Universidad de las Ciencias Informáticas

Facultad 1

Diseñador de plantillas para el módulo Reportes del Sistema de Gestión Académica de Pregrado

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

Autores: Nayilet Martín Soler

Yander Santiesteban Rojas

Tutores: Ing. Norges Sánchez Tumbarell

Ing. Yasmany Tellez Collazo

Ing. Odaymis M. Hechavarría Vargas

La Habana, Cuba

Junio, 2013

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: Diseñador de plantillas para el módulo Reportes del Sistema de Gestión Académica de Pregrado y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente tesis a los 12 días del mes de Junio del año 2013.

Nayilet Martín Soler

_____

Firma de la autora

Yander Santiesteban Rojas

_____

Firma del autor

Ing. Norges Sánchez Tumbarell

_____

Firma del tutor

Ing. Yasmany Tellez Collazo

_____

Firma del tutor

Ing. Odaymis M. Hechavarría Vargas

_____

Firma de la tutora

AGRADECIMIENTOS

Nayilet Martín Soler

A Dios en primer lugar por ser mi guía y mi sustento durante toda mi vida y especialmente durante mi carrera.

A mi mamis Maida que no se que sería sin sus consejos.

A mi titi Norma mi otra madre de corazón.

A mi papi lindo y a tío Manolito mi otro padre.

A mi hermano querido.

A toda la familia linda que Dios me dio por confiar siempre en mí y apoyarme incondicionalmente.

A mi compañero de tesis Yander por su apoyo y comprensión.

A mis tutores Norges, Yasmany y Mercy, no pude tener mejores tutores.

A todos mis amigos y amigas, pero especialmente a mi negri Lonna, mi compañera de toda la carrera.

Y por último, pero no menos especial a mi esposo, porque sin él mi vida no tendría sentido.

Yander Santiesteban Rojas

Siempre resultará difícil agradecer a todos aquellos que de una u otra manera me han acompañado durante el desarrollo de esta investigación, porque nunca alcanza el tiempo, el papel o la memoria para mencionar o dar con justicia todos los créditos y méritos a quienes se lo merecen. Por tanto, quiero agradecerles a todos ellos cuanto han hecho por mí, para que este trabajo saliera adelante de la mejor manera posible.

Partiendo de esta necesidad y diciendo de antemano MUCHAS GRACIAS, primeramente deseo agradecer especialmente a Dios por ser fuente de motivación en los momentos de angustia y después de varios esfuerzos, dedicación, aciertos y reveses que caracterizaron el desarrollo de mi formación profesional y que con su luz divina me guió para no desmayar por este camino que hoy veo realizado. A mi madre por ser mi guía, mi inspiración, por su apoyo y su amor incondicional. A mi padre por estar presente siempre que necesité de su ayuda. A mi abuelo Eugenio por haberme inculcado buenos sentimientos. A mi hermano Rony por su amor. A mi madrastra Marta por su preocupación y apoyo. A Alberto y Clarita por estar siempre pendientes de mí. A mi hermano Roger, aunque no esté presente en vida, por ser mi ejemplo a seguir y la luz que ilumina mis caminos.

A María Teresa, Mayelin y Rolquis por haberme brindado su ayuda en los momentos más difíciles de mi vida.

A mis tutores Norges, Yasmany y Mercedes, por haberme guiado por el buen camino para la culminación satisfactoria de la presente investigación. A mi compañera de tesis por su paciencia, comprensión y por haber depositado toda su confianza en mí. Y a todas las personas que compartieron conmigo durante el transcurso de estos 5 años: aulas, apartamento, proyecto y otras actividades, sinceramente Gracias a todos.

DEDICATORIA

Nayilet Martín Soler

Dedico esta tesis especialmente a mis abuelitos lindos: Martí, Eloisa y Adis por su apoyo y cariño.

A mis 4 padres: Maida, Rodolfo, Norma y Manolito, porque sin ustedes no hubiera llegado hasta aquí.

A mi querido hermanito Rodol por estar siempre presente.

Yander Santiesteban Rojas

Dedico esta investigación a mi madre, y a mi hermano Roger por ser esa luz que ilumina mis caminos.

RESUMEN

El Sistema de Gestión Académica de Pregrado (SGAP) es el encargado de gestionar los procesos de formación de pregrado en la universidad y forma parte del Sistema de Gestión Universitaria (SGU) desarrollado en la Universidad de las Ciencias Informáticas (UCI). El sistema cuenta con un módulo que permite la gestión de los reportes. Estos se obtienen basados en plantillas definidas previamente por el desarrollador de la aplicación, lo que impide al usuario personalizar la organización de los datos que desea mostrar y que el reporte sea visualizado con la apariencia deseada. En la presente investigación se propone el desarrollo de un diseñador de plantillas para gestionar reportes dinámicamente integrado al SGAP, que permita crear plantillas con una salida personalizable de la información y con una interfaz de fácil uso para usuarios con conocimientos básicos en informática. Para lograr tal propósito se realiza un estudio de algunas soluciones informáticas existentes en cuanto al diseño de plantillas para reportes con el objetivo de identificar elementos significativos que pudieran guiar su construcción. En el desarrollo se utiliza PHP 5.3.22 como lenguaje de programación, GUUD 1.0 como marco de trabajo, PostgreSQL 8.4.1 como Sistema Gestor de Bases de Datos, Apache 2.2.2 como servidor web, NetBeans 7.2 como Entorno de Desarrollo Integrado y Visual Paradigm 8.0 como herramienta de modelado. Todas estas se integran en un proceso de desarrollo de software con enfoque ágil que utiliza algunas prácticas de las metodologías ágiles XP y Scrum, basado en el nivel 2 de CMMI.

Palabras clave: diseñador de plantillas, gestión de reportes, salida personalizable.

ÍNDICE

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La revolución tecnológica centrada en la información transformó el modo de pensar, producir, comunicar, gestionar, comerciar y hasta de vivir del hombre. El papel de la información en las organizaciones se transforma debido a los nuevos paradigmas asociados a su gestión y está presente de manera directa en todos los procesos que estas desarrollan. Es por ello, que los responsables en la toma de decisiones han comprendido que la información no es solo un simple activo de la empresa, sino que a su vez da vida a sus principales procesos e influye como factor crítico en la determinación del éxito o fracaso de las mismas. (1)

En la actualidad son utilizados los Sistemas de Información (SI) para la gestión y distribución de la información. Estos cambian la manera en que las organizaciones actuales operan sus procesos, conformados por un conjunto de componentes interrelacionados para recibir (entrada), manipular (procesamiento) y recuperar (salida) datos. Los SI permiten disponer de un mecanismo de retroalimentación útil en pos del cumplimiento de un objetivo y desempeñan un papel fundamental y cada vez más amplio en todas las organizaciones. Con su uso se alcanzan importantes mejoras como la automatización de los procesos operativos, suministro de una plataforma de información necesaria para la toma de decisiones y su implantación logra ventajas competitivas. (2)

Todo SI cuenta con un conjunto de herramientas útiles para la recuperación de la información, entre las cuales es fundamental contar con una que gestione reportes. Un generador de reportes es una herramienta para la gestión de la información de cualquier empresa o institución, mediante el cual se muestra el cumplimiento de las metas establecidas y se contribuye a la toma de decisiones. Proveen una forma transparente al usuario para realizar consultas a la base de datos y obtener información de ella. (3)

Un reporte es aquel documento que se utilizará cuando se quiera mostrar una determinada información. También puede incluir algunos elementos persuasivos, como recomendaciones o sugerencias y además algunas conclusiones a través de las cuales se le indique al lector del mismo alguna acción o conducta a adoptar en el futuro. (4)

La mayoría de las organizaciones los utilizan para registrar y visualizar análisis y resultados. Los reportes son considerados una necesidad principal del negocio, por ello se desarrollan diferentes aplicaciones que viabilizan el proceso de generación de los mismos.

En Cuba existen diferentes instituciones que se dedican al desarrollo de sistemas informáticos. Entre estas se destaca la Universidad de las Ciencias Informáticas (UCI) donde cotidianamente se hace necesario obtener de forma rápida y de acuerdo a determinados criterios de búsqueda, datos que son gestionados por diferentes sistemas. Estos datos pueden ser referentes a diversos aspectos como: conteo

de personas e información nominal de las mismas, resultados académicos-docentes por tantos conceptos como se desee, matrícula, asistencia y evaluaciones diarias, parciales o finales.

En el Sistema de Gestión Académica de Pregrado, que pertenece al Sistema de Gestión Universitaria, se desarrolló el módulo "Generador de Reportes" que permite evitar los problemas mencionados a continuación: (5)

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

Independientemente que el módulo creado permite el manejo de la información de forma clara, sencilla y ordenada, así como la gestión de los reportes, actualmente estos se obtienen basados en plantillas definidas previamente por el desarrollador de la aplicación. Por tanto la salida de los reportes es visualizada en un formato predefinido, haciendo que esta sea estática. De esta forma se restringe la posibilidad de decidir la apariencia final del informe a los usuarios, impidiéndoles personalizar la organización de los datos finales que desea mostrar y que el reporte sea visualizado con la apariencia deseada.

Los usuarios se clasifican en dos grupos: los que tienen conocimientos básicos en informática, por tanto conocen que es posible diseñar una plantilla, tomar datos e insertarlos en esta y especificar la manera en que quieren estructurar dicha plantilla; y los que no poseen conocimientos básicos en informática, desconocen cómo diseñar una plantilla que pueda ser llenada con los datos que se recuperan del sistema y no saben la manera en que desean estructurar la plantilla para obtener el reporte.

Tanto para los dos grupos de usuarios como para los desarrolladores la pérdida de tiempo se convierte en un hecho engorroso, pues para el primer grupo se tiene que invertir tiempo en explicarle al desarrollador cómo se estructura la plantilla para establecer la información obtenida desde el sistema, en la plantilla construida. Y para el segundo grupo, se tienen que expresar los datos que se desean mostrar y esperar que el equipo de desarrollo diseñe la plantilla, la integre al sistema e implemente todas las funcionalidades necesarias para mostrar la información en la plantilla diseñada, corriendo el riesgo que el usuario no quede complacido con el diseño y se tenga que elaborar otro diseño. Además para los dos grupos el desarrollador tiene que invertir tiempo y dejar a un lado sus deberes para satisfacer las necesidades del usuario.

A raíz de la situación problemática planteada se define el siguiente **problema a resolver:** ¿cómo generar plantillas personalizadas para el módulo Reportes del Sistema de Gestión Académica de Pregrado en la

Universidad de las Ciencias Informáticas? En consecuencia se define como **objeto de estudio** el proceso de generación dinámica de reportes.

El **campo de acción** es el diseño de plantillas dentro del proceso de generación dinámica de reportes para el módulo Reportes del Sistema de Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas.

Para dar solución al problema enunciado se propone como **objetivo general** de esta investigación**:** desarrollar un diseñador de plantillas para el módulo Reportes del Sistema de Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas, que permita la personalización de la salida de la información en la gestión de los reportes.

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

1. Identificar los elementos teóricos necesarios para el desarrollo de la investigación.
2. Diseñar la propuesta de solución a partir del proceso de desarrollo de software utilizado.
3. Implementar el diseñador de plantillas a partir de los requisitos identificados.
4. Validar la solución desarrollada a partir de la ejecución de pruebas de software.

Para dar cumplimiento a los objetivos específicos de la investigación se plantean las siguientes **tareas de investigación:**

1. Diagnóstico del estado del arte de los diferentes diseñadores de plantillas existentes en el mundo y en la UCI.
2. Caracterización de las herramientas y el proceso de desarrollo de software a utilizar en la propuesta de solución.
3. Identificación de los requisitos funcionales y no funcionales de la propuesta de solución.
4. Análisis de las áreas configurables de la plantilla de salida para el diseño de los reportes.
5. Definición y análisis de los mecanismos de captura de los valores en la base de datos del SGAP para describir la relación con la fuente de información.
6. Implementación de las funcionalidades del diseñador de plantillas.
7. Caracterización de las pruebas a realizar a la solución.
8. Validación de la propuesta de solución.

La investigación se sustenta en la siguiente **idea a defender**: el diseño de plantillas personalizadas para el módulo Reportes contribuirá a la mejora de la gestión de los mismos en el Sistema de Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas.

Para la realización y culminación de la presente investigación, se hace necesaria la utilización de los siguientes métodos científicos.

Métodos teóricos:

-Histórico-lógico: permite constatar teóricamente cómo ha evolucionado un fenómeno en un período de tiempo dado. Se utiliza para estudiar las formas de solución a problemas similares sobre el diseño de plantillas en la gestión de reportes en el mundo y en la UCI para determinar la evolución de los reportes.

-Analítico-sintético: se utiliza 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 generar plantillas personalizadas para el módulo Reportes en el Sistema de Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas.

Métodos empíricos:

-Método de observación: con la utilización de este método es posible percibir directamente los hechos de la realidad objetiva y observar de forma abierta la población estudiada. También permitió investigar los procesos externamente sin tener que llegar a la esencia de los mismos, lo que ayuda al planteamiento del problema científico. Además posibilitó conocer bien el proceso delimitado como objeto de estudio, lo cual contribuyó a tener un conocimiento más detallado de lo que se quiere, lo que hace falta hacer y cómo hay que hacerlo.

-Revisión documental: se pone de manifiesto al revisar la bibliografía existente y consultar la información en internet para llevar a cabo las tareas de investigación.

Justificación de la investigación:

La propuesta de solución permite crear plantillas personalizadas para generar reportes, donde el usuario puede decidir la apariencia final de su informe. Esto contribuye a que los usuarios puedan diseñar plantillas diferentes a las que actualmente genera el SGAP. Además tributa a un mejor aprovechamiento del tiempo de trabajo del especialista, en función de la evolución del software y lograr una mayor accesibilidad a la información manejada referente al proceso de formación de los estudiantes.

Estructura del documento

El presente trabajo está estructurado en tres capítulos que abarcan todo el proceso para el desarrollo de la solución informática.

Capítulo 1: Fundamentación teórica: en este capítulo se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. Se describen los conceptos fundamentales asociados al dominio del problema. Además se realiza un estudio y análisis de varias herramientas para el diseño de plantillas en el mundo y en la UCI. También, se define el entorno tecnológico para el desarrollo de la propuesta de solución.

Capítulo 2: Descripción de la propuesta de solución: en este capítulo se realiza la descripción de la propuesta de solución, sus requisitos funcionales y no funcionales, así como los actores que intervienen

en ella. También se describe la arquitectura del sistema, haciendo énfasis en los patrones de diseño y de base de datos utilizados. Además, se presenta el modelo de datos y la distribución física o diagrama de despliegue de dicha propuesta.

Capítulo 3: Implementación y validación de la solución: en este capítulo se describe la implementación de la propuesta de solución teniendo en cuenta las técnicas de codificación empleadas. Se presentan las pruebas que servirán para validar la solución como las pruebas unitarias, de integración, del sistema (rendimiento y resistencia) y de aceptación para satisfacer todas las necesidades del cliente.

CAPITULO 1: Fundamentación teórica

El presente capítulo comprende el estudio de varios conceptos y características generales de los diseñadores de plantillas. Además se realiza un análisis de algunas herramientas existentes en el mundo y en la UCI para el diseño de plantillas, dando lugar a una visión general del estado del arte de las mismas. Se caracteriza el proceso de desarrollo de software, lenguajes de programación y tecnologías que se utilizan para el desarrollo de software basado en la web.

1.1. Reportes en el ámbito de la informática

Un reporte es un informe o una noticia. Este tipo de documento puede ser impreso, digital o audiovisual. Pretende transmitir una información, aunque puede tener diversos objetivos. (6)

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. Los reportes tienen diversos niveles de complejidad, desde una lista o enumeración hasta gráficos mucho más desarrollados. Según la herramienta informática y la base de datos en cuestión, los reportes permiten la creación de etiquetas y la elaboración de facturas, entre otras tareas. (6)

Los reportes facilitan a los usuarios observar la marcha de la organización y que de esta forma se logre identificar las dificultades y los logros alcanzados. El carácter determinante de los mismos estimula a un mayor esfuerzo para garantizar su eficiente obtención.

1.2. Generalidades sobre los generadores de reportes

Los generadores de reportes se han creado con el objetivo de complementar los sistemas de información. Utilizan una especie de lenguaje transparente para el usuario por medio del cual éste realiza consultas a la base de datos y obtiene información de ella en forma de reporte. Por lo cual, el acceso a la misma es más fácil y en períodos de tiempo relativamente cortos. (7)Los generadores de reportes están compuestos principalmente por dos elementos básicos, un diseñador o editor de informes y un motor de generación como se muestra en la Figura 1.

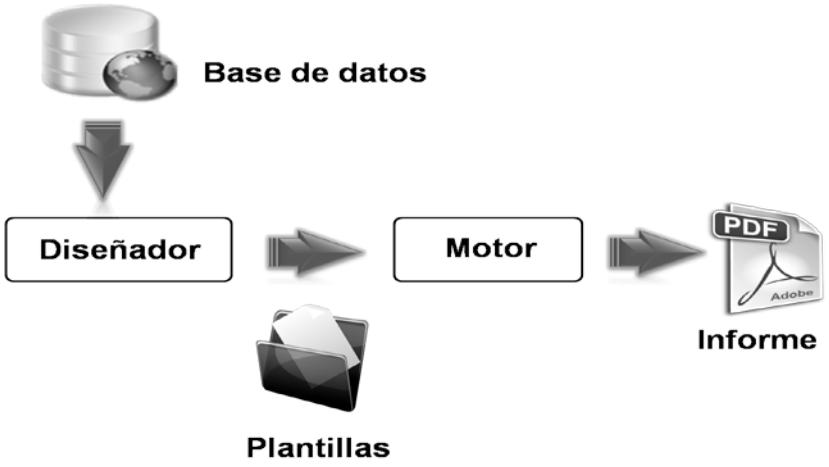


Figura 1: Estructura general de un generador de reportes.

El diseñador visual es un programa que ayuda a los usuarios y desarrolladores a diseñar reportes visualmente. A través de una interfaz simple de usar, se provee las funciones más importantes para crear plantillas en poco tiempo. Una vez concluida la etapa de diseño, comienza la etapa de generación del reporte, donde el motor de generación interpreta la plantilla, toma los datos que el diseñador le entrega y los mezclan con la plantilla diseñada para luego visualizarlos en el formato definido.

1.3. Etapas del generador de reportes

El proceso de generación de reportes consiste en configurar la estructura de los informes en forma de plantilla, la cual contendrá la estructura y apariencia antes de incluir los datos de las consultas y así el reporte se obtiene con la presentación deseada por el cliente.

- **Construcción de la plantilla**

En el diseño se pueden agregar componentes y demás características, en las distintas secciones que se irán almacenando en la plantilla.

Las plantillas generalmente contienen las siguientes secciones:

Título: esta sección se mostrará solo una vez al principio del informe.

Encabezado de página: esta sección es la cabecera de la página, se repite cada vez que se crea una página nueva.

Encabezado de columna: esta sección es la cabecera de las columnas.

Detalles: esta sección es la encargada de mostrar los elementos que tienen alguna repetición.

Pie de la columna: su comportamiento es análogo al Encabezado de columna.

Pie de página: se repite una vez por página. Su comportamiento es análogo al Encabezado de página.

Sumario: solo se repite una vez por informe en la última página del mismo. Su comportamiento es análogo al Título.

- **Recuperación de la información**
  - ✓ ***Interpretación de la plantilla:*** se determinan cuales son los componentes, cómo se leen dichos componentes, cual es la información asociada a cada uno y cómo deben organizarse.
  - ✓ ***Extracción de los datos:*** se realiza la lectura de la información y se confecciona la estructura organizacional de la misma (normalizarla) para el renderizado.
  - ✓ ***Renderización de la información:*** se toma la información normalizada junto a la plantilla a utilizar y se mezclan ambos y se visualiza el informe final.

El generador de reportes constituye una versátil e intuitiva herramienta para usuarios no técnicos que deseen explorar los datos que su empresa almacena por medio de reportes. A través de este se obtiene una salida personalizable de la información, pues el usuario puede escoger la manera en que quiere visualizar el contenido del reporte.

1.4. Diseñadores de plantillas

Para que las diferentes empresas puedan evaluar y mejorar los resultados de su trabajo, es necesario llevar un control detallado del estado de su información, que es de gran importancia para la toma de decisiones. Como una excelente opción están los diseñadores de plantillas, que permiten facilitar y hacer más sencillo la consulta de la información necesitada. Existe una amplia gama de diseñadores que ofrecen varias experiencias globales que hacen que sean una referencia relevante para la concepción de un diseñador de plantillas, aunque presentan características determinadas que no permiten que sean los más convenientes a utilizar. Algunos de estos diseñadores se presentan a continuación:

Crystal Reports

Crystal Reports es la herramienta de elaboración de informes estándar para *Visual Studio .NET* de la empresa alemana SAP[1]**.** Permite crear contenido interactivo con calidad de presentación en la plataforma .NET, lo que ha sido una ventaja fundamental para *Crystal Reports* durante años. (8)

Los desarrolladores pueden utilizar *Crystal Reports* para *Visual Studio* para hacer lo siguiente:

- Diseñar informes ilimitados para uso en aplicaciones de *Visual Studio* con el diseñador de *Crystal Reports* integrado.

[1] Sistemas, Aplicaciones y Productos en Procesamiento de Datos.

El diseñador de plantillas de *Crystal* permite diseñar y modificar los informes del entorno de programación integrado de *Visual Studio .NET*. Dicha herramienta puede programarse directamente desde *Visual Studio .NET* y además no es necesario distribuir el diseñador de plantillas con el informe. Utiliza una funcionalidad de arrastrar y colocar parecida a la que se utiliza en *Visual Studio .NET*. Se arrastra un objeto de informe hasta el diseñador, que puede ser un campo de base de datos o un objeto de texto y se utiliza la ventana propiedades o el menú contextual para dar formato al objeto.

Se utiliza con el sistema operativo *Microsoft Windows*, dispone de licencias por usuario designado para el diseño de plantillas, independientemente de la edición de *Crystal Reports* adquirida, se distribuye bajo los términos de la EULA[2], licencia por la cual el uso de un producto solo está permitido para un único usuario, por lo que es software privativo y de uso restringido mediante el pago de patente. (9)

Active Reports

Active Reports cuenta con las premiadas capacidades significativas que proveen la habilidad de diseñar, crear y desplegar aplicaciones de reportes de forma fácil y rápida.

Está diseñado teniendo en cuenta los desarrolladores y sus diversas necesidades. Soporta el uso de componentes .NET en tiempo de diseño.

- Escrito completamente en C# y provee una completa integración con *Visual Studio .NET*.
- Asistente para la conversión de reportes importados desde *Microsoft Access.*

Incluye filtros para exportar a los formatos más populares, tales como Adobe PDF[3], *Microsoft Excel*, RTF[4], HTML[5], texto plano e imágenes TIFF[6].

- Diseñador de plantillas que permite la inclusión en las aplicaciones con el fin de que los propios usuarios diseñen y modifiquen sus reportes. (10)

Para utilizarlo se requiere tener instalado cualquier versión de Windows a partir de Windows NT 4.0, y se distribuye bajo licencia privativa de la compañía *GrapeCity inc.*

iReport

iReport es una herramienta visual de diseño de informes para *JasperReports* (Librería de Java que facilita y agiliza la generación, la previsualización y la impresión de los reportes) desarrollada por la compañía JasperSoft (compañía especializada en software de Inteligencia de Negocios de código abierto) en Java

[2] *End User Licensing Agreement* (Acuerdo de licencia de usuario final)

[3] *Portable document format* (formato de documento portátil)

[4] *Rich Text Format* (formato de texto enriquecido)

[5] Lenguaje de marcado de hipertexto.

[6] *Tagged Image File Format* (Formato de fichero para imágenes)

para la creación de informes. Tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XSL[7], CSV[8] y XML[9], con una interfaz gráfica intuitiva y fácil de usar. Permite a los usuarios editar visualmente cualquier tipo de informe complejo con gráficas, imágenes y subinformes de manera sencilla y rápida, tanto para usuarios que no están familiarizados con esta tecnología y que desconocen la sintaxis del XML de *JasperReports*, como a usuarios expertos que ya conocían este lenguaje, ahorrándoles tiempo durante el desarrollo de informes muy elaborados. Es multiplataforma, y se distribuye bajo licencia GNU GPL, *General Public License* (Licencia Pública General de GNU). (11)

Generador dinámico de reportes v.1

Generador dinámico de reportes es una aplicación desarrollada sobre el marco de trabajo *Symfony* y escrita en PHP por el centro productivo de desarrollo Centro de tecnologías de gestión de datos (DATEC) de la UCI. Es multiplataforma. Soporta imágenes, gráficas, así como varios orígenes de datos. Proporciona a los usuarios agilizar la toma de decisiones, generar reportes en varios formatos y con gran variedad de opciones en su diseño, marcando una diferencia entre los reportes tradicionales y los reportes dinámicos, objetos de este producto.

El diseñador de plantillas, permite diseñar plantillas de reportes que luego pueden ser salvados hacia la base de datos del Sistema de Generación Dinámica de Reportes para poder ser abiertos y utilizados en el momento que se les necesite. Esta herramienta tiene un área de trabajo hacia la cual se puede arrastrar componentes de una paleta y configurar sus propiedades en un inspector de propiedades. A los reportes se les puede realizar una vista previa desde su propia concepción y de esta forma poder ir acomodando el diseño a las necesidades de los usuarios. (12)

Aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, el usuario aún debe poseer conocimientos básicos. Además, su entorno de trabajo está estructurado de forma tal que es difícil guiarse en la creación y generación de reportes. La aplicación no permite la generación de sub-reportes y debido al motor de generación de reportes con la que fue creada, no es posible agregarle nuevas funcionalidades al sistema.

[7] *Extensible Stylesheet Language* (lenguaje extensible de hojas de estilo)

[8] *Comma-separated values* (son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea)

[9] *Extensible markup language* (lenguaje de marcas extensible)

Análisis de los diseñadores de plantillas estudiados

La elección de la herramienta más conveniente depende de las necesidades de los clientes, así como de las funcionalidades que ofrezcan cada una de estas como base para soportar una implementación que responda a dichas necesidades.

Active Reports es un sistema que satisface algunas necesidades de los clientes, pero una de sus desventajas principales es que al ser un software privativo su adquisición no garantiza, ni contribuye a alcanzar la soberanía tecnológica para la universidad y el país. De igual manera sucede con la herramienta *Crystal Reports*, presenta elevados costes adquisitivos.

El *iReport* representa un alto costo funcional, por lo engorroso en la integración de la tecnología Java con el marco de trabajo del SGU y arquitectónico, por incumplir con las pautas de diseño establecidas para el SGU.

En su mayoría dichas herramientas no están totalmente orientadas al usuario final, pues están orientadas a especialistas en informática con conocimientos en base de datos.

Por su parte el diseñador del generador dinámico de reportes v1.0 tiene como principal inconveniente que está orientado a usuarios avanzados con conocimientos en el área de base de datos, como trabajar con vistas, tablas y estructurar consultas. Además para la creación de los reportes utiliza el motor de generación de reportes phpreport, que no permite agregarle nuevas funcionalidades al sistema y el SGAP no tiene un motor de generación de reportes. También la integración con el SGAP representaría un costo en tiempo igual o mayor que desarrollar un nuevo sistema, pues serían muchas las modificaciones para lograr integrarlo y no se podría reutilizar toda la implementación, ya que el diseñador se basa principalmente en manejo de interfaz y la interfaz del diseñador del GDR esta hecha en con ext-js y la del SGAP con jquery-ui.

Teniendo en cuenta los inconvenientes presentados por las herramientas antes mencionadas, surge la presente investigación para desarrollar una aplicación multiplataforma y que posea gran flexibilidad, permitiendo al usuario final el diseño de plantillas para generar reportes personalizados desde la web. En el presente trabajo se utilizan las características y elementos primordiales de gran fortaleza que brindan los sistemas de su tipo más utilizados en el mundo y en la UCI como la guía para estructurar el reporte y los tipos de etiquetas con sus propiedades, y combinarlas con las ventajas de la web para la realización del mismo. Todo esto permite que al tratarse de una tecnología propia se faciliten los siguientes aspectos: asesoramiento continuo, personalización e integración de la misma con el Sistema de Gestión Académica de Pregrado.

1.5. Entorno tecnológico

El diseñador de plantillas será desarrollado utilizando las tecnologías establecidas por el Grupo de Soporte, Tecnologías e Implantación del Centro de Informatización Universitaria.

1.5.1. **Marco de trabajo**: GUUD 1.0

En la investigación se propone utilizar el marco de trabajo GUUD (acrónimo creado con las iniciales de los departamentos del centro CENIA: Gestión Universitaria, Universidad Digital y Gestión Documental) que es la unión del marco de trabajo *CodeIgniter* con *jQuery*. Este marco es el propuesto por el grupo de arquitectura del centro CENIA.

GUUD (versión 1.0)

El marco de trabajo GUUD incorpora novedades y modificaciones en su infraestructura, a continuación se listan las mismas.

