Universidad de las Ciencias Informáticas

Facultad 1

Desarrollo de un diseñador de plantillas para el Sistema de Impresión de Documentos

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

Autor: Alejandro Vázquez Chávez

Tutores: Ing. Yanio García Vidal
Ing. Adriana Alfonso Luis
Ing. Yander Santiesteban

La Habana, Cuba

Junio, 2014

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Alejandro Vázquez Chávez

_____

Firma del autor

Ing. Yanio García Vidal

_____

Firma del tutor

Ing. Adriana Alfonso Luis

_____

Firma del tutor

Ing. Yander Santiesteban Rojas

_____

Firma del tutor

Datos de contacto

Ing. Yanio García Vidal

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Especialista de la producción y tiene 5 años de experiencia laboral. Actualmente trabaja en el Departamento de Desarrollo de la Dirección de Informatización, Vicerrectoría de Tecnología, Universidad de las Ciencias Informáticas.

yvidal@uci.cu

Ing. Adriana Alfonso Luis:

Graduada de Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2010. Especialista de la producción y profesor instructor, líder del proyecto Sistema de Cooperación Internacional. Tiene 3 años de experiencia laboral. Actualmente trabaja en el Departamento de Desarrollo de la Dirección de Informatización, Vicerrectoría de Tecnología, Universidad de las Ciencias Informáticas.

aluis@uci.cu

Ing. Yander Santiesteban Rojas:

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2013. Recién graduado en adiestramiento y desempeña el rol de desarrollador del proyecto Sistema de Gestión de Residencia. Actualmente trabaja en el Departamento de Desarrollo de la Dirección de Informatización, Vicerrectoría de Tecnología, Universidad de las Ciencias Informáticas.

yander@uci.cu

Dedicatoria

Dedico esta investigación especialmente a mis padres: Adalberto y Sonia por su apoyo incondicional durante toda mi vida estudiantil y a toda mi familia que de una forma u otra me han apoyado también.

Agradecimientos

Siempre es difícil mencionar a todos aquellos que de una u otra manera me han acompañado en el desarrollo de la presente investigación, por tanto quiero agradecerle a todos aquellos que hicieron posible que esta investigación saliera adelante de la mejor manera posible.

Partiendo de esto quisiera agradecer a mi madre Sonia y a mi padre Adalberto por haberme apoyado todos estos años de estudio. Además quisiera agradecerles a ellos por haberme dado todo cuanto han podido y que confiasen en mí. También quiero agradecer a toda mi familia porque desde que comencé en esta universidad me han apoyado incondicionalmente y estado al pendiente de mí estos 5 años. Agradezco a mi novia Ania, a mi cuñada Arianna y a su esposo Jose. Agradezco también a Mercedes, Alelí y a mis tutores Yander, Adriana y Yanio, por haberme guiado a la culminación satisfactoria de la presente investigación. Extiendo también mis agradecimientos a mis compañeros de aula. Sinceramente gracias a todos.

Resumen

En la Universidad de las Ciencias Informáticas (UCI) particularmente en la Dirección de Informatización (DIN) se desarrolla el Sistema de Gestión Universitaria (SGU), el cual permite gestionar los procesos universitarios. En este se desarrolla el componente Documentos Acreditativos (DA). Este componente prepara los documentos para su impresión mediante el Sistema de Impresión de Documentos (SID). La gestión de las plantillas mediante la cual se imprime el documento se realiza dentro del componente DA adjuntando la plantilla XML[1] y su XSL[2] asociado, debido que el mencionado SID utiliza XSL-FO[3] y la librería FOP[4] para la impresión. Para incorporar una plantilla al sistema es necesario tener conocimientos avanzados de los lenguajes XSL y XML. Con la presente investigación se propone desarrollar una solución informática que permita diseñar plantillas personalizadas para el SID. Para lograr tal propósito se realizó un estudio de herramientas informáticas para el diseño de plantillas. La solución se desarrolló utilizando tecnologías establecidas por el Grupo de Arquitectura del Departamento de Desarrollo de la DIN y el proceso de desarrollo de software fue guiado por la metodología Desarrollo Ágil con Calidad (DAC) la cual contribuyó a la obtención de los artefactos. Con la implementación del Sistema Diseñador de Plantillas se dotó a la UCI de un sistema que permite a los usuarios diseñar plantillas personalizadas de documentos acreditativos para su posterior impresión mediante el SID.

Palabras clave: diseñar plantillas, gestión de plantillas, tecnologías libres.

[1] *Extensible Markup Language (lenguaje de marcas extensible).*
[2] *Extensible Stylesheet Language (lenguaje de hojas de estilo extensible).*
[3] *Extensible Stylesheet Language – Formatting Objects (lenguaje de etiquetas basado en XML).*
[4] *Formatting Objects Processor (contiene un controlador de formato de impresión para formateo de objetos XSL (XSL-FO)).*

Índice general

Índice de figuras

Índice de tablas

Introducción

La Gestión de Información (GI) constituye un proceso mediante el cual se planifican, organizan, dirigen y controlan los recursos de información de una organización asegurando un adecuado tratamiento, intercambio y uso de este recurso, que contribuyan al establecimiento de fortalezas organizacionales. Al desarrollar la GI la organización logra identificar y adquirir la información necesaria para satisfacer sus necesidades informativas, y organizar dicha información para un acceso cómodo. (1)

Los sistemas actuales de gestión de la información se basan en gran medida en la tecnología para recopilar y presentar datos. Un Sistema de Gestión de Información (SGI) puede facilitar la colaboración y la comunicación, pero el propósito principal de un SGI es hacer que la toma de decisiones por parte de los gerentes sea más eficiente y productiva mediante la combinación de la información de una variedad de fuentes en una sola base de datos y la presentación de la información en un formato lógico. (2)

La capacidad de gestionar la información referente a documentos acreditativos o a cualquier otro tipo de información es una de las características que presenta un SGI. Un documento acreditativo es aquel documento que obtiene una persona como resultado de su participación en un evento o actividad. Si esto ocurre dentro de alguna de las áreas de procesos de la gestión universitaria como son: pregrado, investigación y postgrado entonces se está en presencia de documentos acreditativos en la gestión universitaria. Estos pueden ser:

- Certificado de participación en eventos.
- Certificado de resultado en eventos.
- Título de graduado en la formación de pregrado.
- Otros diplomas de reconocimiento. (3)

Actualmente en la UCI se encuentra en uso el Sistema de Gestión Universitaria desarrollado por el Departamento de Desarrollo de la DIN para la gestión de los procesos universitarios en la UCI. Dentro de este sistema se encuentra el componente Documentos Acreditativos que permite la configuración y gestión de información de los documentos acreditativos de cualquier tipo, títulos, certificados, diplomas, boletas de asignación de plazas, entre otros. Este componente prepara los documentos para su posterior impresión mediante el Sistema de Impresión de Documentos. Con este sistema se logra imprimir de forma automatizada los títulos otorgados por la universidad. En la UCI desde el año 2007 hasta el 2013 se ha impreso unos 11584 títulos. En la actualidad la gestión de las plantillas mediante la cual se exporta e imprime el documento se realiza dentro del componente de Documentos Acreditativos adjuntando la plantilla XML y su XSL asociado, puesto que el mencionado sistema de impresión utiliza el lenguaje de etiquetas basado en XML: XSL-FO y la librería FOP para la impresión ya que estas contienen un controlador de formato de impresión para formateo de objetos XSL. Esta plantilla XSL debe ser implementada por un programador con conocimientos en este lenguaje y debe ser probada utilizando FOP directamente en el código del SGU, por

lo que para diseñar una plantilla para un documento acreditativo es necesario estar relacionado con la implementación de estos sistemas y tener conocimientos avanzados de XSL-FO. Los usuarios de estas soluciones para incorporar una plantilla al sistema o modificar alguna existente, como ha sido el caso del título otorgado por la UCI en reiteradas ocasiones se realizan de forma manual por un programador con conocimientos en los lenguajes de XML y XSL. Al depender de un desarrollador que les confeccione esta plantilla los usuarios no pueden realizar el diseño de la plantilla fácilmente, es decir de forma personalizada. Por tanto el desarrollador tiene que invertir tiempo y dejar a un lado sus deberes para satisfacer las necesidades del usuario. Por lo que se plantea como **problema a resolver:** ¿Cómo disminuir el nivel de experticia que necesita un usuario para diseñar documentos acreditativos en el Sistema de Gestión Universitaria? En consecuencia se define como **objeto de estudio:** el diseño de plantillas mediante la tecnología XSL-FO.

El **campo de acción** se enmarca en las herramientas informáticas para el diseño de plantillas personalizadas.

Para dar respuesta al problema antes enunciado se plantea como **objetivo general:** desarrollar una solución informática con tecnologías libres, que permita diseñar plantillas personalizadas para el Sistema de Impresión de Documentos.

