crear un reporte de tipo promoción se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. El sistema recoge los datos: versión de plan de estudio, el usuario presiona el botón siguiente, pone título al reporte y selecciona las columnas que desea tenga el reporte de tipo promoción, si las columnas las columnas pertenecen a asignatura o a forma de evaluación puede personalizarlas y adiciona los filtros que desee mediante los campos: campo, signo y valor. Una vez creado el reporte se actualiza el listado y se	Alta	Alta

		muestra un mensaje de información: ¨El elemento ha sido creado satisfactoriamente¨. En el área de íconos internos se muestra la opción: Listar.		

Prototipo





	Campos	Tipos de Datos	Reglas o Restricciones
	Versión de plan de estudio	varchar	▪ Obligatorio. ▪ Campo de selección.
	Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
	Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Personalización de las asignaturas	varchar	▪ Campo de selección.
	Personalización de las formas de	varchar	▪ Campo de selección.

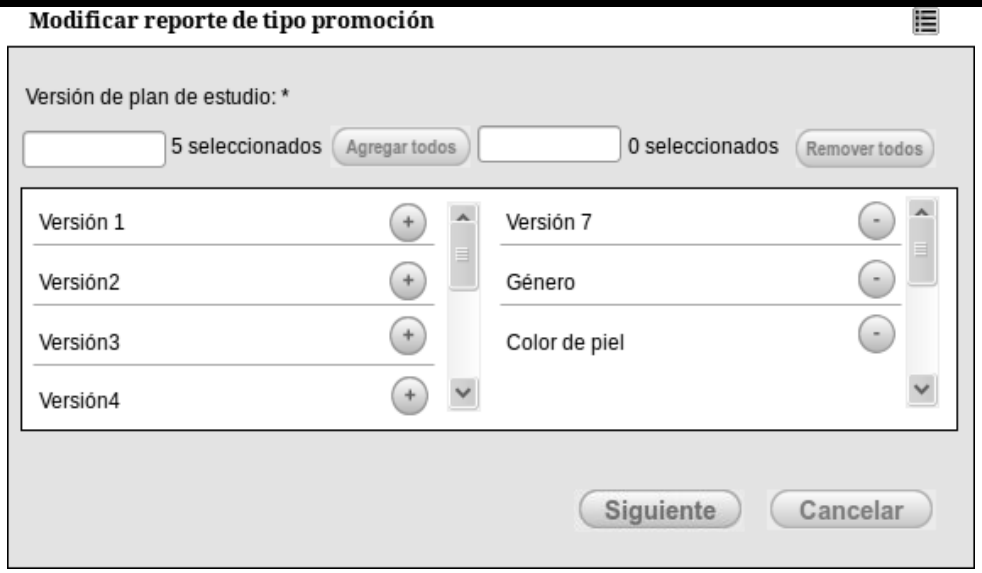
	evaluación		
	Campo	varchar	▪ Campo de selección.
	Signo	varchar	▪ Campo de selección.
	Valor	varchar	▪ Campo de selección.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se crea satisfactoriamente el reporte de tipo promoción el sistema mostrará un mensaje de información “El elemento ha sido creado satisfactoriamente”. 3. En caso que el reporte de tipo promoción exista se muestra un mensaje de error: “El elemento ya existe”. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba crear el” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aun desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.	

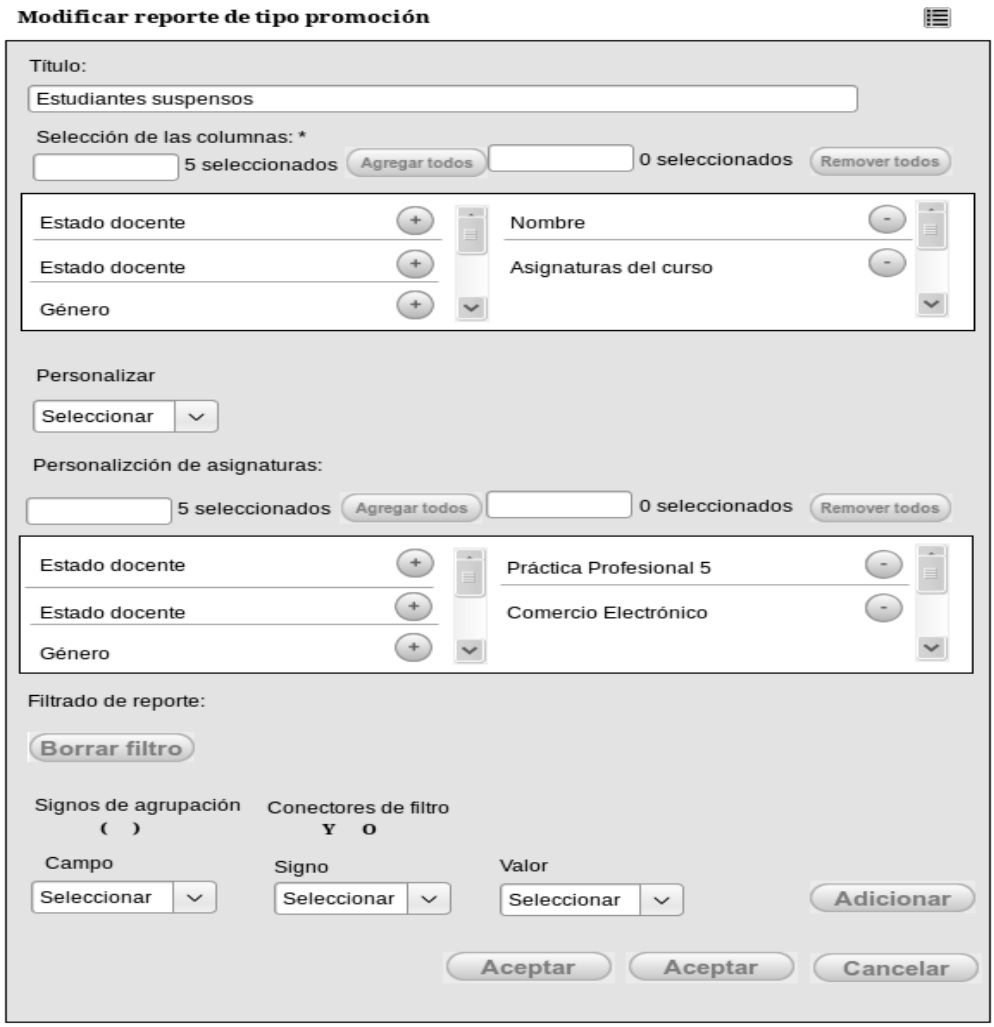
Tabla 19. CENIA_PRE_ESR Modificar reporte de tipo promoción.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR17	Modificar reporte de tipo promoción.	Para modificar un reporte de tipo promoción se selecciona el reporte deseado en el árbol del sistema y selecciona la opción Modificar en la barra de íconos flotantes o clic derecho sobre el reporte y selecciona la opción Modificar. El usuario modifica los datos deseados y presiona Aceptar. El sistema actualiza el listado y se muestra un mensaje de información: ¨El elemento ha sido modificado satisfactoriamente¨.	Alta	Alta

		En el área de íconos internos se muestra la opción: Listar.		

Prototipo



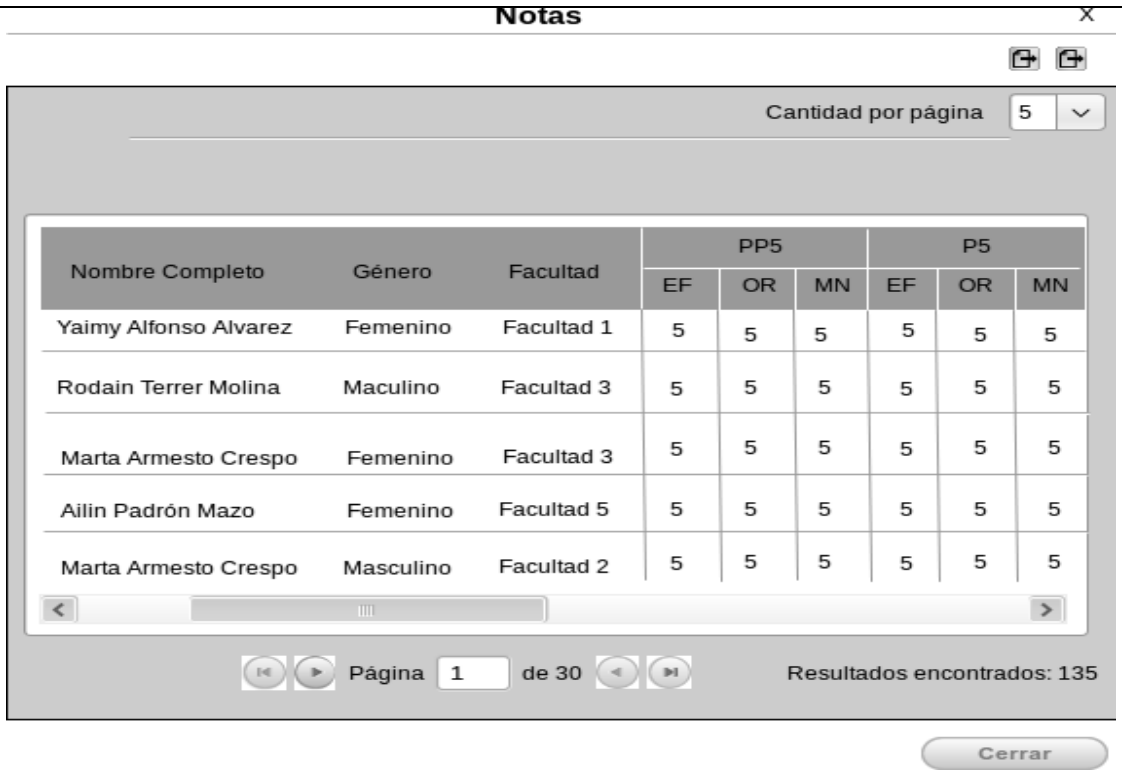


	Campos	Tipos de Datos	Reglas o Restricciones
	Versión de plan de estudio	varchar	▪ Obligatorio. ▪ Campo de selección.
	Título	varchar	▪ Admite entre 3 y 50 caracteres alfanuméricos y el caracter especial _.
	Selección de las columnas	varchar	▪ Obligatorio. ▪ Campo de selección.
	Personalización de las asignaturas	varchar	▪ Campo de selección.
	Personalización de las formas de evaluación	varchar	▪ Campo de selección.
	Campo	varchar	▪ Campo de selección.