Del lado del cliente:

1. Se implementaron una serie de *widgets* para utilizarlos de interfaz de algunos de los *widgets* base de *jquery-ui.*

2. Se le implementó un *plugin* a j*Query* para el manejo de espacios de nombres e internacionalización.

3. Se implementaron funciones comunes para todo el sistema (contenidas en los archivos *core.js* y *common.js*).

Del lado del servidor (hechas a *CodeIgniter*):

1. Se le agregó el manejo de excepciones y mensajes.

2. Se le implementó el IoC[10] para la interacción entre módulos.

3. Se le añadió la característica de la modularidad o sea que una aplicación pueda dividirse en módulos. *CodeIgniter* no cuenta con esta posibilidad.

4. Se añadieron, modificaron y extendieron los *helpers* o asistentes.

La Figura 2 muestra el flujo de información en el marco de trabajo GUUD. Cada petición es recibida por el controlador frontal (*index.php*) que inicializa todos los recursos que necesita el marco de trabajo para procesar la petición, el enrutador examina la solicitud HTTP[11] para determinar qué debería hacer con esta. Si existe este archivo de caché, se lo envía directamente al navegador, sin pasar por la ejecución normal del sistema. Antes de cargar el controlador, por razones de seguridad, se filtra la solicitud HTTP y cualquier otro dato enviado por el usuario. El controlador carga las librerías (que funcionan como

[10] *Inversion of Control* (Inversión de Control)

[11] Protocolo de transferencia de hipertexto.

intermediarias entre las capas de negocio y las de acceso a datos), las bibliotecas del núcleo, *helpers* y cualquier otro recurso requerido para procesar una solicitud, además de procesar las vistas (estas a su vez obtienen información de los *scripts*). Si la caché está habilitada la vista se cachea para que futuras peticiones que la necesiten puedan ser servidas. (13)

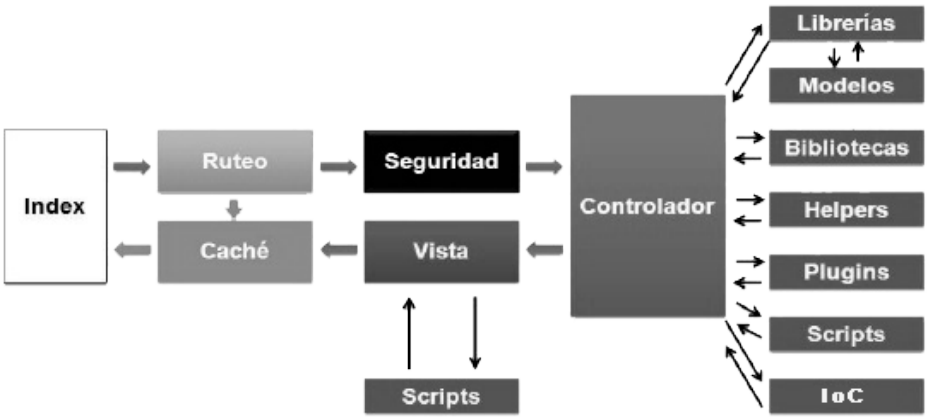


Figura 2: Flujo de información de GUUD.

CodeIgniter (versión 1.7.3)

CodeIgniter es un marco de trabajo para PHP. Es adecuado para desarrollos que requieran mucho rendimiento, que ejecutan muchas versiones de PHP con diferentes configuraciones. Una de sus mayores ventajas es la documentación que se ofrece en internet. Su principal objetivo es ayudar a que los desarrolladores puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero. (14)

CodeIgniter permite concentrase 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, el cual separa en 3 capas distintas: la representación de datos, el interfaz de usuario y el controlador de eventos respectivamente. (14)

jQuery (versión 1.3.2)

jQuery es un nuevo tipo de biblioteca o marco de trabajo de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología *AJAX*[12], al sistema. *jQuery* al igual que otras librerías, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código. Es

[12] *Asynchronous JavaScript and XML* (JavaScript Asíncrono y XML)

decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio. (15)

Con *jQuery* se ahorran muchas líneas de código, se mejora el tiempo de creación y depuración, esta tecnología tiene licencia para uso en cualquier tipo de plataforma, personal o comercial.

1.5.2. Sistema Gestor de Bases de Datos: PostgreSQL 8.4.1

Es un Sistema Gestor de Bases de Datos Objeto-Relacional (de sus siglas en inglés ORDBMS) basado en el proyecto Postgres, de la Universidad de *Berkeley*. Es una derivación libre de este proyecto. 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. 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. (16)

1.5.3. Servidor web: Apache 2.2.2

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. La licencia Apache permite hacer lo que se quiera con el código fuente siempre que les reconozcas su trabajo.

- Corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código abierto.
- Es un servidor altamente configurable de diseño modular, al que es muy fácil ampliar sus capacidades.
- Trabaja con gran cantidad de lenguajes como Perl y PHP.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- Tiene una alta configurabilidad en la creación y gestión de registros. Permite la creación de ficheros de registro a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en el servidor. (17)

1.5.4. Entorno de Desarrollo Integrado: NetBeans 7.2

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento. *NetBeans* IDE[13] es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro

[13] Entorno de desarrollo integrado.

lenguaje de programación. Existe además un número importante de módulos para extender el *NetBeans* IDE. Este es un producto libre y gratuito sin restricciones de uso. (18)

1.5.5. Administrador de base de datos: PgAdmin III 1.10.12

PgAdmin III es una aplicación gráfica para administrar el gestor de bases de datos PostgreSQL. Es capaz de gestionar versiones a partir de la PostgreSQL 1.10.0 ejecutándose en cualquier plataforma. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL[14] simples hasta desarrollar bases de datos complejas. (19)

La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la 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[15] para mayor seguridad. Está soportado por la licencia BSD[16]. (19)

1.5.6. Herramienta CASE: Visual Paradigm 8.0

Herramienta CASE[17] que acelera el desarrollo de aplicaciones, sirviendo de intermediario visual entre arquitectos, analistas y diseñadores de software, mediante un ambiente de modelado superior que posibilita un trabajo más fácil y dinámico. (20)

Es multiplataforma y cuenta con una versión libre para la comunidad (*Comunity Edition*). Genera la documentación del sistema en los formatos PDF, HTML y el formato de documentos de Microsoft Word y permite importar proyectos de otras herramientas de modelado como *Rational Rose, Erwin y Microsoft Visio*. Soporta la revisión ortográfica, brindando sugerencias para los idiomas: inglés, español, francés, alemán y portugués. (20)

Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (20)

Teniendo en cuenta sus características y los beneficios que brinda para la construcción de software, especialmente referente al modelado, se decidió utilizar *Visual Paradigm for UML* para el modelado de la propuesta de solución.

[14] Lenguaje de consulta estructurado.

[15] *Secure Sockets Lay*er (Protocolo de Capa de Conexión Segura)

[16] *Berkeley Software Distribution* (Licencia de software libre permisiva)

[17] *Computer Aided Software Engineering* (Ingeniería de Software Asistida por Computación)

1.5.7. Herramienta para crear prototipos de interfaz: Evolus Pencil 1.3.4

Es una herramienta libre y de código abierto para crear diagramas y prototipos de interfaz gráfica de usuario. Evolus Pencil permite diseñar las ventanas de los prototipos para *Windows* o *Linux*. Puede agregarse como complemento para el navegador *Mozilla Firefox* y tiene un tipo de funcionamiento de arrastrar y soltar. (21)Se decide utilizar la herramienta para el desarrollo de los prototipos de interfaz de la propuesta de solución.

1.5.8. Plataforma de desarrollo: GNU/Linux, Ubuntu 12.10

Algunas características que diferenciaron a Linux de los demás sistemas operativos de su tiempo y que siguen siendo aplicables, y otras heredadas de UNIX[18] son:

-Sistema operativo de código abierto. Cualquiera puede disponer de sus fuentes, modificarlas y crear nuevas versiones que puede compartir bajo la licencia GPL.

-Portabilidad. Tal como el UNIX original, Linux está pensado para depender muy poco de una arquitectura concreta de máquina; consecuentemente, Linux es en su mayor parte, independiente de la máquina de destino y puede portarse a prácticamente cualquier arquitectura que disponga de un compilador C como el GNU GCC[19].

-módulos dinámicamente cargables. Permiten poner partes del sistema operativo, como sistema de archivos, o controladores de dispositivos, como pedazos externos que se cargan (o enlazan) con el núcleo en tiempo de ejecución bajo demanda. (22)

1.5.9. Herramienta para realizar pruebas del sistema: Apache JMeter 2.3.1

Apache JMeter, es una herramienta Open Source realizada en java que se utiliza para realizar pruebas de rendimiento y resistencia, normalmente contra aplicaciones web. JMeter permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su “fuerza” y para analizar el rendimiento ante diferentes tipos de sobrecarga. (23)

Las características principales de Apache JMeter son las siguientes:

- Permite cargar y realizar pruebas sobre distintos tipos de servidores.
- Multiplataforma: Unix (Solaris, Linux), Windows (98, NT, XP), OpenVMS Alpha 7.3.
- Testeo distribuido.
- Multihilo: permite realizar pruebas de forma concurrente.
- Interfaz gráfica: permite realizar las operaciones más rápido.

[18] Es un sistema operativo portable, multitarea y multiusuario.

[19] Colección de compiladores de *GNU.*

- Permite comprobar cómo se comportará una aplicación web ante multitud de usuarios y la velocidad de carga que tendrá. (23)

1.6. Lenguajes

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos de hardware y software existentes. (24)

1.6.1. **Lenguaje de programación**: PHP 5.3.22

PHP es un lenguaje interpretado de propósito general 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. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

Ventajas

- Lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Capacidad de conexión con la mayoría de los motores de bases de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5). (25)

1.6.2. **Lenguaje de programación**: JavaScript 1.2

JavaScript es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo y dinámico.

- Es simple, no hace falta tener conocimientos avanzados de programación para poder hacer un programa en JavaScript.
- Maneja objetos dentro de la página web y sobre ese objeto se pueden definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades y modificar archivos.
- Es dinámico, responde a eventos en tiempo real. Eventos como presionar un botón, pasar el puntero del mouse sobre un determinado texto o el simple hecho de cargar la página o caducar un tiempo. Con esto se puede cambiar totalmente el aspecto de la página al gusto del usuario, evitando tener en el servidor una página para cada gusto, hacer cálculos en base a variables cuyo valor es determinado por el usuario. (26)

1.6.3. **Hojas de estilo en cascada**: CSS 3

CSS[20] es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML. El W3C[21] es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. (27)

1.6.4. **Lenguaje de marcado de hipertexto**: HTML 4

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

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, permite un despliegue rápido, es fácil de aprender y lo admiten todos los exploradores. (28)

1.6.5. **Lenguaje de marcas extensible**: XML 1.0

El lenguaje de marcas extensible es un metalenguaje que permite definir lenguajes de marcado adecuados a usos específicos. Aunque a primera vista un documento XML puede parecer similar a HTML hay una diferencia fundamental: un documento XML contiene datos que se auto-definen, o sea no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea, dependiendo del

[20] *Cascading Style Sheets* (Hojas de Estilo en Cascada).

[21] *World Wide Web Consortium* (Consorcio de la *World Wide Web*)

contenido del documento. En XML se separa el contenido de la presentación de forma total.

Entre sus ventajas se encuentra su aceptación casi universal, su legibilidad y su carácter auto-contenido, si bien el tamaño de los documentos XML es mayor que el de sus equivalentes binarios y su procesamiento requiere más recursos, por lo que no resulta adecuado en aplicaciones en las que la eficiencia sea un objetivo prioritario.

XML permite representar datos de forma homogénea en entornos heterogéneos, lo que facilita la interoperabilidad entre distintos sistemas. (29)

1.6.6. **Lenguaje de hojas de estilo extensible**: XSL 1.0

El lenguaje extensible de hojas de estilo (Extensible Style Sheet Language, XSL) 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.

Esta familia está formada por tres lenguajes:

- XSLT (lenguaje de hojas extensibles de transformación), que permite convertir documentos XML de una sintaxis a otra. Por ejemplo de un XML a un documento HTML.
- XSL-FO (lenguaje de hojas extensibles de formateo de objetos), que permite especificar el formato visual con el cual se quiere presentar un documento XML, es usado principalmente para generar documentos PDF.
- XPath o XML Path Language, es una sintaxis (no basada en XML) para acceder o referirse a porciones de un documento XML.

XSL debe representar de forma independiente a la plataforma utilizada o al medio de representación la información recogida en los documentos XML. También posibilita la ejecución de bucles, sentencias del tipo IF...THEN, selecciones por comparación, operaciones lógicas, ordenaciones de datos y la utilización de plantillas. (30)

1.6.7. **Lenguaje de consulta estructurado**: SQL

SQL o lenguaje estructurado de consultas es un lenguaje declarativo de alto nivel o de no procedimiento de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas. No es más que un lenguaje estándar de comunicación con bases de datos. Se habla por tanto de un lenguaje normalizado que permite trabajar con cualquier tipo de lenguaje (PHP) en combinación con cualquier tipo de base de datos (*MS Access, SQL Server, MySQL*). (31)

1.6.8. **Lenguaje de modelado**: UML 2.0

El Lenguaje Unificado de Modelado (UML) se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es el lenguaje de modelado más conocido y

utilizado en la actualidad. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. UML incluye conceptos semánticos, notación y principios generales. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo, pero no especifica en sí mismo qué metodología o proceso usar. (32)

Un modelo UML está compuesto por tres clases de bloques de construcción:

- Elementos: los elementos son abstracciones de cosas reales o ficticias (objetos, acciones).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: son colecciones de elementos con sus relaciones.

1.7. Proceso de desarrollo de software

El proceso de desarrollo de software es un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un software. (33)

Los procesos de desarrollo de software son aquellos que guían cómo realizar un software correctamente y tienen una implicación total en sus resultados, debido a que influyen en la calidad del mismo. Estos procesos forman un grupo de actividades a realizar y estas a su vez generan un conjunto de artefactos que tributan a la correcta realización del producto.

Las metodologías ágiles o “ligeras” constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración. (34)

1.7.1. Metodología

XP (Programación Extrema)

Programación Extrema es una metodología ágil de desarrollo de software que posee 4 tareas fundamentales: planificación, diseño, desarrollo y pruebas. Esta metodología está basada en la simplicidad durante el desarrollo, la comunicación entre las partes implicadas (clientes y desarrolladores) y la retroalimentación para poder reutilizar el código desarrollado. En su concepción establece entregas frecuentes con posibilidad de refactorización continua, permitiendo mejorar el diseño cada vez que se

añade una funcionalidad. Para su implementación XP establece un conjunto de prácticas que deben ser empleadas en los proyectos de desarrollo, las mismas se mencionan a continuación: (35)

- El juego de la planificación.
- Entregas pequeñas.
- Metáfora.
- 40 horas por semana.
- Diseño simple.
- Pruebas.
- Refactorización.
- Cliente en el lugar.
- Programación en parejas.
- Propiedad colectiva del código.
- Integración continúa.
- Estándares de programación.

Scrum

Scrum es una metodología ágil enfocada a la gestión de proyectos. Sus principales características se pueden resumir en dos: el desarrollo de *sprint o* iteraciones y reuniones a lo largo del desarrollo. Las iteraciones en Scrum tienen una duración máxima de 30 días y el resultado de cada una de ellas define un incremento del producto a desarrollar. La evolución del proyecto por la metodología se define a través de reuniones diarias donde el trabajo del día anterior es revisado por el equipo, previendo además la labor a realizar el día siguiente. Dentro de las prácticas definidas por la metodología Scrum se encuentran: (36)

- Planificación de la iteración o *sprint.*
- Revisión de la iteración o *sprint.*
- Reunión diaria.
- Pila del producto.
- Incremento.
- Propietario del producto.

1.7.2. Modelos de calidad orientados a la mejora de procesos

En el mercado se encuentran modelos como CMMI[22], que se ha convertido en un estándar importante en la industria del software. Tiene el objetivo de contribuir al control y mejora de los procesos para alcanzar o superar las expectativas de los clientes asegurando que los productos cumplen con los requisitos que los definen. (37)

CMMI tiene dos representaciones, la representación continua y la escalonada. Son equivalentes y cada empresa puede optar por adoptar la que se adapte a sus características y prioridades de mejora.

-La representación continua es una aproximación que permite que una organización seleccione un área específica para hacerle una mejora. Utiliza niveles de capacidad para caracterizar una mejora relativa a un área de proceso individual. (37)

-La representación por niveles o escalonada es una aproximación que usa un conjunto predefinido de áreas de procesos para definir un camino para la mejora de una organización.

Tiene cinco niveles de madurez: Inicial o nivel 1, Gestionado o nivel 2, Definido o nivel 3, Gestionado Cuantitativamente o nivel 4 y Optimizado o nivel 5. El nivel dos posee siete áreas de procesos: Planificación de Proyectos, Seguimiento y Control del Proyecto, Gestión de Acuerdos con Proveedores,

[22] *Capability maturity model integration* (Integración de modelos de madurez de capacidades)

Medición y Análisis, Aseguramiento de la Calidad del Producto y el Proceso, Gestión de la Configuración y Administración de Requisitos. (37)

1.7.3. Proceso de desarrollo con enfoque ágil en el 2do nivel de CMMI

La Universidad de las Ciencias Informáticas llevó a cabo a partir del 2008 un proceso de mejora encaminado a alcanzar el nivel 2 del modelo CMMI. Con la gestión ágil de proyectos adoptando como guía en el desarrollo del software la integración de algunas prácticas de las metodologías ágiles Scrum y Programación Extrema (XP), que tiene como objetivo dar garantías a las cuatro demandas principales de la industria en la que se ha generado: valor, reducción del tiempo de desarrollo, agilidad y fiabilidad; y CMMI, modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de Administración como de Ingeniería de Sistemas y Software. Para determinar la correspondencia entre la aplicación de CMMI y las metodologías ágiles XP y Scrum se realizó un análisis donde se evaluó la posibilidad de implementación de las prácticas del modelo y las metodologías ágiles tratadas. (38)

A partir de dicho análisis se realizó un resumen que contiene las prácticas de las metodologías XP y Scrum presentes en el área de proceso de Administración de Requisitos del nivel 2 de CMMI, el mismo se encuentra en la Tabla 1. (38)

Tabla 1: Prácticas de las metodologías XP y Scrum utilizadas en el área de proceso de Administración de Requisitos.

Área de proceso del nivel 2 de CMMI	Prácticas de la metodología XP	Prácticas de la metodología Scrum
Administración de Requisitos	Cliente en el lugar.	Reunión diaria.
	Juego de planificación.	Planificación de la iteración.
	Diseño simple	Revisión de la iteración.
	Pruebas.	Pila del producto.
	Estándares de programación.	Incremento.
	Entregas pequeñas	Propietario del producto.

La Universidad de las Ciencias Informáticas establece el modelo de ciclo de vida de los proyectos de desarrollo de software donde se definen las fases (Inicio, Desarrollo, Transición y Cierre) por las que transitará el proyecto como se muestra en la Figura 3 y las disciplinas (Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación, Despliegue y Soporte) involucradas en el desarrollo de software. (39)

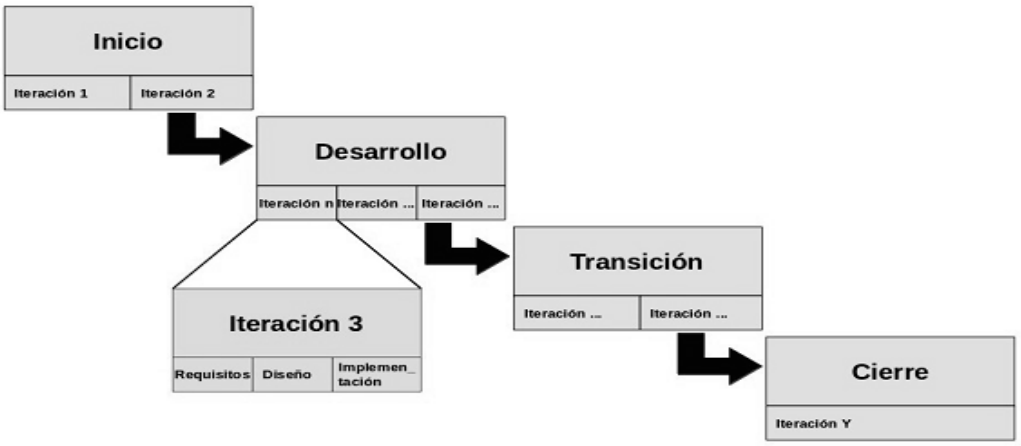


Figura 3: Relación entre las fases y disciplinas del ciclo de vida.

Por lo antes expuesto, el CENIA utiliza para el guiar la producción de software el proceso de desarrollo de software con enfoque ágil basado en el nivel 2 de CMMI utilizando el ciclo de vida que define la UCI. De igual forma, la propuesta de solución será desarrollada y documentada con el mismo proceso de desarrollo de software.

El análisis de los principales conceptos relacionados con el tema, permitió comprender los elementos teóricos que sustentan la presente investigación. El estudio de sistemas utilizados en la actualidad para el diseño de plantillas, en el mundo y en la UCI, permitió adquirir un conjunto de conocimientos previos sobre las características fundamentales que tienen estos sistemas. A pesar de que los mismos no satisfacen todas las necesidades y condiciones requeridas, constituyen una base de estudio para llegar a la obtención de la propuesta de solución. La caracterización de las herramientas, lenguajes y el proceso de desarrollo de software definido por el centro CENIA, permitió la familiarización con los elementos del entorno de desarrollo y la obtención de los conocimientos necesarios sobre sus características, para poder utilizarlos en la construcción de la propuesta de solución.

CAPÍTULO 2: Descripción de la propuesta de solución

En este capítulo se presenta la propuesta de solución al problema planteado en el diseño teórico de la investigación, con el objetivo de ayudar a comprender los conceptos con los que deberá trabajar la misma. Se desarrolla el modelado del dominio y se describen los conceptos fundamentales. Se especifican las técnicas de obtención de requisitos, identificando así los requisitos funcionales y no funcionales del diseñador de plantillas. Además se realiza la descripción de la arquitectura y patrones empleados, así como se muestra el modelo físico de la base de datos y el diagrama de despliegue con la descripción de sus componentes.

2.1. Modelo del dominio

Un modelo de dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción. Se presenta como uno o más diagramas de clases y no contiene conceptos propios de un sistema de software sino de la propia realidad física. (40)

La propuesta que se presenta formará parte del SGAP, específicamente del módulo Reportes, y básicamente su función será diseñar plantillas para generar reportes que obtendrán y filtrarán información gestionada por los otros módulos de dicho sistema. Por tanto no existe un negocio bien definido, propiciando que se haga un modelado de dominio en lugar del negocio, como se muestra en la Figura 4.

Figura 4: Representación del dominio.

El usuario requiere una información en un formato determinado, por tanto, entrega dicho formato al desarrollador. Este construye la plantilla y la integra al sistema, para luego crear todas las funcionalidades necesarias para visualizar la plantilla elaborada con la información que el usuario necesita y finalmente entregarla al mismo.

Para una mejor comprensión, a continuación se describen cada una de las clases que intervienen en el diagrama mostrado.

- **Usuario:** es el administrador del sistema o directivos de la alta gerencia que tengan conocimientos básicos de informática.
- **Información:** datos que necesita obtener el usuario en el formato determinado.
- **Formato plantilla:** estructura que el usuario entrega de cómo desea obtener la información.
- **Desarrollador:** persona capacitada que utiliza el sistema para construir la plantilla con el formato predefinido.
- **Plantilla:** modelo que define la presentación de los datos en los reportes.
- **Sistema:** aplicación informática con la que se construye la plantilla y se introduce información solicitada.
- **Información en plantilla:** plantilla digitalizada con información asociada.

2.2. Integración de la propuesta de solución al SGU

El SGU es un sistema que está compuesto por los sistemas Pregrado (SGAP), Postgrado, Cooperación, Residencia, Ingreso, Investigación, Producción, Laboratorios, Biblioteca, Extensión, Teleformación y Egreso, que agrupan las diferentes áreas de procesos de la UCI. Este sistema cuenta con módulos que son horizontales para todos los sistemas entre los que se encuentran: Seguridad, Configuración y Trazas.

Por su parte el SGAP está conformado por los módulos: Carrera, Personal y Secretaría, Tesis y títulos, Control Docente, Estudiante y Reportes.

El módulo Reportes se encarga de la gestión de reportes, gráficos y estadísticas, y cuenta con las principales funcionalidades de gestionar reportes y directorios. Los reportes a gestionar se clasifican en varios tipos dependiendo de las necesidades del usuario. Además dicho módulo permite elaborar gráficas a partir de los datos obtenidos de los reportes generados por el usuario, así como exportar dichos datos a diferentes formatos (pdf y xsl) para su almacenamiento físico en dispositivos. El módulo Reportes permite al usuario realizar determinadas configuraciones a la hora de crear el reporte sobre los datos que desea mostrar en el mismo, pero al final la salida del reporte es estática, es decir, el usuario no puede decidir la apariencia final de su informe. Por ello, se decide desarrollar el diseñador de plantillas para dicho módulo como un flujo alternativo con el objetivo de generar plantillas personalizadas con una interfaz de fácil uso para usuarios con conocimientos básicos en informática. En la Figura 5 se muestra un mapa de navegación de la propuesta de solución.

Figura 5: Integración de la propuesta de solución.

2.3. Propuesta de solución

El diseñador de plantillas forma parte del módulo Reportes del SGAP que a su vez es un sistema del SGU. Este permite al usuario poder diseñar una plantilla para el reporte a generar, permitiéndole elegir la apariencia de sus informes tal y cual desee, con un área de trabajo intuitiva. Lo cual facilita la interacción de los usuarios con la aplicación, sin que estos tengan avanzados conocimientos informáticos. En esta plantilla el usuario puede introducir componentes como: imágenes, etiquetas de texto, tablas y áreas de texto, así como modificar las propiedades de acuerdo a cada componente. El área de diseño se visualiza en un formato definido A4. Las plantillas diseñadas se pueden guardar para su posterior utilización. A continuación se explican las partes que tiene el diseñador de plantillas y se muestra la Figura 6 que visualiza la propuesta de solución.

- **Diseñador de plantillas:** aplicación visual para el diseño de plantillas.
- **Área de diseño:** región del diseñador de plantillas que permite la creación y edición de plantillas.
- **Plantilla:** modelo que define la presentación de los datos en los reportes.
- **Componentes de la plantilla:** elementos que se insertan en la plantilla del reporte a generar.
- **Propiedades:** características que adoptan los componentes.

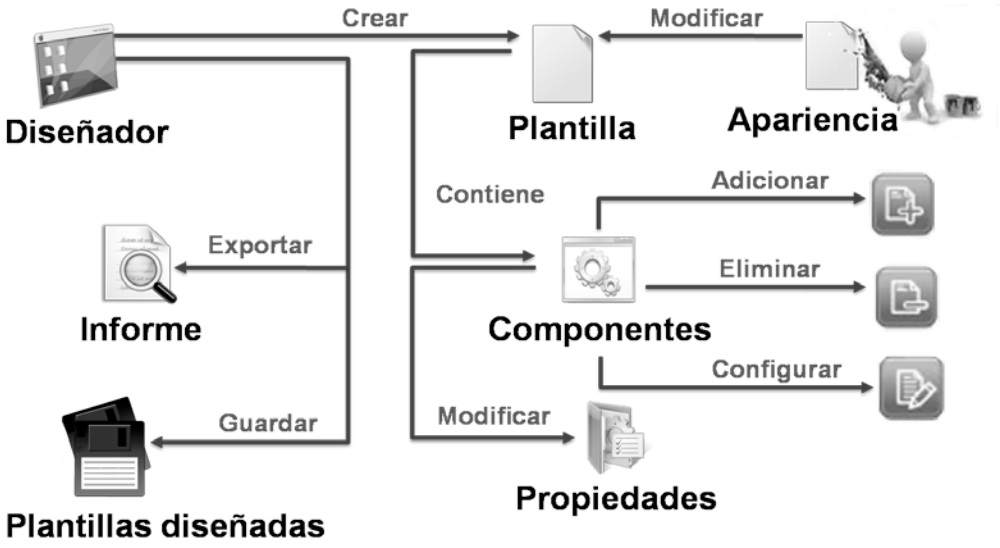


Figura 6: Mapa conceptual de la propuesta de solución.

2.3.1. Roles que interactúan con el diseñador de plantillas

Dado que el diseñador de plantillas que se propone permitirá obtener información, en ocasiones sensible, relacionada con la formación de los estudiantes, no todos los usuarios del SGU podrán interactuar con el mismo. El módulo solo será visible para los siguientes roles:

- ✓ Administrador del sistema.
- ✓ Directivos de la alta gerencia con conocimientos básicos de informática.

Esto se logrará mediante la integración con el módulo Seguridad del sistema anteriormente mencionado que es horizontal para todos los sistemas que lo conforman, incluyendo el SGAP.