Para garantizar el cumplimiento del objetivo general, el mismo se ha desglosado en los siguientes **objetivos específicos:**

- Identificar los elementos del diseño teórico necesarios, como fundamentación teórica de la investigación.
- Conceptualizar una propuesta de solución que permita diseñar documentos acreditativos para su posterior impresión mediante el SID.
- Implementar la propuesta de solución a partir de los requisitos identificados utilizando las tecnologías y técnicas definidas.
- Validar la solución implementada aplicando una estrategia de ejecución de pruebas de software.

Para lograr un correcto cumplimiento de los objetivos específicos y obtener los resultados esperados 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 para obtener buenas prácticas e identificar posibles deficiencias.
2. Caracterización de las herramientas y del proceso de desarrollo de software a utilizar en la propuesta de solución para obtener un producto de software libre que cumpla con las indicaciones de la UCI para el desarrollo de software.

3. Identificación de los requisitos funcionales y no funcionales de la propuesta de solución para definir las funcionalidades a desarrollar.
4. Modelado de la base de datos para representar y describir la estructura de datos y las relaciones entre ellos.
5. Implementación de las funcionalidades del diseñador de plantillas que cumplan con la arquitectura.
6. Realización de pruebas a la solución implementada para comprobar su correcto funcionamiento.

La **idea a defender** que sustenta la investigación es: el diseño de plantillas personalizadas para el Sistema de Impresión de Documentos contribuirá a mejorar la gestión de las plantillas en el Sistema de Gestión Universitaria.

Los métodos científicos utilizados para la realización de la presente investigación son:

Métodos teóricos:

- **Histórico-lógico:** permite constatar teóricamente la trayectoria real de un fenómeno en un período de tiempo dado. Es utilizado para estudiar las formas de solución a problemas análogos sobre el diseño de plantillas en el mundo y en la UCI.
- **Analítico-sintético:** permite hacer un análisis de las teorías en los documentos para extraer los elementos más importantes de cada uno de los aspectos esenciales de las herramientas de la literatura seleccionada para generar plantillas personalizadas para el Sistema de Impresión de Documentos en el Sistema de Gestión Universitaria.

Métodos empíricos:

- **Método de observación:** con este método es posible distinguir directamente los hechos de la realidad objetiva. Permite conocer el proceso delimitado como objeto de estudio, lo cual contribuyó a tener un registro visual más detallado de lo que se quiere y hace falta hacer; y cómo hay que hacerlo.
- **Revisión documental:** es utilizada para organizar, consultar y revisar la bibliografía existente en la universidad, así como la de diferentes fuentes de información en internet para la realización de las tareas de investigación.

Estructura del documento

El presente trabajo de diploma está estructurado en tres capítulos que abarcan todo el proceso para el desarrollo de la herramienta 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 las diferentes herramientas existentes

para el diseño de plantillas tanto en el mundo como en la UCI. También se describe el entorno tecnológico utilizado para el desarrollo de la presente investigación.

Capítulo 2: Descripción de la propuesta de solución: en este capítulo se plasma la descripción y el análisis de la propuesta de solución, sus requisitos funcionales y no funcionales y los actores involucrados en la propuesta de solución. También se describe la arquitectura del sistema, los patrones de diseño y de base de datos utilizados. Además se presenta el modelo de datos.

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 definen los estándares de codificación a utilizar en la implementación de la propuesta así como las pruebas que servirán para validar la solución.

Capítulo 1: Fundamentación teórica

En el presente capítulo se refleja el estudio de los diferentes conceptos relacionados con el diseño y gestión de las plantillas de documentos acreditativos en el SGU y las características generales de los diseñadores de plantillas. Además se realiza un estudio de los sistemas homólogos existentes en el mundo y en la UCI para realizar una valoración crítica de los mismos. También se caracteriza el proceso de desarrollo de software, los lenguajes de programación y tecnologías que se utilizan para el desarrollo de la propuesta de solución.

1.1. Diseño de plantillas

El diseño de plantillas es un proceso o labor destinado a proyectar, coordinar, seleccionar y organizar un conjunto de elementos para producir y crear objetos visuales. Siendo la plantilla diseñada una forma de objeto visual que proporciona una separación entre la forma o estructura y el contenido. Una plantilla agiliza el trabajo de reproducción de muchas copias idénticas o casi idénticas por lo que si se quiere un trabajo más refinado, más creativo, la plantilla no es más que un punto de partida, es decir, una idea aproximada de lo que se quiere hacer. A partir de la plantilla pueden diseñarse y fabricarse nuevas plantillas. El diseño de plantillas puede ser mediante la tecnología XSL-FO, siendo esta precisamente la tecnología que utiliza el SID.

1.2. Tecnología XSL-FO

XSL-FO es un lenguaje de etiquetas basado en XML que describe cómo se deben de disponer en una página el texto, las imágenes, las líneas y otros elementos gráficos. Con XSL-FO se pueden crear productos de impresión de alta calidad en papel o en pantalla. Al contrario que el formato XHTML[5]/HTML[6], que está especialmente indicado para aplicaciones de navegador, XSL-FO se utiliza sobre todo en el ámbito de la impresión y el archivado, es decir, en los casos en los que dentro de un documento se acumulan muchas páginas. XSL-FO incluye, entre otros, los siguientes elementos y atributos (selección):

- Regiones, márgenes y áreas.
- Ancho, alto y secuencia de las páginas.
- Paginación.
- Marcos, separaciones, múltiples columnas y bloques.
- Párrafos, listas y tablas.
- Formateo del texto como formatos de párrafo, saltos de línea y tabulación.

[5] *Extensible Hypertext Markup Language (lenguaje de marcado de hipertexto extensible).*
[6] *Hypertext Markup Language (lenguaje de marcado de hipertexto).*

- Líneas, imágenes.

Ventajas de XSL-FO:

- Es útil para la estructuración de textos/documentos y la definición de categorías como dirección, nombre, apellidos, sexo, etc.
- Las soluciones XSL-FO se pueden integrar en procesos XML ya existentes porque están basadas en XML. Esto resulta ventajoso en los documentos que se crean automáticamente. (4)

1.3. Diseñadores de plantillas

Los diseñadores de plantillas se le presentan a las organizaciones como una excelente opción para que el diseño de sus documentos sea de una manera más sencilla. Actualmente en el mundo existe una amplia variedad de sistemas que apoyan a los usuarios a diseñar plantillas, aunque todos fueron creados para un mismo fin no significa que sean los más convenientes a utilizar. A continuación se presentan los diseñadores de plantillas estudiados.

Crystal Reports

Crystal Reports para *Visual Studio .NET* es la herramienta de elaboración de informes estándar para *Visual Studio .NET*. Permite crear contenido interactivo con calidad de presentación en la plataforma .NET, lo que ha supuesto una ventaja fundamental para *Crystal Reports* durante años. (5)

El diseñador de plantillas de *Crystal Reports* presenta la interfaz de usuario a través de la cual un usuario, programador o no, puede diseñar un informe y guardarlo en un fichero para su posterior reutilización. Dicha herramienta puede programarse directamente desde *Visual Studio .NET* y además no es necesario distribuir el diseñador de plantillas con el informe. En general para crear un informe se necesita cargar los datos y posteriormente de una forma muy parecida a la mezcla de *PowerPoint* con *Excel* ir colocando y formateando los datos. Este informe se puede guardar como plantilla, visualizarlo o exportarlo en PDF[7], XML y otros populares formatos. (5)

Crystal Reports se utiliza con el sistema operativo *Microsoft Windows*, se debe adquirir una licencia de *Crystal Reports* por cada usuario que instale la aplicación, independientemente de la edición de *Crystal Reports* adquirida, por tanto se distribuye bajo los términos de la EULA[8], por lo que es software privativo y de uso restringido mediante el pago de patente.

ActiveReports 8

ActiveReports es un componente de informes .NET para aplicaciones *Windows Forms* y de formularios Web.

[7] *Portable Document Format (formato de documento portátil).*

[8] *End User Licensing Agreement (acuerdo de licencia de usuario final).*

El diseñador de informes de *ActiveReports* se integra perfectamente en los entornos de desarrollo *Visual Studio .NET*. Una vez instalado el producto en el equipo del desarrollador, la adición de un informe a un proyecto es tan fácil como añadir una clase o un formulario. Entre sus principales características se destaca que está escrito en C# e incluye filtros para la exportación a formatos más frecuentes como *Adobe PDF*, *Microsoft Excel*, RTF[9], HTML, entre otros. *ActiveReports* es totalmente administrado, no presenta dependencias de aplicaciones de terceros. También incluye un control de diseñador de plantillas ofreciéndoles a los usuarios finales capacidad para crear y modificar informes. (6)

ActiveReports se utiliza con el sistema operativo *Microsoft* y se distribuye bajo licencia privativa de la compañía GrapeCity inc.

iReport

iReport es una herramienta de diseño de informes visual, poderoso, intuitivo y fácil de usar para *JasperReports* (librería de Java que facilita y agiliza la generación, la previsualización y la impresión de los informes) escrito en las secuencias de comandos de código de Java. Este software permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. *iReport* está además integrado con *JFreeChart*, una de la biblioteca gráficas código abierto más difundida para Java. (7)

Esta herramienta fue desarrollada por la compañía *JasperSoft* (compañía especializada en software de Inteligencia de Negocios de código abierto) y es compatible con la salida en PDF, XML, XLS[10], CSV[11], HTML, pero no es compatible con la salida de XSL. Es multiplataforma, y se distribuye bajo la licencia GNU GPL[12].