	Signo	varchar	▪ Campo de selección.
	Valor	varchar	▪ Campo de selección.
	Observaciones	1. El usuario debe estar autenticado en el sistema. 2. Si se modifica satisfactoriamente el reporte de tipo promoción el sistema mostrará un mensaje de información “El elemento ha sido modificado satisfactoriamente”. 3. En caso que el reporte de tipo promoción exista se muestra un mensaje de error: “El elemento ya existe”. 4. Si ocurre un error durante la operación se muestra una ventana de error con el título: Error y el mensaje: “Ha ocurrido un error cuando intentaba crear el reporte” mostrando cual fue el error. 5. Si el usuario introduce caracteres extraños el sistema no permitirá escribirlos. 6. En caso de cancelar la acción se muestra un mensaje de confirmación “Perderá la información que no ha sido guardada. ¿Aún desea cancelar?”. 7. En caso de que deje campos obligatorios vacíos se mostrará un mensaje en rojo encima del campo que debe ser llenado obligatorio “Campo requerido”. 8. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 9. En caso de que el usuario en el campo Título introduzca menos cantidad de caracteres de los que están definidos se muestra un mensaje de error: “Entre al menos 3 caracteres” y se muestra en rojo el nombre del campo. 10. Si excede el número de letras permitidas por palabra el sistema muestra el mensaje: "Ha excedido el número de letras permitidas para una palabra" y se muestra en rojo el nombre del campo que debe ser llenado correctamente.	

Tabla 20. CENIA_PRE_ESR Mostrar reporte de tipo promoción.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RFR19	Mostrar reporte de tipo promoción.	Para mostrar un reporte de tipo promoción se selecciona en el árbol del sistema el reporte deseado dando clic sobre él o clic derecho y selecciona la opción Abrir. Se muestran todos los datos del reporte de tipo promoción seleccionado. En el área de íconos internos se muestran las opciones: Exportar en documento pdf y Exportar en documento excel.	Alta	Alta
	Prototipo			



	Campos	Tipos de Datos	Reglas o Restricciones
		No procede	No procede
	Observaciones	4. El límite máximo de datos del reporte que se ven por página es de 20. 5. El límite mínimo de datos del reporte que se ven por página es de 5. 6. La cantidad de datos del reporte predeterminada es 5.	

Anexo No.6 Organización de los reportes. Árbol jerárquico.

Figura 9. Árbol jerárquico de directorios.

Anexo No.7 Descripción de las clases implementadas

Tabla 21. Descripción de la clase reporte.php

Nombre: reporte	
Tipo de clase: controladora	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**index**
Descripción:	Función que carga la vista que define el área de contenido donde se van a hacer todas las operaciones del módulo.
Nombre:	**verReporte**
Descripción:	Función que visualiza un reporte.
Nombre:	**exportReporteToXLS**
Descripción:	Función que dado un reporte, lo exporta a formato de hoja de cálculo (.xls).
Nombre:	**exportarReportetoPDF**
Descripción:	Función que dado un reporte, lo exporta a formato PDF.
Nombre:	**checkPersonaLogueada**
Descripción:	Función que verifica si existe una persona logueada en el sistema.
Nombre:	**obtenerDinamicHtml**
Descripción:	Función que dado un campo crea un lista de selección con todos los valores posibles que puede tomar dicho campo.

Nombre:	**getDatosReporte**
Descripción:	Función que devuelve los datos de un reporte.
Nombre:	**cargarGrafica**
Descripción:	Función que carga la vista donde se construye la gráfica.
Nombre:	**cargarFormRegistrarCarpeta**
Descripción:	Función que carga la vista con el formulario necesario para registrar una carpeta o directorio.
Nombre:	**registrarCarpeta**
Descripción:	Función que recibe los datos entrados por el usuario para el registro o la modificación de un directorio.
Nombre:	**renombrarCarpeta**
Descripción:	Función que carga la vista con el formulario necesario para modificar un directorio.
Nombre:	**registrar**
Descripción:	Función que carga la vista con el formulario necesario para crear un reporte.
Nombre:	**modificarReporte**
Descripción:	Función que carga la vista con el formulario necesario para modificar un reporte.
Nombre:	**registrarReporte**
Descripción:	Función que recibe los datos entrados por el usuario para el registro o modificación de un reporte.
Nombre:	**efectuarSobreDirectorio**
Descripción:	Función que determina que acción realizar sobre un directorio (Copiar, Mover, Eliminar).
Nombre:	**datosReporte**
Descripción:	Función que carga una vista con detalles de un reporte.
Nombre:	**datosDirectorio**
Descripción:	Función que carga una vista con detalles de un directorio.
Nombre:	**arbolReportes**
Descripción:	Función que obtiene los datos necesarios para construir el árbol de reportes.
Nombre:	**buildTree**
Descripción:	Función que es llamada por la función anterior para construir el árbol de reportes.
Nombre:	**modificarUbicacionContenido**
Descripción:	Función que modifica la ubicación de un contenido (directorio o reporte) dentro del árbol.

Tabla 22. Descripción de la clase reporte_lib.php

Nombre: reporte_lib	
Tipo de clase: librería	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**obtenerPersonaLogueada**
Descripción:	Función que obtiene todos los datos de la persona que está logueada en el sistema.
Nombre:	**construirReporte**
Descripción:	Función que determina si se va a registrar un nuevo reporte o modificar uno ya existente.
Nombre:	**registrarReporte**
Descripción:	Función que efectúa la lógica necesaria para registrar un reporte.
Nombre:	**modificarReporte**
Descripción:	Función que efectúa la lógica necesaria para modificar un reporte.
Nombre:	**registrarCarpeta**
Descripción:	Función efectúa la lógica necesaria para registrar una carpeta.
Nombre:	**renombrarCarpeta**
Descripción:	Función efectúa la lógica necesaria para renombrar una carpeta.
Nombre:	**obtenerDatosCarpeta**
Descripción:	Función que obtiene los datos de una carpeta.
Nombre:	**preparandoScript**
Descripción:	Función que construye la consulta sql que obtiene la información a mostrar en el reporte.
Nombre:	**obtenerConfigReporte**
Descripción:	Función que obtiene la configuración de un reporte.
Nombre:	**obtenerDatos**
Descripción:	Función que obtiene y formatea la información a mostrar en un reporte.
Nombre:	**eliminarReporteTemporal**
Descripción:	Función que elimina los reportes temporales.
Nombre:	**eliminarReporte**

Descripción:	Función que efectúa la lógica necesaria para eliminar un reporte.
Nombre:	**eliminarCarpeta**
Descripción:	Función que efectúa la lógica necesaria para eliminar una carpeta.
Nombre:	**copiarReporte**
Descripción:	Función que efectúa la lógica necesaria para copiar un reporte.
Nombre:	**copiarCarpeta**
Descripción:	Función que efectúa la lógica necesaria para copiar una carpeta.
Nombre:	**moverReporte**
Descripción:	Función que efectúa la lógica necesaria para mover un reporte.
Nombre:	**moverCarpeta**
Descripción:	Función que efectúa la lógica necesaria para mover una carpeta.
Nombre:	**obtenerExploracion**
Descripción:	Función que efectúa la lógica necesaria para obtener todos los contenidos de una carpeta.
Nombre:	**obtenerDatosCarpeta**
Descripción:	Función que obtiene todos los datos de una carpeta.
Nombre:	**getDinamicHtmlValorFiltro**
Descripción:	Función utilizada para construir el filtro y datos un campo seleccionado obtiene los posibles valores que puede tomar ese campo.

Tabla 23. Descripción de la clase reporte_config_lib.php

Nombre: reporte_config_lib	
Tipo de clase: librería	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**dependencias**
Descripción:	Función que devuelve un arreglo con la dependencias existentes entre cada una de las tablas a partir de las cuales se obtendrán los datos del reporte.
Nombre:	**condicionesObligatorias**
Descripción:	Función que devuelve un arreglo con las condiciones que son obligatorias para construir la consulta a la base de datos que obtendrá los datos solicitados por

	el usuario.
Nombre:	**enlacesLeftJoin**
Descripción:	Establece entre cuales de las tablas se realizarán operaciones de tipo LEFT JOIN.
Nombre:	**tablasAlias**
Descripción:	Función que devuelve un arreglo que establece la configuración de todas las tablas que empleará el reportador, entendiéndose por configuración el esquema más la tabla o una subconsulta y adicionalmente un alias para la misma.
Nombre:	**arrayConfigConceptos**
Descripción:	Devuelve la configuración de los conceptos o campos empleados para los reportes nominales (listados, cuantitativos y cruzados), entendiéndose por configuración un nombre para mostrar, el campo al que hace referencia y de cuál tabla.
Nombre:	**arrayConfigConceptosPromocion**
Descripción:	Igual que arrayConfigConceptos pero solo para los reportes de promoción.
Nombre:	**arrayConfigConceptosEspecialesPromocion**
Descripción:	Devuelve la configuración de los campos dependientes (asignaturas y tipo de evaluación) de los reportes de promoción.

Tabla 24. Descripción de la clase tb_dcarpeta_mdl.php

Nombre: tb_dcarpeta_mdl	
Tipo de clase: entidad	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**copiarDirectorio**
Descripción:	Función que realiza la copia de un directorio o carpeta.
Nombre:	**nombreAdecuado**
Descripción:	Función que determinar si el nombre provisto por el usuario para la carpeta es el adecuado y en caso que no sea así establece un nombre válido para dicho directorio.

Nombre:	**pegarCarpeta**
Descripción:	Función que pega un directorio previamente copiado.
Nombre:	**eliminarDirectorio**
Descripción:	Función que elimina de la base de datos una carpeta.
Nombre:	**getAttributesTable**
Descripción:	Devuelve los atributos de la tabla.
Nombre:	**getShemaTable**
Descripción:	Devuelve el esquema y la tabla - *Example: public.table1.*
Nombre:	**getSchema**
Descripción:	Devuelve el esquema - *Example: public.*
Nombre:	**getTable**
Descripción:	Devuelve la tabla - *Example: table1.*
Nombre:	**getSqlSelectAttributes**
Descripción:	Devuelve una cadena con los atributos de la tabla a seleccionar.
Nombre:	**obtenerPorPagina**
Descripción:	Devuelve un arreglo con los datos acorde a dos parámetros que indican el inicio y la cantidad deseada a partir del inicio.
Nombre:	**registrar**
Descripción:	Registra un record y devuelve el id generado.
Nombre:	**modificar**
Descripción:	Actualiza los datos de un registro.
Nombre:	**obtenerDadoId**
Descripción:	Devuelve los datos de un registro.
Nombre:	**obtenerCantidad**
Descripción:	Devuelve la cantidad de elementos existentes en la tabla.
Nombre:	**eliminarDadoId**
Descripción:	Elimina dado un id.
Nombre:	**obtenerXml**
Descripción:	Devuelve un documento xml con los resultados de la consulta.