2.4. Requisitos del diseñador de plantillas

Los requisitos de software le brindan al usuario la posibilidad de que el sistema cumpla y tenga las condiciones y propiedades que él necesita, siendo estos una manera de verificar la calidad del producto. Por esto, el levantamiento de requisitos se hace fundamental a la hora de desarrollar cualquier sistema. Estos requisitos se dividen en dos grupos, funcionales, que son las capacidades o condiciones que debe cumplir el sistema, y los no funcionales que son cualidades o propiedades que el producto debe tener. (41)

2.4.1. Técnicas de obtención de requisitos

Una cuestión básica en ingeniería de requisitos es la manera de averiguar lo que los usuarios realmente necesitan. La ingeniería de requisitos comprende las actividades de obtención (captura, descubrimiento y adquisición), análisis (negociación), especificación y validación de requisitos. La obtención de requisitos es

el proceso de recoger información sobre el sistema propuesto y los existentes, y extraer de esta los requisitos del usuario y del sistema. Existen diferentes técnicas para la obtención de los requisitos, de las cuales fueron utilizadas:

-Sesiones de tormentas de ideas (*Brainstorming*): el diseñador de plantillas forma parte del módulo Reportes del SGAP por lo que se realizaron reuniones de grupo, con la participación del cliente e integrantes del proyecto, con el objetivo de generar ideas respecto a las funcionalidades e integración del diseñador con el sistema. A partir de estas reuniones se encontraron algunas funcionalidades e interfaces que sirvieron de base para la propuesta de solución.

-Etnografía: es una técnica de observación que se puede utilizar para entender los requisitos sociales y organizacionales. A través de esta técnica el analista se sumerge por sí mismo en el entorno laboral donde será aplicado el sistema, observa el trabajo y anota las tareas reales en las que los participantes están involucrados. Con la etnografía se descubrieron los requisitos implícitos que reflejan los procesos reales más que los formales en los que los usuarios del diseñador de plantillas están involucrados.

-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 conocer más acerca del problema y sus posibles soluciones. Estos se le mostraron al cliente, quienes proporcionaron los requisitos adicionales o modificaron algunos de los anteriores.

2.4.2. Requisitos funcionales

En la Tabla 2 se muestran los requisitos funcionales que debe cumplir el diseñador de plantillas para un total de 23 requisitos funcionales de los cuales tienen una complejidad Alta 19 y Media 4, así como una prioridad Alta 21 y Media 2.

Tabla 2: Requisitos funcionales.

No.	Requisitos	Complejidad	Prioridad
RF1	Insertar componente de tipo texto.	Alta	Alta
RF2	Modificar texto.	Alta	Alta
RF3	Eliminar texto.	Media	Alta
RF4	Insertar componente de tipo tabla.	Alta	Alta
RF5	Modificar tabla.	Alta	Alta
RF6	Eliminar tabla.	Media	Alta
RF7	Insertar componente de tipo imagen.	Alta	Alta
RF8	Modificar imagen.	Alta	Alta
RF9	Eliminar imagen.	Media	Alta
RF10	Insertar componente de tipo área de texto.	Alta	Alta

RF11	Modificar área de texto.	Alta	Alta
RF12	Eliminar área de texto.	Media	Alta
RF13	Agregar componentes al componente de tipo tabla.	Alta	Media
RF14	Insertar columna y fila al componente de tipo tabla.	Alta	Alta
RF15	Combinar celdas en el componente de tipo tabla.	Alta	Alta
RF16	Crear plantilla.	Alta	Alta
RF17	Modificar plantilla.	Alta	Media
RF18	Exportar la plantilla.	Alta	Alta
RF19	Guardar plantilla.	Alta	Alta
RF20	Aplicar estilos de plantilla predefinidos.	Alta	Alta
RF21	Mostrar listado de plantillas.	Alta	Alta
RF22	Asociar componente a una variable de una fuente de información.	Alta	Alta
RF23	Acotar la fuente de información.	Alta	Alta

Especificación de requisitos

Con el objetivo de presentar los requisitos funcionales de una manera más consistente y por ende más entendible, se realizó la especificación de los mismos. Para esto se definieron secciones o escenarios para describir detalladamente cada requisito, 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. A continuación se muestra en la Tabla 3 una de las especificaciones diseñadas, para consultar las restantes remitirse al **Anexo 1**.

Tabla 3: Especificación del requisito Insertar componente de tipo texto.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF1.	Insertar componente de tipo texto.	Permite insertar un componente de tipo texto en el área de diseño de la plantilla al dar clic encima del componente y luego en el área de diseño en cualquiera de las regiones (Encabezado de página, Cuerpo o Pie de página), tomando por defecto los valores de las propiedades del texto (*Nombre, Área, Posición, Contenido, Fuente, Tamaño, Color*), además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Alta
	Prototipo			

Prototipo Insertar componente de tipo texto.

Campos	**Tipos de Datos**	**Reglas o Restricciones**
Nombre	character varying	De solo lectura.
Área	character varying	De solo lectura.
Posición	integer	De solo lectura.
Contenido	character varying	Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos para una palabra es de 30. La cantidad de caracteres permitidos es 100.
Fuente	character varying	Campo de selección.
Tamaño	integer	Solo dígitos del 8 al 32.
Color	character varying	Campo de selección.
Observaciones	*1.* Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.	

2.4.3. Requisitos no funcionales

En la Tabla 4 se muestran los requisitos no funcionales que debe cumplir el diseñador de plantillas para un total de 15 requisitos no funcionales.

Tabla 4: Requisitos no funcionales.

Usabilidad	
RNF 1	Desarrollar una solución web integrada al Sistema de Gestión Académica de Pregrado.
RNF 2	El sistema debe presentar una interfaz que permita la fácil interacción por parte de los usuarios, los cuales deben poder acceder de manera rápida y efectiva a la información solicitada.
Seguridad	
RNF 3	El sistema debe solicitar una verificación sobre acciones irreversibles (eliminaciones).
Disponibilidad	
RNF 4	El sistema estará disponible las 24 horas del día y los siete días de la semana.
Eficiencia	
RNF 5	El tiempo de respuesta del sistema debe ser como máximo de 8 segundos y soportar una conexión simultánea de hasta 100 usuarios.
Software	
RNF 6	El lenguaje de programación a utilizar es PHP 5.3.22.
RNF 7	Como Entorno de Desarrollo Integrado se emplea NetBeans 7.2.
RNF 8	Como servidor web se utiliza Apache 2.2.2.
RNF 9	El Sistema Gestor de Bases de Datos debe ser PostgreSQL 8.4.1.
RNF 10	El diseño de la base de datos se realiza con Visual Paradigm 8.0.
RNF 11	El sistema operativo a utilizar en el entorno de desarrollo deberá ser: GNU Linux.
Hardware	
RNF 12	Para el desarrollo se requiere de una PC Intel Pentium 4, CPU 3GHZ, de 2 GB RAM, 160 GB HDD.
RNF 13	Para la utilización del cliente se requiere de una PC Pentium 3, CPU 133 MHZ, 1 GB RAM recomendada.
Soporte	
RNF 14	El sistema brinda como apoyo una Ayuda contextual en la cual se refleja la explicación de cada una de las pantallas con sus respectivas funcionalidades.
RNF15	Se precisa que la documentación del sistema esté actualizada en todos los aspectos, fases de trabajo y ciclos de desarrollo del mismo, permitiendo un respaldo tanto ingenieril como legal del desarrollo de dicho sistema.

2.5. Descripción de la arquitectura

Entre las definiciones más reconocidas de la Arquitectura de software se encuentra la de Paul Clements:[23] *"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 la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones".* (42)

Teniendo en cuenta que la propuesta de solución a desarrollar está integrada al SGAP del SGU, la misma deberá cumplir con todas las pautas especificadas para el desarrollo de dicho sistema. Por esta razón se ajustará a la arquitectura definida, 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 seleccionada en la cual se está desarrollando este sistema.

2.5.1. Arquitectura

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debido a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

Cliente: el cliente es el proceso que permite al usuario formular los requisitos y pasarlos al servidor. El cliente normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario, además de acceder a los servicios distribuidos en cualquier parte de una red.

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

• Administrar la interfaz de usuario.

• Interactuar con el usuario.

• Procesar la lógica de la aplicación y hacer validaciones locales.

• Generar requisitos de bases de datos.

[23] Paul Clements es un alto miembro del personal técnico del Instituto de Ingeniería de Software (SEI) en E.U., donde trabaja en arquitectura de software e ingeniería de la línea de productos. Él es el autor de cinco libros y más de tres docenas de documentos sobre estos temas.

• Recibir resultados del servidor.

• Formatear resultados.

Servidor: es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

• Aceptar los requisitos de bases de datos que hacen los clientes.

• Procesar requisitos de bases de datos.

• Formatear datos para trasmitirlos a los clientes.

• Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un solo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, servidor de bases de datos, entre otros.

La ventaja más importante de este modelo es que es una arquitectura distribuida. Se puede hacer un uso efectivo de los sistemas en red con muchos procesadores distribuidos. 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. La Figura 7 muestra una representación de la arquitectura cliente-servidor.

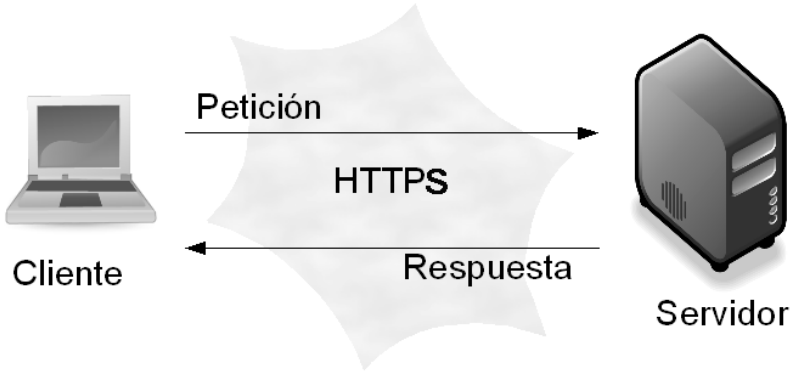


Figura 7: Arquitectura cliente-servidor.

2.5.2. Patrón de arquitectura

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 y representa una plantilla de construcción que provee un conjunto de subsistemas aportando las normas para su organización.

El Modelo-Vista-Controlador es un patrón para el desarrollo del software que se basa en separar los datos, la interfaz del usuario y la lógica interna. Es mayormente usado en aplicaciones web, donde la vista es la página HTML, el modelo es el sistema de gestión de base de datos y la lógica interna y el controlador es el responsable de recibir los eventos y darles solución.

A continuación se explica mejor cada elemento:

Modelo: es la representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedido por el controlador para añadir, eliminar, consultar o actualizar datos.

Vista: es la que presenta al modelo en un formato adecuado para que el usuario pueda interactuar con él, casi siempre es la interfaz de usuario.

Controlador: es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactúa tanto con el modelo como con la vista.

Implementación del MVC que realiza GUUD

La Figura 8 ilustra la particularidad del MVC implementado por el GUUD y que a continuación se describe. El marco de trabajo GUUD 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[24] 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 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. Posteriormente, el controlador carga estos datos a archivos escritos en HTML los cuales pueden incluir llamadas a archivos escritos en

[24] 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).

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.

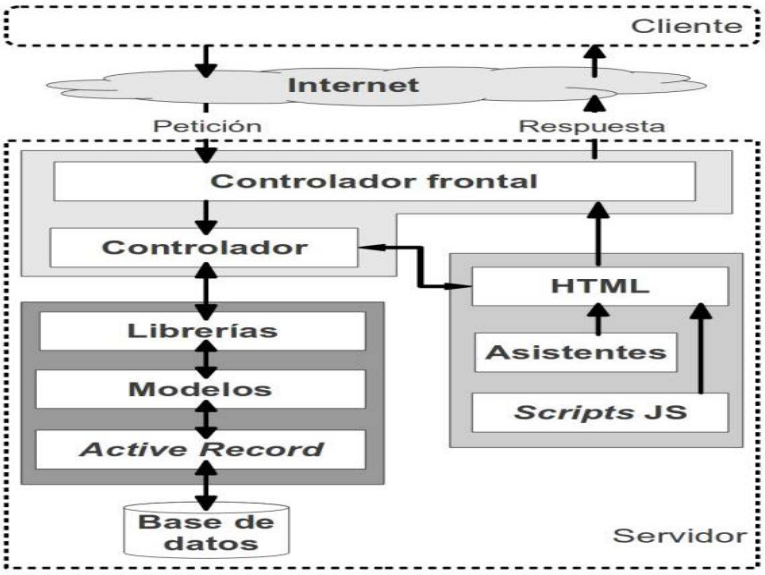


Figura 8: Funcionamiento del MVC en GUUD.

Ventajas de utilizar este patrón arquitectónico:

- Diseño modular y poco acoplado, favoreciendo la reutilización.
- Mayor cohesión ya que cada elemento del patrón está altamente especializado en su tarea, la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo del negocio.
- Las vistas proveen mayor flexibilidad y agilidad pues se pueden crear múltiples vistas de un modelo, las vistas pueden anidarse y sincronizarse.
- Más claridad de diseño.
- Facilita el mantenimiento.
- Mayor escalabilidad.

2.5.3. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo. El patrón de diseño tiene como finalidad precisar en detalle los subsistemas y componentes de la aplicación.

Los beneficios que provee el uso de patrones en el proceso de desarrollo de software incluyen: reutilización de diseño, reutilización potencial de código, mayor comprensión de la organización global de

un sistema y mejor interoperabilidad con otros sistemas por medio de la introducción de estándares. Los patrones permiten establecer un vocabulario común de diseño, cambiando el nivel de abstracción a colaboraciones entre clases y permitiendo comunicar experiencia sobre dichos problemas y soluciones. Son también un gran mecanismo de comunicación para transmitir la experiencia de los ingenieros y diseñadores experimentados a los novatos, convirtiéndose en unas de las vías para la gestión del conocimiento. (43)

Patrones para asignar responsabilidades (GRASP)

Los patrones GRASP[25] describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento.

Para el diseño de la propuesta de solución se tuvo en cuenta los patrones GRASP: Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador. El marco de trabajo GUUD busca un máximo rendimiento y flexibilidad en sus soluciones y pone en práctica estos patrones para lograr un sistema reusable y flexible. A continuación se realiza una descripción de los patrones antes mencionados:

- **Experto:** asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.
- **Creador:** asignarle a la clase B la responsabilidad de crear una instancia de clase A.
- **Controlador:** asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.
- **Bajo acoplamiento**: asignar una responsabilidad para mantener bajo acoplamiento. El grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.
- **Alta cohesión:** asignar una responsabilidad de modo que la cohesión siga siendo alta. Este patrón define que la información que almacena una clase debe de ser coherente y está en mayor medida relacionada con la clase.

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 clases y la formación de estructuras de mayor complejidad. Permiten crear grupos de objetos que ayudan

[25] *General Responsibility Assignment Software Patterns*, Patrones Generales de Software para Asignar Responsabilidades.

a realizar tareas complejas. Estos patrones pueden ser de tres tipos: de creación, estructurales y de comportamiento.

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.

- **Fábrica abstracta (*Abstract Factory*):** permite trabajar con objetos de distintas familias de manera que estas no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- **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.

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

- **Mediador (*Mediator*):** define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- **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.

Aplicaciones de los patrones

GRASP

- **Experto:** el uso de este patrón se evidencia en las clases librerías, que son las que cuentan con la información necesaria para cumplir las responsabilidades sobre los elementos de negocio.
- **Creador:** el uso de este patrón se evidencia en la clase *loader* que es el objeto load de las clases controladoras, se encarga de cargar los elementos del marco de trabajo dígase, librerías, modelos.
- **Controlador:** el uso de este patrón se evidencia en las clases controladoras que se encargan de obtener los datos y enviarlos a las librerías y las vistas.
- **Bajo acoplamiento y Alta cohesión:** la propia implementación de codeigniter contiene estos patrones nivelados pues permite el uso de los componentes de forma individual, evidenciando el bajo acoplamiento y así como la dependencia entre ellos o alta cohesión.

GoF

- **Fábrica abstracta:** en el módulo Seguridad, en la librería fabrica_ma_lib, que se encarga de crear los objetos de los modos de autenticación (ma) que heredan de la clase autenticacion_lib, que son ma servicio web, ma base de datos, ma ldap y ma open ldap.

- **Instancia única:** Todas las clases controladoras, son instancias únicas. La Ioc para la interacción entre módulos.
- **Mediador:** las librerías que funcionan como mediadoras entre las clases controladoras y las modelos o acceso a datos.
- **Observador:** este patrón se evidencia en la clase loader que es el objeto load de las clases controladoras, se encarga de cargar los elementos del marco de trabajo dígase, librerías, modelos y se encarga de actualizar la controladora instanciada.

2.5.4. Patrones de base de datos

Los patrones de diseño de una Base de Datos permiten al usuario crear una BD más fortalecida ya que constituyen una guía que especifica cómo debe ser la misma. El diseño y construcción de una base de datos requiere del mayor esfuerzo y análisis posible, ya que a partir de este diseño es que esta se crea y la calidad con que se obtenga determinará su comportamiento futuro. (44)

- **Llaves subrogadas**

Este patrón es muy utilizado pues facilita la interacción con la BD en un futuro. El mismo plantea que se genere una llave primara única para cada entidad, en vez de usar un atributo identificador en el contexto dado. Normalmente se usan enteros en columnas *identity* o GUID (*Global UniqueIdentifier*) que están demostradas que no se repiten o con una probabilidad extremadamente baja. Esto permite que las tablas sean más fáciles de consultar a partir del identificador, pues todos tienen el mismo tipo en cada una de las tablas. (44)

Mediante este patrón, fueron generados los identificadores de todas las tablas pertenecientes al modelo de datos que se muestra más adelante.

- **Entidad-Atributo-Valor**

Es la representación de un modelo flexible donde se pueden representar objetos con sus atributos. Es un acercamiento al modelo orientado a objeto representado en el modelo relacional, donde la entidad *Class* representa las clases, la entidad *Attribute* representa los atributos de las clases, por su parte la entidad *Object* representa las instancias de las clases, mientras que la entidad *Value* representa los valores de cada atributo para cada objeto dado. (44)

El modelo de datos propuesto en el acápite 2.5.6 presenta dicho patrón entre las tablas:
tb_dcomponente -> tb_rcomponente_propiedad -> tb_npropiedad.

- **Árboles fuertemente codificados**

A cada nivel del árbol se le asocia una entidad. Normalmente constituyen relaciones de 1 a muchos (n). Se utiliza para representar jerarquías donde es bien conocida la estructura y es importante representar la

correspondencia, por ejemplo las estructuras organizacionales. Además admite tantos niveles como requiera la jerarquía que se vaya a representar(44), como se muestra en la Figura 9.

El modelo de datos propuesto en el acápite 2.5.6 presenta dicho patrón entre las tablas:

tb_dplantilla -> tb_dcomponente -> tb_rcomponente_propiedad.

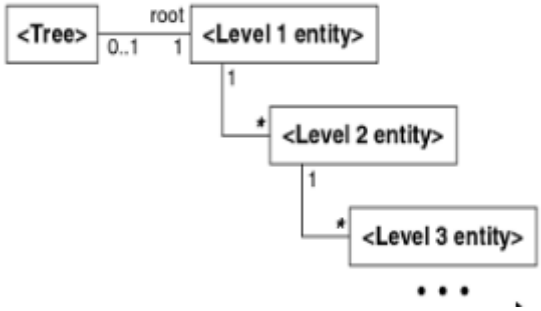


Figura 9: Árbol fuertemente codificado.

2.5.5. Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Este diagrama muestra las relaciones físicas entre los componentes de hardware y software en el sistema final. Describe la topología del sistema: la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos, dicha estructura se muestra en la Figura 10.

Figura 10: Diagrama de despliegue.

Descripción de elementos e interfaces de comunicación

PC cliente: su función es acceder al sistema 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, incluyendo computadoras personales, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN).

2.5.6. Modelo de datos

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre ellos y las restricciones que deben cumplirse sobre los mismos. Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas y actualizaciones de datos. A continuación se muestra la Figura 11 que representa el modelo físico de la propuesta de solución con un total de 7 clases persistentes que son descritas en el diccionario de datos en el **Anexo 2**.

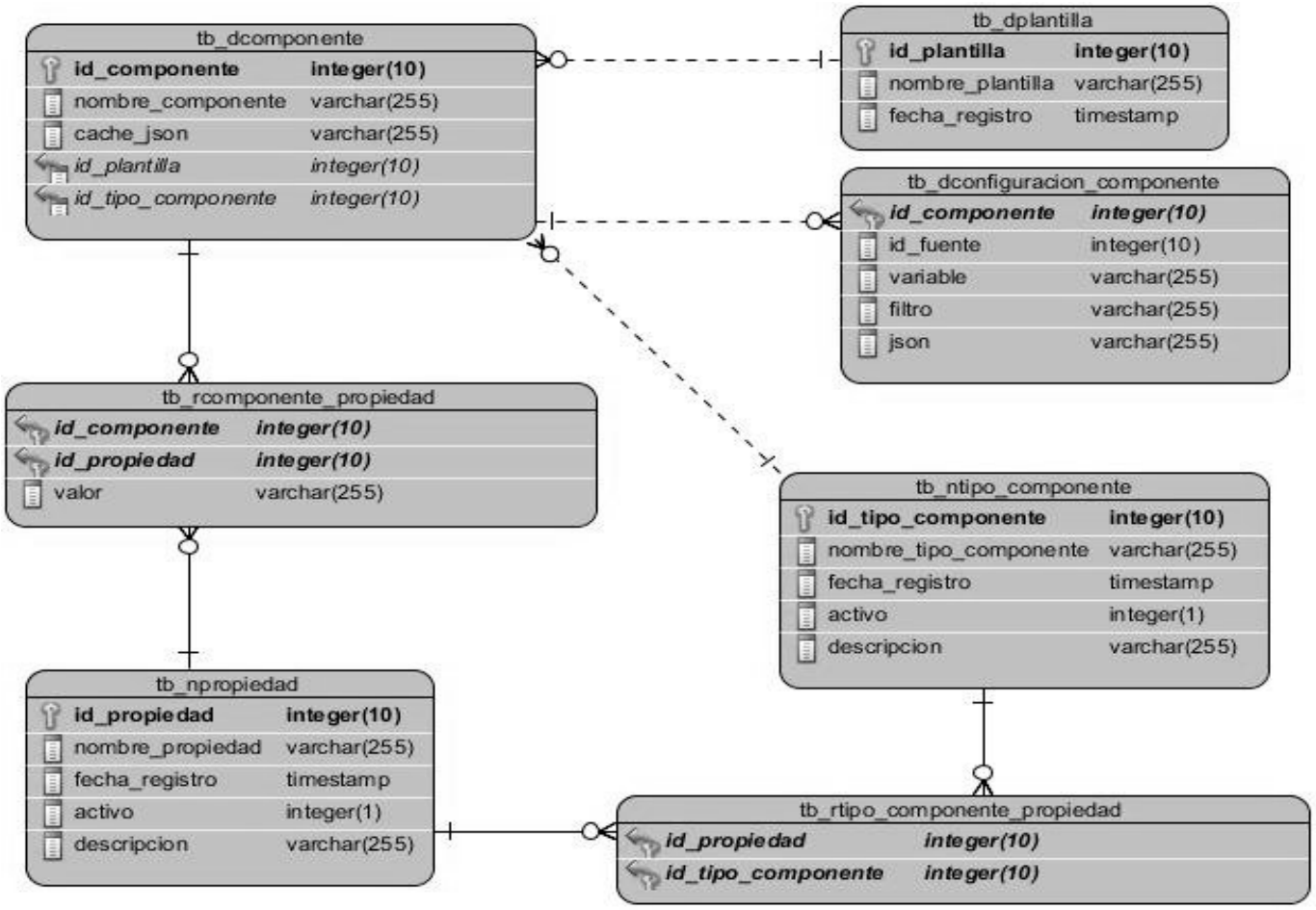


Figura 11: Modelo de datos físico.

La descripción de las características fundamentales de la propuesta de solución y las personas que se relacionan con el diseñador de plantillas, propició un mejor entendimiento con el cliente. El estudio de las técnicas de obtención de requisitos, permitió la definición de los requisitos funcionales y no funcionales. Con la caracterización de la arquitectura cliente servidor, el patrón MVC, los patrones de diseño y los de base de datos, así como la obtención del modelo de datos y la distribución física de la propuesta de solución, se logró comprender toda la arquitectura para el desarrollo de la propuesta de solución. Al finalizar este capítulo, se logró crear las condiciones necesarias para la implementación y validación del diseñador de plantillas para la generación de reportes del SGAP.

CAPITULO 3: Implementación y validación de la solución

En la evaluación de un software el número de tipos de pruebas varía tanto, como el número de enfoques que se le pueda dar al desarrollo. Las pruebas de software son un elemento crítico para la garantía de la calidad del mismo y representan una revisión final de las especificaciones del diseño y la codificación. En el presente capítulo se describe la implementación de la propuesta de solución descrita en el capítulo anterior teniendo en cuenta las técnicas de codificación empleadas. Además se presentan las pruebas a realizar como unitarias, de integración, del sistema y de aceptación para validar que dicha solución satisface todas las necesidades del cliente y cumple con todas sus especificaciones.

3.1. Técnicas de codificación

Una técnica de codificación completa comprende todos los aspectos de la generación de código. Un código fuente completo debe estar organizado, como si un único programador hubiera escrito todo el código de una sola vez. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer una técnica de codificación sobre la que se efectuarán luego revisiones del código de rutinas.

Para el desarrollo del diseñador de plantillas se utilizaron las técnicas de codificación establecidas por el centro CENIA, con el propósito de estandarizar las nomenclaturas en la implementación del sistema y obtener un producto estable.

3.1.1. Indentación, llaves de apertura y cierre, y tamaño de las líneas

Se debe usar la indentación sin tabulaciones, con un equivalente a 4 espacios, para mantener integridad en las revisiones svn[26]. El uso de las llaves "{,}" será en una nueva línea. La longitud de las líneas de código es aproximadamente de 75-80 caracteres, para mantener la legibilidad del código.

Ejemplo:

```
....{
    ....if($parametro)
    ....{
        ....//Bloque de instrucciones
    ....}
....}
```

Figura 12: Indentación y llaves.

[26] Sistema de control de versiones.

3.1.2. Convención de nomenclatura

Variables locales

Los nombres de las variables se rigen por la nomenclatura *CamelCase*[27], 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;
```

Figura 13: Variables.

Clases

El nombre debe ser descriptivo, evitando abreviaturas y siempre comienzan con mayúscula. En caso de nombres compuestos, se usará el guión bajo "_" para separar las palabras.

Ejemplo:

```
....class Clase_nombre_compuesto
....{
    ....//Bloque de instrucciones
....}
```

Figura 14: Clases.

Funciones

Las funciones se rigen por la nomenclatura CamelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa y aquellos con valores por defecto deben ser colocados al final de la lista.

Ejemplo:

```
....function funcionNombreCompuesto()
....{
    ....//Bloque de instrucciones
....}
```

Figura 15: Funciones.

Ficheros: siempre se escriben en minúscula y en caso de nombres compuestos se usa el guión bajo"_".

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

[27] 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.

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

3.1.3. Estructuras de control

Se incluye un espacio entre las estructuras de control (if, for, foreach, while, switch) y los paréntesis. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos. Si las condiciones son muy largas que sobrepasan el tamaño de la línea, estas se dividen en varias líneas.
Ejemplo:

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

Figura 16: Estructuras de control.

3.1.4. 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
 *
 * PHP versión #
 *
 * @category Categoría de la clase implementada “Libreria,
 * Controladora, Modelo”
 * @package Nombre del paquete o módulo al que pertence
 * @author Nombre y Apellidos del autor y correo electrónico
 */
```

Figura 17: Documentación de clases.

```
/**
 * Breve descripción de la función
 *
 * @param tipo y nombre del paremetro(por cada parametro que
 * recibe la función)
 * @return tipo que retorna
 * @author Nombre y Apellidos del autor y correo electrónico
 */
```

Figura 18: Documentación de funciones.

3.1.5. Llamadas a funciones

Las funciones deben ser llamadas sin espacio entre el nombre de la función, el paréntesis de apertura y el primer parámetro. En caso de varios parámetros, separar con espacios entre la coma y cada parámetro y sin espacios entre el último parámetro, el paréntesis de cierre y el punto y coma.

3.1.6. Inclusión de código

Salvo casos específicos y puntuales, utilizar **require_once** para incluir código incondicionalmente e **include_once** para los casos condicionales (o sus equivalentes en otros lenguajes).

3.1.7. 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++ (//).

3.1.8. Buenas prácticas

Los valores booleanos y nulos siempre se escriben con mayúscula, para facilitar la legibilidad del código, se usa una línea en blanco antes de las estructuras de control y definición de las funciones.

Ejemplo:

```
....$variableBooleana = FALSE;
....$variableNula = NULL;
```

Figura 19: Buenas prácticas.

3.2. Validación de los requisitos

La validación de requisitos es el proceso por el cual se determina si los requisitos identificados son consistentes con las necesidades del cliente y se realiza antes de comenzar la implementación. Además examina las especificaciones para asegurar que los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones y que el resultado del trabajo se ajusta a los estándares establecidos.

3.2.1. Criterios de validación de requisitos del cliente

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

Interrogantes para validar los requisitos del cliente

- ¿El proveedor del requisito es un proveedor válido?

- ¿El requisito está identificado como único?
- ¿El requisito es modificable?
- ¿El requisito no es ambiguo?
- ¿El requisito está completo?
- ¿El requisito es congruente con otros REQ relacionados?
- ¿El requisito puede ser implementado?
- ¿El requisito puede ser probado?
- ¿El resultado de la evaluación de impacto es positivo?
- ¿El requisito está correcto?
- ¿El requisito es traceable?

Resultado de aplicar los criterios de validación

Con la aplicación de los criterios para validar requisitos del cliente, se logró obtener el 100 % de los requisitos aprobados como se refleja en el documento “Criterios para validar requisitos del cliente” en el expediente del proyecto Pregrado.