El Diseñador de Modelos para el Generador Dinámico de Reportes (GDR) v1.8

El GDR está conformado por un conjunto de módulos que brindan las funcionalidades para dar soporte a todo el ciclo de vida de los reportes. El Diseñador de Modelos (DM) es uno de estos módulos. El diseñador permite gestionar orígenes de datos en varios gestores de base de datos y diseñar los modelos semánticos que serán utilizados en la confección de los reportes. En el ámbito del GDR, un modelo semántico es: una abstracción de la base de datos que contiene las entidades (tablas, vistas y rutinas). Almacena en forma de fichero XML toda la información de los metadatos de los objetos de la base de datos. Las principales características son:

- Aplicación desarrollada sobre el marco de trabajo *Symfony*.

---

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

[10] *Extensible Language Spreadsheets (lenguaje extensible de hojas de cálculo).*

[11] *Comma-separated values (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).*

[12] *General Public License (Licencia Pública General de GNU).*

- Escrita en el lenguaje de desarrollo PHP[13].
- Multiplataforma.
- Soporta imágenes y gráficas.
- Soporta varios orígenes de datos. (8)

Esta herramienta fue desarrollada por el centro productivo de desarrollo Centro de Tecnologías de Gestión de Datos (DATEC) de la UCI y cuenta con 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. (8)

Su entorno de trabajo está estructurado de forma tal que es difícil guiarse en el diseño de reportes. El DM del GDR aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, obliga al usuario a dominar conocimientos básicos de bases de datos, por tanto los usuario que no tengan estos conocimientos no podrán diseñar un documento y guardarlo en un fichero para su posterior reutilización.

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

El Diseñador de Plantillas forma parte del módulo Reportes del Sistema de Gestión Académica de Pregrado (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. (9)

El Diseñador de plantillas fue desarrollado sobre el marco de trabajo GUUD[14] v1.0 que es la unión del marco de trabajo CodeIgniter con jQuery. Está escrito en los lenguajes de PHP, JavaScript[15], CSS[16], HTML, XML, XSL y SQL[17]. El sistema no permite la integración con el Sistema de Impresión de Documentos ya que este está escrito en java y el diseñador de plantillas esta hecho sobre el marco de trabajo GUUD.

---

[13] *Hypertext Preprocessor (lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor).*

[14] *Acrónimo creado con las iniciales de los departamentos del antiguo centro CENIA: Gestión Universitaria, Universidad Digital y Gestión Documental.*

[15] *Lenguaje de programación interpretado.*

[16] *Cascading Style Sheets (hojas de estilo en cascada).*

[17] *Structured Query Language (lenguaje de consulta estructurado).*

1.3.1. Análisis de los diseñadores de plantillas estudiados

Los diseñadores de informes estudiados anteriormente presentan propiedades similares en lo que al diseño de documentos respecta, permitiendo que se generen con un diseño personalizado y enriquecido visualmente.

Active Reports es una herramienta que satisface determinadas necesidades de los clientes, pero una de sus desventajas principales es que al ser un software privativo no cumple con las políticas establecidas por el país para la migración a Software Libre (SL), por tanto no contribuye a obtener la soberanía tecnológica de la nación. De igual manera ocurre con el sistema *Crystal Reports* que tiene elevados costos adquisitivos.

iReport es el diseñador visual de código libre para *JasperReports* escrito en *Java*. Es utilizado para crear diseños muy complejos que contienen gráficos, imágenes, subinformes, tablas cruzadas y mucho más. No permite exportar la plantilla XSL, así como tampoco permite incorporar fuentes, siendo estos unos de los requisitos de la propuesta de solución.

El Diseñador de Modelos del GDR aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, obliga al usuario a dominar conocimientos básicos de base de datos y con la propuesta de solución lo que se busca es diseñar plantillas de una forma fácil.

El Diseñador de plantillas para el módulo Reportes del Sistema de Gestión Académica de Pregrado es una aplicación web implementada únicamente para el SGU, por lo que no es posible incorporarlo a ningún otro sistema.

Los sistemas estudiados anteriormente aunque permiten de una forma u otra diseñar plantillas presentan inconvenientes, por tal motivo surge la presente investigación para desarrollar una aplicación multiplataforma y que posea gran flexibilidad, permitiendo al usuario final el diseño de plantillas personalizadas para generar documentos acreditativos y que sea visto como un sistema independiente que genere plantillas en los formatos XML y XSL.

1.4. Entorno tecnológico utilizado para la implementación de la solución

A continuación se describen brevemente las tecnologías utilizadas para el desarrollo de la solución a partir de los requerimientos definidos para el desarrollo de un Diseñador de Plantillas para el Sistema de Impresión de Documentos en la Universidad de las Ciencias Informáticas. Las herramientas que se describen a continuación con sus versiones son las definidas a utilizar para el desarrollo de la solución por el Grupo de Arquitectura del Departamento de Desarrollo de la Dirección de Informatización, ya que estas herramientas son libres.

1.4.1. Lenguajes

Un lenguaje de programación en el ámbito de la informática es un elemento que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; permitiendo la comunicación tanto entre el programador, como el hardware y el software. A continuación se detallan los lenguajes utilizados para la implementación de la propuesta de solución.

1.4.1.1. Lenguaje de programación: *Java 7.0*

Java es un lenguaje de programación que puede ser ejecutado en múltiples plataformas. Es uno de los escasos lenguajes cuyos programas pueden ser transportados de sistema operativo, computadora o entorno, sin necesidad de cambiar el código. Esta característica ha sido la que alimentó su auge en la última década. Hasta la fecha, la plataforma *Java* ha atraído a más de 9 millones de desarrolladores de software. Este lenguaje es utilizado en los principales sectores industriales y está presente en una amplia gama de dispositivos, computadoras y redes. (10)

Algunas de las características principales que ofrece *Java* son:

- **Simple:** ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de estos. *Java* reduce en un 50% los errores más comunes de programación con lenguajes como C[18] y C++[19]. Además, el intérprete completo de *Java* que hay en este momento es muy pequeño, solamente ocupa 215 Kb de RAM[20].
- **Orientado a objetos:** trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. *Java* incorpora funcionalidades inexistentes en C++ como por ejemplo, la resolución dinámica de métodos.
- **Distribuido:** se ha construido con extensas capacidades de interconexión TCP/IP[21]. Existen librerías de rutinas para acceder e interactuar con protocolos como http[22] y ftp[23]. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.
- **Robusto:** realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en *Java* ayuda a detectar errores, lo antes posible,

[18] *lenguaje de programación orientado a la implementación de Sistemas Operativos, concretamente Unix.*

[19] *lenguaje de programación orientado a objetos derivado del lenguaje C.*

[20] *Random Access Memory (memoria de acceso aleatorio).*

[21] *Protocolo de Control de Transmisión / Protocolo de Internet.*

[22] *HyperText Transfer Protocol (protocolo de transferencia de hipertexto).*

[23] *File Transfer Protocol (protocolo de transferencia de archivos).*

en el ciclo de desarrollo. *Java* obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

- **Arquitectura neutral:** para establecer *Java* como parte integral de la red, el compilador *Java* compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (*run-time*) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.
- **Seguro:** en el lenguaje, características como los punteros o el *casting* implícito que hacen los compiladores de C y C++ se eliminan para prevenir el acceso ilegal a la memoria. Cuando se usa *Java* para crear un navegador, se combinan las características del lenguaje con protecciones de sentido común aplicadas al propio navegador.
- **Dinámico:** se beneficia todo lo posible de la tecnología orientada a objetos. *Java* no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. (10)

Este lenguaje les permite a los programadores tener fluidez con el lenguaje, debido a su diseño simple. Además de trabajar con sus datos como objetos y con interfaces a estos, proporciona las librerías y herramientas necesarias para que los programas puedan ser distribuidos.

1.4.1.2. Lenguaje de hojas de estilo extensible: XSL 1.0

XSL establece el lenguaje de estilo del documento XML permitiendo modificar el aspecto del mismo. Permite la visualización de tablas, tipos y tamaños de letra diferentes. Es más potente que las hojas de estilo CSS. Utiliza:

- XSLT (XML Stylesheets Transformation Language, o lenguaje de transformación basado en hojas de estilo) que es un lenguaje utilizado para convertir documentos XML en otros documentos XML o en otros formatos diferentes.
- XPath (XML Path Language) que es un lenguaje que permite navegar dentro de un documento XML y, para ello, permite construir expresiones que recorren y procesan el documento.
- XSL-FO (XSL Formatting Objects) que permite especificar el formato visual con el que se quiere presentar un documento XML. Se utiliza, sobre todo, para generar documentos PDF. (11)

Vale destacar que el lenguaje de hojas de estilo extensible no es un único lenguaje, sino toda una familia de recomendaciones del *World Wide Web Consortium* para expresar Hojas de estilos en lenguaje XML.