Tabla 25. Descripción de la clase tb_dconfig_reportes_mdl.php

Nombre: tb_config_reportes_mdl

Tipo de clase: entidad	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**obtenerExploracion**
Descripción:	Función que dado un directorio o carpeta obtiene el contenido de la misma.
Nombre:	**eliminarReportesTemporales**
Descripción:	Función que elimina todos los reportes temporales.
Nombre:	**eliminarReporte**
Descripción:	Función que elimina de la base de datos un determinado reporte.
Nombre:	**copiarReporte**
Descripción:	Función que realiza la copia de un reporte.
Nombre:	**moverReporte**
Descripción:	Función que mueve un reporte de un directorio hacia otro.
Nombre:	**getAttributesTable**
Descripción:	Devuelve los atributos de la tabla.
Nombre:	**getShemaTable**
Descripción:	Devuelve el esquema y la tabla - *Example: public.table1.*
Nombre:	**getSchema**
Descripción:	Devuelve el esquema - *Example: public.*
Nombre:	**getTable**
Descripción:	Devuelve la tabla - *Example: table1.*
Nombre:	**getSqlSelectAttributes**
Descripción:	Devuelve una cadena con los atributos de la tabla a seleccionar.
Nombre:	**obtenerPorPagina**
Descripción:	Devuelve un arreglo con los datos acorde a dos parámetros que indican el inicio y la cantidad deseada a partir del inicio.
Nombre:	**registrar**
Descripción:	Registra un reporte y devuelve el id generado.
Nombre:	**modificar**
Descripción:	Actualiza los datos de un registro.
Nombre:	**obtenerDadoId**

Descripción:	Devuelve los datos de un registro.
Nombre:	**obtenerCantidad**
Descripción:	Devuelve la cantidad de elementos existentes en la tabla.
Nombre:	**eliminarDadoId**
Descripción:	Elimina dado un id.
Nombre:	**obtenerXml**
Descripción:	Devuelve un documento xml con los resultados de la consulta.

Tabla 26. Descripción de la clase tb_dcontenido_mdl.php

Nombre: tb_dcontenido_mdl	
Tipo de clase: entidad	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**getAttributesTable**
Descripción:	Devuelve los atributos de la tabla.
Nombre:	**getShemaTable**
Descripción:	Devuelve el esquema y la tabla - *Example: public.table1.*
Nombre:	**getSchema**
Descripción:	Devuelve el esquema - *Example: public.*
Nombre:	**getTable**
Descripción:	Devuelve la tabla - *Example: table1.*
Nombre:	**getSqlSelectAttributes**
Descripción:	Devuelve una cadena con los atributos de la tabla a seleccionar.
Nombre:	**obtenerPorPagina**
Descripción:	Devuelve un arreglo con los datos acorde a dos parámetros que indican el inicio y la cantidad deseada a partir del inicio.
Nombre:	**registrar**
Descripción:	Registra un record y devuelve el id generado.
Nombre:	**modificar**
Descripción:	Actualiza los datos de un registro.
Nombre:	**obtenerDadoId**

Descripción:	Devuelve los datos de un registro.
Nombre:	**obtenerCantidad**
Descripción:	Devuelve la cantidad de elementos existentes en la tabla.
Nombre:	**eliminarDadoId**
Descripción:	Elimina dado un id.
Nombre:	**obtenerXml**
Descripción:	Devuelve un documento xml con los resultados de la consulta.

Tabla 27. Descripción de la clase reporte.js

Nombre: reporte	
Tipo de clase: librería javascript	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	**init**
Descripción:	Función que crea el área que contendrá todo lo relacionado con los reportes. Además, inicializa la barra de navegación o de íconos flotantes.
Nombre:	**initTooltip**
Descripción:	Función que inicializa el componente Tooltip.
Nombre:	**initMenu**
Descripción:	Función que crea e inicializa el menú contextual.
Nombre:	**initReporte**
Descripción:	Función javascript que utiliza la librería filetree.js para construir el árbol contenedor de los reportes.
Nombre:	**initDragAndDrop**
Descripción:	Función que hace posible que los elementos del árbol puedan ser arrastrados y soltados dentro de los directorios.
Nombre:	**explorarContenido**
Descripción:	Función que muestra los resultados del reporte.
Nombre:	**seleccionar**
Descripción:	Función que selecciona un elemento dentro del árbol.
Nombre:	**deshacerSeleccion**

Descripción:	Función que desmarca los elementos seleccionados.
Nombre:	**sistemaOperativoActivo**
Descripción:	Función que determina sobre qué sistema operativo se está trabajando.
Nombre:	**cargarNavBar**
Descripción:	Función que construye la barra de navegación o de íconos flotantes.
Nombre:	**initCrear**
Descripción:	Función que inicializa todos los componentes del formulario para registrar y modificar reportes.
Nombre:	**eliminarObjeto**
Descripción:	Función que elimina un objeto del filtro.
Nombre:	**campoSeleccionado**
Descripción:	Función que determina el campo que el usuario ha seleccionado para construir el filtro y a partir de este crear un select de HTML con todos lo valores que puede tomar el mismo.
Nombre:	**agregarFiltro**
Descripción:	Función que agrega un nuevo filtro al ya existente o crea uno nuevo.
Nombre:	**graficar**
Descripción:	Función que determina el tipo de gráfica que ha solicitado el usuario y a partir de esta manda a construirla mediante la llamada a métodos especializados en la construcción de un tipo de gráfica determinada.
Nombre:	**graficarColumnasSimples**
Descripción:	Método especializado en la construcción de gráficas de columnas simples.
Nombre:	**graficarBarrasSimples**
Descripción:	Método especializado en la construcción de gráficas de barras simples.
Nombre:	**graficarAreaSpline**
Descripción:	Método especializado en la construcción de gráficas de áreas con splines cúbicos.
Nombre:	**graficarAreasApiladas**
Descripción:	Método especializado en la construcción de gráficas de áreas apiladas.
Nombre:	**graficarLineasBasicas**
Descripción:	Método especializado en la construcción de gráficas de líneas básicas.
Nombre:	**graficarPie**

Descripción:	Método especializado en la construcción de gráficas de pasteles.
Nombre:	**clicRegistrarNavBar**
Descripción:	Define la acción a realizar al hacer clic en el botón “registrar” de la barra de navegación o íconos flotantes.
Nombre:	**clicCrearCarpetaNavBar**
Descripción:	Define la acción a realizar al hacer clic en el botón “crear carpeta” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicModificarReporte**
Descripción:	Define la acción a realizar al hacer clic en el botón “modificar” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicRenombrarCarpeta**
Descripción:	Define la acción a realizar al hacer clic en el botón “renombrar carpeta” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicCopiarReporte**
Descripción:	Define la acción a realizar al hacer clic en el botón “copiar reporte” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicCopiarCarpeta**
Descripción:	Define la acción a realizar al hacer clic en el botón “copiar carpeta” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicMoverReporte**
Descripción:	Define la acción a realizar al hacer clic en el botón “mover reporte” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicMoverCarpeta**
Descripción:	Define la acción a realizar al hacer clic en el botón “mover carpeta” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicEliminarReporte**
Descripción:	Define la acción a realizar al hacer clic en el botón “eliminar reporte” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicEliminarCarpeta**
Descripción:	Define la acción a realizar al hacer clic en el botón “eliminar carpeta” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicPegarReporte**

Descripción:	Define la acción a realizar al hacer clic en el botón “pegar reporte” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**clicPegarCarpeta**
Descripción:	Define la acción a realizar al hacer clic en el botón “pegar carpeta” de la barra de navegación o en la opción de igual nombre del menú contextual.
Nombre:	**validarFormulario**
Descripción:	Función que valida todos los formularios empleados por el módulo.

Anexo No.8 Descripción de la técnica: *Prueba del camino básico*

Esta técnica se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe, y representa un límite superior para el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada camino del programa. Los pasos a realizar para aplicar esta técnica son:

Paso 1. Representar el programa en un grafo de flujo.
Paso 2. Calcular la complejidad ciclomática.
Paso 3. Determinar el conjunto básico de caminos independientes.
Paso 4. Derivar los casos de prueba que fuerzan la ejecución de cada camino.

A continuación se detallan cada uno de estos pasos.

Paso 1. Representar el programa en un grafo de flujo

El grafo de flujo se utiliza para representar flujo de control lógico de un programa. Para ello se utilizan los tres elementos siguientes:

- Nodos: representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión (bifurcación).
- Aristas: líneas que unen dos nodos.
- Regiones: áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más.
- Nodos predicado: cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR,...) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina nodo predicado.

Cada construcción lógica de un programa tiene una representación. La figura 10 muestra dichas representaciones.

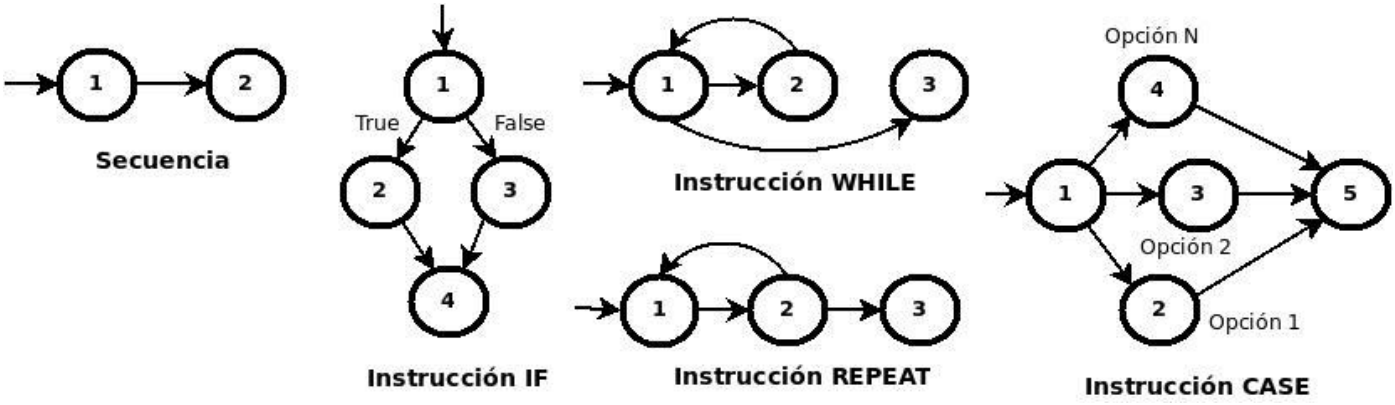


Figura 10. Representación en grafo de flujo de las estructuras lógicas de un programa.

Paso 2. Calcular la complejidad ciclomática

La complejidad ciclomática es una métrica del *software* que proporciona una medida cuantitativa de la complejidad lógica de un programa. En el contexto del método de prueba del camino básico, el valor de la complejidad ciclomática define el número de caminos independientes de dicho programa, y por lo tanto, el número de casos de prueba a realizar. Primero se mostrará cómo calcular ciclomática, a partir de un grafo de flujo, para obtener el número de caminos a identificar. Posteriormente se demostrará cómo se identifican esos caminos.