3.2.2. Técnicas de validación de los requisitos

Existen algunas técnicas que permiten que el proceso de validación tenga una mejor calidad, de ellas se utilizaron:

- **Revisión de requisitos:** se realizaron reuniones entre varios miembros del proyecto, fundamentalmente analistas donde se revisaron los documentos de especificación de requisitos y se detectaron un número de inconsistencias, que pueden ser 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 usuarios finales y 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 requisitos especificados, permitiendo verificar que todos se pudieran probar e identificar en medida de la complejidad del diseño de caso de prueba, requisitos que deberían ser reconsiderados y cuáles pueden ser los más difíciles de implementar.

Resultado de la revisión de los requisitos

Durante la revisión de requisitos se realizaron verificaciones en el documento “Especificaciones de requisitos”. Este proceso comprendió:

- Verificaciones de validez: los requisitos 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 requisitos no deben contradecirse en las especificaciones escritas, no debe haber restricciones o descripciones que estén opuestas a las reglas definidas.
- Verificaciones de completitud: los requisitos 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 requisitos estuvieran descritos de manera que puedan ser verificables, o sea, que se puedan diseñar casos de pruebas orientados a estos y que demuestren que el sistema a entregar responde a las necesidades del cliente.

Como resultado de este proceso se identificaron inconsistencias en las especificaciones tales como:

- ✓ Falta de concordancia entre la complejidad de la especificación y la registrada en el documento de Evaluación de Requisitos.
- ✓ Componentes innecesarios en los prototipos de interfaz.
- ✓ Descripciones poco detalladas de algunos de los requisitos.
- ✓ Errores ortográficos.

3.3. Pruebas de software

En la investigación se utilizan algunas de las pruebas definidas por Pressman[28] en su libro “Ingeniería de software: Un enfoque práctico”, donde se pretende que el producto se pruebe desde la parte más pequeña del software hasta la más grande en forma de espiral usando los niveles, tipos, técnicas y herramientas de prueba.

Para comprobar que la solución implementada es válida y que cumple con los requisitos acordados con el cliente, se realizaron pruebas unitarias, pruebas de integración, pruebas del sistema y pruebas de aceptación como se muestra en la Tabla 3.

[28] Roger Pressman, es una autoridad internacionalmente reconocida en la mejora de procesos de software y en tecnologías de Ingeniería de software. Por más de tres décadas, ha trabajado como ingeniero, gerente, profesor, autor y consultor de software en temas de ingeniería de software. Actualmente es presidente de R.S. Pressman *and Associates*, Inc., una firma consultora especialista en métodos y entrenamiento en ingeniería de software.

Tabla 5: Pruebas de software.

Pruebas	Tipo de prueba	Método	Técnica
Unitarias	Funcional	Caja blanca	Camino básico
		Caja negra	Particiones de equivalencia
Integración	Funcional	Caja negra	Incremental
Sistema	Rendimiento y resistencia	Caja negra	Automática
Aceptación	Funcional	Caja negra	Alfa

3.3.1. Pruebas unitarias

Las pruebas de unidad se centran en la verificación de los elementos más pequeños del software que se puedan probar. Se prueba la interfaz para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada. Además, se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente conservan su integridad mediante los pasos de ejecución del algoritmo. La interfaz del sistema puede ser probada mediante el método de **caja negra**, también denominada prueba de comportamiento, esta se centra en los requisitos funcionales del software. Las estructuras de datos pueden ser probadas mediante el método de **caja blanca** o estructural, 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.

Pruebas de caja blanca o estructural

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, se pueden obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada funcionalidad. (41)

La técnica utilizada en la investigación para realizar el método de caja blanca es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida corno guía para la definición de un conjunto básico de caminos de ejecución. (41)

Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. (41)

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.

CodeIgniter posee una librería o clase especializada en la ejecución de pruebas estructurales, 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 ejecutar 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 debe devolver 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*".

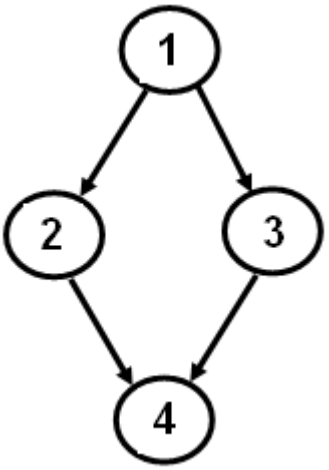
En la Tabla 6 se muestra un caso de prueba de caja blanca utilizando la técnica de camino básico:

Tabla 6: Caso de prueba de caja blanca CENIA_PRE_R_DP_OP.

<table>
<tr><th>Prueba estructural de caja blanca</th><th>Código caso de prueba: CENIA_PRE_R_DP_OP.</th></tr>
<tr><td colspan="2">Probador: Yander Santiesteban Rojas</td></tr>
<tr><td colspan="2">Código al que se aplica:

```
public function obtenerPlantilla($post_vars) {
    $this->load->helper('grid');
    if (isset($post_vars['cadena']) && !empty($post_vars['cadena'])) {
        $datos_buscador = json_decode($post_vars['cadena']);
        echo grid_json($this->generador_reporte_lib, 'obtenerCantidadPlantillas',
                'obtenerPlantillas', array($datos_buscador));
    } else {
        $datos_buscador = array();
        echo grid_json($this->generador_reporte_lib, 'obtenerCantidadPlantillas',
                'obtenerPlantillas', array($datos_buscador));
    }
}
```

</td></tr>
<tr><td>Complejidad ciclomática:
V (G) = (A - N) + 2 = (4 - 4) + 2 = 2
Caminos independientes
1) 1 – 3 – 4
2) 1 – 2 – 4</td><td>Representación del grafo:</td></tr>
</table>



Caso de prueba para el camino básico 1	
Descripción: los datos de entrada es el atributo a buscar.	
Condición de ejecución: los datos de entrada no se envían.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo vacío.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array." $nombrePrueba="Prueba:'Obtener Plantillas'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->obtenerPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 2	
Descripción: los datos de entrada es el atributo a buscar.	
Condición de ejecución: se envían los datos de entrada.	
Procedimiento prueba automatizada	
Datos de entrada:	un arreglo de datos para buscar.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array." $nombrePrueba="Prueba:'Obtener Plantillas'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->obtenerPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	

Evaluación del caso de prueba:	satisfactoria
Resultado final de la prueba: satisfactoria en su totalidad.	

En el **Anexo 3** se muestra el resto de los casos de prueba de caja blanca realizados a la solución. Al aplicar la prueba de caja blanca se alcanzaron los siguientes resultados en cada una de las iteraciones, como se muestra en la Tabla 7. Las no conformidades encontradas fueron resueltas en su totalidad.

Tabla 7: No conformidades por iteraciones caja blanca.

Iteraciones	No. No conformidades	Asociadas a
1ra	9	Errores de validación, entradas y salidas incorrectas.
2da	3	Errores de validación
3ra	0	-

Pruebas de caja negra o funcional

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Estas permiten genera casos de prueba que demuestran que las funciones del software son operativas, que la entrada sea aceptada de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. (41)Este método se utiliza a través de la realización de los casos de prueba basados en requisitos, utilizando la técnica de particiones de equivalencia. Esta técnica divide el campo de entrada en clases de datos que ejercitan determinadas funciones del software, es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software. El objetivo del diseño de casos de prueba es que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requisitos. Para diseñar un caso de prueba, se selecciona una característica del sistema o componente que se está probando.

En la Tabla 8 se muestra un caso de prueba de caja negra utilizando la técnica particiones de equivalencia:

Tabla 8: Caso de prueba de caja negra.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Insertar elemento correctamente.	Mediante este escenario el usuario puede insertar un componente imagen correctamente.	El sistema inserta el componente imagen en el área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Pregrado y luego el módulo Reporte. El sistema muestra las opciones de menú y

			el usuario selecciona la agrupación funcional "Diseñador de plantillas" la funcionalidad ”Diseñar plantilla”. Seguidamente se selecciona la opción aplicar plantilla en el listado de plantillas o crear plantilla en el área de íconos flotantes y luego se selecciona en la paleta de componentes el componente imagen, dando clic encima del componente y luego en el área de diseño.
EC 1.2 Insertar elemento incorrectamente.	Mediante este escenario el usuario no puede insertar un componente imagen en el área de diseño de la plantilla.	El sistema no inserta el componente imagen en el área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Pregrado y luego el módulo Reporte. El sistema muestra las opciones de menú y el usuario selecciona la agrupación funcional "Diseñador de plantillas" la funcionalidad “Diseñar plantilla”. Seguidamente se selecciona la opción aplicar plantilla en el listado de plantillas o crear plantilla en el área de íconos flotantes y luego se selecciona en la paleta de componentes el componente imagen, se arrastra el componente hacia el área de diseño.

Para ver los diseños de casos de pruebas en su totalidad, consultar el expediente del proyecto Pregrado.

Al aplicar la prueba de caja negra se alcanzaron los siguientes resultados en cada una de las iteraciones, como se muestra en la Tabla 9. Las no conformidades encontradas fueron resueltas en su totalidad.

Tabla 9: No conformidades por iteraciones caja negra.

Iteraciones	No. No conformidades	Asociadas a
1ra	18	Errores de interfaz, de validación y ortografía.
2da	9	Errores de interfaz, de validación y ortografía.
3ra	0	-

3.3.2. Pruebas de integración

Aún 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. 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, en la integración 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 solución. Estas pruebas aparecen relacionadas en el **Anexo 4**. Al finalizar las pruebas de integración no fueron detectados errores asociados a la interacción del diseñador de plantillas con los módulos Control Docente, Tesis y Título, Personal y Secretaría y Estudiante del SGAP.

3.3.3. 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. (41)

Pruebas de resistencia (Stress)

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. (41)

Pruebas de rendimiento (Carga)

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. (41)

Resultados de las pruebas de rendimiento y stress

Para llevar a cabo las pruebas de carga y stress se utilizó la herramienta JMeter 2.3.1 descrita en el acápite 1.5.9. A continuación se muestran en la Tabla 10 los resultados obtenidos con la misma y que fueron extraídos de las imágenes relacionadas en el **Anexo 5**.

Tabla 10: Resultados de las pruebas del sistema.

Aplicado a	Usuarios(Muestras)	Tiempo de ejecución (ms)				Rendimiento		
		Mín.	Max.	Media	Mediana	% Error	Pet/seg	Kb/seg
Propiedades del texto	1	175	175	175	175	0.0%	5.3/sec	27.5
	50	447	8807	4931	6305	0.0%	5.2/sec	24.9
	100	433	8807	4898	6282	0.0%	40.9/min	3.3
Obtener objeto	1	212	212	212	212	0.0%	4.7/sec	0.6
	50	835	3849	2620	3250	0.0%	3.9/sec	0.5

	100	771	3987	2609	3142	0.0%	40.0/min	0.1
Obtener propiedades del objeto dado Id	1	239	239	239	239	0.0%	4.2/sec	3.8
	50	909	4075	2785	3198	0.0%	3.3/sec	3.0
	100	909	4075	2865	3331	0.0%	39.4/min	0.6
Guardar plantilla	1	235	235	235	235	0.0%	4.3/sec	2.8
	50	1067	3534	2673	2908	0.0%	3.0/sec	1.9
	100	927	3573	2628	2864	0.0%	38.9/min	0.4
Obtener plantilla	1	214	214	214	214	0.0%	4.7/sec	1.5
	50	1103	2901	2023	1994	0.0%	2.9/sec	0.9
	100	784	3211	2047	2055	0.0%	38.8/min	0.2
Obtener propiedades del componente dado Idplantilla	1	197	197	197	197	0.0%	5.1/sec	4.3
	50	522	3216	1791	1741	0.0%	3.0/sec	2.5
	100	522	3216	1800	1696	0.0%	38.9/min	0.5

Columna Mínimo (Mín.): indica el mínimo de tiempo de ejecución invertido para una petición con *n* (columna *Muestras*) usuarios haciendo peticiones de manera concurrente.

Columna Máximo (Máx.): indica el máximo de tiempo de ejecución invertido para una petición con *n* (columna *Muestras*) usuarios haciendo peticiones de manera concurrente.

Media: representa el tiempo de ejecución promedio de una petición con *n* usuarios.

Mediana: significa que el 50% de las peticiones realizadas por *n* usuarios tardaron menos del valor reflejado.

Error: indica la relación entre el total de peticiones y el número de peticiones que originaron errores.

Rendimiento (Pet./seg): hace referencia al número de peticiones que el servidor puede procesar en un segundo.

Rendimiento (Kb./seg): hace referencia a la cantidad de datos que el servidor puede procesar en un segundo.

Análisis de los resultados de las pruebas del sistema

De manera general en la Tabla 10 se puede observar que las peticiones son ejecutadas en tiempos (inferiores a 5 segundos en la mayoría de los casos). Para todas las muestras de usuarios la ocurrencia de errores se mantiene en 0.0%, lo que quiere decir que todas las peticiones hechas se ejecutan satisfactoriamente. Si se tiene en cuenta que el sistema esta pensado para determinados usuarios (consultar acápite 2.3.1) y que actualmente en la universidad son solo 2 personas que cumplen con las especificaciones para poder tener acceso al diseñador, además de un administrador del sistema, se

puede asumir entonces que este resultado es satisfactorio, ya que 100 es 33.3 veces mayor que 3, lo que indica que el sistema es capaz de soportar 33.3 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 100, mayor será la probabilidad de ocurrencia de errores. Por tanto, se puede considerar que el límite permisible de procesamiento de peticiones concurrentes es de 100.

3.3.4. Pruebas de aceptación

Las pruebas de aceptación son básicamente pruebas funcionales sobre el sistema completo, ya que tienen el objetivo de comprobar que se satisfacen los requisitos establecidos. Su ejecución es facultativa del cliente y en el caso de que no se realicen explícitamente, se dan por incluidas dentro de las pruebas de sistema. Las pruebas de aceptación son, a menudo, responsabilidad del usuario o del cliente, aunque cualquier persona involucrada en el negocio puede realizarlas. Estas se realizan a través de las técnicas alfa y beta. La técnica alfa son realizadas por el usuario con el desarrollador como observador en un entorno controlado (simulación de un entorno de producción) y la beta son realizadas por el usuario en su entorno de trabajo y sin observadores. Para realizar las pruebas de aceptación a la solución se escogió la técnica alfa, tomando como referencia la especificación de requisitos y comprobando que el sistema cumple satisfactoriamente los requisitos del cliente, emitiéndose el acta de aceptación, garantizando evaluar el grado de calidad del software.

La caracterización de las técnicas de codificación empleadas en la implementación del diseñador de plantillas, permitió una mejor organización y comprensión del código. Con los resultados de validación de los requisitos identificados, haciendo uso de criterios para validar requisitos del cliente propuestos por el proceso de desarrollo de software utilizado y de técnicas de validación de requisitos, se logró obtener un listado de requisitos correctamente redactados y especificados. La descripción de las pruebas utilizadas para asegurar la calidad del software, permitió obtener resultados satisfactorios, asegurando que el diseñador de plantillas implementado no contiene errores y tiene la aceptación requerida.

CONCLUSIONES

Una vez finalizada la fundamentación teórica que sustentó la presente investigación, definidas las características del diseñador de plantillas y efectuado su desarrollo y validación, se obtuvieron resultados que permiten arribar a las siguientes conclusiones:

- El análisis realizado a diferentes diseñadores de plantillas en el mundo y en la UCI, demostró que los sistemas estudiados no están en correspondencia con las políticas de migración a software libre de la universidad y del país. Además se ajustan a las necesidades y características de las instituciones que las desarrollaron o para las que fueron desarrolladas y no están orientadas a todo tipo de usuarios, aunque poseen características que sirvieron como base para elaborar la solución.
- Con la aplicación del proceso de desarrollo de software con enfoque ágil basado en el nivel 2 de CMMI, se logró obtener y especificar los requisitos, así como elaborar los artefactos propuestos y el diseño de la solución.
- Con la implementación del diseñador de plantillas, se logró personalizar la salida de los reportes, a través de una interfaz de fácil uso, integrando al SGAP una herramienta que complementa el trabajo del módulo Reportes.
- Una vez realizada las pruebas de software pertinentes se logró una aplicación que cumple con todos los requisitos y que satisface las necesidades del cliente.
- Se contribuyó a que los usuarios puedan diseñar plantillas diferentes a las que actualmente genera el SGAP, se tributó a un mejor aprovechamiento del tiempo de trabajo del especialista en función de la evolución del software y se logró una mayor accesibilidad a la información manejada referente al proceso de formación de los estudiantes.

RECOMENDACIONES

- Desarrollar futuras versiones del diseñador de plantillas del SGAP que incorporen nuevas funcionalidades que perfeccionen el trabajo con el mismo como:
  - Eliminar filas y columnas en una tabla.
  - Visualizar la plantilla en tiempo de diseño.
  - Agregar nuevas métricas de las fuentes de texto en el FOP[29].
  - Adicionar nuevas propiedades a los componentes como: cambiar el color de fondo en las tablas, poner en negrita, cursiva o subrayado a los textos.

[29] *Formatting Objects Processor* es una aplicación Java (formateador de impresión) que convierte archivos *XSL Formatting Objects* (XSL-FO) a PDF u otros formatos imprimibles.

BIBLIOGRAFÍA REFERENCIADA

1. GUTIÉRREZ, R. H. G. Y. J. M. G. Conocimiento, innovación y desarrollo. San José, Costa Rica 2011, vol. 1ra edición, nº p. 290. Disponible en: http://catedrainnovacion.ucr.ac.cr/librocid.pdf.
2. HERNÁNDEZ, H. La importancia de un Sistema de Información en las empresas. 5 de diciembre de 2012 2012, nº Disponible en: http://hectorhernandezadm.blogspot.com/.
3. MEDINA, A. J. A. Generador dinámico de reportes versión 1.8.0. 2012, nº Disponible en: http://publicaciones.uci.cu/index.php/SC/article/view/889.
4. ABC, D. Comunicación-Reporte. 1 de noviembre de 2012 2012, nº Disponible en: http://www.definicionabc.com/comunicacion/reporte.php.
5. MOLINA, R. T. "Generador dinámico de reportes para la gestión académica de Pregrado". La Habana, Junio, 2012, nº
6. DEFINICIÓN.DE. Definición de reporte. 25 de enero de 2013 2008, nº Disponible en: http://definicion.de/reporte.
7. HERNANDEZ, I. A. S. Generador Automático de Reportes Dinámicos. 20 de enero de 2013 2003, nº Disponible en: http://www.cs.cinvestav.mx/TesisGraduados/2003/tesisSilvaHernandez.pdf.
8. CORPORATION, M. Crystal Reports 24 de enero de 2013 2013, nº Disponible en: http://msdn.microsoft.com/es-s/library/aa287920%28VS.71%29.aspx.
9. DOCUMENTATION, B. O. Manual del usuario de Crystal Reports XI. 23 de enero de 2013 2005, nº Disponible en: http://www.wiener.edu.pe/manuales2/5tociclo/PROGRAMACIONVISUAL4/ManualCrystalReports11XI.pdf.
10. GRAPECITY. ASP.NET ActiveReports GrapeCity. .NET Reporting Tool for Silverlight, WinForms. 25 de enero de 2013 2013, nº Disponible en: http://www.datadynamics.com/Products/ActiveReports/Overview.aspx.
11. IREPORT. iReport. 30 de enero de 2013 2013, nº Disponible en: http://jasperforge.org/projects/ireport.
12. LABORDE, M. Generador Dinámico de Reportes V 2.0: Análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes. 2011, nº
13. VIDAL, Y. G. Documento de Arquitectura de Software del proyecto. 5 de febrero de 2013 nº
14. INC, C. E. L. CODEIGNITER. 30 de enero de 2013 2013, nº Disponible en: http://codeigniter.com/.
15. PROJECT, J. T. J. The jQuery Project. 3 de febrero de 2013 2010, nº Disponible en: http://docs.jquery.com/Release:jQuery_1.3.2.
16. GROUP, P. G. D. A brief history of PostgreSQL. 2 de febrero de 2013 2013, nº Disponible en: http://www.postgresql.org/docs/8.2/interactive/intro-whatis.html.

17. CIBERAULA. Una introducción a Apache. 4 de febrero de 2013 2010, nº Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro/.
18. AFFILIATES, O. C. A. O. I. ¿Qué es NetBeans? . 3 de febrero de 2013 2012, nº Disponible en: http://pencil.evolus.vn/.
19. NORIEGA, J. 2 de febrero de 2013 2010, nº Disponible en: http://noriegatic.blogspot.com/2010/03/postgre-sql.html.
20. PARADIGM, V. UML tool, business process modeler and database designer for software development team. 3 de febrero de 2013 nº Disponible en: http://www.visual-paradigm.com.
21. PROJECT, P. Evolus Pencil. 3 de febrero de 2013 2012, nº Disponible en: http://pencil.evolus.vn/.
22. ESTEVE, J. J. Administración avanzada de GNU/Linux. 3 de febrero de 2013 2004, nº Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/Administracin_avanzada_de_GNU_Linux.pdf.
23. WWW.NOSOLOUNIX.COM. GNU Linux Android, software libre, tecnología y mucho más. 18 de marzo de 2013 2010, nº Disponible en: http://www.nosolounix.com/2010/02/herramientas-test-software.html.
24. MARIN, M. D. A. Definición de lenguaje de programación. . 15 de febrero de 2013 2008, vol. catedraprogramacion, nº Disponible en: http://catedraprogramacion.foroactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.
25. PHP, S. O. PHP. 4 de febrero de 2013 2007, nº Disponible en: http://www.php.net/archive/2007.php.
26. MAESTROSDELWEB. ¿Qué es JavaScript? 4 de febrero de 2013 nº Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.
27. LIBROSWEB.ES. Introducci'on a CSS. 3 de febrero de 2013 2013, nº Disponible en: http://www.librosweb.es/css/.
28. ECURED. HTML. 4 de febrero de 2013 nº Disponible en: http://www.ecured.cu/index.php/HTML.
29. WALSH, N. A Technical Introduction to XML. 3 de febrero de 2013 1998, nº Disponible en: http://www.xml.com/pub/a/98/10/guide0.html.
30. WEB, U. XSL. 3 de febrero de 2013 2012, nº Disponible en: http://www.w3.org/Style/XSL/.
31. ÁLVAREZ, R. Qué es y para qué sirve SQL. 4 de febrero de 2013 2001, nº Disponible en: http://www.desarrolloweb.com/articulos/262.php.
32. SCHMULLER, J. UML en 24 Horas. 3 de febrero de 2013 nº Disponible en: http://www.scribd.com/doc/38392190/UML-en-24-Horas.
33. FRANCISCO RUIZ, U. C. Ingeniería de Software 1. 5 de febrero de 2012 nº Disponible en: http://www.ctr.unican.es/asignaturas/is1/is1-t02-trans.pdf.

34. MSDA. Documento Metodologías de Desarrollo de Software Ágiles. 5 de febrero de 2013 nº Disponible en: http://www.google.com.cu/url?sa=t&rct=j&q=Metodologias+agiles&source=web&cd=2&cad=rja&ved=0CDcQFjAB&url=http%3A%2F%2Finterfaces-siges.googlecode.com%2Ffiles%2FMetodologias%2520Agiles.doc&ei=Ci4RUaOGAeyP0QGkm4HoCg&usg=AFQjCNGMUGSqvAEuNie9qsc_fLNWw1qW_A&bvm=bv.41867550,d.dmQ.
35. LUIS CALABRIA, P. P. Metodología XP. *Cátedra de Ingeniería de Software,Universidad ORT, Uruguay*, 15 de febrero de 2013 2003, nº Disponible en: http://fi.ort.edu.uy/innovaportal/file/2021/1/metodologia_xp.pdf.
36. PERALTA, A. Metodología Scrum. *Cátedra de Ingeniería de Software,Universidad ORT, Uruguay*, 15 de febrero de 2013 2003, nº Disponible en: http://fi.ort.edu.uy/innovaportal/file/2021/1/scrum.pdf.
37. CONTRERAS, M. F. Modelo CMMI. 15 de febrero de 2013 2008, nº Disponible en: http://www.monografias.com/trabajos56/modelo-cmmi/modelo-cmmi.shtml.
38. CASTRO, I. E. L. Propuesta para la integración de prácticas de las metodologías XP y Scrum con el proceso de administración de requisitos del nivel 2 de CMMI. 2013, nº
39. ESTRADA, D. A. F. Indicación 0505 de la vicerectoría de Producción, Ciclo de vida de proyectos de desarrollo de software. 1 de marzo de 2013 2013, nº
40. SYNERGIX, T. Y. Modelo de dominio. 4 de febrero de 2013 2011, nº Disponible en: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.
41. PRESSMAN, R. S. Ingeniería del software, un enfoque práctico 5ta edición. nº
42. CLEMENTS, P. “A Survey of Architecture Description Languages”. Proceedings of the International Workshop on Software Specification and Design. 1996, nº
43. E. GAMMA, R. H., R. JOHNSON, AND J. VLISSIDES. Desing Patterns. 1995, nº
44. EVA. Patrones de diseño de base de datos. 28 de abril de 2013 2013, nº Disponible en: http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf.

BIBLIOGRAFÍA CONSULTADA

1- COPE, Rop, OpenLoig, 2013, Comparing Open Source Reporting Tools for Use in the Enterprise (9 de octubre de 2008), [En línea] [ref. de 28 de enero de 2013], Disponible en web: http://www.openlogic.com/wazi/bid/188176/Comparing-Open-Source-Reporting-Tools-for-Use-in-the-Enterprise.

2- ESPINOSA, Roberto, Reporting en Pentaho con Pentaho Report Designer (5 de Julio de 2010), [En línea] [ref. de 28 de enero de 2013], Disponible en web:http://churriwifi.wordpress.com/2010/07/15/17-4-reporting-en-pentaho-con-pentaho-report-designer-otras-posibilidades-de-reporting-birt-y-jasperreports/.

3- Introducción a jasperreports e ireport (primera parte). [En línea] 2010. [ref. de 30 de enero de 2013.] Disponible en web:http://www.mygnet.net/articulos/java/introduccion_a_jasperreports_e_ireport_primera_parte.301/Pagina/2.

4- Manual for iReport. [En línea] 2013 [ref. de 30 de enero de 2013] Disponible en web:http://ireport.sourceforge.net/manual0.2.0.html#1.0.

5- Scribd.2013. [En línea] 2013 [ref. de 28 de enero de 2013] Disponible en web:http://es.scribd.com/doc/81591886/Manual-de-iReport.

6- SOMERVILLE, Ian. Ingeniería del Software. [En línea]. 7a. ed. Madrid: Pearson Education S.A, 2005. 661p. [En línea], [ref. de 3 de febrero de 2013]. Disponible en web:http://books.google.com.cu/books?id=gQWd49zSut4C&pg=PA80&dq=Herramientas+Case&hl=es&sa=X&ei=uOG3T-_cE8jpgged2PSiCg&ved=0CEQQ6AEwAw#v=onepage&q=Herramientas%20Case&f=false ISBN: 84-7829-074-5.

7- GUTIÉRREZ, R. H. G. Y. J. M. G. Conocimiento, innovación y desarrollo. San José, Costa Rica 2011, vol. 1ra edición, nº p. 290. Disponible en: http://catedrainnovacion.ucr.ac.cr/librocid.pdf.

8- HERNÁNDEZ, H. La importancia de un Sistema de Información en las empresas. 5 de diciembre de 2012 2012, nº Disponible en: http://hectorhernandezadm.blogspot.com/.

9- MEDINA, A. J. A. Generador dinámico de reportes versión 1.8.0. 2012, nº Disponible en: http://publicaciones.uci.cu/index.php/SC/article/view/889.

10- ABC, D. Comunicación-Reporte. 1 de noviembre de 2012 2012, nº Disponible en: http://www.definicionabc.com/comunicacion/reporte.php.

11- MOLINA, R. T. “Generador dinámico de reportes para la gestión académica de Pregrado”. La Habana, Junio, 2012, nº.

12- DEFINICIÓN.DE. Definición de reporte. 25 de enero de 2013 2008, nº Disponible en: http://definicion.de/reporte.

13- HERNANDEZ, I. A. S. Generador Automático de Reportes Dinámicos. 20 de enero de 2013 2003, nº Disponible en: http://www.cs.cinvestav.mx/TesisGraduados/2003/tesisSilvaHernandez.pdf.

14- CORPORATION, M. Crystal Reports 24 de enero de 2013 2013, nº Disponible en: http://msdn.microsoft.com/es-s/library/aa287920%28VS.71%29.aspx.

15- DOCUMENTATION, B. O. Manual del usuario de Crystal Reports XI. 23 de enero de 2013 2005, nº Disponible en: http://www.wiener.edu.pe/manuales2/5tociclo/PROGRAMACIONVISUAL4/ManualCrystalReports11XI.pdf.

16- GRAPECITY. ASP.NET ActiveReports GrapeCity. .NET Reporting Tool for Silverlight, WinForms. 25 de enero de 2013 2013, nº Disponible en: http://www.datadynamics.com/Products/ActiveReports/Overview.aspx.

17- IREPORT. iReport. 30 de enero de 2013 2013, nº Disponible en: http://jasperforge.org/projects/ireport.

18- LABORDE, M. Generador Dinámico de Reportes V 2.0: Análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes. 2011, nº.