1.4.1.3. Lenguaje de marcas extensible: *XML 1.0*

XML es un metalenguaje que permite definir lenguajes de marcado con objetivos determinados. Además XML permite representar información estructurada en forma de documentos, de modo que pueda ser almacenada, transmitida, procesada, presentada e impresa por diferentes tipos de aplicaciones y dispositivos. (11)

Algunas de las características principales que presenta XML son:

- Permite la creación de etiquetas propias y permite asignar atributos a las etiquetas.
- Trabaja con los llamados DTDs[24], que a su vez estos contienen la estructura de los datos.
- Como consecuencia de los puntos anteriores, XML permite la creación de nuevos DTDs. Por ese motivo, entre otros, este metalenguaje se llama extensible.
- En un documento XML la estructura y el diseño están completamente separados.
- XML se almacena en formato texto (no binario) lo cual hace que los documentos sean directamente entendibles. Es decir, los documentos tienen una estructura entendible tanto por los ordenadores como por las personas.
- Permite la exportabilidad a otros formatos de publicación de datos (HTML, PDF, RTF, entre otros). (11)

Este metalenguaje permite que los diseñadores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones. La ventaja principal es que al ser extensible el diseñador puede añadir nuevas etiquetas tras el diseño del documento y facilita el procesamiento de los documentos XML creados por terceros.

1.4.1.4. Lenguaje de modelado: *UML 2.0*

Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. (12)

El lenguaje unificado de modelado es una de las herramientas más conocida y utilizada en el mundo actual del desarrollo de sistemas. Esto se debe a que permite a los creadores entender, diseñar, hojear, configurar,

[24] *Definición de Tipo de Documento (son archivos de texto cuyo contenido son las definiciones de las etiquetas y sus atributos con los que se puede trabajar en un determinado documento)*

mantener y controlar la información sobre los sistemas permitiéndole generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas. (12)

Es válido señalar que no es un lenguaje de programación, sino que es un lenguaje de propósito general para el modelado orientado a objetos. Se ha convertido en la herramienta ideal para modelar el ciclo de vida de un proyecto de software utilizando la tecnología orientada a objetos. También puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes.

1.4.1.5. Lenguaje de consulta estructurado: SQL

SQL es un lenguaje estándar de comunicación con bases de datos. El hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos. En efecto, determinadas bases de datos implementan funciones específicas que no tienen necesariamente que funcionar en otras. Se habla por tanto de un lenguaje normalizado que permite trabajar con cualquier tipo de lenguaje como PHP en combinación con cualquier tipo de base de datos (*MS Access*, *SQL Server*, *MySQL*). (13)

El LDD[25] de SQL tiene comandos para el borrado de relaciones y modificaciones de los esquemas de relación. Además de incluir comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos y para definir las vistas, tiene comandos para especificar el comienzo y el final de una transacción.

1.4.2. Herramientas

A continuación se describen las herramientas que se utilizaron para la implementación de la propuesta de solución.

1.4.2.1. Entorno de Desarrollo Integrado (IDE): NetBeans 7.4

NetBeans IDE 7.4 es un IDE gratuito de código abierto para Oracle Solaris, Oracle Linux, otras distribuciones de Linux, Mac y Windows que permite a los desarrolladores crear rápidamente aplicaciones Web, empresariales, de escritorio y móviles utilizando principalmente Java, así como PHP y C/C++. Ofrece mejoras considerables para crear interfaces gráficas de usuario *Swing* (*Swing GUI Builder*), admite CSS3 y herramientas para la depuración visual de interfaces de usuario (IU). (14)

Este IDE admite múltiples tecnologías clave como:

[25] *Data Definition Language (lenguaje de definición de datos).*

- Java: las herramientas de refactorización a granel (*bulk refactoring*) permiten correcciones y actualizaciones de proyecto en el código. Un nuevo depurador visual ayuda a controlar el flujo de ejecución en términos de estructuras de nivel en la IU.
- Lenguajes Web: admite CSS3 en el editor *NetBeans* CSS, con completación de código, resaltado de sintaxis y documentación para los nuevos elementos CSS3, y nuevas configuraciones de propiedades específicas de navegador.
- Control de versión y rastreo de *Bug*: soporte innovador integrado de *Git*, la nueva solapa de historia en la ventana del editor de archivos muestra el historial de cambios, archivos de bloqueo y desbloqueo en el directorio de trabajo *Subversion*, y soporte branches y *tags* de mercurial. (14)

Este es un producto libre y gratuito sin restricciones de uso. El código fuente está disponible para su reutilización de acuerdo con la CDDL[26] vl.0 y la GNU GPL v2, por lo que cumple con las políticas de migración a Software Libre que está impulsando el país.

1.4.2.2. Herramienta CASE: Visual Paradigm para UML 8.0

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML. Es ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de Sistemas que tiene como objetivo la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. *Visual Paradigm* también ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Ambiente visualmente superior de modelado.
- Sincronización de código fuente en tiempo real. (15)

Esta herramienta CASE es multiplataforma y su licencia se encuentra disponible de forma gratuita solo con fines académicos y no comerciales. Además brinda un completo conjunto de herramientas de equipos de desarrollo de software necesario para la captura de requisitos, software de planificación y el modelado de datos. También proporciona abundantes tutoriales de UML.

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

Evolus Pencil es una herramienta gratuita, de código abierto, multiplataforma y permite la creación de prototipos. Desarrollada por la compañía *Evolus Co*. con el propósito de proporcionar una herramienta de prototipado que las personas puedan instalar y utilizar para crear maquetas en plataformas de escritorio

[26] *Common Development and Distribution License (licencia común de desarrollo y distribución).*

populares fácilmente. Brinda la capacidad de editar texto enriquecido y trabajar a través de procedimientos estándares de dibujo, tales como alineación**,** escalado**,** rotación y dimensión. Permite exportar el proyecto de manera directa en un solo archivo, contando con diferentes formatos como pdf, html y png. (16)

Esta herramienta de creación de prototipos de interfaz de usuario es una de las más usadas en el Sistema de Gestión Académica de Pregrado por tal motivo es la que más conocimientos tiene todo el personal ante cualquier duda que se presente y cumple con la políticas de migración a Software Libre.

1.4.2.4. Sistema Gestor de Bases de Datos: PostgreSQL 8.4

PostgreSQL es un Sistema de Gestión de Bases de Datos (SGBD) objeto-relacional, distribuido bajo licencia BSD[27] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. (17)

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (17)

Entre las características más importantes y soportadas por este sistema gestor de bases de datos se destacan que es unicode, realiza copias de seguridad en caliente (*Online/hot backups*), tiene una documentación completa, y está disponible para Linux y UNIX[28] en todas sus variantes y Windows 32/64bit. Sirve de soporte a muchos de los lenguajes más utilizados hoy día, tal es el caso de PHP, C, C++, *Java*, *Python*, entre otros. Es dirigido por una comunidad de desarrolladores y organizaciones comerciales la cual trabaja en su desarrollo y perfeccionamiento, esta comunidad es denominada *PostgreSQL Global Development Group*.

1.4.2.5. Administrador de base de datos: PgAdmin III 1.12

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos postgresql, siendo la más completa y popular con licencia *Open Source*. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico 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

[27] *Berkeley Software Distribution (distribución de software Berkeley, sistema operativo derivado del sistema Unix).*

[28] *Sistema operativo portable, multitarea y multiusuario.*

servidor, un agente para lanzar *scripts* programados. La conexión al servidor puede hacerse mediante conexión TCP/IP y puede encriptarse mediante SSL[29] para mayor seguridad. (18)

Es un software libre publicado bajo la licencia *PostgreSQL*. Este administrador de base de datos se encuentra disponible en más de una docena de lenguajes y para varios sistemas operativos como: Windows, Linux, entre otros. Esta desarrollado por una comunidad de expertos de *PostgreSQL* en todo el mundo.

1.4.2.6. Herramienta para realizar pruebas: JUnit 3.8.2

Es un *Framework Open Source* para la automatización de las pruebas tanto unitarias como de integración en los proyectos de software. Provee al usuario de herramientas, clases y métodos que facilitan la realización de las pruebas, de modo que asegura la consistencia y funcionalidad del sistema. En función de algún valor de entrada se evalúa el valor de retorno esperado, si la clase cumple con la especificación entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. (19)

Esta herramienta Open Source está hecha para la realización de pruebas en la fase de desarrollo para el entorno *Java*, por lo que la convierte en la más utilizada en entornos *Java*. Además los principales IDEs de desarrollo (*Eclipse*, *NetBeans*) incorporan *plugins* para su utilización.