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

1. El número de regiones del grafo coincide con la complejidad ciclomática, V (G).
2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como:
   V (G) = Aristas – Nodos + 2
3. La complejidad ciclomática, V(G), de un grafo de flujo G se define como:
   V (G) = Nodos Predicado + 1

La complejidad ciclomática determina el número de casos de prueba que deben ejecutarse para garantizar que todas las sentencias de un programa se han ejecutado al menos una vez, y que cada condición se habrá ejecutado en sus vertientes verdadera y falsa.

Paso 3. Determinar el conjunto básico de caminos independientes

Un camino independiente es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una condición, respecto a los caminos existentes. En términos del diagrama de flujo, un camino independiente está constituido por lo menos por una arista que no haya

sido recorrida antes de la definición del camino. En la identificación de los distintos caminos de un programa para probar se debe tener en cuenta que cada nuevo camino debe tener el mínimo número de sentencias nuevas o condiciones nuevas respecto a los que ya existen. De esta manera, se intenta que el proceso de depuración sea más sencillo.

El conjunto de caminos independientes de un grafo no es único. No obstante, a continuación se muestran algunas heurísticas para identificar dichos caminos:

a) Elegir un camino principal que represente una función válida que no sea un tratamiento de error. Debe intentar elegirse el camino que atraviese el máximo número de decisiones en el grafo.
b) Identificar el segundo camino mediante la localización de la primera decisión en el camino de la línea básica alternando su resultado mientras se mantiene el máximo número de decisiones originales del camino inicial.
c) Identificar un tercer camino, colocando la primera decisión en su valor original a la vez que se altera la segunda decisión del camino básico, mientras se intenta mantener el resto de decisiones originales.
d) Continuar el proceso hasta haber conseguido tratar todas las decisiones, intentando mantener como en su origen el resto de ellas.

Paso 4. Derivar los casos de prueba que fuerzan la ejecución de cada camino

El último paso es construir los casos de prueba que fuerzan la ejecución de cada camino. Una forma de representar el conjunto de casos de prueba es como se muestra en la siguiente tabla.

Número del camino	Caso de prueba	Resultado esperado

Anexo No.9 Descripción de la técnica: *Particiones de equivalencia*

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realzada con

cualquier otro valor de dicha clase. Esto quiere decir que si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. Y viceversa, si un caso de prueba no ha detectado ningún error, es de esperar que ninguno de los casos de prueba correspondientes a la misma clase de equivalencia encuentre ningún error.

El diseño de casos de prueba según esta técnica consta básicamente de dos pasos:

1. Identificar las clases de equivalencia.
2. Identificar los casos de prueba.

A continuación se detallan cada uno de estos pasos.

Paso 1. Identificar las clases de equivalencia

Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Las clases de equivalencia se identifican examinando cada condición de entrada (normalmente una frase en la especificación) y dividiéndola en dos o más grupos. Se definen dos tipos de clases de equivalencia, las clases de equivalencia válidas, que representan entradas válidas al programa, y las clases de equivalencia no válidas, que representan valores de entrada erróneos. Estas clases se pueden representar en una tabla como la siguiente.

Condición externa	Clases de equivalencia válidas	Clases de equivalencia no válidas

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

1. Si una condición de entrada especifica un *rango*, se define una clase de equivalencia válida y dos no válidas.
2. Si una condición de entrada requiere un *valor* específico, se define una clase de equivalencia válida y dos no válidas.
3. Si una condición de entrada especifica un miembro de un *conjunto*, se define una clase de equivalencia válida y una no válida.
4. Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

Paso 2. Identificar los casos de prueba

El objetivo es minimizar el número de casos de prueba, así cada caso de prueba debe considerar tantas condiciones de entrada como sea posible. No obstante, es necesario realizar con cierto cuidado los casos de prueba, de manera que no se enmascaren faltas. Así, para crear los casos de prueba a partir de las clases de equivalencia se han de seguir los siguientes pasos:

1. Asignar a cada clase de equivalencia un número único.
2. Hasta que todas las clases de equivalencia hayan sido cubiertas por los casos de prueba, se tratará de escribir un caso que cubra tantas clases válidas no incorporadas como sea posible.
3. Hasta que todas las clases de equivalencia no válidas hayan sido cubiertas por casos de prueba, escribir un caso para cubrir una única clase no válida no cubierta.

La razón de cubrir con casos individuales las clases no válidas es que ciertos controles de entrada pueden enmascarar o invalidar otros controles similares. Por ejemplo, si tenemos dos clases válidas: *"introducir cantidad entre 1 y 99"* y *"seguir con letra entre A y Z"*, el caso *"105 1"* (dos errores) puede dar como resultado *105 fuera de rango de cantidad*, y no examinar el resto de la entrada no comprobando así la respuesta del sistema ante una posible entrada no válida.

Anexo No.10 Mecanismos empleados para la validación del filtro

Gramática que genera todas las formas posibles que pueden tener los filtros:

```
<filtros> → <filtro><más_filtros>
<más_filtros> → tkAnd <filtro><más_filtros> | tkOr <filtro><más_filtro> | ε
<filtro> → tkOpenPar <filtros> tkClosePar | tkCadena <signos> tkCadena
<signos> → tkCompIgualdad | tkCompDiff | tkCompContiene | tkMayor | tkMenor
```

En donde:

ε: representa una cadena vacía, es decir, una cadena sin elementos.

tkAnd: representa el operador lógico *And* (en un lenguaje natural equivale a la conjunción copulativa *Y*).

tkOr: representa el operador lógico *Or* (en un lenguaje natural equivale a la conjunción disyuntiva *O*).

tkOpenPar: representa el paréntesis abierto "(".

tkClosePar: representa el paréntesis cerrado ")".

tkCadena: representa una cadena formada por solo letras (a…z o A…Z).

tkCompIgualdad: representa la igualdad entre dos elementos.

tkCompDiff: representa la desigualdad entre dos elementos.

tkCompContiene: representa que un elemento está contenido en otro.

tkMayor: representa que un elemento es mayor que otro.

tkMenor: representa que un elemento es menor que otro.

Representación de la clase interprete_filtro

```
Clase interprete_filtro
INICIO
     protegido:
                String elemento_actual;    /* Elemento del filtro que se
está
                                           Analizando */
                Array filtros;      /* Arreglo que contiene cada unos de los
                                    elementos que componen el filtro */
                Lista errores;      /* Lista de los errores sintácticos */
                Integer pos_elemento_actual;     /* Posición del elemento
del
                                                  que se está analizando */
     publico:
                interprete_filtro()
                INICIO
                     pos_elemento_actual = 0;
                     elemento_actual = filtros[pos_elemento_actual];
                FIN
                Match(Tipo)
                INICIO
                     SI elemento_actual.tipo = Tipo ENTONCES
                          pos_elemento_actual = pos_elemento_actual + 1;
                          elemento_actual = filtros[pos_elemento_actual]
                     SINO
                          errores.adicionar("Error: se esperaba" + Tipo
                          + "y se encontró" + elemento_actual.tipo)
                     FIN SI
                FIN

/* Regla 1: <filtros> → <filtro><más_filtros> */
                Filtros()
                INICIO
                     Filtro();
                     Más_Filtros();
                FIN

/* Regla 2: <<más_filtros> → tkAnd <filtro><más_filtros> | tkOr
<filtro><más_filtro> | ε */
```

```
                Más_Filtros()
                INICIO
                     SI elemento_actual.tipo = tkAnd ENTONCES
                          Match(tkAnd);
                          Filtro();
                          Más_Filtros();
                     SINO SI elemento_actual.tipo = tkOr ENTONCES
                          Match(tkOr);
                          Filtro();
                          Más_Filtros();
                     SINO
                          errores.adicionar("Error: se esperaba" + Tipo
                          + "y se encontró" + elemento_actual.tipo)
                     FIN SI
                FIN

/* Regla 3: <filtro> → tkOpenPar <filtros> tkClosePar | tkCadena <signos>
tkCadena  */
                Filtro()
                INICIO
                     SI elemento_actual.tipo = tkOpenPar ENTONCES
                          Match(tkOpenPar);
                          Filtros();
                          Match(tkClosePar);
                     SINO SI elemento_actual.tipo = tkCadena ENTONCES
                          Match(tkCadena);
                          Signos();
                          Match(tkCadena);
                     SINO
                          errores.adicionar("Error: se esperaba" + Tipo
                          + "y se encontró" + elemento_actual.tipo)
                     FIN SI
                FIN

/* Regla 4: <signos> → tkCompIgualdad | tkCompDiff | tkCompContiene */
                Signos()
                INICIO
                     SI elemento_actual.tipo = tkCompIgualdad ENTONCES
                          Match(tkCompIgualdad);
                     SINO SI elemento_actual.tipo = tkCompContiene ENTONCES
                          Match(tkCompContiene);
                     SINO SI elemento_actual.tipo = tkCompDiff ENTONCES
                          Match(tkCompDiff);
```

```
                    SINO SI elemento_actual.tipo = tkMayor ENTONCES
                         Match(tkMayor);
                    SINO SI elemento_actual.tipo = tkMenor ENTONCES
                         Match(tkMenor);
                    SINO
                         errores.adicionar("Error: se esperaba" + Tipo
                         + "y se encontró" + elemento_actual.tipo)
                    FIN SI
               FIN

FIN
```