19- VIDAL, Y. G. Documento de Arquitectura de Software del proyecto. 5 de febrero de 2013 nº.

20- INC, C. E. L. CODEIGNITER. 30 de enero de 2013 2013, nº Disponible en: http://codeigniter.com/.

21- PROJECT, J. T. J. The jQuery Project. 3 de febrero de 2013 2010, nº Disponible en: http://docs.jquery.com/Release:jQuery_1.3.2.

22- GROUP, P. G. D. A brief history of PostgreSQL. 2 de febrero de 2013 2013, nº Disponible en: http://www.postgresql.org/docs/8.2/interactive/intro-whatis.html.

23- CIBERAULA. Una introducción a Apache. 4 de febrero de 2013 2010, nº Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro/.

24- MARIN, M. D. A. Definición de lenguaje de programación. . 15 de febrero de 2013 2008, vol. catedraprogramacion, nº Disponible en: http://catedraprogramacion.foroactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.

25- AFFILIATES, O. C. A. O. I. ¿Qué es NetBeans? . 3 de febrero de 2013 2012, nº Disponible en: http://pencil.evolus.vn/.

26- NORIEGA, J. 2 de febrero de 2013 2010, nº Disponible en: http://noriegatic.blogspot.com/2010/03/postgre-sql.html.

27- PARADIGM, V. UML tool, business process modeler and database designer for software development team. 3 de febrero de 2013 nº Disponible en: http://www.visual-paradigm.com.

28- PROJECT, P. Evolus Pencil. 3 de febrero de 2013 2012, nº Disponible en: http://pencil.evolus.vn/.

29- ESTEVE, J. J. Administración avanzada de GNU/Linux. 3 de febrero de 2013 2004, nº Disponible en:http://sunshine.prod.uci.cu/gridfs/sunshine/books/Administracin_avanzada_de_GNU_Linux.pdf.

30- WWW.NOSOLOUNIX.COM. GNU Linux Android, software libre, tecnología y mucho más. 18 de marzo de 2013 2010, nº Disponible en: http://www.nosolounix.com/2010/02/herramientas-test-software.html.

31- PHP, S. O. PHP. 4 de febrero de 2013 2007, nº Disponible en: http://www.php.net/archive/2007.php.

32- MAESTROSDELWEB. ¿Qué es JavaScript? 4 de febrero de 2013 nº Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

33- LIBROSWEB.ES. Introducci'on a CSS. 3 de febrero de 2013 2013, nº Disponible en: http://www.librosweb.es/css/.

34- ECURED. HTML. 4 de febrero de 2013 nº Disponible en: http://www.ecured.cu/index.php/HTML.

35- LUIS CALABRIA, P. P. Metodología XP. *Cátedra de Ingeniería de Software,Universidad ORT, Uruguay*, 15 de febrero de 2013 2003, nº Disponible en: http://fi.ort.edu.uy/innovaportal/file/2021/1/metodologia_xp.pdf.

36- PERALTA, A. Metodología Scrum. *Cátedra de Ingeniería de Software,Universidad ORT, Uruguay*, 15 de febrero de 2013 2003, nº Disponible en: http://fi.ort.edu.uy/innovaportal/file/2021/1/scrum.pdf.

37- WALSH, N. A Technical Introduction to XML. 3 de febrero de 2013 1998, nº Disponible en: http://www.xml.com/pub/a/98/10/guide0.html.

38- CONTRERAS, M. F. Modelo CMMI. 15 de febrero de 2013 2008, nº Disponible en: http://www.monografias.com/trabajos56/modelo-cmmi/modelo-cmmi.shtml.

39- ÁLVAREZ, R. Qué es y para qué sirve SQL. 4 de febrero de 2013 2001, nº Disponible en: http://www.desarrolloweb.com/articulos/262.php.

40- SCHMULLER, J. UML en 24 Horas. 3 de febrero de 2013 nº Disponible en: http://www.scribd.com/doc/38392190/UML-en-24-Horas.

41- FRANCISCO RUIZ, U. C. Ingeniería de Software 1. 5 de febrero de 2012 nº Disponible en: http://www.ctr.unican.es/asignaturas/is1/is1-t02-trans.pdf.

42- MSDA. Documento Metodologías de Desarrollo de Software Ágiles. 5 de febrero de 2013 nº Disponible en: http://www.google.com.cu/url?sa=t&rct=j&q=Metodologias+agiles&source=web&cd=2&cad=rja&ved=0CDcQFjAB&url=http%3A%2F%2Finterfaces-siges.googlecode.com%2Ffiles%2FMetodologias%2520Agiles.doc&ei=Ci4RUaOGAeyP0QGkm4HoCg&usg=AFQjCNGMUGSqvAEuNie9qsc_fLNWw1qW_A&bvm=bv.41867550,d.dmQ.

43- CASTRO, I. E. L. Propuesta para la integración de prácticas de las metodologías XP y Scrum con el proceso de administración de requisitos del nivel 2 de CMMI. 2013, nº.

44- ROMERO, G. M. P. Metodología ágil para proyectos de software libre. Junio 2008, Ciudad de La Habana 2008, nº.

45- ESTRADA, D. A. F. Indicación 0505 de la vicerectoría de Producción, Ciclo de vida de proyectos de desarrollo de software. 1 de marzo de 2013 2013, nº.

46- SYNERGIX, T. Y. Modelo de dominio. 4 de febrero de 2013 2011, nº Disponible en: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

47- CLEMENTS, P. "A Survey of Architecture Description Languages". Proceedings of the International Workshop on Software Specification and Design. 1996, nº.

48- BAHIT, E. El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC. 15 de febrero de 2013 nº Disponible en:http://www.monografias.com/trabajos89/poo-y-mvc-php/poo-y-mvc-php2.shtml.

49- E. GAMMA, R. H., R. JOHNSON, AND J. VLISSIDES. Desing Patterns. 1995, nº.

50- EVA. Patrones de diseño de base de datos. 28 de abril de 2013 2013, nº Disponible en: http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf

51- PRESSMAN, R. S. Ingeniería del software, un enfoque práctico 5ta edición. nº.

GLOSARIO DE TÉRMINOS

Fuente de información: diversos tipos de documentos que contienen datos útiles para satisfacer una demanda de información o conocimiento, o los recursos necesarios para poder acceder a la información y al conocimiento en general.

Flujo: secuencia de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan y cuál es su orden de ejecución.

Información: es un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto o sistema que recibe dicho mensaje.

Indentación: es un anglicismo (de la palabra inglesa *indentation*) de uso común en informática que significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente, lo que en el ámbito de la imprenta se ha denominado siempre como sangrado o sangría.

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

Multiplataforma: término usado para referirse a la característica de aplicaciones de ejecutarse en diversas plataformas o sistemas operativos.

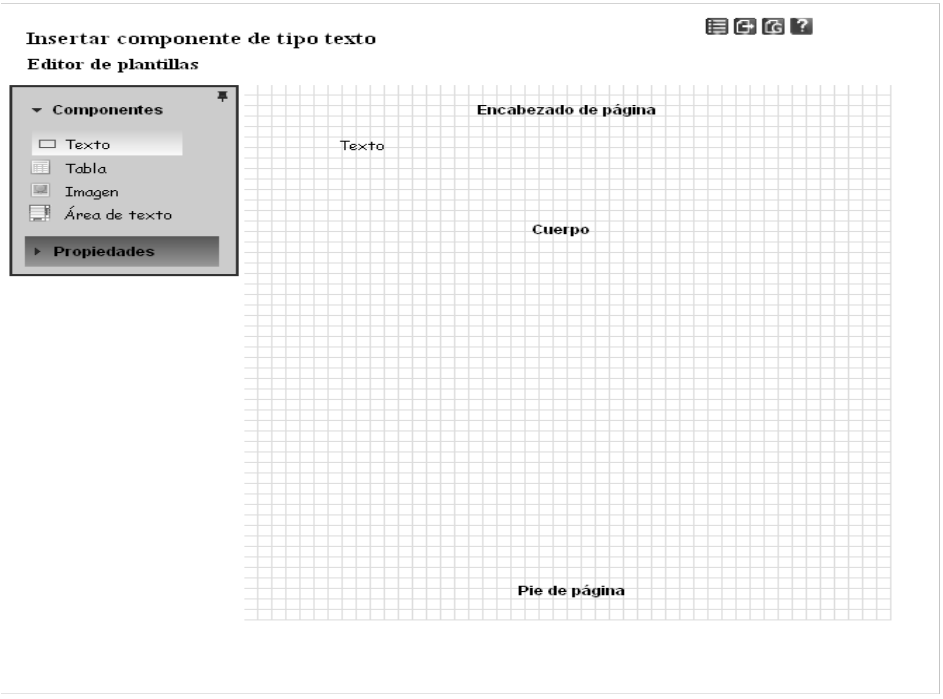
Plantilla: de dispositivo o de interfaz, que suele proporcionar una separación entre la forma o estructura y el contenido. Es un medio o aparato o sistema, que permite guiar, portar, o construir, un diseño o esquema predefinido.

Proceso: conjunto de actividades relacionadas lógicamente que producen una salida o resultado.

Renderizado, renderización: es una adaptación al castellano del vocablo inglés *"rendering"* y que define un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen o secuencia de imágenes.

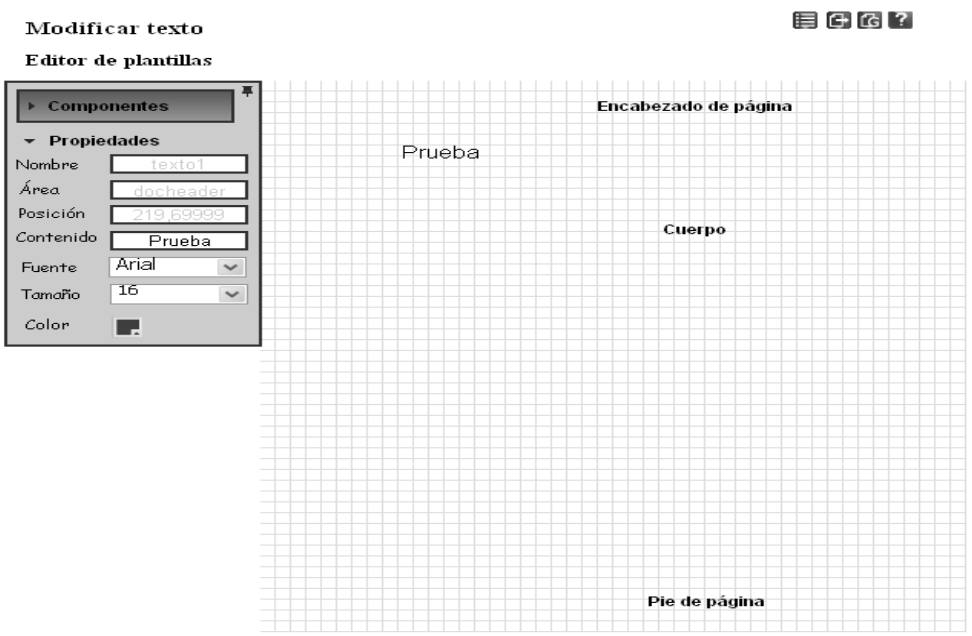
ANEXOS

Anexo 1: Especificación de requisitos

<table>
<tr><th>Nº</th><th>Nombre</th><th>Descripción</th><th>Complejidad</th><th>Prioridad para cliente</th></tr>
<tr><td rowspan="6">RF1.</td><td>Insertar componente de tipo texto.</td><td>Permite insertar un componente de tipo texto en el área de diseño de la plantilla al dar clic encima del componente y luego en el área de diseño en cualquiera de las regiones (Encabezado de página, Cuerpo o Pie de página), tomando por defecto los valores de las propiedades del texto (Nombre, Área, Posición, Contenido, Fuente, Tamaño, Color), además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.</td><td>Alta</td><td>Alta</td></tr>
<tr><td colspan="4">Prototipo</td></tr>
<tr><td colspan="4"> 

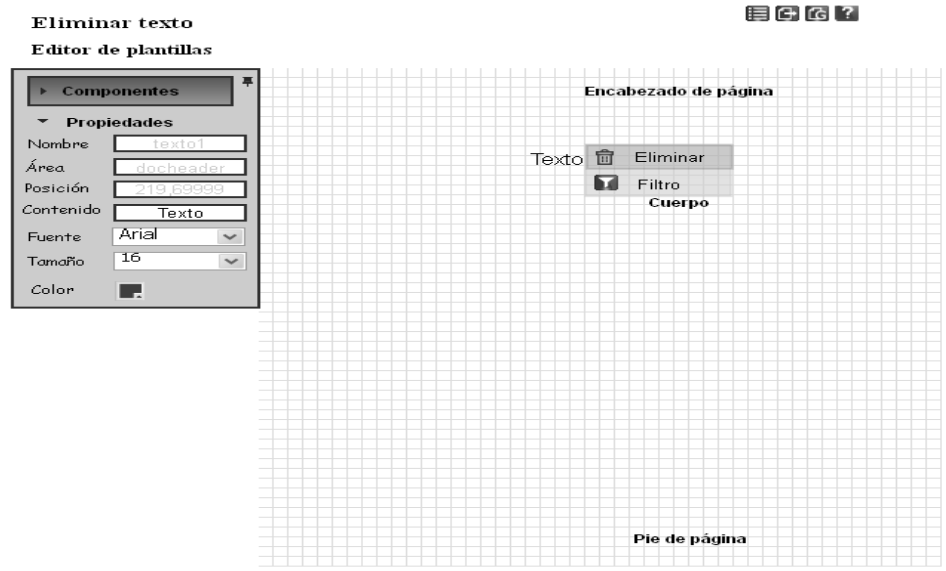
Prototipo Insertar componente de tipo texto.</td></tr>
<tr><td>Campos</td><td>Tipos de Datos</td><td colspan="2">Reglas o Restricciones</td></tr>
<tr><td>Nombre</td><td>character varying</td><td colspan="2">De solo lectura.</td></tr>
<tr><td>Área</td><td>character varying</td><td colspan="2">De solo lectura.</td></tr>
</table>

	Posición	integer	De solo lectura.	
	Contenido	character varying	Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos para una palabra es de 30. La cantidad de caracteres permitidos es 100.	
	Fuente	character varying	Campo de selección	
	Tamaño	integer	Solo dígitos del 8 al 32	
	Color	character varying	Hasta 3 caracteres	
	Observaciones	*2.* Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.		
RF2.	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Modificar texto.	Permite modificar las propiedades (*Nombre, Área*, *Posición, Contenido, Fuente, Tamaño, Color*) del componente de tipo texto en el área de diseño de la región donde fue creado (Encabezado de página, Cuerpo o Pie de página), además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Alta
	Prototipo			



Prototipo Modificar texto.

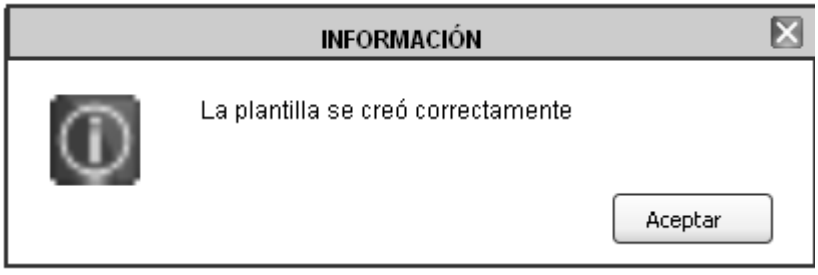
Campos	**Tipos de Datos**	**Reglas o Restricciones**
Nombre	character varying	De solo lectura.
Área	character varying	De solo lectura.
Posición	integer	De solo lectura.
Contenido	character varying	Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos para una palabra es de 30. La cantidad de caracteres permitidos es 100. Campo obligatorio.
Fuente	character varying	Campo de selección
Tamaño	integer	Solo dígitos del 8 al 32. Campo obligatorio.
Color	character varying	Hasta 3 caracteres.
Observaciones	1. Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2. En caso de que se introduzcan datos incorrectos (porciento o comillas simples) no permite crear el elemento y se muestra el mensaje en rojo encima del componente: “No se admiten apóstrofo (') o porciento (%)".	

<table>
<tr><td></td><td></td><td colspan="3">3. Si se introduce un tamaño del texto fuera del rango de 8 a 32 o se introduce otro tipo de caracteres, el sistema muestra un mensaje en rojo encima del componente: "Un número entre 8 y 32."
4. Si se introduce menos de 2 caracteres, el sistema muestra un mensaje en rojo encima del componente: "Entre al menos 2 caracteres".
5. Si se introduce más caracteres de los permitidos para una palabra el sistema muestra el mensaje en rojo encima del componente: “Ha excedido el número de letras permitidas para una palabra”.
6. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.
7. En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo “Campo requerido” en el campo que debe ser llenado obligatorio.</td></tr>
<tr><td rowspan="4">RF3.</td><td>Nombre</td><td>Descripción</td><td>Complejidad</td><td>Prioridad para cliente</td></tr>
<tr><td>Eliminar texto.</td><td>Permite eliminar el texto del área de diseño de la plantilla, dando clic derecho sobre el texto y seleccionando la opción eliminar.</td><td>Media</td><td>Alta</td></tr>
<tr><td colspan="4">Prototipo</td></tr>
<tr><td colspan="4"> 

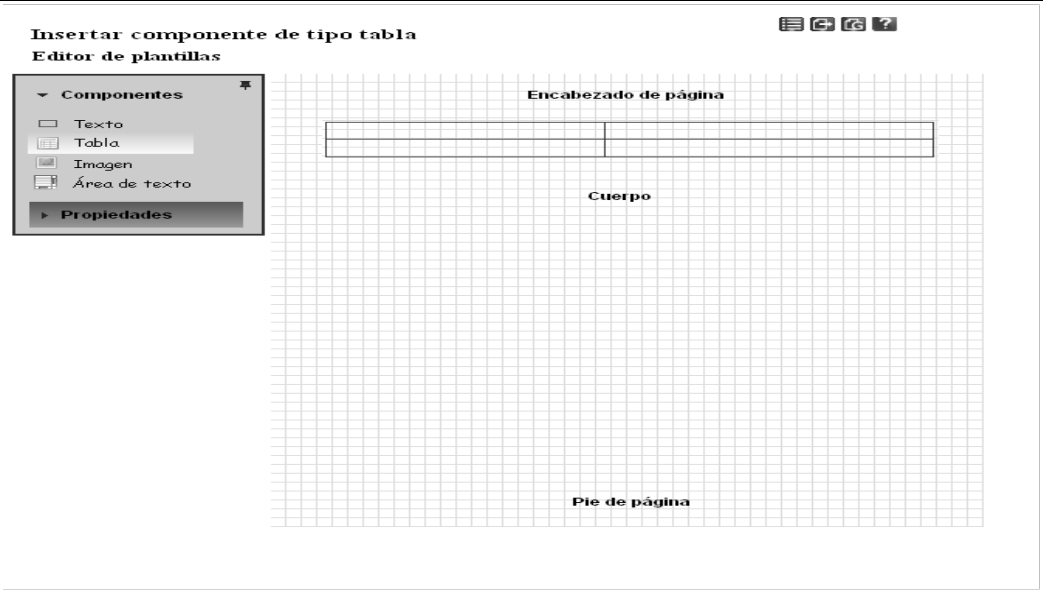
Prototipo Eliminar el texto.</td></tr>
<tr><td></td><td>Campos</td><td>Tipos de Datos</td><td colspan="2">Reglas o Restricciones</td></tr>
<tr><td></td><td>Observaciones</td><td colspan="3">1. Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.</td></tr>
</table>

		2. En caso de cancelar la acción se muestra un mensaje de advertencia "¿Está seguro de realizar la acción? ".		
RF4.	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Insertar componente de tipo tabla.	Permite insertar un componente de tipo tabla en el área de diseño de la plantilla al dar clic encima del componente, luego se activa una ventana, donde se seleccionan todos los valores para crear la tabla (*Columna, Fila, Borde, Área*) en cualquiera de las regiones (Encabezado de página, Cuerpo o Pie de página), tomando por defecto los valores de las propiedades de la tabla (*Nombre, Área, Posición, Arrastrar, Redimensionar*), y se muestra un mensaje de información "La tabla se creó correctamente", además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Alta
	Prototipo			

Prototipo 1 Insertar componente de tipo tabla.

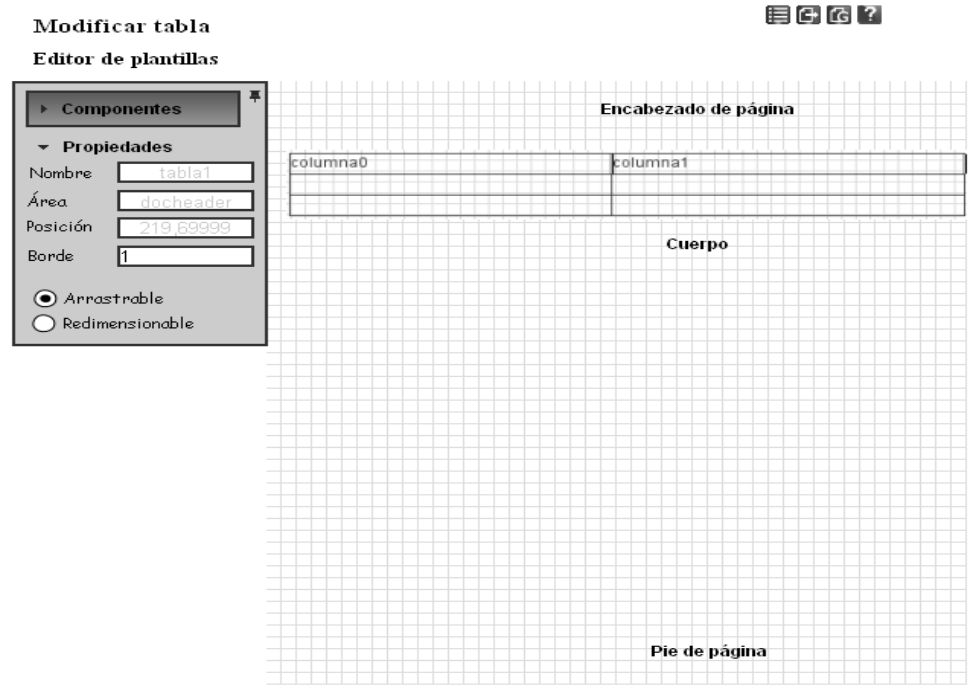


Prototipo 2 Insertar componente de tipo tabla.



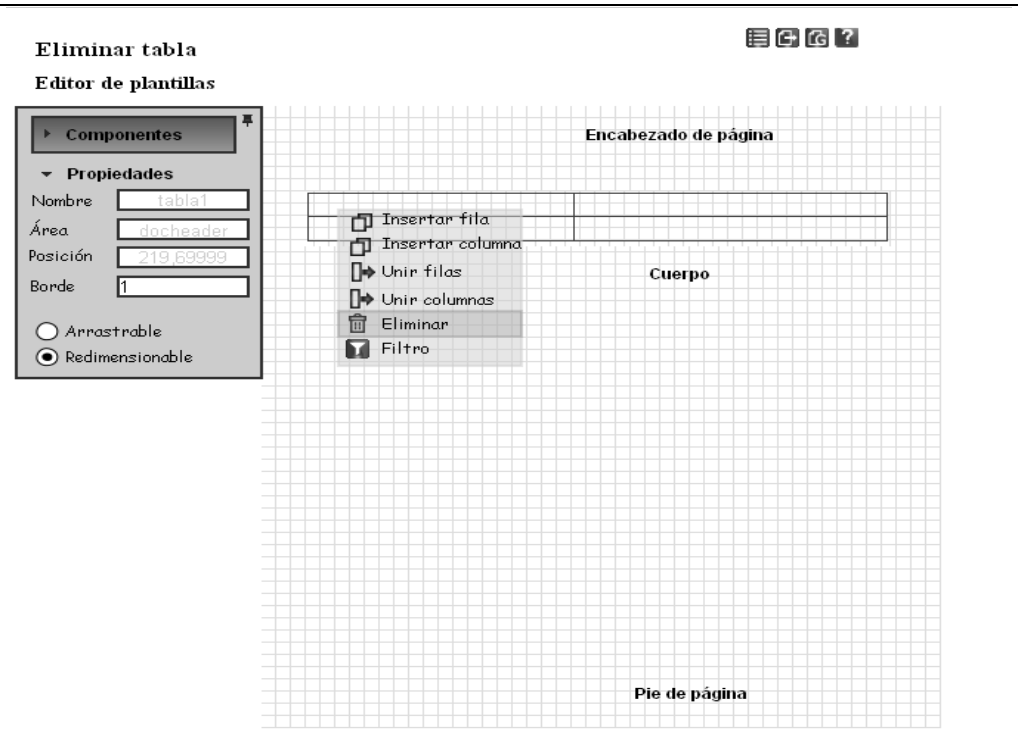
Prototipo 3 Insertar componente de tipo tabla.

Campos	Tipos de Datos	Reglas o Restricciones
Columna	integer	Campo de texto. Solo dígitos. Hasta 2 caracteres. Campo obligatorio.
Fila	integer	Campo de texto. Solo dígitos. Hasta 2 caracteres. Campo obligatorio.
Borde	integer	Campo de texto. Solo admite un 0 ó un 1. Campo obligatorio.
Posición	character varying	Lista desplegable.
Nombre	character varying	De solo lectura.
Área	character varying	De solo lectura.
Posición	integer	De solo lectura.
Arrastrar	booleano	Campo de selección.
Redimensionar	booleano	Campo de selección.
Observaciones	1. Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2. Cuando se crea la tabla el sistema muestra un mensaje de confirmación: “La tabla se creó correctamente”. 3. En caso de cancelar la acción se muestra un mensaje de advertencia “¿Está seguro de realizar la acción? “.	

		4. En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo “Campo requerido” en el campo que debe ser llenado obligatorio. 5. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no deja seguir entrando caracteres. 6. En caso de no introducir dígitos en el campo requerido el sistema muestra un mensaje de error: “Entre un número válido”. 7. En caso de entrar un valor diferente a 0 ó 1 en el campo borde el sistema muestra un mensaje en rojo encima del campo: “Solo admite un 0 ó un 1”.		
RF5.	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Modificar tabla.	Permite modificar las propiedades (*Nombre, Área, Posición, Borde, Arrastrar, Redimensionar*) de la tabla en el área de diseño de la región donde fue creado (Encabezado de página, Cuerpo o Pie de página), además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Alta
	Prototipo			
				

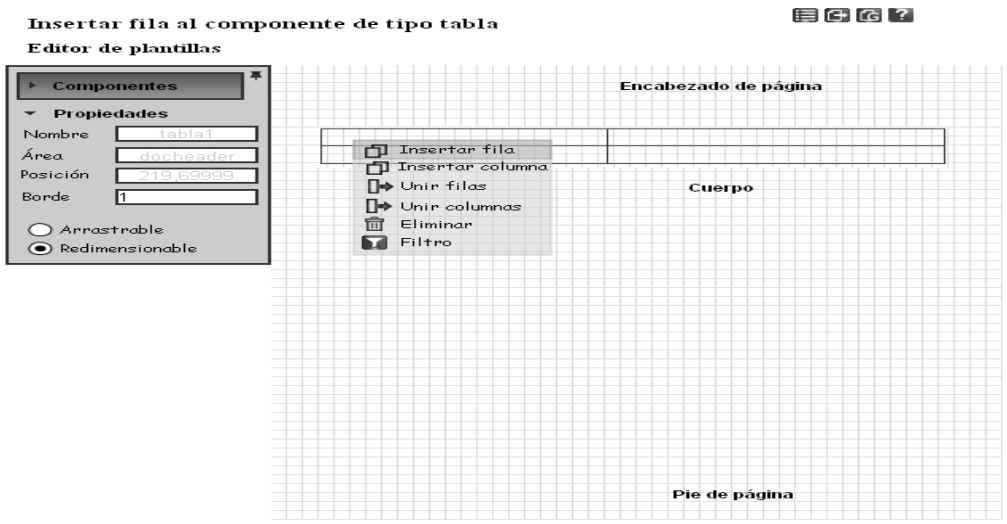
Prototipo Modificar tabla.