1.4.2.7. Plataforma de desarrollo: GNU/Linux, Ubuntu 12.4

GNU/Linux es un sistema operativo multitarea, multiusuario y multiplataforma. Pueden ejecutarse varias tareas al mismo tiempo y varios usuarios conectados al mismo tiempo cada uno ejecutando sus propias aplicaciones. Además puede ejecutarse en una gran variedad de microprocesadores y plataformas distintas. (20)

Es multitarea ya que varios programas (realmente procesos) se ejecutan al mismo tiempo, es multiplataforma ya que funciona en muchas CPUs[30] distintas, no solo en Intel, funciona en modo protegido 386. Tiene protección de la memoria entre procesos, de manera que uno de ellos no pueda colgar el sistema. Linux solo lee de disco aquellas partes de un programa que están siendo usadas actualmente. Tiene una política de copia en escritura para la compartición de páginas entre ejecutables: esto significa que varios procesos pueden usar la misma zona de memoria para ejecutarse. Cuando alguno intenta escribir en esa memoria, la página (4Kb de memoria) se copia a otro lugar. Esta política de copia en escritura tiene dos beneficios: aumenta la velocidad y reduce el uso de memoria. (20)

[29] *Secure Sockets Layer (protocolo de capa de conexión segura).*

[30] *Central Processing Unit (Unidad Central de Procesamiento).*

Es usada esta plataforma para el desarrollo de la solución ya que es un sistema operativo de código abierto. Cualquiera puede disponer de sus fuentes, modificarlas y crear nuevas versiones que puede compartir bajo la licencia GPL. Además cumple con las políticas de migración a Software Libre para lograr la independencia tecnológica.

1.4.3. Proceso de desarrollo de software: Desarrollo Ágil con Calidad

La metodología Desarrollo Ágil con Calidad (DAC) es un proceso de desarrollo de software que combina las metas y prácticas de las áreas de procesos del nivel 2 de CMMI[31] con las buenas prácticas de la dirección y desarrollo ágil de proyectos de software. Es un proceso colaborativo, recursivo-iterativo, incremental y guiado por procesos y requisitos. Su modelo del proceso es una adaptación del modelo en Cascada a los modelos Programación Extrema y Desarrollo Concurrente. Está enfocado a proyectos pequeños o proyectos grandes divididos en sub-proyectos que desarrollan software de gestión basado en componentes. (21)

Este proceso tiene 8 actividades del marco de trabajo del proceso común llamadas Fases o Procesos del Ciclo de Vida: Inicio, Análisis y Diseño Arquitectónico, Requisitos, Construcción, Cierre de iteración (opcional), Liberación, Transición y Cierre, ocurriendo las iteraciones concurrentes entre las fases de Requisitos, Construcción y Cierre de iteración. Además entre las fases de Requisitos y Construcción puede ocurrir un ciclo pues a medida que los requisitos son especificados estos pueden ir entrando a la fase de Construcción. El proceso tiene también dos Áreas de Procesos de Protección: Gestión de proyectos y Soporte así como dos Fases o Procesos Horizontales cuyas tareas están presentes en varias de las fases del proceso común en forma de subprocesos: Arquitectura y Planificación. En la Figura 2 se puede ver el modelo del proceso DAC. (21)

[31] *Capability Maturity Model Integration (Modelo de Capacidad y Madurez Integrado).*

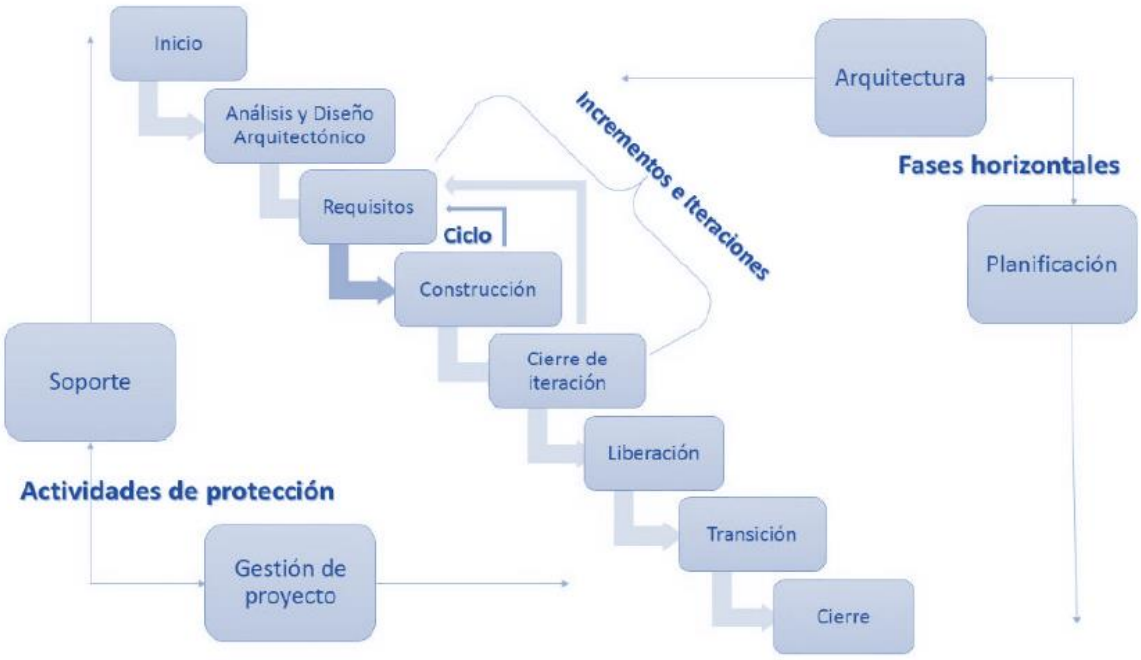


Figura 1. Modelo del proceso DAC.

El proceso de desarrollo de software planteado anteriormente es la metodología a seguir por los proyectos que se desarrollan con un enfoque ágil en el Departamento de Desarrollo de la Dirección de Informatización de la UCI. Esta metodología incluye a grandes rasgos la especificación de las actividades y tareas de cada una de las fases del ciclo de vida de los proyectos teniendo en cuenta los procesos de CMMI nivel 2 para la UCI, en fin permite detallar los artefactos a generar en cada momento del ciclo de vida del proyecto.

Conclusiones parciales

En el capítulo quedan caracterizados los principales elementos de la fundamentación teórica de la investigación. El estudio realizado a los diferentes sistemas homólogos existentes tanto en el ámbito internacional como nacional, permitió obtener conocimientos y puntualizar las características fundamentales de dichos sistemas para tenerlos en cuenta en el desarrollo de la propuesta de solución y así incorporarles nuevas ideas que faciliten la creación de plantillas personalizadas. Además el estudio de las tecnologías seleccionadas por el Grupo de Arquitectura del Departamento de Desarrollo de la Dirección de Informatización permitió un acercamiento a los elementos del entorno de desarrollo y sus características, para hacer más fácil su uso durante el desarrollo de la propuesta de solución.

Capítulo 2: Descripción de la propuesta de solución

En el presente capítulo se caracteriza la propuesta de solución al problema planteado. Se expone las clases del dominio involucradas determinando las personas que interactúan con el sistema y la responsabilidad asignada a cada uno. Se especifican las técnicas utilizadas para la obtención de requisitos del diseñador de plantillas. Además se describe la arquitectura, los patrones de diseño y de base de datos a utilizar, así como el modelo físico de la base de datos y el diagrama de despliegue. También se detalla los diferentes tipos de mensajes que muestra la propuesta de solución y las vistas principales.

2.1. Modelo conceptual

Un modelo conceptual es un conjunto de conceptos y de reglas destinados a representar de forma global los aspectos lógicos de los diferentes tipos de elementos existentes en la realidad que está siendo analizada. Es una representación figurada de una experiencia empírica, que tiene como objetivo ayudar a comprender la realidad. Son marcos o estructuras que representan la realidad, pero no son la realidad sino su abstracción. (22)

La propuesta que se presenta permite diseñar plantillas de documentos acreditativos para exportar dichas plantillas en los formatos XML y XSL. Como dicha propuesta no tiene un modelo de negocio bien definido propicia el uso del modelo conceptual; permitiendo identificar las relaciones entre todos los conceptos comprendidos en el ámbito sometido al análisis. A continuación en la Figura 2 se muestra la representación del modelo conceptual a partir de las entidades más importantes y sus relaciones.