Anexo No.11 Resultados de las pruebas realizadas con la herramienta jmeter 2.6

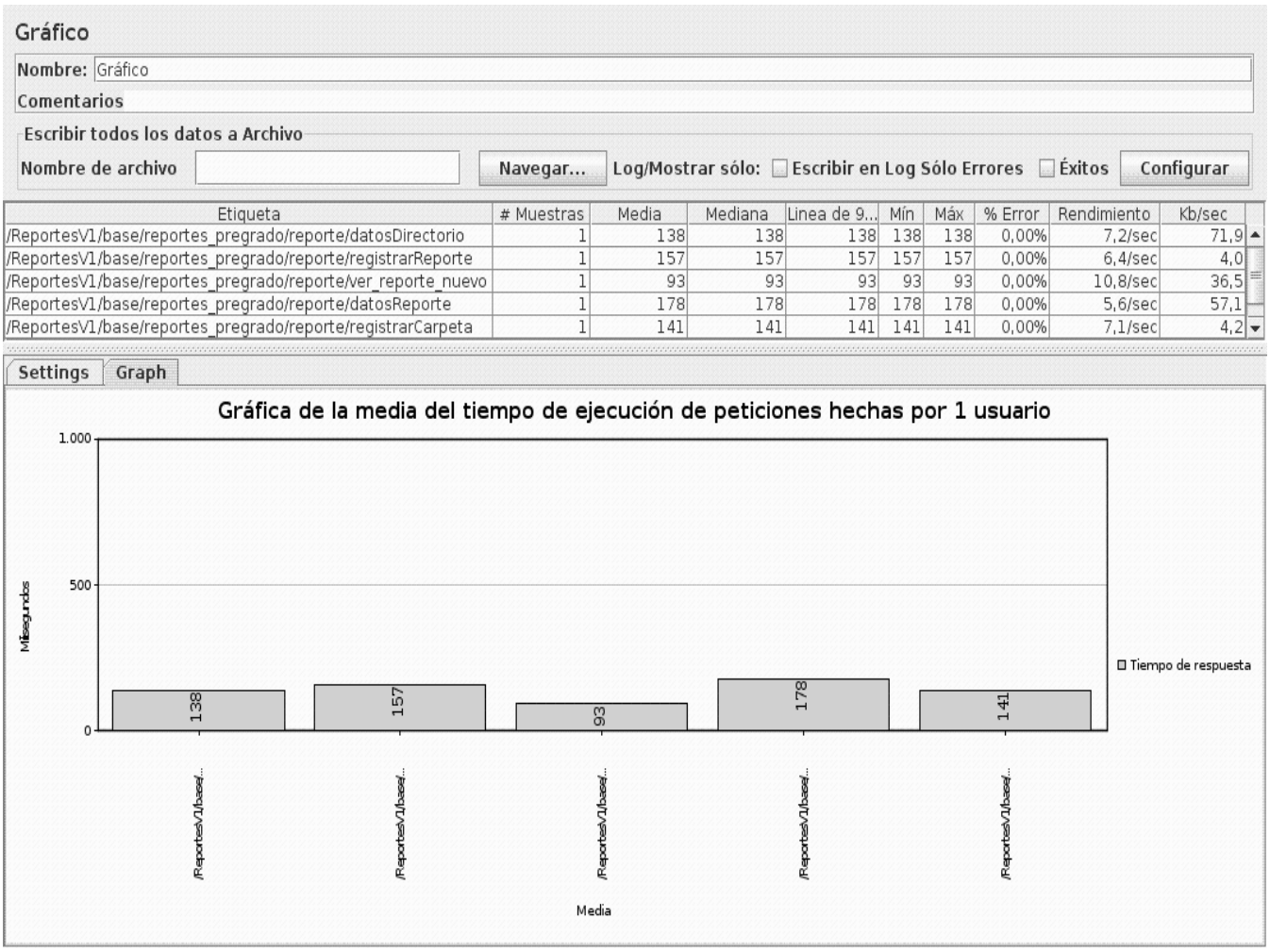


Figura 11. Resultados mostrados por jmeter 2.6 con una muestra de 1 usuario.

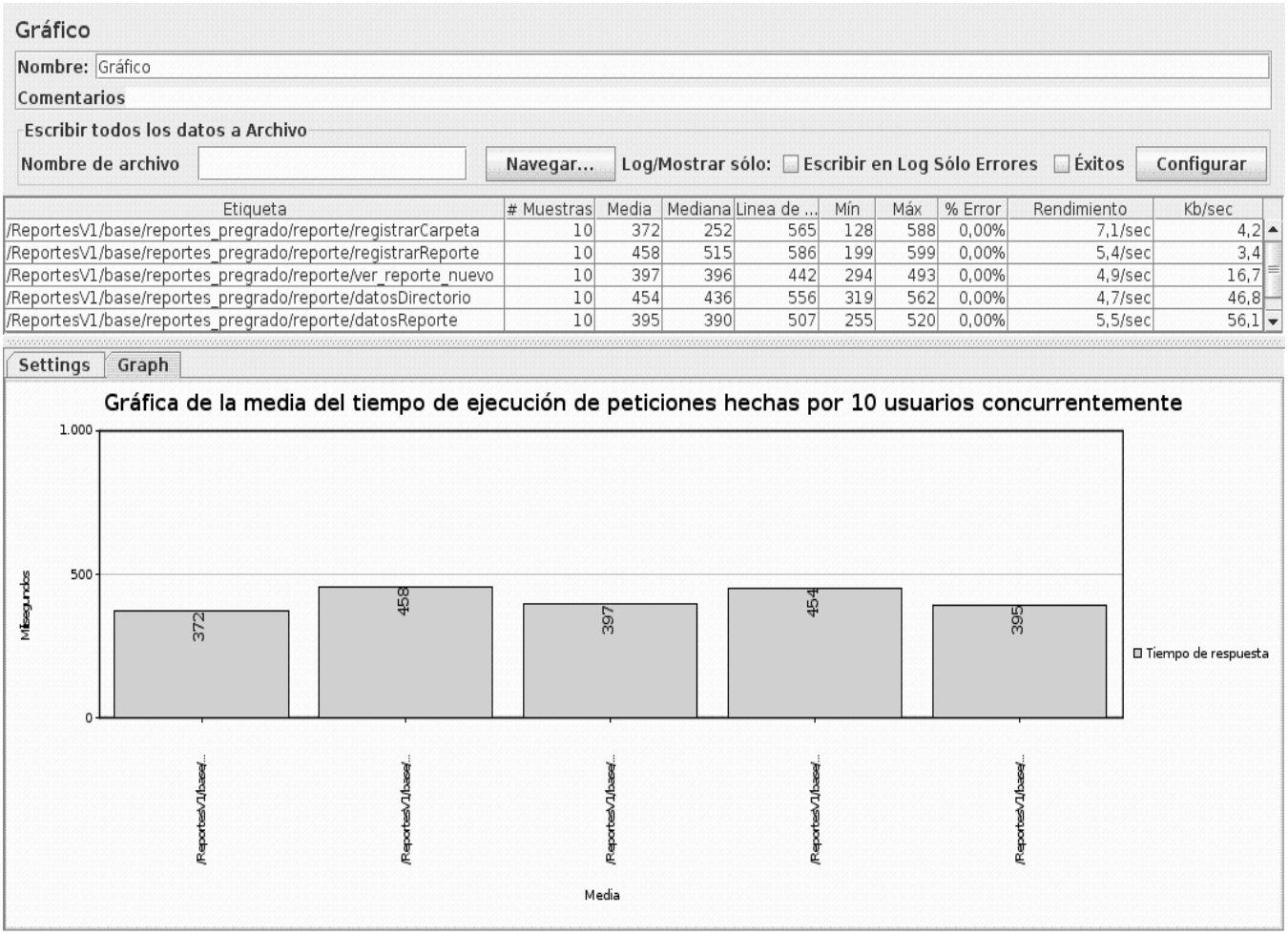
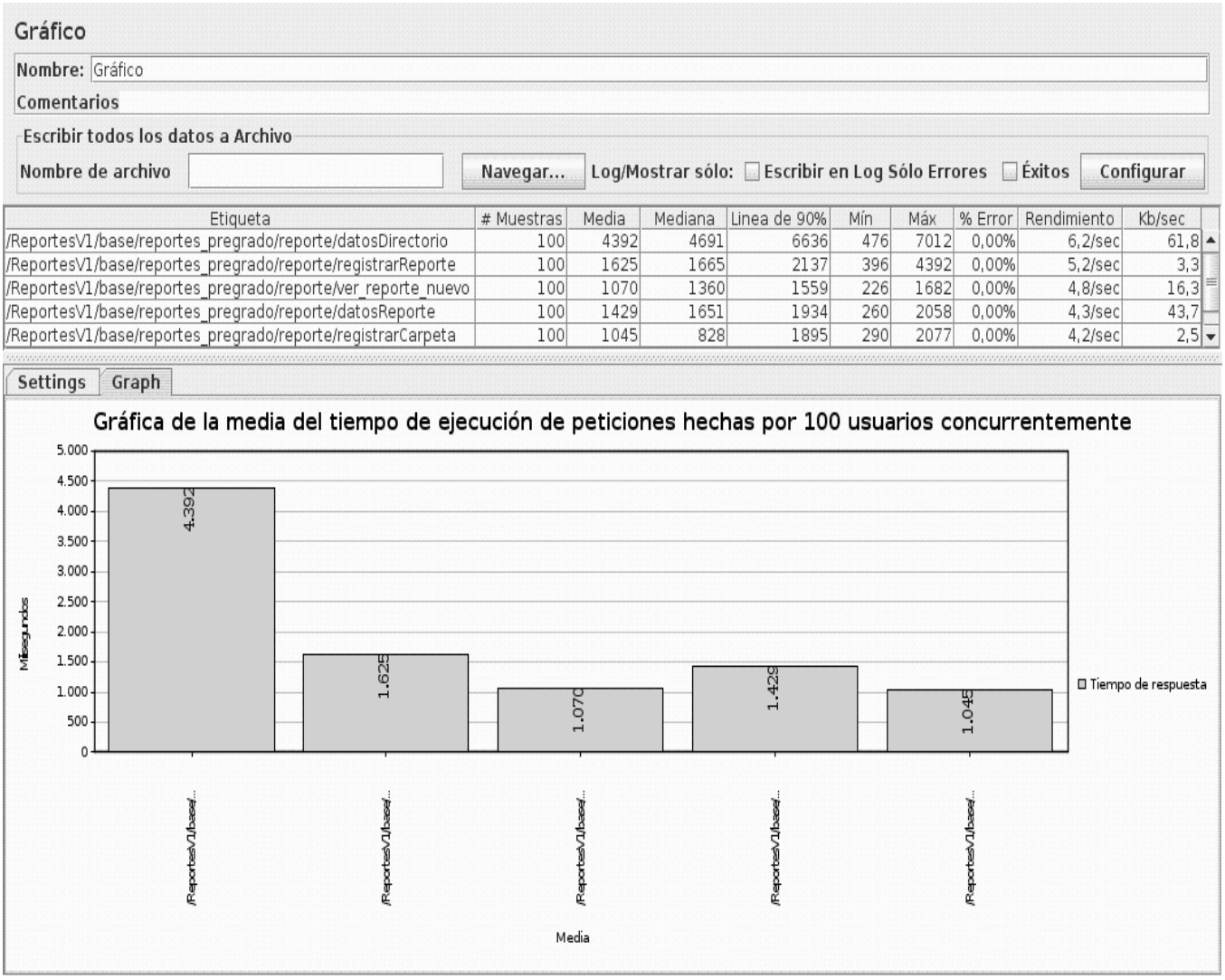
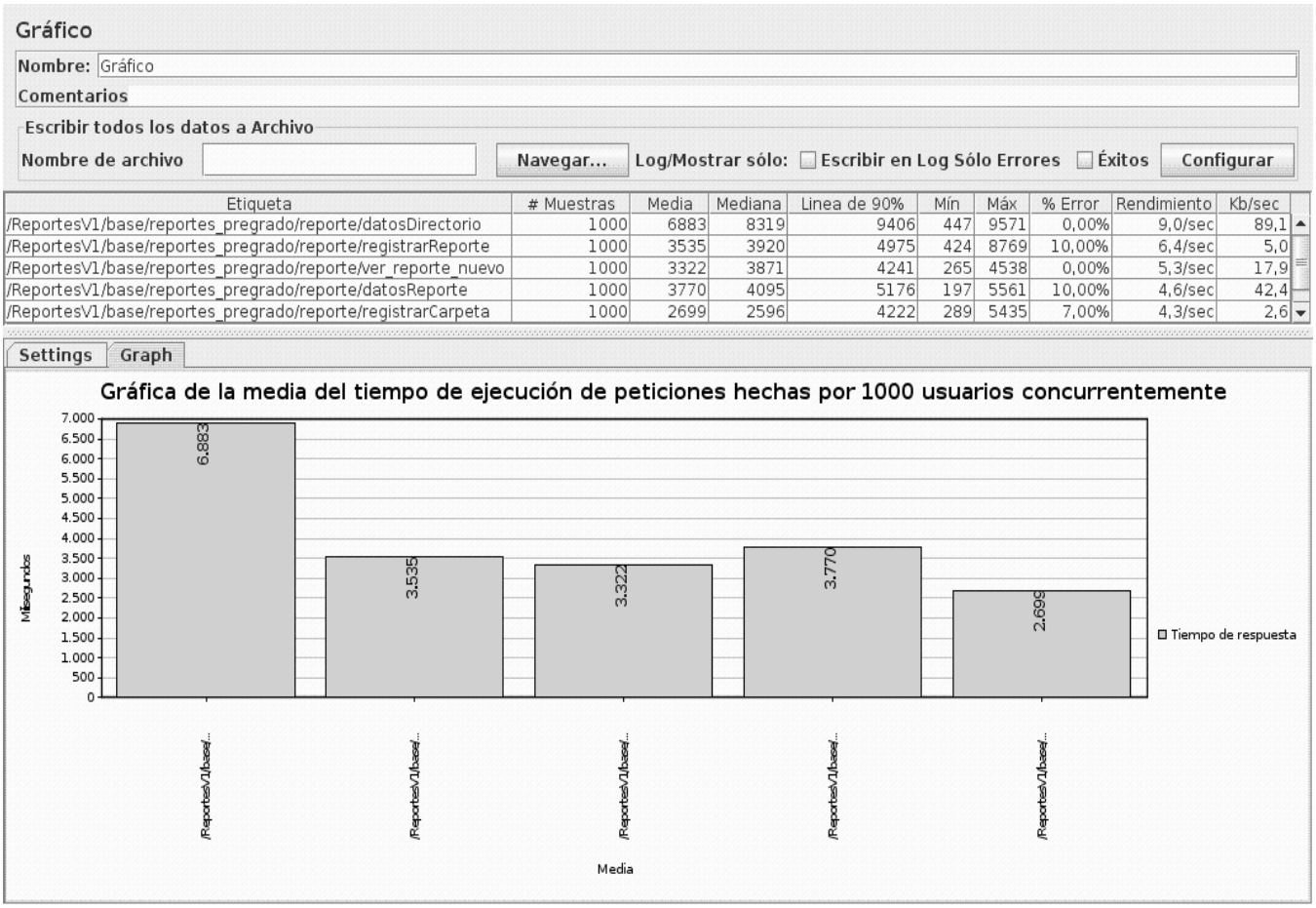