	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Nombre	character varying	De solo lectura.	
	Área	character varying	De solo lectura.	
	Posición	integer	De solo lectura.	
	Borde	character varying	Solo admite un 0 ó un 1. Campo obligatorio.	
	Arrastrar	booleano	Campo de selección.	
	Redimensionar	booleano	Campo de selección.	
	Observaciones	1. Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2. En caso de no introducir dígitos en el campo requerido el sistema muestra un mensaje de error: “Entre un número válido”. 3. En caso de entrar un valor diferente a 0 ó 1 en el campo borde el sistema muestra un mensaje en rojo encima del campo: “Solo admite un 0 ó un 1”. 4. En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo “Campo requerido” en el campo que debe ser llenado obligatorio.		
RF6	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Eliminar la tabla.	Permite eliminar la tabla del área de diseño de la plantilla, dando clic derecho sobre la tabla y seleccionando la opción eliminar.	Media	Alta
	Prototipo			

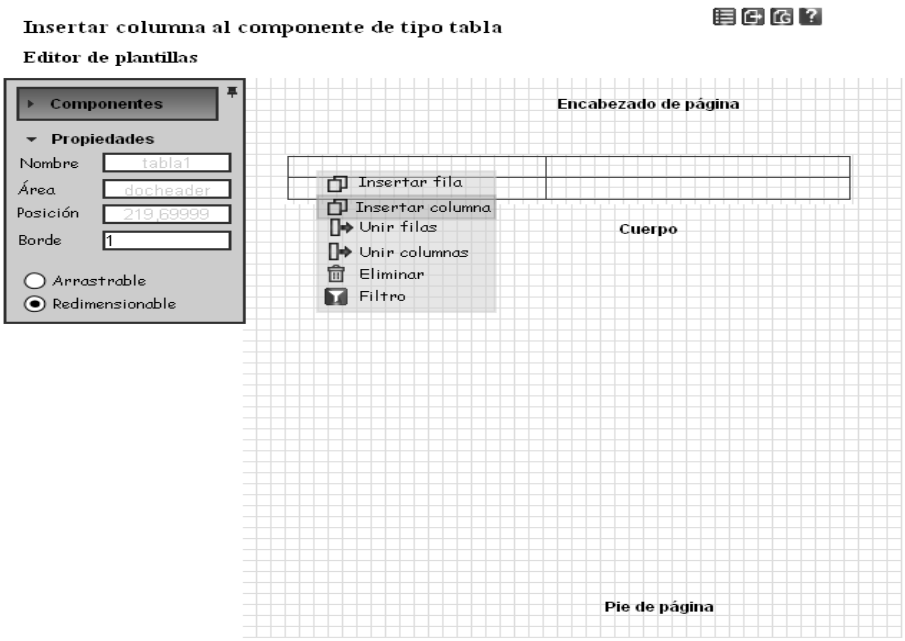


Prototipo Eliminar la tabla.

	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones	1. Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2. En caso de cancelar la acción se muestra un mensaje de advertencia "¿Está seguro de realizar la acción? "		
RF7	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Insertar columnas y filas al componente de tipo tabla.	Permite insertar una columna o fila al componente tabla, al dar clic derecho sobre la tabla y seleccionar la opción insertar fila o insertar columna, además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Alta
	Prototipo			

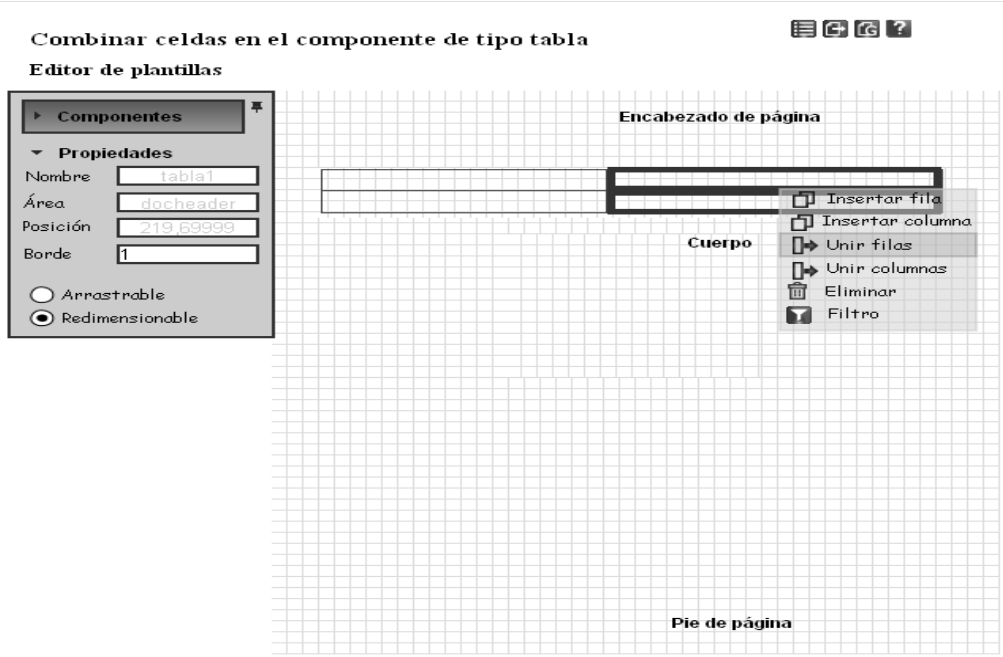


Prototipo 1 Insertar columnas y filas al componente de tipo tabla.

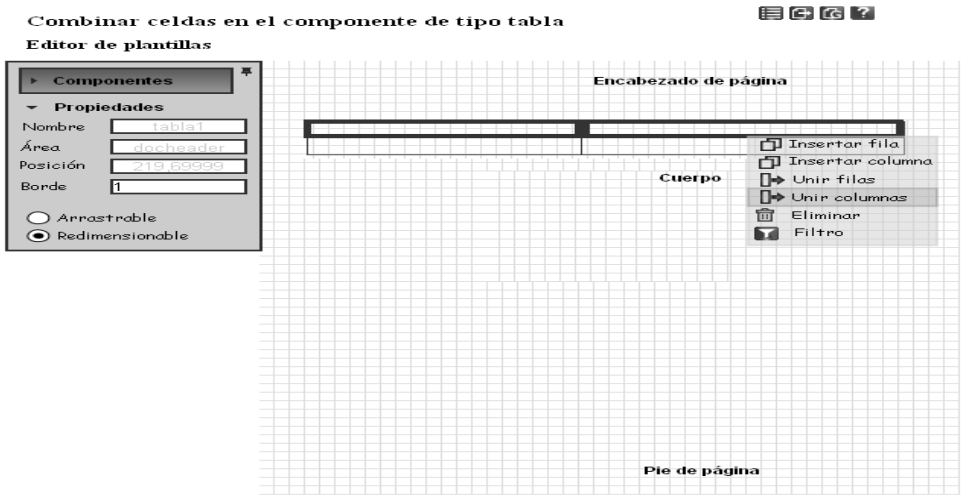


Prototipo 2 Insertar columnas y filas al componente de tipo tabla.

Campos	Tipos de Datos	Reglas o Restricciones
Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.	

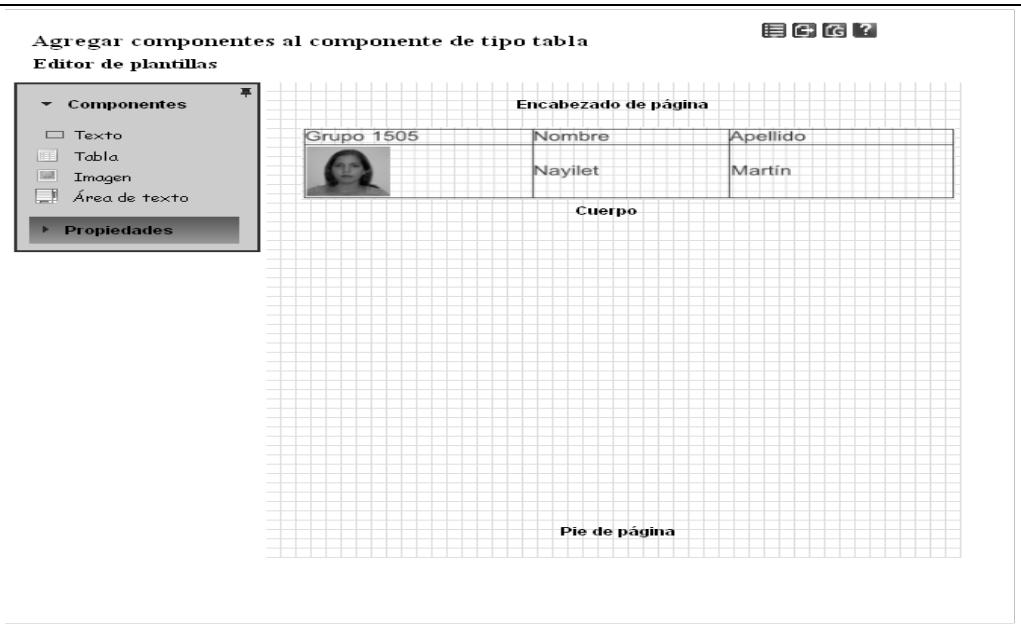
<table>
<tr><td rowspan="3">RF8</td><td>Nombre</td><td>Descripción</td><td>Complejidad</td><td>Prioridad para cliente</td></tr>
<tr><td>Combinar celdas en el componente de tipo tabla</td><td>Permite combinar celdas en el componente tabla, dando clic derecho sobre la tabla y seleccionar la opción Unir filas o Unir columnas, además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.</td><td>Alta</td><td>Alta</td></tr>
<tr><td colspan="4">Prototipo</td></tr>
<tr><td></td><td colspan="4"> 

Prototipo 1 Combinar celdas en el componente de tipo tabla.</td></tr>
</table>



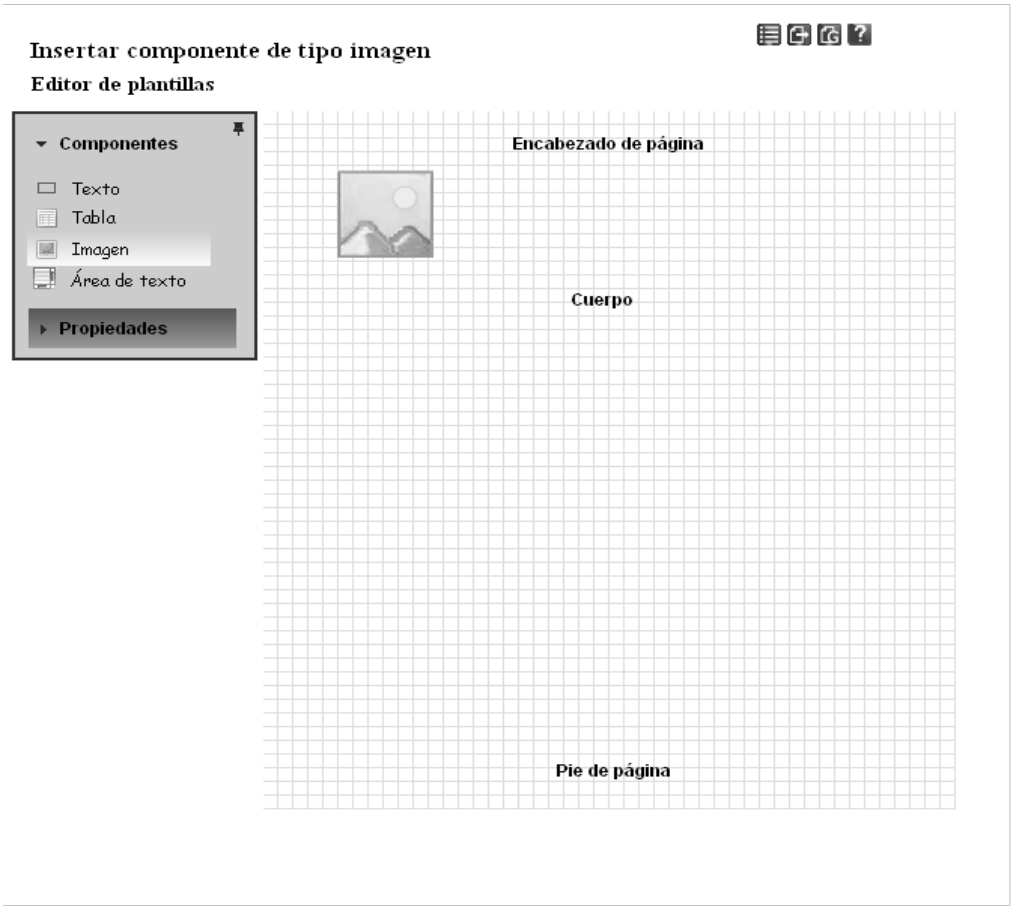
Prototipo 2 Combinar celdas en el componente de tipo tabla.

	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.		
RF9	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Agregar componentes al componente de tipo tabla	Permite insertar dentro de una tabla otros componentes (Texto, Imagen y área de texto), además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Media
	Prototipo			



Prototipo Agregar componentes al componente de tipo tabla.

	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.		
RF10	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Insertar componente de tipo imagen.	Permite insertar un componente de tipo imagen en el área de diseño de la plantilla al dar clic encima del componente y luego en el área de diseño en cualquiera de las regiones (Encabezado de página, Cuerpo o Pie de página), tomando por defecto los valores de las propiedades de la imagen (*Nombre, Área, Posición, Ancho, Alto, Ruta*), además se muestran las opciones: Ayuda y Exportar en el área de íconos flotantes.	Alta	Alta
	Prototipo			

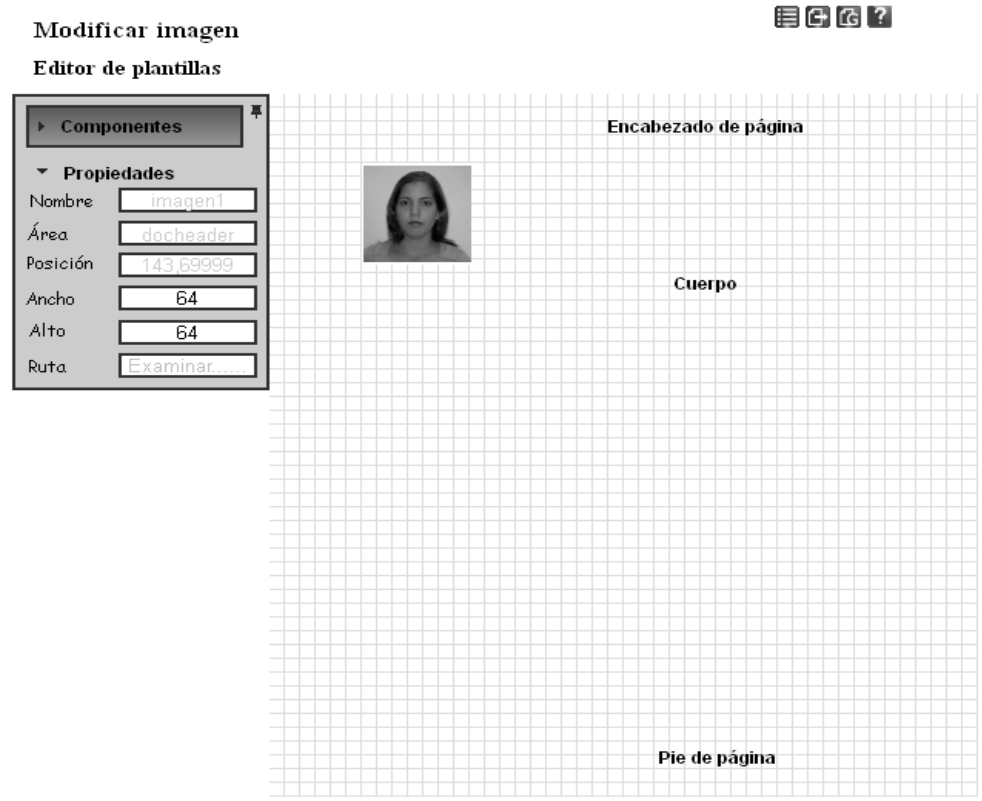


Prototipo Insertar componente de tipo imagen.

Campos	Tipos de Datos	Reglas o Restricciones
Nombre	character varying	De solo lectura.
Área	character varying	De solo lectura.
Posición	integer	De solo lectura.
Alto	integer	Solo dígitos y en un rango de 64 a 128 píxeles. Campo requerido.
Ancho	integer	Solo dígitos y en un rango de 64 a 128 píxeles. Campo requerido.
Ruta	Archivo adjunto	Campo de selección.
Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.	

RF11	Nombre	Descripción	Complejidad	Prioridad para cliente
	Modificar imagen	Permite modificar las propiedades (*Nombre, Área, Posición, Alto, Ancho, Ruta*) de la imagen en el área de diseño de la región donde fue creado (Encabezado de página, Cuerpo o Pie de página), además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Alta

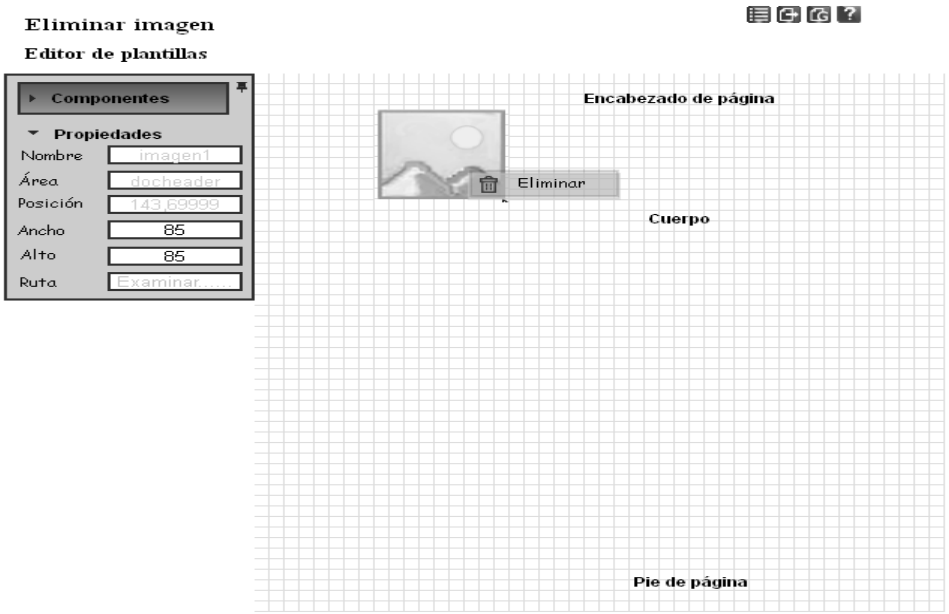
Prototipo

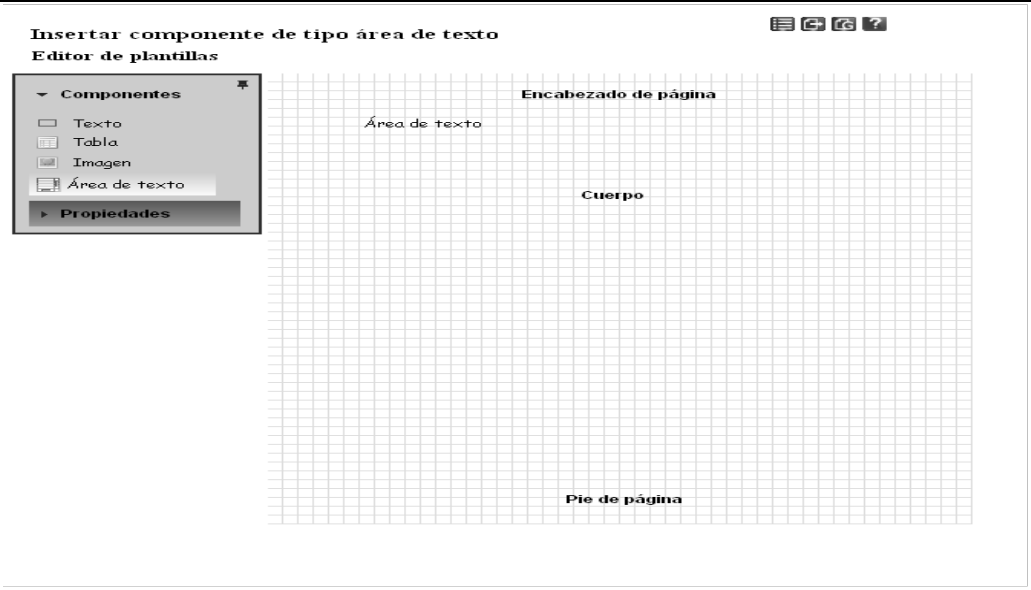


Prototipo Modificar imagen.

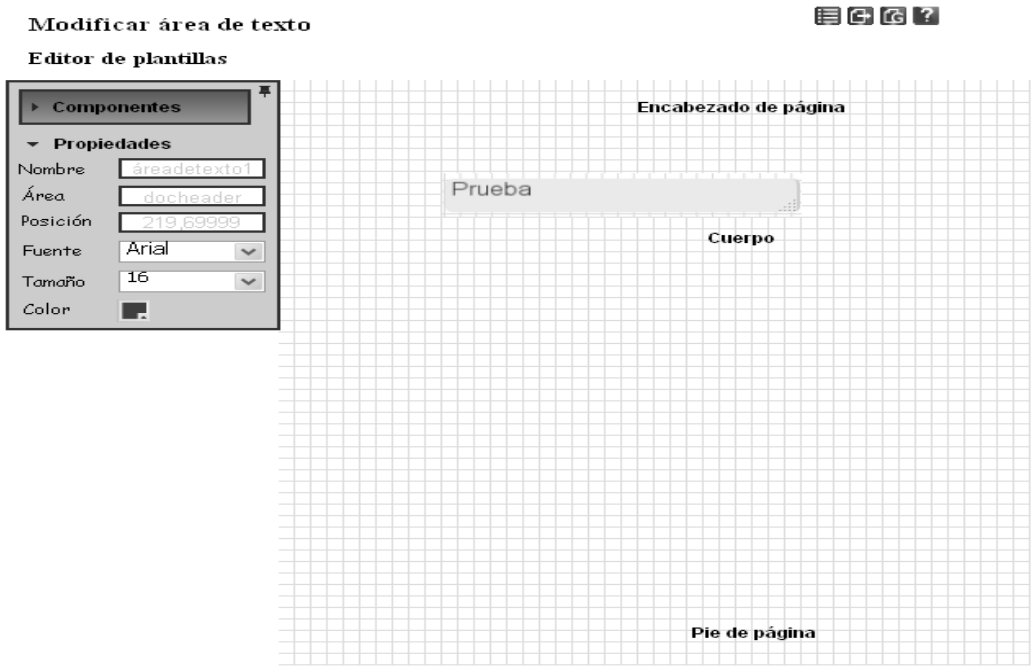
Campos	Tipos de Datos	Reglas o Restricciones
Nombre	character varying	De solo lectura.
Área	character varying	De solo lectura.

	Posición	integer	De solo lectura.	
	Alto	integer	Solo dígitos del 64 a 128 píxeles. Campo requerido.	
	Ancho	integer	Solo dígitos del 64 a 128 píxeles. Campo requerido.	
	Ruta	Archivo adjunto	Campo de selección.	
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2- En caso de no introducir dígitos en el campo requerido el sistema muestra un mensaje de error: “Entre un número válido”. 3- En caso de introducir un Alto o Ancho menor que 64 y mayor que los 128 píxeles el sistema muestra un mensaje de error: Un número de 64 a 128. 4- En caso de dejar en blanco el campo, el sistema muestra un mensaje en rojo encima del campo: Campo requerido.		
RF12	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Eliminar imagen	Permite eliminar la imagen del área de diseño de la plantilla, dando clic derecho sobre la imagen y seleccionando la opción eliminar.	Media	Alta
	Prototipo			



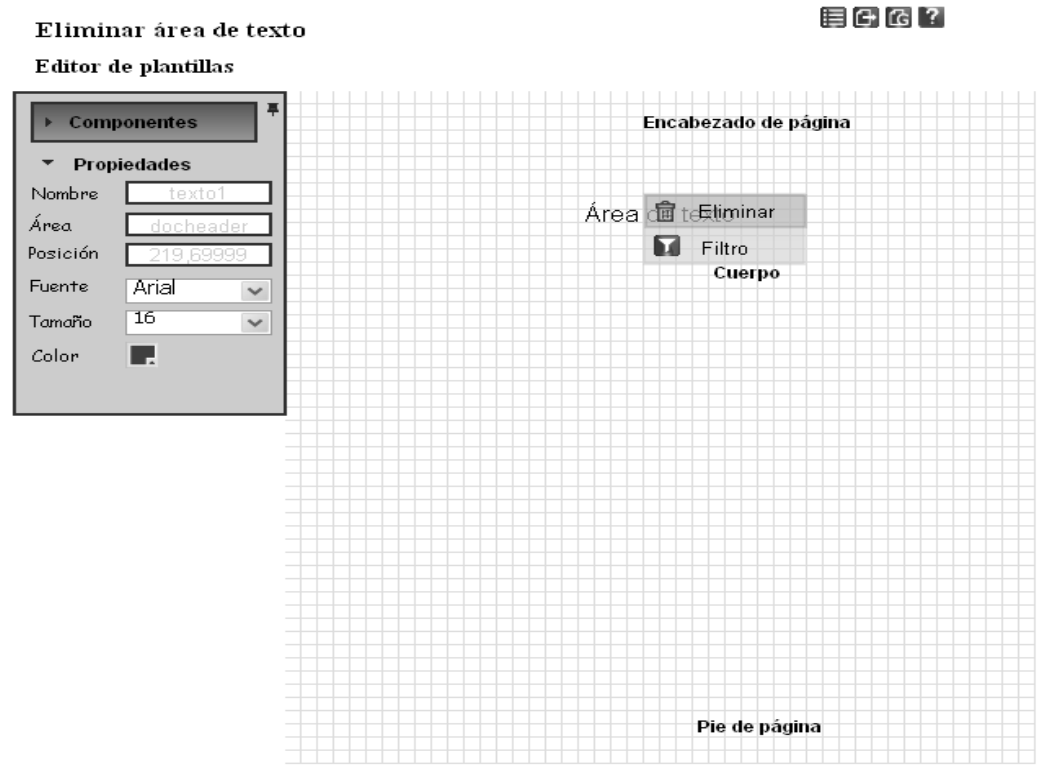
	Prototipo Eliminar imagen.			
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2- En caso de cancelar la acción se muestra un mensaje de advertencia "¿Está seguro de realizar la acción? "		
RF13	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Insertar componente de tipo área de texto	Permite insertar un componente de tipo área de texto en el área de diseño de la plantilla al dar clic encima del componente y luego en el área de diseño en cualquiera de las regiones (Encabezado de página, Cuerpo o Pie de página), tomando por defectos los valores de las propiedades del área de texto (*Nombre, Área, Posición, Fuente, Tamaño, Color*), además se muestran las opciones: Ayuda y Exportar en el área de íconos flotantes.	Alta	Alta
	Prototipo			
				
	Prototipo Insertar componente de tipo área de texto.			
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Nombre	character varying	De solo lectura.	

	Área	character varying	De solo lectura.	
	Posición	integer	De solo lectura.	
	Fuente	character varying	Campo de selección	
	Tamaño	integer	Solo dígitos del 8 al 32. Campo requerido.	
	Color	character varying	Hasta 3 caracteres.	
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.		
RF14	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Modificar área de texto	Permite modificar las propiedades (*Nombre, Área*, *Posición, Fuente, Tamaño, Color*) del componente de tipo área de texto en el área de diseño de la región donde fue creado (Encabezado de página, Cuerpo o Pie de página), para modificar el contenido del área de texto se presiona control + clic y se escribe el texto deseado, además se muestran las opciones: Listar, Exportar, Guardar y Ayuda en el área de íconos flotantes.	Alta	Alta
	Prototipo			



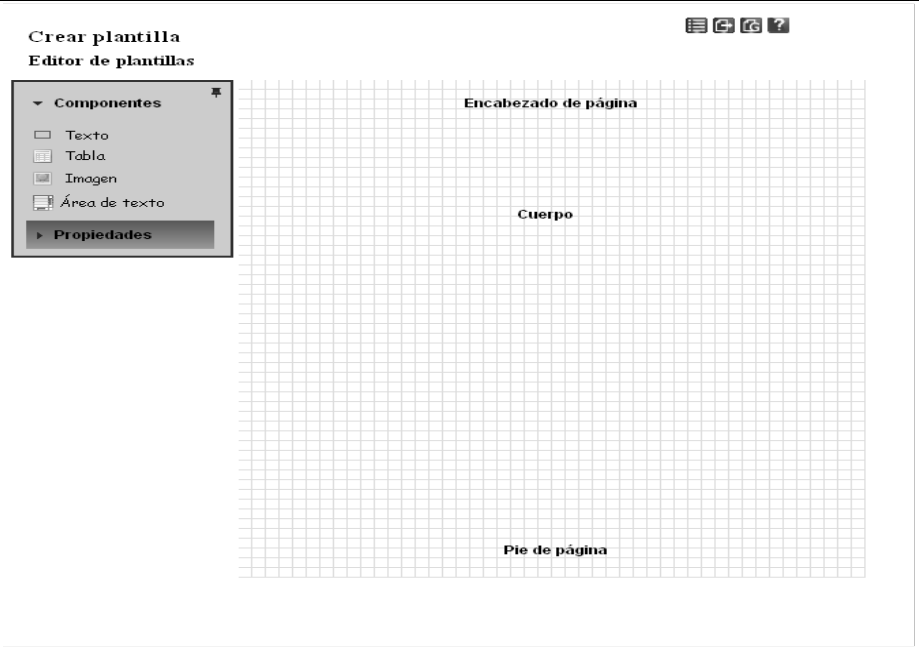
Prototipo Modificar área de texto.