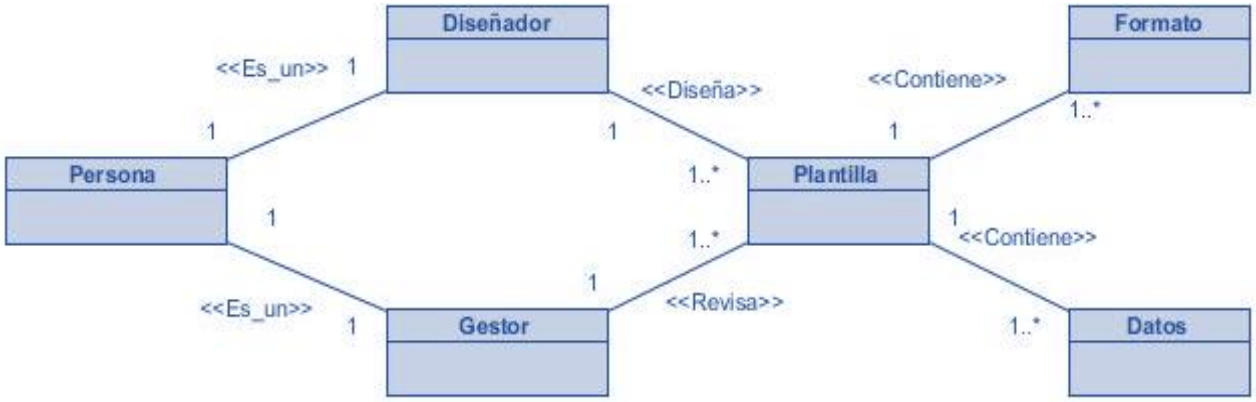


Figura 2. Diagrama de representación del modelo conceptual.

Las personas con el rol de diseñador tienen la tarea de diseñar plantillas de documentos acreditativos y el gestor de revisar las plantillas previamente diseñadas. Estas plantillas diseñadas contienen un formato definido por el diseñador y datos.

2.1.1. Descripción de las entidades del modelo conceptual

Para una mejor compresión, a continuación se describen cada uno de los conceptos identificados durante el modelado de conceptualización de la realidad que está siendo analizada.

- **Persona:** representa la entidad que contiene todos los datos de los individuos involucrados en el diseño y gestión de plantillas de documentos acreditativos.
- **Diseñador:** es el responsable de realizar el diseño de las plantillas de los documentos acreditativos.
- **Gestor:** es el responsable de revisar las plantillas de los documentos acreditativos previamente diseñados.
- **Plantilla:** modelo que define la presentación de la estructura del documento acreditativo.
- **Formato:** representa la entidad que contiene la definición del estilo de la plantilla previamente diseñada.
- **Datos:** representa la entidad que contiene las etiquetas almacenadas de la plantilla previamente diseñada.

2.2. Presentación de la propuesta de solución

El diseñador de plantillas es una aplicación de escritorio que permite al usuario gestionar la seguridad del sistema y diseñar una plantilla para un documento acreditativo a generar, permitiéndole también modificar la apariencia de dichos documentos y visualizarlos, con un área de trabajo fácil de usar. Esto proporciona la interacción de los usuarios con la aplicación, sin que estos tengan conocimientos informáticos avanzados. En este diseñador de plantillas el usuario puede adicionar y eliminar componentes tales como: imágenes, etiquetas de texto y áreas de texto, así como modificar las propiedades de acuerdo a cada componente. También se pueden generar los respectivos XML y XSL asociados a cada plantilla.

Estas plantillas pueden ser exportadas en un espacio destinado en el servidor del Sistema de Gestión Universitaria para su posterior uso por el Sistema de Impresión de Documentos, el cual se nutre de los datos y configuraciones gestionados en el módulo Documentos Acreditativos del SGU.

Las plantillas diseñadas se pueden guardar para su posterior utilización. El área de diseño se visualiza en el formato definido por el usuario de acuerdo al tipo de hoja y se le puede incorporar diferentes tipos de fuentes al diseñador de plantillas. En la Figura 3 se visualiza las partes que integran la propuesta de solución.

Figura 3. Mapa conceptual de la propuesta de solución.

2.3. Roles que interactuarán con la propuesta de solución

El diseñador de plantillas será utilizado por usuarios que de una forma u otra interactuarán con el sistema, ya sea para su configuración o la utilización del sistema en sí para el diseño de plantillas. El diseñador contará con 3 roles:

- ***Administrador del sistema:*** realiza las funcionalidades de seguridad y configuración del sistema en general.
- ***Diseñador:*** realiza solamente el diseño, modificación y visualización de las plantillas de los documentos acreditativos.
- ***Gestor:*** realiza la impresión de las plantillas, puede revisar la calidad y hacer observaciones a las mismas, así como exportar a los formatos XML y XSL dichas plantillas.

2.4. Requisitos de la propuesta de solución

Los requisitos del software son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un sistema que ayude

a resolver algún problema, en fin, un requisito del software es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. (23)

2.4.1. Técnicas de obtención de requisitos

En el proceso de desarrollo de software el área del conocimiento de gran importancia es la Ingeniería de Requisitos (IR). Esta área comprende las actividades de obtención, análisis, especificación, y validación de requisitos con el objetivo de proporcionar el mecanismo adecuado para entender y poder llevar a la luz los requisitos que el cliente desea. 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. La IR ofrece numerosas técnicas para identificar y extraer los requisitos. A continuación se describen las técnicas utilizadas:

- ***Sesiones de tormentas de ideas (Brainstorming):*** esta técnica de reuniones en grupo se puso en práctica entre el cliente y el autor de la propuesta de solución con el objetivo de generar varias ideas y opiniones al desarrollar la creatividad en un ambiente libre de críticas. También contribuyó a generar una gran variedad de vistas del problema y a formularlo de diferentes formas al inicio del proceso, cuando los requisitos son todavía muy difusos. (24)
- ***Prototipos:*** un prototipo es una versión inicial del sistema de software que le permite a los usuarios del sistema experimentar, es decir ver cómo este ayuda a su trabajo. Además de contribuir al desarrollo de ideas que convergen a nuevos requerimientos, permite mostrar las funciones del sistema. También se identifican las discrepancias entre los desarrolladores y los usuarios; y aunque limitado, se dispone rápidamente de un sistema que funciona y demuestra la factibilidad y usabilidad de la aplicación. (24)
- ***Observación directa (Etnografía):*** esta técnica de observación se utiliza para entender los requerimientos sociales y organizacionales. Mediante la aplicación de esta técnica el analista se sumerge por sí mismo en el entorno laboral donde la propuesta de solución se utilizará y con el trabajo diario a través de la observación detectará las tareas reales en las que los usuarios están involucrados. Esta técnica es especialmente efectiva para los requerimientos que se derivan de la forma en la que las personas trabajan realmente y en los requerimientos que se derivan de la cooperación y conocimiento de las actividades de las personas. (24)

2.4.2. Requisitos funcionales de alto nivel

Los requisitos funcionales de alto nivel (RFAN) son los requisitos que se dividirán posteriormente en requisitos funcionales y no funcionales, permitiendo así obtener los requisitos a implementar en la propuesta de solución. A continuación se muestran en la Tabla 1 los RFAN definidos para el diseñador de plantillas.

Tabla 1. Requisitos funcionales de alto nivel.

No.	Descripción
RFAN1	Diseñar plantillas de acuerdo al formato y los campos que la componen.
RFAN2	Visualizar plantillas diseñadas.
RFAN3	Revisar plantillas diseñadas para su aceptación o rechazo.
RFAN4	Controlar el acceso al sistema mediante roles y usuarios.
RFAN5	Exportar las plantillas y sus componentes a un servidor local.
RFAN6	Configurar las opciones del sistema para las diferentes conexiones que se realizan.
RFAN7	Imprimir plantillas diseñadas.

2.4.2.1. Requisitos funcionales de la propuesta de solución

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (23)

En la Tabla 2 se muestran los requisitos funcionales definidos que debe cumplir el diseñador de plantillas.

Tabla 2. Requisitos funcionales.

No.	Descripción	Complejidad	Prioridad
RF1	Autenticar usuario.	Baja	Alta
RF2	Cambiar contraseña.	Baja	Baja
RF3	Configurar sistema externo.	Baja	Baja
RF4	Configurar base de datos.	Baja	Baja
RF5	Crear rol.	Baja	Alta
RF6	Modificar rol.	Baja	Alta
RF7	Eliminar rol.	Baja	Alta
RF8	Ver detalles de rol.	Baja	Alta
RF9	Crear usuario.	Baja	Alta
RF10	Modificar usuario.	Baja	Alta
RF11	Eliminar usuario.	Baja	Alta