Figura 12. Resultados mostrados por jmeter 2.6 con una muestra de 10 usuarios.



Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/ReportesV1/base/reportes_pregrado/reporte/datosDirectorio	100	4392	4691	6636	476	7012	0,00%	6,2/sec	61,8
/ReportesV1/base/reportes_pregrado/reporte/registrarReporte	100	1625	1665	2137	396	4392	0,00%	5,2/sec	3,3
/ReportesV1/base/reportes_pregrado/reporte/ver_reporte_nuevo	100	1070	1360	1559	226	1682	0,00%	4,8/sec	16,3
/ReportesV1/base/reportes_pregrado/reporte/datosReporte	100	1429	1651	1934	260	2058	0,00%	4,3/sec	43,7
/ReportesV1/base/reportes_pregrado/reporte/registrarCarpeta	100	1045	828	1895	290	2077	0,00%	4,2/sec	2,5

Figura 13. Resultados mostrados por jmeter 2.6 con una muestra de 100 usuarios.



Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/ReportesV1/base/reportes_pregrado/reporte/datosDirectorio	1000	6883	8319	9406	447	9571	0,00%	9,0/sec	89,1
/ReportesV1/base/reportes_pregrado/reporte/registrarReporte	1000	3535	3920	4975	424	8769	10,00%	6,4/sec	5,0
/ReportesV1/base/reportes_pregrado/reporte/ver_reporte_nuevo	1000	3322	3871	4241	265	4538	0,00%	5,3/sec	17,9
/ReportesV1/base/reportes_pregrado/reporte/datosReporte	1000	3770	4095	5176	197	5561	10,00%	4,6/sec	42,4
/ReportesV1/base/reportes_pregrado/reporte/registrarCarpeta	1000	2699	2596	4222	289	5435	7,00%	4,3/sec	2,6

Figura 14. Resultados mostrados por jmeter 2.6 con una muestra de 1000 usuarios.

Anexo No.12 Pruebas de integración realizadas a la propuesta de solución

Tabla 28. CENIA_PRE_CPI-Personal y Secretaría.

Caso de Prueba: INT-PS
Módulo al que se integra: Personal y Secretaría
Condiciones de ejecución: El módulo de Personal y Secretaría haya introducido los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba: Comprobar que el módulo de Reportes es capaz de confeccionar reportes con información gestionada por el módulo de Personal y Secretaría.
Entradas/Pasos de ejecución: El módulo de Personal y Secretaría introduce en la base de datos central los datos y el módulo de Reportes consulta estos datos y crea el reporte.
Resultado esperado: Se crea el reporte con los datos.
Evaluación: Prueba satisfactoria.

Tabla 29. CENIA_PRE_CPI-Control Docente.

Caso de Prueba: : INT-CD
Módulo al que se integra: Control Docente.
Condiciones de ejecución: El módulo de Control Docente haya introducido los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba: Comprobar que el módulo de Reportes es capaz de confeccionar reportes con información gestionada por el módulo de Control Docente.
Entradas/Pasos de ejecución: El módulo de Control Docente introduce en la base de datos central los datos y el módulo de Reportes consulta estos datos y crea el reporte.
Resultado esperado: Se crea el reporte con los datos.
Evaluación: Prueba satisfactoria.

Tabla 30. CENIA_PRE_CPI-Diseño de Carrera.

Caso de Prueba: INT-DC
Módulo al que se integra: Diseño de Carrera.
Condiciones de ejecución: El módulo de Diseño de Carrera haya introducido los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba: Comprobar que el módulo de Reportes es capaz de confeccionar reportes con información gestionada por el módulo de Diseño de Carrera.
Entradas/Pasos de ejecución: El módulo de Diseño de Carrera introduce en la base de datos central los datos y el módulo de Reportes consulta estos datos y crea el reporte.
Resultado esperado: Se crea el reporte con los datos.
Evaluación: Prueba satisfactoria.

Tabla 31. CENIA_PRE_CPI-Tesis y Título.

Caso de Prueba: INT-TT
Módulo al que se integra: Tesis y Título.
Condiciones de ejecución: El módulo de Tesis y Título haya introducido los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba: Comprobar que el módulo de Reportes es capaz de confeccionar reportes con información gestionada por el módulo de Tesis y Título.
Entradas/Pasos de ejecución: El módulo de Tesis y Título introduce en la base de datos central los datos y el módulo de Reportes consulta estos datos y crea el reporte.
Resultado esperado: Se crea el reporte con los datos.

Evaluación: Prueba satisfactoria.

Anexo No.13 Pruebas unitarias realizadas a la propuesta de solución

13.1 Casos de pruebas de caja blanca

Antes de proceder con la descripción de cada uno de los casos de pruebas es necesario tener en cuenta las siguientes consideraciones:

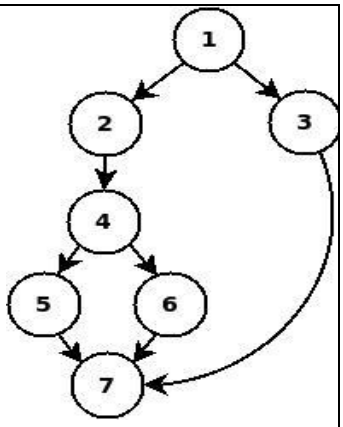
1. Para la gestión del tipo de contenido *reporte* se emplean las mismas funciones (*registrarReporte*, *modificarReporte*, *mostrarReporte* y *verDatosReportes*) independientemente del tipo de reporte que sea (cruzado, cuantitativo, listado o promoción). Por esta razón para cada una de estas funcionalidades solo se presentará un caso de prueba en representación de todos. En estos casos el caso de prueba es identificado con un código con el formato:
   CENIA_PRE_PUCPCB *+ identificador de la acción que se realiza (**MR:** modificar reporte, **RR:** registrar reporte, **VR:** mostrar reporte, **DR:** detalles del reporte, **RC:** registrar carpeta, **DC:** detalles de carpeta) +* ***identificadores de los requerimientos a los que hace referencia*** *(separados por comas).*
2. Los requerimientos relacionados con la modificación de los tipos de contenidos (reportes y carpetas) utilizan la misma función que los relacionados con la creación de estos. Esto se debe a que la función dicha función internamente es capaz de determinar si se está creando o modificando un contenido.

Tabla 32. CENIA_PRE_PUCPCB-RR: RFR1, RFR5, RFR9, RFR13.

Prueba estructural o caja blanca	**Código caso de prueba :** CENIA_PRE_PUCPCB-RR: RFR1, RFR5, RFR9, RFR13
Probador: Rodain Terrer Molina	
Código al que se aplica:	**Representación en grafo de flujo:**

```
public function registrarReporte($all_vars) {
    if (isset($all_vars['columnas']) && !empty($all_vars['columnas'])) {        1
        $response = array();
        $result = $this->reporte_lib->construir_reporte($all_vars, $response);  2

        if ($result) {          4
            $response['success'] = TRUE;                                        5
            $response['titulo_reporte'] = $result["titulo_reporte"];            5
            $response['id_configuracion'] = $result["id_configuracion"];        5
            return $response['success'];                                        5
        }
        else
            return "Ocurrió un error durante la operación. Intente más tarde."; 6
    }
    else
        return "Uno o varios elementos se ha introducido de forma incorrecta."; 3
}   7
```



Complejidad ciclomática:

V (G) = (A - N) + 2 = (8 - 7) + 2 = 3

Caminos independientes:

1. 1 - 3 -7
2. 1 - 2 - 4 - 5 -7
3. 1 - 2 - 4 - 6 -7

Caso de prueba para el camino básico No. 1	
Descripción: los datos de entrada serán atributos del reporte a crear.	
Condición de ejecución: los datos de los atributos no son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	Arreglo vacío.
Tipo de dato esperado:	is_string

Función de evaluación:
`$resultEsperado = "Uno o varios elementos se ha introducido de forma incorrecta."`
`$nombrePrueba = "Prueba: 'Crear reporte de tipo listado'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No. 2	
Descripción: los datos de entrada serán atributos del reporte a crear.	
Condición de ejecución: los datos de los atributos son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	Arreglo con los datos del reporte insertados por el usuario.
Tipo de dato esperado:	is_bool

Función de evaluación:
`$resultEsperado = TRUE`
`$nombrePrueba = "Prueba: 'Crear reporte de tipo listado'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No. 3	
Descripción:	
Condición de ejecución:	

Procedimiento prueba automatizada	
Datos de entrada:	Arreglo con los datos del reporte insertados por el usuario.
Tipo de dato esperado:	is_string

Función de evaluación:
`$resultEsperado = "Ocurrió un error durante la operación. Intente más tarde."`
`$nombrePrueba = "Prueba: 'Crear reporte de tipo listado'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Resultado final de la prueba: SATISFACTORIA EN SU TOTALIDAD	

Tabla 33. CENIA_PRE_PUCPCB-MR: RFR2. RFR6, RFR10, RFR14.