Campos	**Tipos de Datos**	**Reglas o Restricciones**
Nombre	character varying	De solo lectura.
Área	character varying	De solo lectura.
Posición	integer	De solo lectura.
Fuente	character varying	Campo de selección.
Tamaño	integer	Solo dígitos del 8 al 32. Campo requerido.
Color	character varying	Hasta 3 caracteres.
Observaciones	1. Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2. Si se introduce un tamaño del texto fuera del rango de 8 a 32 o se introduce otro tipo de caracteres, el sistema muestra un mensaje de error: "Un número entre 8 y 32". 3. Si se trata de introducir más de 2 caracteres para el campo tamaño el sistema no permite seguir introduciéndolos. 4. Si se trata de introducir más de 3 caracteres para 6 de los campos del color y más de 3 caracteres para 1 campo del mismo, el sistema no deja seguir	

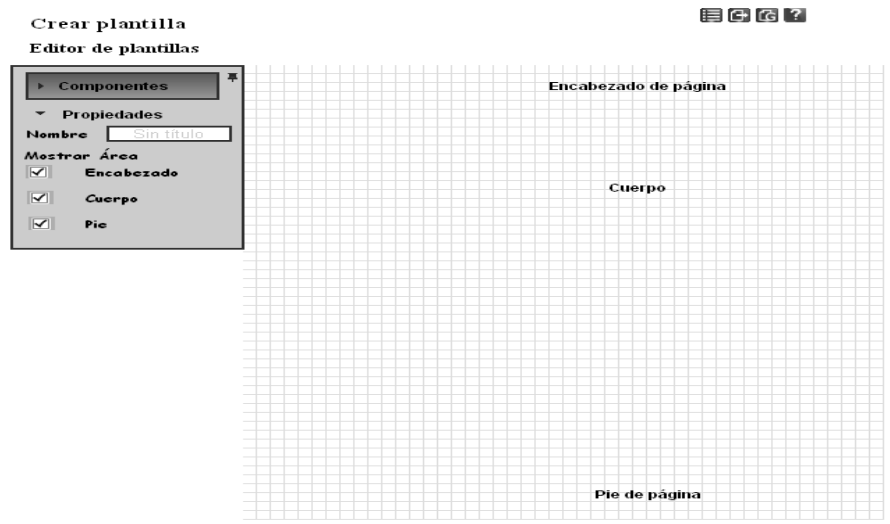
<table>
<tr><td></td><td></td><td colspan="3">introduciéndolos.
5. En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo “Campo requerido” en el campo que debe ser llenado obligatorio.</td></tr>
<tr><td rowspan="4">RF15</td><td>Nombre</td><td>Descripción</td><td>Complejidad</td><td>Prioridad para cliente</td></tr>
<tr><td>Eliminar área de texto.</td><td>Permite eliminar el área de texto del área de diseño de la plantilla, dando clic derecho sobre el texto y seleccionando la opción eliminar.</td><td>Media</td><td>Alta</td></tr>
<tr><td colspan="4">Prototipo</td></tr>
<tr><td colspan="4"> 

Prototipo Eliminar área de texto.</td></tr>
<tr><td></td><td>Campos</td><td colspan="2">Tipos de Datos</td><td>Reglas o Restricciones</td></tr>
<tr><td></td><td>Observaciones</td><td colspan="3">1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.
2- En caso de cancelar la acción se muestra un mensaje de advertencia “¿Está seguro de realizar la acción? “</td></tr>
</table>

RF16	Nombre	Descripción	Complejidad	Prioridad para cliente
	Crear plantilla.	Permite crear una plantilla dando clic en la opción Crear en el área de los íconos flotantes. En las propiedades de la plantilla se le pone el nombre a la plantilla y se selecciona la región que estará en el área de diseño.	Alta	Alta
	Prototipo			



Prototipo 1 Crear plantilla.

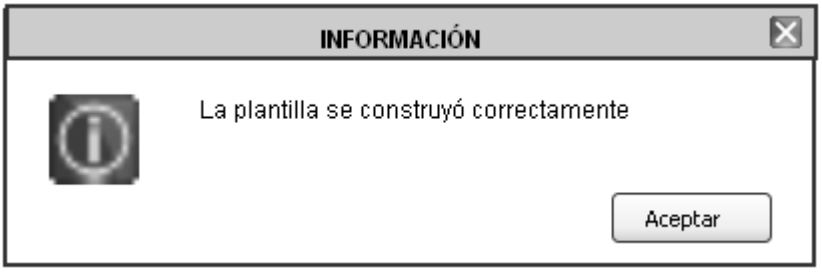


Prototipo 2 Crear plantilla.

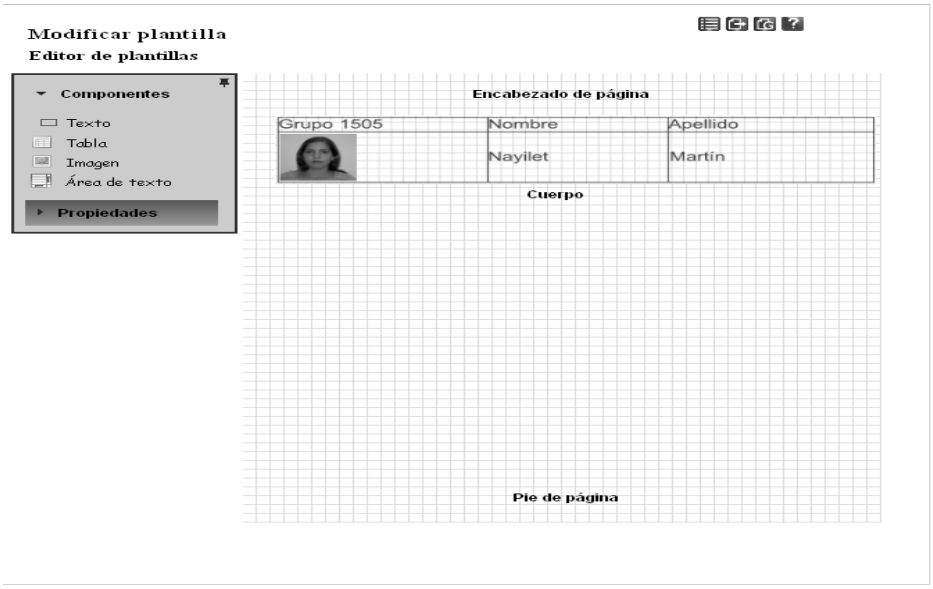
	Campos	**Tipos de Datos**	**Reglas o Restricciones**
	Nombre	character varying	Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos para una palabra es de 30. La cantidad de caracteres permitidos es 100.Campo requerido.
	Encabezado	booleano	Campo de selección.
	Cuerpo	booleano	Campo de selección.
	Pie	booleano	Campo de selección.
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2- En caso que se deje un campo de los obligatorios vacío se muestra un mensaje de error de color rojo “Campo requerido” en el campo que debe ser llenado obligatorio.	

	Nombre	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF17	Modificar plantilla.	Permite modificar una plantilla ya creada, dando clic en la opción aplicar plantilla en el listado de plantilla sobre la plantilla que se desea modificar.	Alta	Baja
	Prototipo			

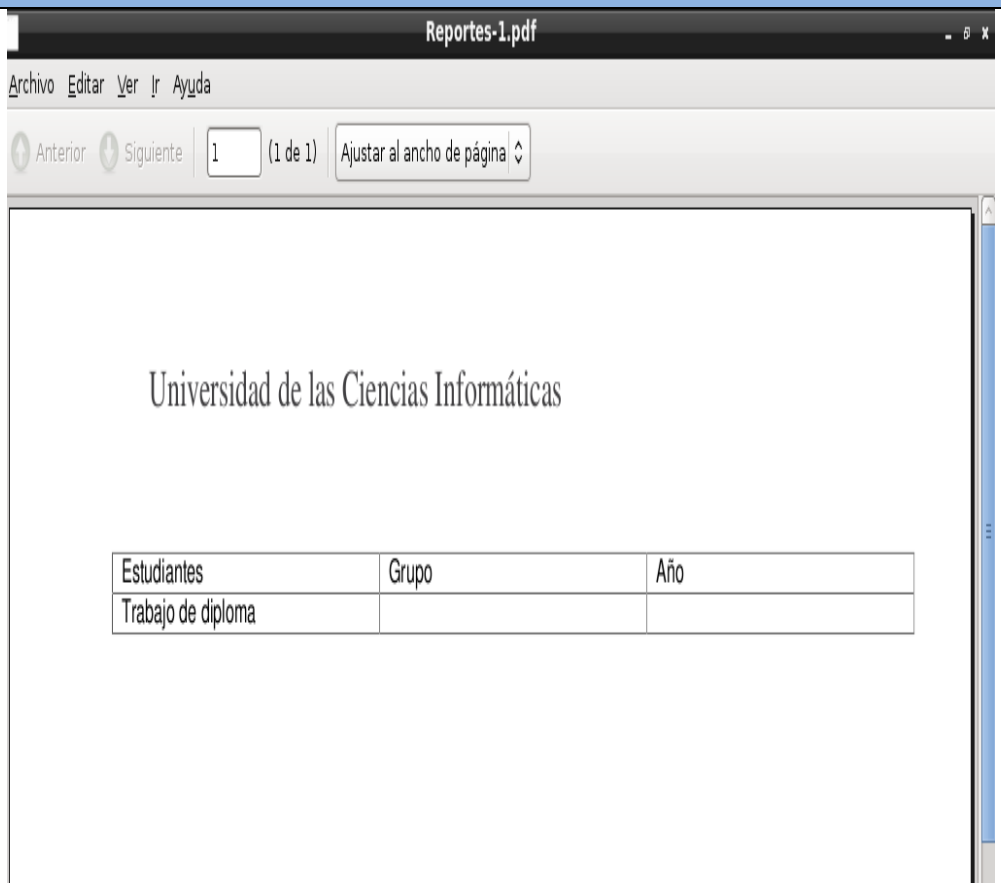
Prototipo 1 Modificar plantilla.



Prototipo 2 Modificar plantilla.



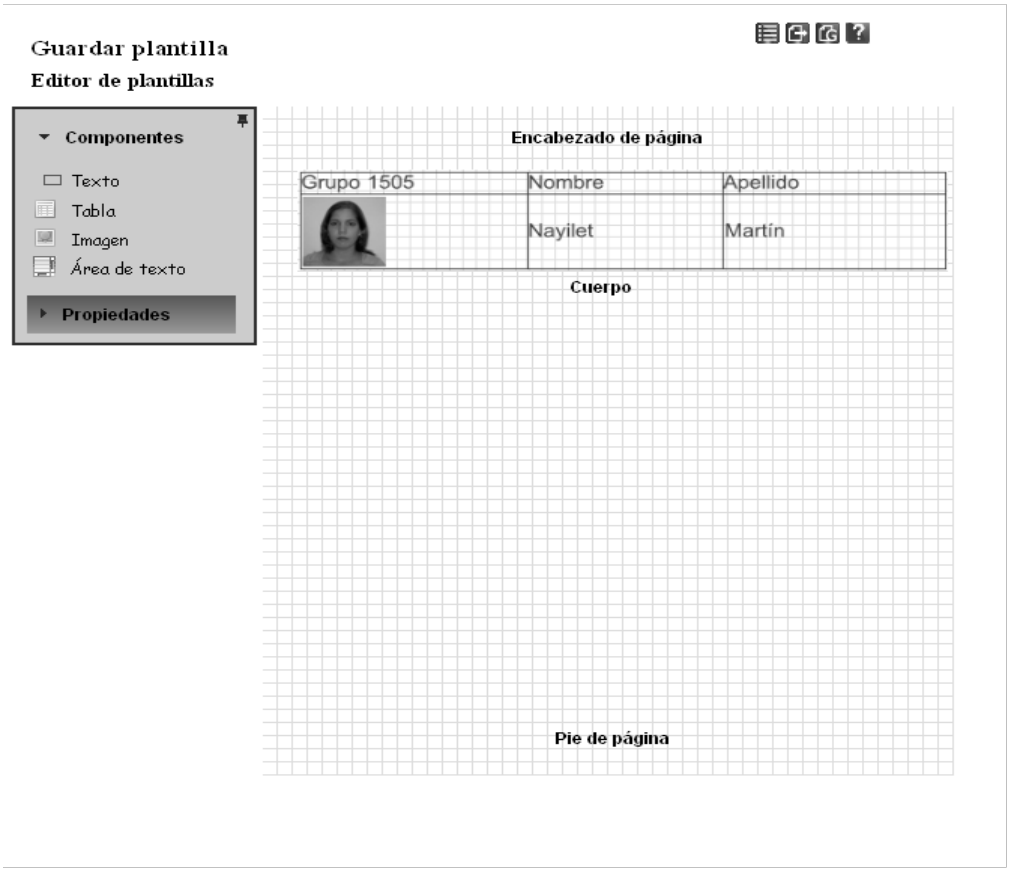
	Prototipo 3 Modificar plantilla.			
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática. 2- Cuando se aplica la plantilla el sistema muestra un mensaje de información: "La plantilla se construyó correctamente".		
RF18	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Exportar plantilla.	Permite mostrar en formato pdf como queda la plantilla que se esta construyendo, a través de la opción Exportar en el área de íconos flotantes.	Alta	Alta
	Prototipo			



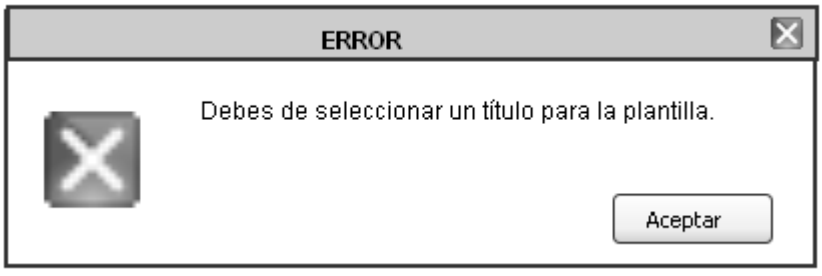
Prototipo Exportar la plantilla.

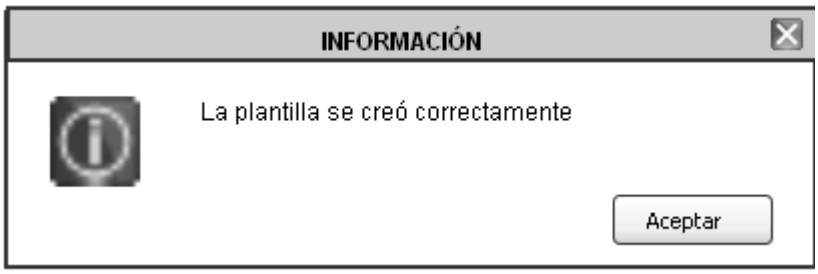
Campos	**Tipos de Datos**	**Reglas o Restricciones**

	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.		
RF19	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
	Guardar plantilla.	Permite guardar la plantilla construida, a través de la opción Guardar en el área de íconos flotantes.	Alta	Alta
	Prototipo			



Prototipo 1 Guardar plantilla.



<table>
<tr><td></td><td colspan="4">Prototipo 2 Guardar plantilla.
 

Prototipo 3 Guardar plantilla.</td></tr>
<tr><td></td><td>Campos</td><td colspan="2">Tipos de Datos</td><td>Reglas o Restricciones</td></tr>
<tr><td></td><td>Observaciones</td><td colspan="3">1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.
2- Si la plantilla no tienen nombre se muestra el mensaje de error: “Debes de seleccionar un título para la plantilla”.
3- Al guardar la plantilla se muestra un mensaje de información: “La plantilla se creó correctamente”.</td></tr>
<tr><td>RF20</td><td>Nombre</td><td>Descripción</td><td>Complejidad</td><td>Prioridad para cliente</td></tr>
<tr><td></td><td>Aplicar estilos de plantilla predefinidos.</td><td>Permite aplicar un estilo de plantilla predefinido buscando con el filtro Categoría y seleccionando la opción Predefinidas, luego se aplica la plantilla predefinida seleccionada, además se muestran las opciones: Crear y Ayuda en el área de íconos flotantes.</td><td>Alta</td><td>Alta</td></tr>
<tr><td></td><td colspan="4">Prototipo</td></tr>
</table>

Prototipo Aplicar estilos de plantilla predefinidos.

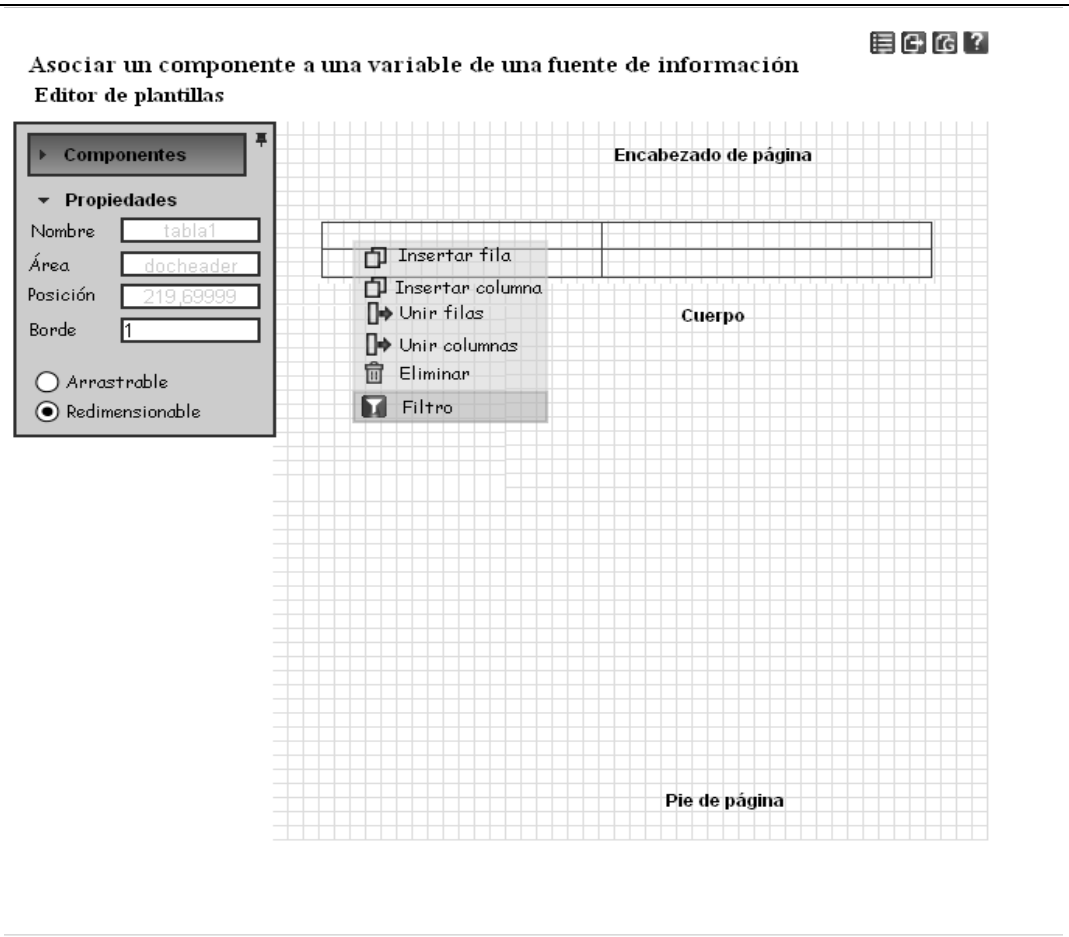
Campos	**Tipos de Datos**	**Reglas o Restricciones**
Filtrar búsqueda	character varying	Campo de selección.
Categoría	character varying	Campo de selección.
Opción	booleano	Campo de selección.
Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.	

	Nombre	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF21	Mostrar listado de plantillas.	Permite mostrar un listado de las plantillas creadas, también se muestra un buscador por el título de la plantilla, además se muestran las opciones: Crear y Ayuda en el área de íconos flotantes.	Alta	Alta
	Prototipo			

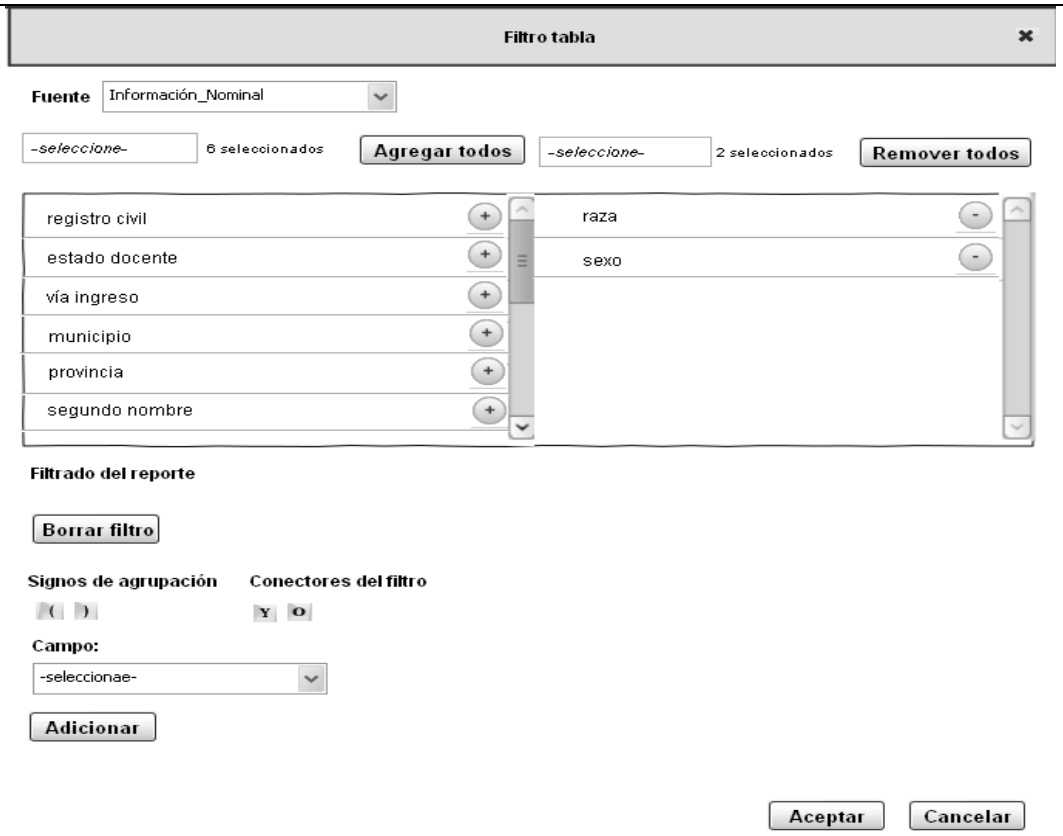
Prototipo Mostrar listado de plantillas.

Campos	**Tipos de Datos**	**Reglas o Restricciones**
Buscar	character varying	Campo de texto.
Cantidad por página	Lista desplegable	Permite la cantidad de elementos a escoger por páginas. Puede ser de 5, 10, 15, 20.
Número de página	Campo de texto.	Permite escribir el número de página del listado a la que se desea acceder, muestra el número de la página del listado que se está mostrando.
Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.	

	Nombre	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF22	Asociar un componente a una variable de una fuente de información.	Permite asociar un componente a una variable de una fuente de información al dar clic derecho sobre el componente (tabla, texto y área de texto) y seleccionar la opción Filtro, luego selecciona la fuente y dada la fuente la variable para dicho componente.	Alta	Alta
	Prototipo			

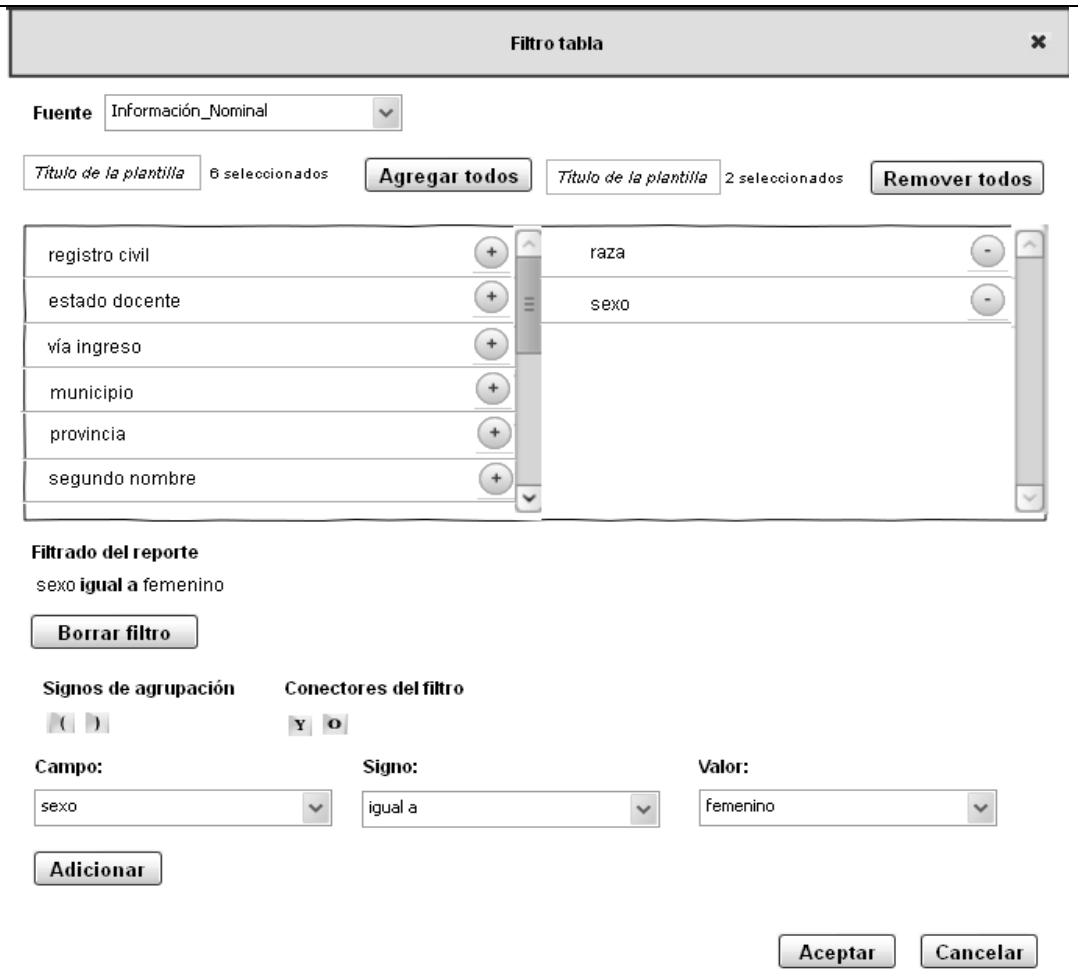


Prototipo 1 Asociar un componente a una variable de una fuente de información.

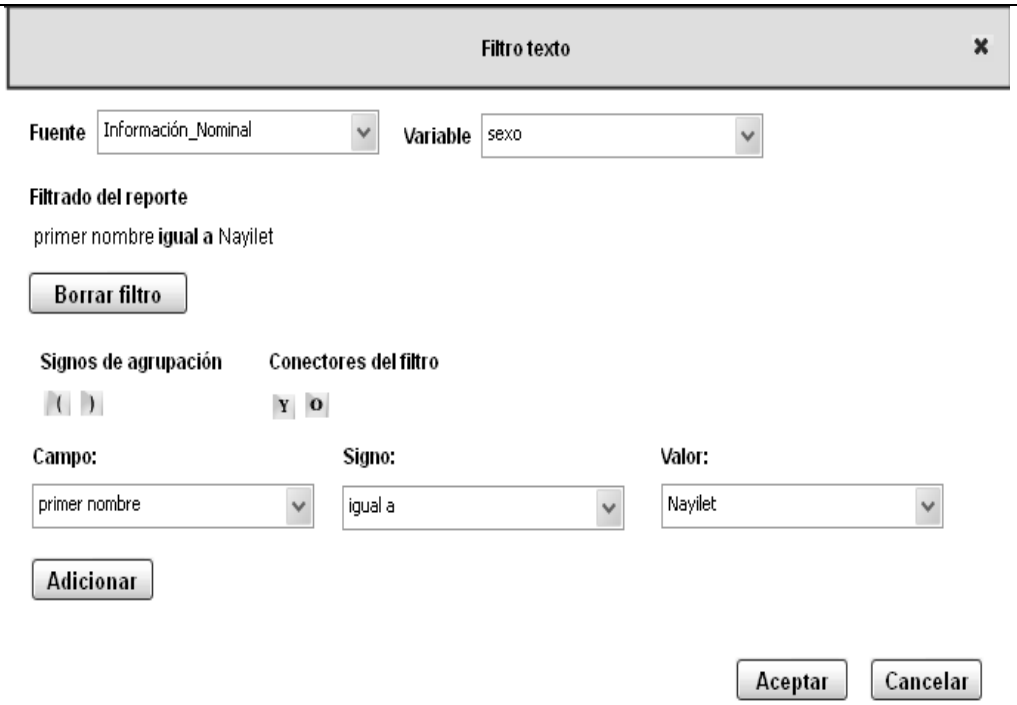


Prototipo 2 Asociar un componente a una variable de una fuente de información.

<table>
<tr><td></td><td colspan="5">Prototipo 3 Asociar un componente a una variable de una fuente de información.</td></tr>
<tr><td rowspan="5"></td><td>Campos</td><td>Tipos de Datos</td><td colspan="3">Reglas o Restricciones</td></tr>
<tr><td>Fuente de información</td><td>character varying</td><td colspan="3">Campo obligatorio</td></tr>
<tr><td>Variable</td><td>character varying</td><td colspan="3">Campo obligatorio</td></tr>
<tr><td>Selección de las columnas</td><td>varchar</td><td colspan="3">Campo obligatorio</td></tr>
<tr><td>Observaciones</td><td colspan="4">1. Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.</td></tr>
<tr><td rowspan="3">RF23</td><td>Nombre</td><td colspan="2">Descripción</td><td>Complejidad</td><td>Prioridad para cliente</td></tr>
<tr><td>Acotar la fuente de información.</td><td colspan="2">Permite acotar la variable de la fuente de información seleccionada a través de los filtros con la opción Campo, Signo y Valor, además de los signos de agrupación y los conectores de filtro, para construir el filtro. (Diferente solo para las tablas, el componente área de texto se comporta igual que el texto)</td><td>Alta</td><td>Alta</td></tr>
<tr><td></td><td colspan="5">Prototipo</td></tr>
</table>



Prototipo 1 Permitir acotar la fuente de información.