RF12	Ver detalles de usuario.	Baja	Alta
RF13	Listar rol.	Baja	Media
RF14	Listar usuario.	Baja	Media
RF15	Definir primera configuración del sistema.	Media	Media
RF16	Insertar componente de tipo texto.	Alta	Alta
RF17	Modificar componente de tipo texto.	Alta	Alta
RF18	Eliminar componente de tipo texto.	Media	Alta
RF19	Insertar componente de tipo área de texto.	Alta	Alta
RF20	Modificar componente de tipo área de texto.	Alta	Alta
RF21	Eliminar componente de tipo área de texto.	Media	Alta
RF22	Insertar componente de tipo tabla.	Alta	Alta
RF23	Modificar componente de tipo tabla.	Alta	Alta
RF24	Eliminar componente de tipo tabla.	Media	Alta
RF25	Insertar componente de tipo imagen.	Alta	Alta
RF26	Modificar componente de tipo imagen.	Alta	Alta
RF27	Eliminar componente de tipo imagen.	Media	Alta
RF28	Crear plantilla.	Alta	Alta
RF29	Modificar plantilla.	Alta	Alta
RF30	Eliminar plantilla.	Baja	Alta
RF31	Listar plantilla del diseñador.	Baja	Media
RF32	Ver detalles de plantilla.	Baja	Alta
RF33	Configurar márgenes de la página de la plantilla.	Baja	Media
RF34	Configurar orientación de la página de la plantilla.	Baja	Media
RF35	Configurar tamaño de la página de la plantilla.	Baja	Media
RF36	Exportar plantilla en espacio local.	Alta	Alta
RF37	Exportar plantilla en servidor.	Alta	Alta
RF38	Insertar fuentes.	Media	Alta
RF39	Visualizar la plantilla.	Alta	Alta
RF40	Listar plantilla del gestor.	Baja	Media
RF41	Revisar plantilla.	Media	Media
RF42	Imprimir plantilla.	Alta	Media
RF43	Insertar fila al componente tipo tabla.	Alta	Media
RF44	Eliminar fila al componente tipo tabla.	Media	Alta
RF45	Insertar columna al componente tipo tabla.	Media	Alta
RF46	Eliminar columna al componente tipo tabla.	Media	Alta

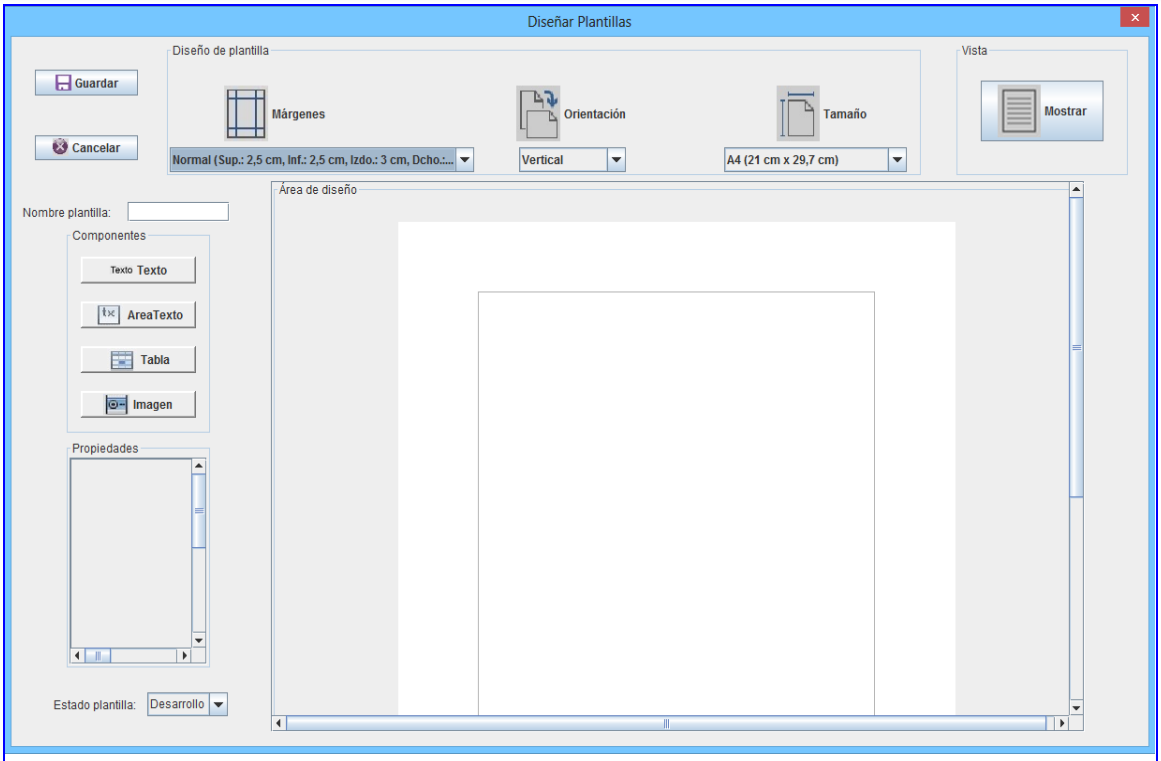
RF47	Alinear texto del componente texto y área de texto.	Media	Alta
RF48	Guardar plantilla.	Alta	Alta

2.4.2.2. Descripción textual y prototipo de los requisitos funcionales

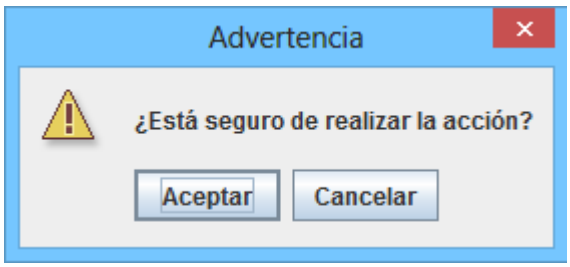
La descripción textual y prototipo del requisito contiene la información necesaria que ayuda a los desarrolladores del software a analizar y entender los requisitos y requerimientos que nuestro cliente desea, en el mismo se describe lo que realmente se desea obtener, y de esta manera lograr tener un documento cuya información sirva para el desarrollo del software. A continuación se muestra en la Tabla 3 una de las descripciones de requisitos, para consultar las restantes más importantes ir a los **Anexos (Tabla 10-25)**.

Tabla 3. Descripción textual y prototipo del requisito funcional Guardar plantilla.

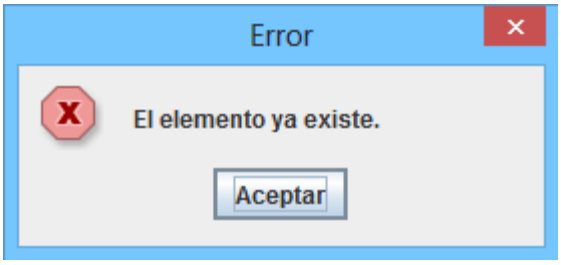
Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RFDP_48	Guardar plantilla.	• El requisito permite guardar una plantilla. • El Diseñador diseña la plantilla, llena el campo del nombre de la plantilla y puede cambiar el estado de la plantilla y luego se guarda la plantilla o cancela la acción.	Alta	Alta
	• **Prototipo**			



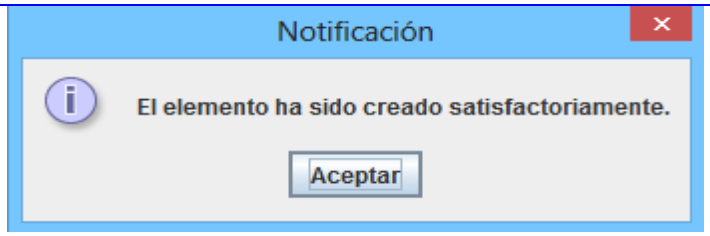
Prototipo 1



Prototipo 2



Prototipo 3



Prototipo 4

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.
• Estado	• character varying	• Campo de selección. Campo requerido.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 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. 3. En caso de que se escriba apóstrofos y porcientos se muestra un mensaje de error de color rojo "No se admite apóstrofos ni porcientos.". 4. En caso de que la plantilla exista se muestra un mensaje de error: “El elemento ya existe”. 5. Si ocurre un error durante la operación se muestra un mensaje de error con el título: “Error” indicando el error que produjo el problema. 6. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 7. En caso de cancelar la acción se muestra un mensaje de confirmación “¿Está seguro de realizar la acción?” 8. Si se crea satisfactoriamente el elemento se mostrará un mensaje de información “El elemento ha sido creado satisfactoriamente.”	

2.4.2.3. Requisitos no funcionales de la propuesta de solución

Los requisitos no funcionales (RNF) son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no

funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. (23)

En la Tabla 4 se muestran los requisitos no funcionales establecidos que debe cumplir el diseñador de plantillas.

Tabla 4. Requisitos no funcionales.

Usabilidad	
RNF1	Facilidad de uso por parte de los usuarios: el sistema debe presentar una interfaz que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.
RNF2	Emplear perfiles de usuario: diferenciar las interfaces y opciones para los usuarios que accedan al sistema según los diferentes roles que estos tengan dentro del sistema.
RNF3	Especificación de la terminología utilizada: el sistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.
Seguridad	
RNF4	Seguridad de la base de datos: la base de datos deberá estar fraccionada en esquemas que permitan un mejor uso de la información y la división de forma lógica de las funcionalidades del sistema, trayendo consigo además la protección de la información al ocurrir un incidente sobre una parte de la base de datos. El SGBD escogido debe presentar facilidades de administración de roles y usuarios restringiendo el acceso a los datos.
RNF5	Políticas de seguridad por usuarios y roles: el sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.
Soporte	
RNF6	Grupo de soporte y asesoría: el sistema contará con un grupo de soporte y asesoría al cliente del producto destinado a brindar asesoría y soporte técnico al mismo.
Restricciones del diseño	
RNF7	El Sistema Gestor de Bases de Datos que la propuesta de solución utilizará es PostgreSQL 8.4.
RNF8	El diseño de la base de datos se realiza con la herramienta CASE Visual Paradigm 8.0.
RNF9	Se utilizará como Entorno de Desarrollo Integrado el NetBeans 7.4.