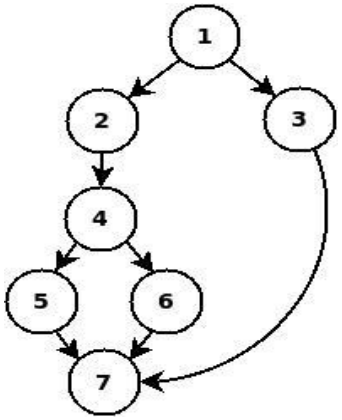
Prueba estructural o caja blanca	**Código caso de prueba:** CENIA_PRE_PUCPCB-MR: RFR2. RFR6, RFR10, RFR14
Probador: Rodain Terrer Molina	

Código al que se aplica:

```
public function registrarReporte($all_vars) {                              
    if (isset($all_vars['columnas']) && !empty($all_vars['columnas'])) {   1
        $response = array();
        $result = $this->reporte_lib->construir_reporte($all_vars, $response); 2

        if ($result) {                                                     4
            $response['success'] = TRUE;                                   5
            $response['titulo_reporte'] = $result["titulo_reporte"];       5
            $response['id_configuracion'] = $result["id_configuracion"];   5
            return $response['success'];                                   5
        }
        else
            return "Ocurrió un error durante la operación. Intente más tarde."; 6
    }
    else
        return "Uno o varios elementos se ha introducido de forma incorrecta."; 3
}   7
```

Representación en grafo de flujo:



Complejidad ciclomática:

V (G) = (A - N) + 2 = (8 - 7) + 2 = 3

Caminos independientes:

1. 1 - 3 -7
2. 1 - 2 - 4 - 5 -7
3. 1 - 2 - 4 - 6 -7

Caso de prueba para el camino básico No. 1	
Descripción: los datos de entrada serán atributos del reporte a crear.	
Condición de ejecución: los datos de los atributos no son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	Arreglo vacío.
Tipo de dato esperado:	is_string

Función de evaluación:
`$resultEsperado = "Uno o varios elementos se ha introducido de forma incorrecta."`
`$nombrePrueba = "Prueba: 'Modificar reporte'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	

Caso de prueba para el camino básico No. 2	
Descripción: los datos de entrada serán atributos del reporte a crear.	
Condición de ejecución: los datos de los atributos son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	Arreglo con los datos del reporte insertados por el usuario.
Tipo de dato esperado:	is_bool

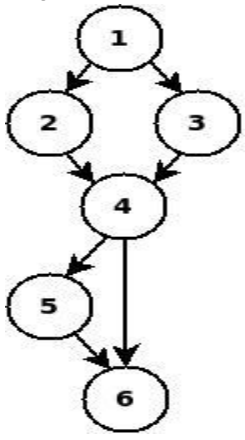
Función de evaluación:
`$resultEsperado = TRUE`
`$nombrePrueba = ""Prueba: 'Modificar reporte'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No. 3	
Descripción: los datos de entrada serán atributos del reporte a crear.	
Condición de ejecución: ocurre un error de base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	Arreglo con los datos del reporte insertados por el usuario.
Tipo de dato esperado:	is_string

Función de evaluación:
`$resultEsperado = "Ocurrió un error durante la operación. Intente más tarde."`
`$nombrePrueba = "Prueba: 'Modificar reporte'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Resultado final de la prueba: SATISFACTORIA EN SU TOTALIDAD	

Tabla 34. CENIA_PRE_PUCPCB-DR: RFR3. RFR7, RFR11, RFR15.

Prueba estructural o caja blanca	**Código caso de prueba:** CENIA_PRE_PUCPCB-DR: RFR3, RFR7, RFR11, RFR15
Probador: Rodain Terrer Molina	
Código al que se aplica:	**Representación en grafo de flujo:** 

```
public function datosReporte() {
    $post_var = $this->input->all_post(true);                                          1
    $datos_reporte = $this->reporte_lib->ObtenerReporteDadoIdContenido($post_var['id_contenido']);   1
    $nombre_reporte = $datos_reporte->titulo_reporte;                                  1
    $usuario_reporte = $datos_reporte->realizado_por;
    $usuario = $this->reporte_lib->obtenerPersonaDadoUsuario($usuario_reporte);
    if (!empty($usuario)) {   1
    $nombre_propietario = $usuario->primer_nombre.' '. $usuario->segundo_nombre.' '.$usuario->  primer_apellido .' '.   2
$usuario->segundo_apellido;
    } else {
        $nombre_propietario = "Datos personales no definidos.";                        3
    }
    $filtros = $datos_reporte->filtro_html;                                            4
    $filtros = strip_tags($filtros, '<b> </b> <br/> <br>');
    if (empty($filtros))   4
        $filtros = "Este reporte no posee filtros para mostrar.";   5
    $this->template->set_data('nombre_reporte', $nombre_reporte);                      6
    $this->template->set_data('nombre_propietario', $nombre_propietario);              6
    $this->template->set_data('usuario_reporte', $usuario_reporte);                    6
    $this->template->set_data('filtros', $filtros);                                    6
    $this->template->set_data('usuario', $usuario);                                    6
    echo $this->template->render('datos_reporte_view');                                6
    }
```

Complejidad ciclomática:	**Caminos independientes:**
V (G) = (A - N) + 2 = (7 - 6) + 2 = 3	**1.** 1 - 2 - 4 - 6 **2.** 1 - 3 - 4 - 6 **3.** 1 - 2 - 4 - 5 - 6

Caso de prueba para el camino básico No. 1	
Descripción: el dato de entrada será el identificador del reporte. Con este identificador se obtiene quién es el propietario del mismo y se tratan de obtener los datos de este y los filtros del reporte.	
Condición de ejecución: existen datos registrados para ese usuario y no posee filtros para mostrar.	
Procedimiento prueba automatizada	
Datos de entrada:	identificador del reporte.
Tipo de dato esperado:	is_bool

Función de evaluación:
`$resultEsperado = TRUE`
`$nombrePrueba = "Prueba: 'Ver detalles reporte'"`
`echo $this->unit->run($this->datosReporte($id_reporte),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	

Caso de prueba para el camino básico No. 2	
Descripción: el dato de entrada será el identificador del reporte. Con este identificador se obtiene quién es el propietario del mismo y se tratan de obtener los datos de este y los filtros del reporte.	
Condición de ejecución: no existen datos registrados para ese usuario.	
Procedimiento prueba automatizada	
Datos de entrada:	identificador del reporte.
Tipo de dato esperado:	is_bool
Función de evaluación: `$resultEsperado = TRUE`	

`$nombrePrueba = ""Prueba: 'Modificar reporte'"` **`echo $this->unit->run($this->datosReporte($id_reporte),$resultEsperado,$nombrePrueba);`**
Evaluación del caso de prueba: Satisfactoria
Caso de prueba para el camino básico No. 3
Descripción: el dato de entrada será el identificador del reporte. Con este identificador se obtiene quién es el propietario del mismo y se tratan de obtener los datos de este y los filtros del reporte.
Condición de ejecución: el reporte no posee filtros.
Procedimiento prueba automatizada

Datos de entrada:	identificador del reporte.
Tipo de dato esperado:	is_bool

Función de evaluación:
`$resultEsperado = TRUE`
`$nombrePrueba = "Prueba: 'Ver detalles reporte'"`
`echo $this->unit->run($this->datosReporte($id_reporte),$resultEsperado,$nombrePrueba);`

Evaluación del caso de prueba: Satisfactoria
Resultado final de la prueba: SATISFACTORIA EN SU TOTALIDAD

Tabla 35. CENIA_PRE_PUCPCB-RC: RFR17, RFR18.

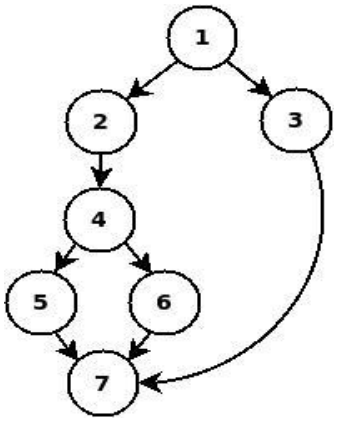
Prueba estructural o caja blanca	**Código caso de prueba:** CENIA_PRE_PUCPCB-RC: RFR17, RFR18
Probador: Rodain Terrer Molina	

Código al que se aplica:

```
public function registrarCarpeta() {                                    1
       if ($this->input->is_post_back(array('carpeta'))) {              2
           $all_post = $this->input->all_post(true);
           $response = array();
           $result = $this->reporte_lib->creando_carpeta($all_post, $response); 2

           if ($result) {                                               4
               $response['carpetando'] = true;
               $response['success'] = true;                             5
               echo json_encode($response);                             5
           } else {
               throw new Exception('SYS006');                           6
           }
       }
       else
           throw new Exception('SYS007');                               3
   }  7
```

Representación en grafo de flujo:



Complejidad ciclomática:

V (G) = (A - N) + 2 = (8 - 7) + 2 = 3

Caminos independientes:

1. 1 - 3 -7
2. 1 - 2 - 4 - 5 -7
3. 1 - 2 - 4 - 6 -7

Caso de prueba para el camino básico No. 1
Descripción: los datos de entrada serán atributos del directorio o carpeta a crear.
Condición de ejecución: los datos de los atributos no son válidos.
Procedimiento prueba automatizada

Datos de entrada:	Arreglo vacío.