Prototipo 2 Permitir acotar la fuente de información.

	Campos	**Tipos de Datos**	**Reglas o Restricciones**
	Signos de agrupación	character varying	Campo de selección
	Conectores de filtro	character varying	Campo de selección
	Campo	character varying	Campo de selección
	Signo	character varying	Campo de selección
	Valor	character varying	Campo de selección
	Observaciones	1- Interactúa con esta acción el administrador y directivos de la alta gerencia con conocimientos básicos de informática.	

Anexo 2: Diccionario de datos

Nombre de la entidad		tb_dcomponente			
Descripción de la entidad		Tabla de datos para almacenar los componentes creados.			
Nombre	**Descripció**	**Tipo**	**Puede**	**Restricciones**	**Criterio de**

del atributo	n		ser nulo			Selección	
				Clases válidas	Clases no válidas	Múltiple	Única
Id_componente	Llave primaria	serial	No	Numérico	Letras	No	Si
nombre_componente		character varying	No	Alfanumérico	Caracteres especiales	No	Si
cache_json		text	No	Alfanumérico	Caracteres especiales	No	Si
id_plantilla	Lave foránea	integer	No	Numérico	Letras	No	Si
id_tipo_componente	Lave foránea	integer	No	Numérico	Letras	No	Si
Nombre de la entidad		tb_dplantilla					
Descripción de la entidad		Tabla de datos para almacenar la plantilla creada.					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
Id_plantilla	Llave primaria	serial	No	Numérico	Letras	No	Si
nombre_plantilla		character varying	No	Alfanumérico	Caracteres especiales	No	Si
fecha_registro		timestamp	No	-	Letras	No	Si
Nombre de la entidad		tb_npropiedad					
Descripción de la entidad		Tabla nomenclador para almacenar las propiedades de los componentes.					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
Id_propiedad	Llave primaria	serial	No	Numérico	Letras	No	Si

nombre_propiedad		character varying	No	Alfanumérico	Caracteres especiales	No	Si
fecha_registro		timestamp	No	-	Letras	No	Si
activo		boolean	No	True False		No	Si
descripción		text	No	Alfanumérico	Caracteres especiales	No	Si
Nombre de la entidad		tb_ntipo_componente					
Descripción de la entidad		Tabla nomenclador para almacenar los tipos de componentes.					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
Id_tipo_componente	Llave primaria	serial	No	Numérico	Letras	No	Si
nombre_tipo_componente		character varying	No	Alfanumérico	Caracteres especiales	No	Si
fecha_registro		timestamp	No	-	Letras	No	Si
activo		boolean	No	True False	Alfanumérico	No	Si
descripción		text	No	Alfanumérico	Caracteres especiales	No	Si
Nombre de la entidad		tb_rcomponente_propiedad					
Descripción de la entidad		Tabla para guardar las relaciones entre las tablas tb_dcomponente y tb_npropiedad.					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
Id_componente	Llave primaria	integer	No	Numérico	Letras	No	Si
Id_propiedad	Llave	integer	No	Numérico	Letras	No	Si

	primaria						
valor		text	No	Alfanumérico	Caracteres especiales	No	Si
Nombre de la entidad		tb_rtipo_componente_propiedad					
Descripción de la entidad		Tabla para guardar las relaciones entre las tablas tb_tipo_componente y tb_npropiedad.					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
Id_tipo_componente	Llave primaria	integer	No	Numérico	Letras	No	Si
Id_propiedad	Llave primaria	integer	No	Numérico	Letras	No	Si
Nombre de la entidad		tb_dconfiguracion_componente					
Descripción de la entidad		Tabla para guardar las configuraciones de los componentes.					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_componente	Llave primaria	integer	No	Numérico	Letras	No	Si
id_fuente		integer	No	Numérico	Letras	No	Si
variable		character varying	No	Alfanumérico	Caracteres especiales	No	Si
filtro		text	No	Alfanumérico	Caracteres especiales	No	Si
json		text	No	Alfanumérico	Caracteres especiales	No	Si

Anexo 3: Casos de prueba de caja blanca

Tabla 11: Caso de prueba de caja blanca CENIA_PRE_R_DP_GP.

Prueba estructural de caja blanca	**Código caso de prueba: CENIA_PRE_R_DP_GP**
Probador: Yander Santiesteban Rojas	

Código al que se aplica:

```
function guardarPlantilla($post_vars) {
     if (isset($post_vars) && !empty($post_vars)) {
         if (!$this->generador_reporte_lib->existeElemento(json_decode($post_vars['nombrePlantilla']))) {
             if ($this->generador_reporte_lib->guardarPlantilla(json_decode($post_vars['datos'])) != false) {
                 return "El elemento ha sido modificado satisfactoriamente";
             }
             else
                 return "Ocurrió un error durante la operación. Intente más tarde";
         }else
             return "El elemento ya existe.";
     }else
         return "Uno o varios elementos se ha introducido de forma incorrecta.";
 }
```

Complejidad ciclomática:

V (G) = (A - N) + 2 = (10 -8) + 2 = 4

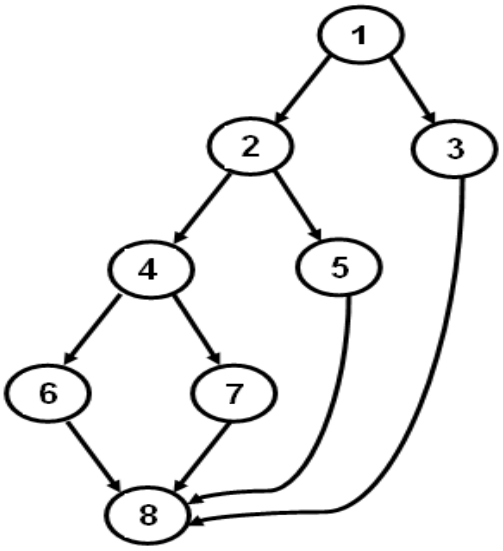
Caminos independientes

1) 1 – 3 – 8

2) 1 – 2 – 5 – 8

3) 1 – 2 – 4 – 7 – 8

4) 1 – 2 – 4 – 6 – 8



Caso de prueba para el camino básico 1	
Descripción: los datos de entrada serán los atributos de la plantilla a guardar.	
Condición de ejecución: los datos de entrada no son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo vacío.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Uno o varios elementos se ha introducido de forma incorrecta."	

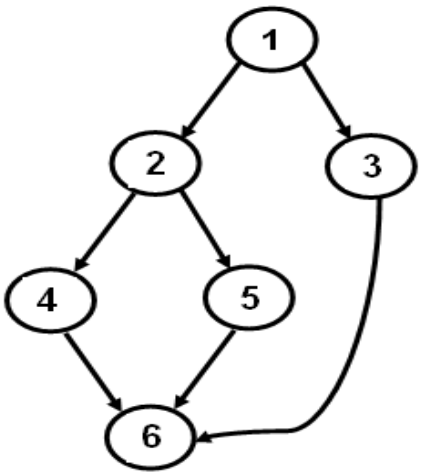
$nombrePrueba="Prueba:'Guardar Plantilla'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 2	
Descripción: los datos de entrada serán los atributos de la plantilla a guardar.	
Condición de ejecución: los datos de entrada ya están registrados en la base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos de la plantilla insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="El elemento ya existe." $nombrePrueba="Prueba:'Guardar Plantilla'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 3	
Descripción: los datos de entrada serán los atributos de la plantilla a guardar.	
Condición de ejecución: ocurre un error a la hora de guardar los datos en la base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos de la plantilla insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Ocurrió un error durante la operación. Intente más tarde." $nombrePrueba="Prueba:'Guardar Plantilla'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 4	
Descripción: los datos de entrada serán los atributos de la plantilla a guardar.	
Condición de ejecución: los datos de entrada no están registrados en la base de datos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos de la plantilla insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación:	

$resultadoEsperado="El elemento ha sido modificado satisfactoriamente" $nombrePrueba="Prueba:'Guardar Plantilla'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);
Evaluación del caso de prueba: satisfactoria
Resultado final de la prueba: satisfactoria en su totalidad.

Tabla 12: Caso de prueba de caja blanca CENIA_PRE_R_DP_E.

Prueba estructural de caja blanca	**Código caso de prueba: CENIA_PRE_R_DP_E**
Probador: Yander Santiesteban Rojas	

Código al que se aplica:
`function exportar($post_vars) {`
`  if (isset($post_vars) && !empty($post_vars)) {`
`  if ($this->generador_reporte_lib->exportarPlantilla($post_vars) != true) {`
`  return "Ocurrió un error durante la operación. Intente más tarde.";`
`  }`
`  else return true;`
`  }`
`  else`
`  return "Uno o varios elementos se ha introducido de forma incorrecta.";`
`}`	

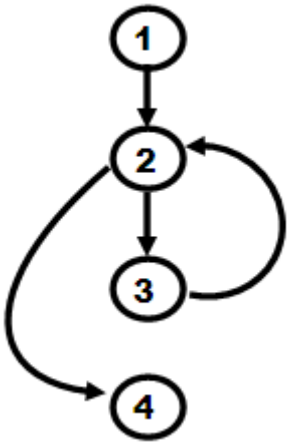
Complejidad ciclomática: V (G) = (A - N) + 2 = (7 - 6) + 2 = 3 **Caminos independientes** 1) 1 – 3 – 6 2) 1 – 2 – 5 – 6 3) 1 – 2 – 4 – 6	
Caso de prueba para el camino básico 1	
Descripción: los datos de entrada serán los atributos de la plantilla a exportar.	
Condición de ejecución: los datos de entrada no son válidos.	

Procedimiento prueba automatizada	
Datos de entrada:	arreglo vacío.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Uno o varios elementos se ha introducido de forma incorrecta." $nombrePrueba="Prueba:'Exportar Plantilla'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->exportar($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	Satisfactoria
Caso de prueba para el camino básico 2	
Descripción: los datos de entrada serán los atributos de la plantilla a exportar.	
Condición de ejecución: ocurre un error en la construcción de la plantilla xsl.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos de la plantilla insertados por el usuario.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Ocurrió un error durante la operación. Intente más tarde." $nombrePrueba="Prueba:' Exportar Plantilla'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->exportar($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 3	
Descripción: los datos de entrada serán los atributos de la plantilla a exportar.	
Condición de ejecución: los datos de entrada son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos de la plantilla insertados por el usuario.
Tipo de dato esperado:	is_true
Función de evaluación: $resultadoEsperado="TRUE" $nombrePrueba="Prueba:' Exportar Plantilla'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->exportar($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Resultado final de la prueba: satisfactoria en su totalidad.	

Tabla 13: Caso de prueba de caja blanca CENIA_PRE_R_DR_OO.

<table>
<tr><td colspan="2">Prueba estructural de caja blanca</td><td>Código caso de prueba: CENIA_PRE_R_DR_OO.</td></tr>
<tr><td colspan="3">Probador: Yander Santiesteban Rojas</td></tr>
<tr><td colspan="3">Código al que se aplica:

```
public function obtenerObjetos() {
    $objeto = $this->generador_reporte_lib->obtenerObjetos();
    $fuente = array();
    foreach ($objeto as $value) {
        $fuente[] = array('id_objeto' => $value->id_objeto,
            'nombre_objeto' => $value->nombre_objeto);
    }
    echo json_encode($fuente);
}
```

</td></tr>
<tr><td colspan="2">Complejidad ciclomática:
V (G) = (A - N) + 2 = (4 - 4) + 2 = 2
Caminos independientes
1) 1 – 2 – 4
2) 1 – 2 – 3 – 2 – 4</td><td> 
</td></tr>
<tr><td colspan="3">Caso de prueba para el camino básico 1</td></tr>
<tr><td colspan="3">Descripción: los datos de entrada serán los atributos para la fuente de información.</td></tr>
<tr><td colspan="3">Condición de ejecución: los datos de entrada no son válidos.</td></tr>
<tr><td colspan="3">Procedimiento prueba automatizada</td></tr>
<tr><td>Datos de entrada:</td><td colspan="2">arreglo vacío.</td></tr>
<tr><td>Tipo de dato esperado:</td><td colspan="2">is_string</td></tr>
<tr><td colspan="3">Función de evaluación:
$resultadoEsperado="Uno o varios elementos se ha introducido de forma incorrecta."
$nombrePrueba="Prueba:' Obtener Fuentes'";
echo $this->ejecutarPrueba($this->generador_reporte_lib->exportar($post_vars),
$resultadoEsperado,$nombrePrueba);</td></tr>
<tr><td>Evaluación del caso de prueba:</td><td colspan="2">satisfactoria</td></tr>
<tr><td colspan="3">Caso de prueba para el camino básico 2</td></tr>
<tr><td colspan="3">Descripción: los datos de entrada serán los atributos para la fuente de información.</td></tr>
</table>

Condición de ejecución: los datos de entrada son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo con los datos de las fuentes de información.
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array" $nombrePrueba="Prueba:' Obtener Fuentes'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->exportar($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Resultado final de la prueba: satisfactoria en su totalidad.	

Tabla 14: Caso de prueba de caja blanca CENIA_PRE_R_DP_OPODI.

Prueba estructural de caja blanca	**Código caso de prueba: CENIA_PRE_R_DP_OPODI.**
Probador: Yander Santiesteban Rojas	
Código al que se aplica:	

```
public function obtenerPropiedadesObjetosDadoId() {
    $post_vars = $this->input->all_post(true);
    $variables = array();
    if (isset($post_vars['id']) && !empty($post_vars['id'])) {
        $propiedades = $this->generador_reporte_lib->obtenerPropiedadesObjetosDadoId($post_vars['id']);
        foreach ($propiedades as $key => $value) {
            if ($value->referencia_nomenclador == 't') {
                $variables[] = array('id_propiedad' => $value->campo_visual_nomenclador,
                    'nombre_propiedad' => $value->nombre_visual);
            } else {
                $variables[] = array('id_propiedad' => $value->nombre_campo,
                    'nombre_propiedad' => $value->nombre_visual);
            }
        }
    }
    echo json_encode($variables);
}
```

Complejidad ciclomática:

V (G) = (A - N) + 2 = (11 -9) + 2 = 4

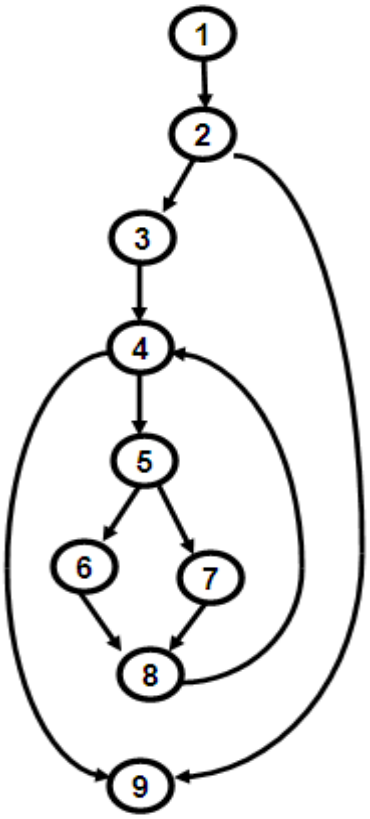
Caminos independientes

1) 1 – 2 – 9

2) 1 – 2 – 3 – 4 – 9

3) 1 – 2 – 3 – 4 – 5 – 7 – 8 – 4 – 9

4) 1 – 2 – 3 – 4 – 5 – 6 – 8 – 4 – 9



Caso de prueba para el camino básico 1	
Descripción: los datos de entrada será un arreglo con el identificador de la fuente de información.	
Condición de ejecución: los datos de entrada no son válidos.	
Procedimiento prueba automatizada	
Datos de entrada:	arreglo vacío.
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Uno o varios elementos se ha introducido de forma incorrecta." $nombrePrueba="Prueba:'Obtener Variables"'; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 2	
Descripción: los datos de entrada será un arreglo con el identificador de la fuente de información.	
Condición de ejecución: ocurre un error de base de datos obteniendo las variables de la fuente de información.	
Procedimiento prueba automatizada	
Datos de entrada:	cadena de caracteres con los con el identificador de la fuente de

	información
Tipo de dato esperado:	is_string
Función de evaluación: $resultadoEsperado="Uno o varios elementos se ha introducido de forma incorrecta." $nombrePrueba="Prueba:'Obtener Variables'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 3	
Descripción: los datos de entrada será un arreglo con el identificador de la fuente de información.	
Condición de ejecución: el campo referencia nomenclador tiene el valor de false	
Procedimiento prueba automatizada	
Datos de entrada:	los datos de entrada son válidos
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado="Array." $nombrePrueba="Prueba:'Obtener Variables'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Caso de prueba para el camino básico 4	
Descripción: los datos de entrada será un arreglo con el identificador de la fuente de información.	
Condición de ejecución: el campo referencia nomenclador tiene el valor de true	
Procedimiento prueba automatizada	
Datos de entrada:	los datos de entrada son válidos
Tipo de dato esperado:	is_array
Función de evaluación: $resultadoEsperado=" Array " $nombrePrueba="Prueba:'Obtener Variables'"; echo $this->ejecutarPrueba($this->generador_reporte_lib->guardarPlantilla($post_vars), $resultadoEsperado,$nombrePrueba);	
Evaluación del caso de prueba:	satisfactoria
Resultado final de la prueba: satisfactoria en su totalidad.	

Anexo 4: Casos de prueba de integración

Tabla 15: Prueba de integración-módulo Tesis y Título.

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 diseñador de plantillas 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 diseñador de plantillas consulta estos datos y crea el reporte.
Resultado esperado	Se crea el reporte con los datos.
Evaluación	Prueba satisfactoria.

Tabla 16: Prueba de integración-módulo Personal y Secretaría.

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 diseñador de plantillas 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 diseñador de plantillas consulta estos datos y crea el reporte.
Resultado esperado	Se crea el reporte con los datos.
Evaluación	Prueba satisfactoria.

Tabla 17: Prueba de integración-módulo Control Docente.

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 diseñador de plantillas 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 diseñador de plantillas consulta estos datos y crea el reporte.

Resultado esperado	Se crea el reporte con los datos.
Evaluación	Prueba satisfactoria.

Tabla 18: Prueba de integración-módulo Estudiante.

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

Anexo 5: Resultados de pruebas de carga y stress

Aggregate Graph

Nombre: Aggregate Graph

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/proyecto/base...	1	175	175	175	175	175	0,00%	5,7/sec	27,5
/proyecto/base...	1	212	212	212	212	212	0,00%	4,7/sec	,6
/proyecto/base...	1	239	239	239	239	239	0,00%	4,2/sec	3,8
/proyecto/base...	1	235	235	235	235	235	0,00%	4,3/sec	2,8
/proyecto/base...	1	214	214	214	214	214	0,00%	4,7/sec	1,5
/proyecto/base...	1	197	197	197	197	197	0,00%	5,1/sec	4,3
TOTAL	6	212	214	239	175	239	0,00%	4,7/sec	6,0

Max length of x-axis label

Width 800

Height 600

Column Media | Display Graph | Save Graph | Save Table Data

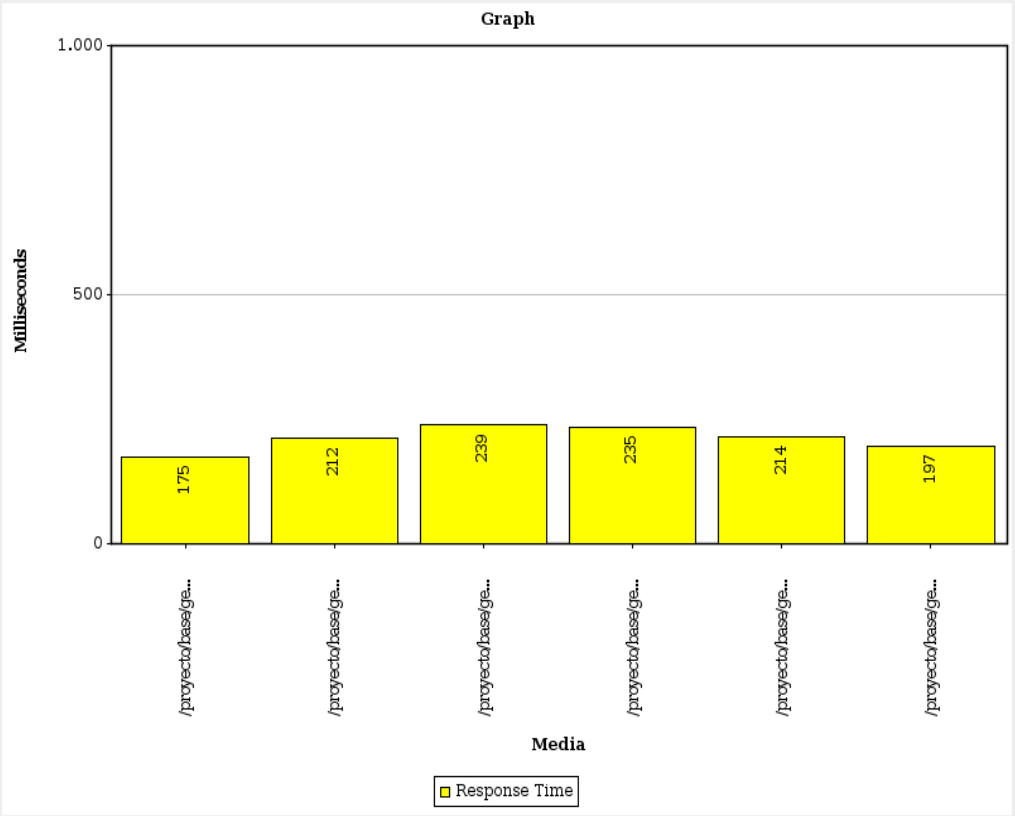


Figura 20: Resultados para 1 muestra.

Aggregate Graph

Nombre: Aggregate Graph

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/proyecto/base...	50	4931	6305	7991	447	8807	0,00%	5,2/sec	24,9
/proyecto/base...	50	2620	3250	3674	835	3849	0,00%	3,9/sec	,5
/proyecto/base...	50	2785	3198	3656	909	4075	0,00%	3,3/sec	3,0
/proyecto/base...	50	2673	2908	3284	1067	3534	0,00%	3,0/sec	1,9
/proyecto/base...	50	2023	1994	2725	1103	2901	0,00%	2,9/sec	,9
/proyecto/base...	50	1791	1741	2711	522	3216	0,00%	3,0/sec	2,5
TOTAL	300	2804	2706	5114	447	8807	0,00%	14,2/sec	18,1

Title for Graph

Max length of x-axis label

Width 800

Height 600

Column Media | Display Graph | Save Graph | Save Table Data

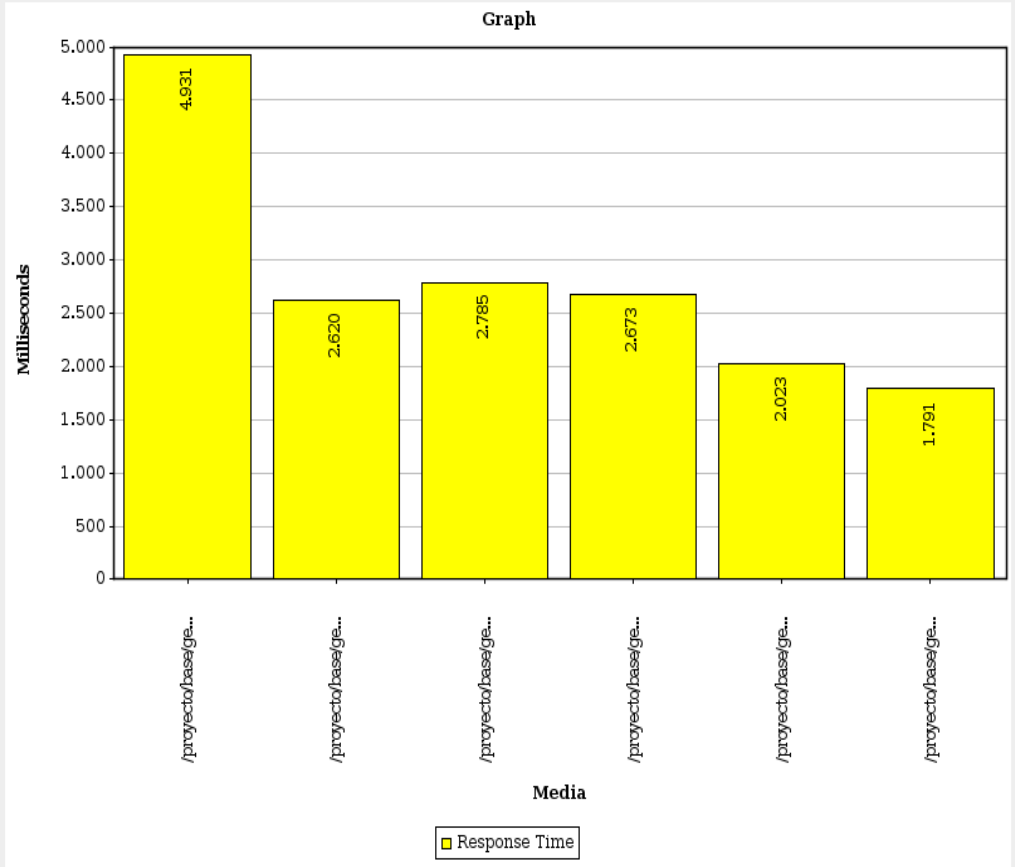


Figura 21: Resultados para 50 muestras.

Aggregate Graph

Nombre: Aggregate Graph

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/proyecto/base...	100	4898	6282	7882	433	8807	0,00%	40,9/min	3,3
/proyecto/base...	100	2609	3142	3721	771	3987	0,00%	40,0/min	,1
/proyecto/base...	100	2865	3331	3695	909	4075	0,00%	39,4/min	,6
/proyecto/base...	100	2628	2864	3263	927	3573	0,00%	38,9/min	,4
/proyecto/base...	100	2047	2055	2717	784	3211	0,00%	38,8/min	,2
/proyecto/base...	100	1800	1696	2699	522	3216	0,00%	38,9/min	,5
TOTAL	600	2808	2676	4995	433	8807	0,00%	3,8/sec	4,8

Statistical Graphs

Title for Graph

Max length of x-axis label

Width 800

Height 600

Column Media | Display Graph | Save Graph | Save Table Data

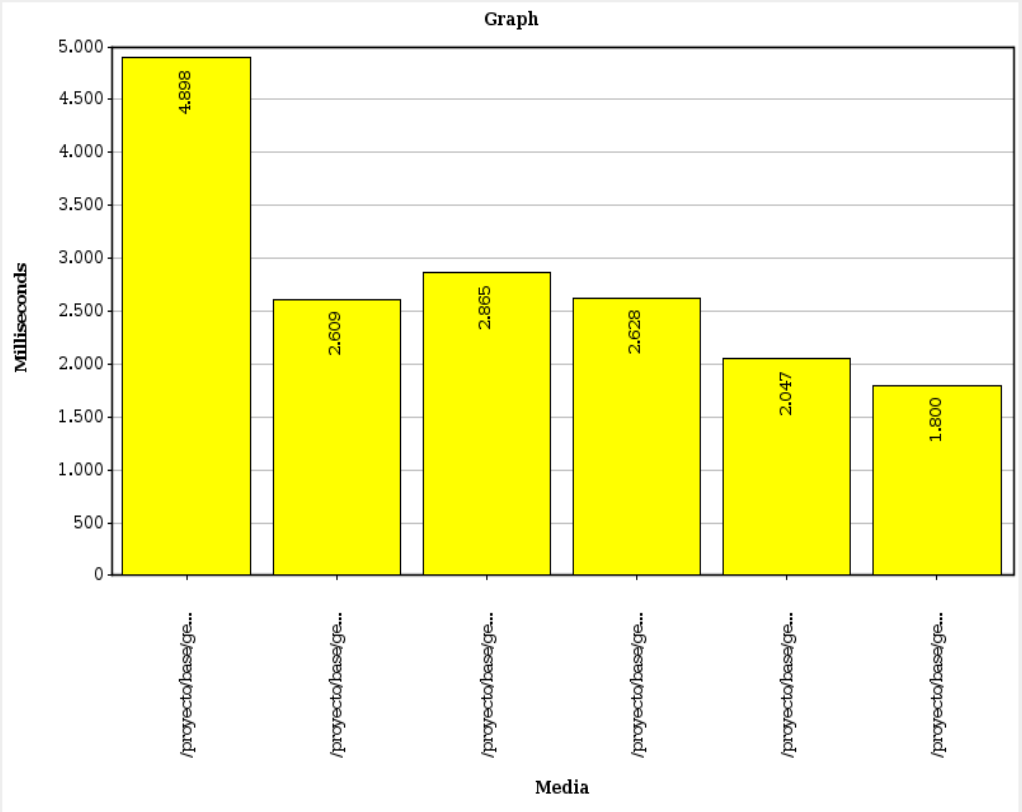


Figura 22: Resultados para 100 muestras.