Tipo de dato esperado:	is_string

Función de evaluación:
`$resultEsperado = "Uno o varios elementos se ha introducido de forma incorrecta."`
`$nombrePrueba = "Prueba: 'Registrar o Modificar carpeta'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No. 2	
Descripción: los datos de entrada serán atributos del directorio o carpeta a crear.	
Condición de ejecución: los datos de los atributos son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	Arreglo con los datos del directorio insertados por el usuario.
Tipo de dato esperado:	is_bool

Función de evaluación:
`$resultEsperado = TRUE`
`$nombrePrueba = "Prueba: 'Registrar o Modificar carpeta'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No. 3	
Descripción: los datos de entrada serán atributos del directorio o carpeta a crear.	
Condición de ejecución: ocurre un error de base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	Arreglo con los datos del reporte insertados por el usuario.
Tipo de dato esperado:	is_string

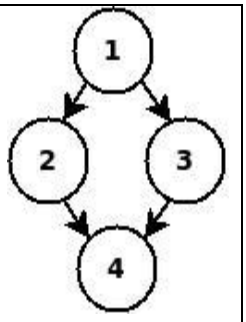
Función de evaluación:
`$resultEsperado = "Ocurrió un error durante la operación. Intente más tarde."`
`$nombrePrueba = "Prueba: 'Modificar reporte'"`
`echo $this->unit->run($this->registrarReporte($datos),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	
Resultado final de la prueba: SATISFACTORIA EN SU TOTALIDAD	

Tabla 36. CENIA_PRE_PUCPCB-DC: RFR19.

Prueba estructural o caja blanca	**Código caso de prueba:** CENIA_PRE_PUCPCB-DC: RFR19
Probador: Rodain Terrer Molina	
Código al que se aplica:	**Representación en grafo de flujo:**

```
public function datosDirectorio() {
    $post_var = $this->input->all_post(true);                                                    1
    $datos_directorio = $this->reporte_lib->ObtenerDirectorioDadoIdCarpeta($post_var['id_contenido']);
    $nombre_directorio = $datos_directorio->nombre_carpeta;                                      1
    $titulo = 'directorio ' . $nombre_directorio;
    $datos_usuario = $this->reporte_lib->obtenerPersonaDadoUsuario($datos_directorio->propietario);  1
    if (!empty($datos_usuario)) {    1
        $nombre_propietario = $datos_usuario->primer_nombre . ' ' . $datos_usuario->segundo_nombre . ' '  2
                . $datos_usuario->primer_apellido . ' ' . $datos_usuario->segundo_apellido;
        $usuario_propietario = $datos_usuario->usuario;
    } else {
        $nombre_propietario = "Datos no definido de esta persona.";                              3
        $usuario_propietario = "Datos no definido de esta persona.";
    }
    $this->template->set_data('nombre_directorio', $nombre_directorio);                          4
    $this->template->set_data('titulo', $titulo);                                                4
    $this->template->set_data('nombre_propietario', $nombre_propietario);                        4
    $this->template->set_data('usuario_propietario', $usuario_propietario);                      4
    $this->template->set_data('usuario', $datos_usuario);
    echo $this->template->render('datos_directorio_view');                                       4
}
```



Complejidad ciclomática:

V (G) = (A - N) + 2 = (4 - 4) + 2 = 2

Caminos independientes:

1. 1 - 2 - 4
2. 1 - 3 - 4

Caso de prueba para el camino básico No. 1	
Descripción: el dato de entrada será el identificador del directorio o carpeta. Con este identificador se obtiene quién es el propietario del mismo y se tratan de obtener los datos de este.	
Condición de ejecución: existen datos registrados para ese usuario.	
Procedimiento prueba automatizada	
Datos de entrada:	identificador de la carpeta.
Tipo de dato esperado:	is_bool

Función de evaluación:
`$resultEsperado = TRUE`
`$nombrePrueba = "Prueba: 'Ver detalles carpeta'"`
`echo $this->unit->run($this->datosReporte($id_carpeta),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria	

Caso de prueba para el camino básico No. 2	
Descripción: el dato de entrada será el identificador del directorio o carpeta. Con este identificador se obtiene quién es el propietario del mismo y se tratan de obtener los datos de este.	
Condición de ejecución: no existen datos registrados para ese usuario.	
Procedimiento prueba automatizada	
Datos de entrada:	identificador de la carpeta.
Tipo de dato esperado:	is_bool

Función de evaluación:
`$resultEsperado = TRUE`
`$nombrePrueba = ""Prueba: 'Modificar reporte'"`
`echo $this->unit->run($this->datosReporte($id_reporte),$resultEsperado,$nombrePrueba);`	

Evaluación del caso de prueba: Satisfactoria
Resultado final de la prueba: SATISFACTORIA EN SU TOTALIDAD

13.2 Casos de pruebas de caja negra

- **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema.
- **Nombre caso de prueba:** Crear reporte de tipo listado.

Escenario	Descripción	Variable 1 (Título)	Variable 2 (Selección de las columnas)	Variable 3 (Campos disponibles)	Variable 4 (Ordenar por)	Variable 5 (Ordenar descendente)	Variable 6 (Campo)	Variable 7 (Signo)	Variable 8 (Valor)	Variable 9 (Signos de agrupación)	Variable 10 (Conectores de filtro)	Respuesta del sistema	Flujo central
Insertar datos correctamente.	Este escenario permite crear correctamente un reporte de tipo listado utilizando datos correctos.	V “baja”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	No procede	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema una vez creado el reporte de tipo listado actualiza el árbol del sistema y muestra el mensaje de información:” El elemento ha sido creado satisfactoriamente”.	- Al autenticarse el usuario, el sistema por defecto muestra el árbol del sistema, el usuario selecciona el módulo “Reportes”, selecciona la funcionalidad “Listados” de la agrupación funcional Reportes Nominales. - El sistema muestra un árbol con los reportes de tipo listado existentes. - Se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte de tipo listado y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. - Se introducen los datos correctamente para un reporte de tipo listado.
		V “baja1”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”		
		V “Número de caracteres entre 1 y 50”	V “Seleccionar una o varias”	No procede	No procede	No procede	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”		

													- Se selecciona el botón Aceptar.
Insertar elemento repetido	Este escenario permite crear un reporte de tipo listado que ya exista en el sistema.	V “baja”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	No procede	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema crea el nuevo reporte y como nombre le pone el definido por el usuario y adicionalmente un número que representa la cantidad de reportes con el mismo nombre.	- Al autenticarse el usuario, el sistema por defecto muestra el árbol del sistema, el usuario selecciona el módulo “Reportes”, selecciona la funcionalidad “Listados” de la agrupación funcional Reportes Nominales. - El sistema muestra un árbol con los reportes de tipo listado existentes. - Se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte de tipo listado y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. - Se introducen los datos repetidos para un reporte de tipo listado. - Se selecciona el botón Aceptar.
		V “baja1”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”		
Insertar datos incompletos	Mediante este escenario se permite no introducir datos que son obligatorios.	V “baja”	I “Vacío”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema muestra el mensaje en rojo en la parte superior del componente que debe	- Al autenticarse el usuario, el sistema por defecto muestra el árbol del sistema, el usuario

												ser llenado obligatoriamente” Campo requerido”	selecciona el módulo “Reportes”, selecciona la funcionalidad “Listados” de la agrupación funcional Reportes Nominales. - El sistema muestra un árbol con los reportes de tipo listado existentes. - Se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte de tipo listado y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. - Se introducen los datos dejando campos obligatorios vacíos para un reporte de tipo listado. - Se selecciona el botón Aceptar.
Insertar datos incorrectos	Mediante este escenario se permite introducir datos incorrectos.	I “%baja”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	No procede	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema muestra el mensaje en rojo en la parte superior del componente ” Entre solo letras, números, espacios y el carácter especial _ ”	- Al autenticarse el usuario, el sistema por defecto muestra el árbol del sistema, el usuario selecciona el módulo “Reportes”, selecciona la funcionalidad “Listados” de la agrupación funcional

		I “Entrar más de 200 caracteres”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema muestra el mensaje en rojo en la parte superior del componente” No más de 200 caracteres” y no permite continuar escribiendo.	Reportes Nominales. - El sistema muestra un árbol con los reportes de tipo listado existentes. - Se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte de tipo listado y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. - Se introducen los datos incorrectos para un reporte de tipo listado. - Se selecciona el botón Aceptar
		I “Entrar una palabra con más de 30 caracteres”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema muestra el mensaje en rojo en la parte superior del componente” Ha excedido el número máximo permitido de caracteres por palabra” y no permite continuar escribiendo.	
		I “Copiar mas de 50 caracteres”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema muestra el mensaje en rojo en la parte superior del componente” No más de 50” y no permite continuar escribiendo.	
		I “Entrar menos de 2 caracteres”	V “Seleccionar una o varias”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno”	V “Seleccionar uno o escribir un valor”	V “Seleccionar uno”	V “Seleccionar uno”	El sistema muestra el mensaje en rojo en la parte superior del componente “Entre al menos 2 caracteres”.	
Cancelar operación	En este escenario se cancela la operación realizada.	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	El sistema muestra un mensaje de advertencia: “¿Está seguro de realizar la acción?”.	- Al autenticarse el usuario, el sistema por defecto muestra el árbol del sistema, el usuario selecciona el módulo “Reportes”, selecciona la funcionalidad “Listados” de la agrupación funcional

													Reportes Nominales. - El sistema muestra un árbol con los reportes de tipo listado existentes. - Se selecciona en el árbol del sistema la carpeta dentro de la cual se desea crear el reporte de tipo listado y se selecciona la opción Crear en la barra de íconos flotantes o clic derecho sobre la carpeta y selecciona la opción Nuevo reporte. - Se introducen los datos correctos para un reporte de tipo listado. - Se selecciona el botón Cancelar.

- **Descripción de las variables.**

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Título	Campo de texto	Si	Nombre del elemento que permita breve descripción. Admite hasta 50 caracteres alfanuméricos y el carácter especial _.
2	Selección de las columnas	Campo de selección	No	Contiene las columnas o datos que puede contener el reporte de tipo listado.
3	Campos disponibles	Campo de selección	Si	Contiene todos los campos seleccionados por el usuario del reporte de tipo listado.
4	Ordenar por	Campo de selección	Si	Contiene los campos seleccionados por el usuario para ser ordenados de forma ascendente del reporte de tipo listado.
5	Ordenar descendente	Campo de selección	Si	Contiene los campos seleccionados por el usuario para ser ordenados de forma descendente del reporte de tipo listado.

6	Campo	Campo de selección	Si	Contiene los campos de datos para el filtrado del reporte de tipo listado.
7	Signo	Campo de selección	Si	Contiene los signos de asociación para el filtrado del reporte de tipo listado.
8	Valor	Campo de selección	Si	Contiene los valores de los campos de datos para el filtrado del reporte de tipo cruzado cuando el Campo es un nomenclador, de lo contrario es un campo de texto.
9	Signos de agrupación	Campo de selección	Si	Contiene los signos de agrupación para el filtrado del reporte de tipo listado.
10	Conectores de filtro	Campo de selección	Si	Contiene los conectores para el filtrado del reporte de tipo listado.