RNF10	La plataforma de desarrollo en la cual se desarrollará el sistema es GNU/Linux y el sistema operativo Ubuntu 12.4.
RNF11	El sistema debe ser implementado en el lenguaje de programación de *Java*.
Requisitos para la documentación de usuarios en línea y ayuda del sistema	
RNF12	Manual de usuario: el sistema deberá presentar un manual de usuario, permitiendo con ello un correcto uso de sus funcionalidades y brindarle al usuario una mayor experiencia del trabajo con el mismo.
RNF13	Documentación actualizada del grupo de desarrollo: se precisa que la documentación del sistema esté actualizada en todos los aspectos, fases de trabajo y ciclos de desarrollo del mismo, permitiendo con ello un respaldo tanto ingenieril como legal del desarrollo de dicho sistema.
Interfaz	
RNF14	Interfaz de escritorio: la interfaz deberá ser sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo de su empleo.
RNF15	Interfaz interna: la interfaz interna estará determinada por los desarrolladores, construyendo así una vista escalable de las clases o agrupaciones de clases que permitirán un mejor encapsulamiento de las funcionalidades y una mayor abstracción modular del sistema.
Hardware	
RNF16	Para el desarrollo: PC Intel Pentium 4 o superior, un CPU de 3GHZ o superior, 2 GB de memoria RAM o superior y 160 GB de HDD o superior.
RNF17	Para explotación del cliente: PC Intel Pentium 4 o superior, un CPU de 3GHZ o superior, un 1GB de memoria RAM o superior y 1GB de HDD disponible para su instalación.
Software	
RNF18	El componente reutilizado de otra aplicación que fue necesario en el sistema fue el componente Seguridad del Sistema de Impresión de Documentos.
RNF19	El sistema deberá ser soportado en una PC cliente con los sistemas operativos Windows XP o superior y GNU/Linux con ambiente gráfico.
RNF20	La PC cliente deberá tener instalada la máquina virtual de java versión 6 o superior.
Estándares Aplicables	
RNF21	Referirse al documento de arquitectura: 010216_2_Arquitectura_vista_de_sistema_DAC.doc (en el mismo se especifican los estándares aplicables)

2.5. Descripción de la arquitectura

La arquitectura de software hace alusión a la especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos; los requerimientos que debe satisfacer el sistema y las restricciones a las que está sujeto, así como las propiedades no funcionales del sistema y su impacto sobre la calidad del mismo; las reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas. (25)

2.5.1. Arquitectura

La arquitectura utilizada es cliente/servidor. Esta es un modelo para el desarrollo de sistemas de información, en el cual las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Esta arquitectura consiste en que el 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. La separación entre clientes 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.

Cliente: conjunto de software y hardware que invoca los servicios de uno o varios servidores. Los clientes interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad. Entre sus funciones comunes se destacan:

- Mantener y procesar todo el diálogo con el usuario.
- Menús e interpretación de comandos.
- Entrada de datos y validación.
- Procesamiento de ayudas.
- Recuperación de errores.
- Generación de consultas e informes sobre las bases de datos. (26)

Servidor: conjunto de hardware y software que responde a los requerimientos de un cliente. Los servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallas, y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Entre sus funciones comunes se destacan:

- Acceso, almacenamiento y organización de datos.
- Actualización de datos almacenados.
- Administración de recursos compartidos.
- Ejecución de toda la lógica para procesar una transacción.
- Procesamiento común de elementos del servidor (datos, capacidad de CPU, almacenamiento en disco, capacidad de impresión, manejo de memoria y comunicación).
- Gestión de periféricos compartidos. (26)

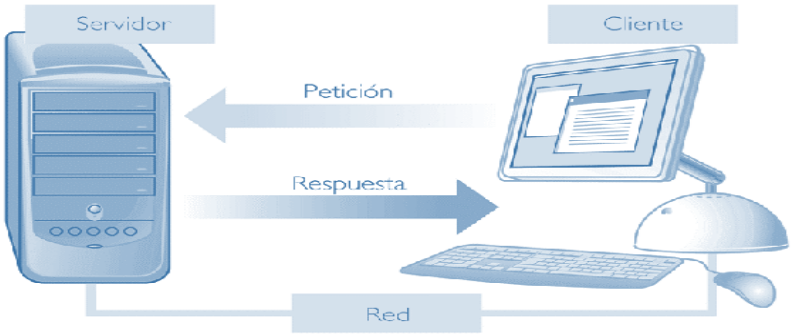


Figura 4. Arquitectura cliente-servidor.

2.5.2. Patrón de arquitectura

El patrón arquitectónico describe la estructura fundamental del sistema de software, expresando un conjunto de subsistemas predefinidos, especificando responsabilidades y organizando las relaciones entre ellos. Su importancia radica en que es la decisión fundamental de diseño cuando se desarrolla un sistema de software.

El patrón arquitectónico utilizado es Modelo-Vista-Controlador (MVC). Este es un patrón que define la organización independiente del Modelo, la Vista y el Controlador; esto permite dividir el sistema en tres capas donde, se tiene la encapsulación de los datos a un lado, la interfaz o vista por otro y por último la lógica interna o controlador. A continuación se explica cada uno de los elementos identificados:

- **Modelo:** puede visualizarse como la capa que conecta la aplicación con los datos, es decir representa toda la información con la que opera la aplicación. También responde a las peticiones de información sobre el estado, que vienen de la Vista o a las instrucciones de cambio de estado provenientes del Controlador, en fin se encarga de realizar todo el tratamiento sobre los datos que la aplicación manejará.

- **Vista:** es la capa encargada de mostrar los datos al usuario o de requerir datos. Esta capa gestiona todo lo relativo a la parte visible de la aplicación (interfaz de usuario) con la que el usuario interactúa y ejecuta las acciones disponibles.
- **Controlador:** es la capa crucial de la aplicación. En esta capa se aplica la lógica de la aplicación misma. El controlador se encarga de recibir peticiones del usuario, procesar, invocar métodos necesarios definidos en el modelo y retornar los datos al usuario por medio de la vista, en fin esta capa interpreta las entradas del usuario, informando al modelo y/o a la vista de los cambios que supongan esas entradas. (27)

El HMVC (por sus siglas en inglés, Herarchical-Model-View-Controller) es una variante mejorada del MVC que se forma mediante una colección de estos, siendo cada MVC independiente de los otros, y siendo un aspecto importante la reutilización de código, por lo que la localización física de los MVC no es importante. El HMVC es muy efectivo ya que reduce las dependencias entre las distintas partes de la aplicación y permite que la aplicación sea más extensible al añadir o remover módulos sin sacrificar la facilidad de mantenimiento.

Debido a las características mencionadas se seleccionó este patrón para aplicarlo a la arquitectura del sistema quedando diseñado en la Figura 5 de la forma siguiente:

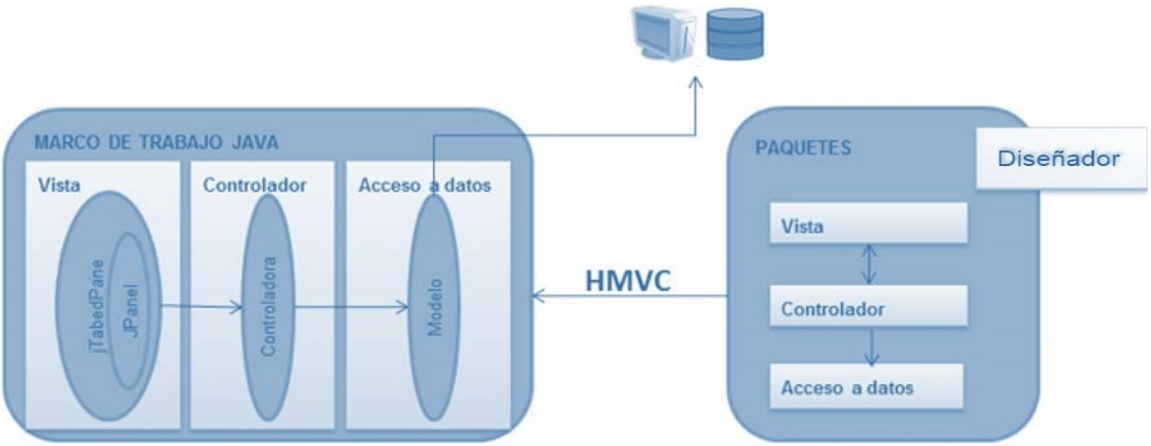


Figura 5. Arquitectura cliente-servidor. Tomado de Arquitectura del Software SGU-IVES 1.1.

2.5.3. Diagrama de despliegue

El diagrama de despliegue es utilizado para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes, es decir modela la arquitectura en tiempo de ejecución del sistema desde el punto de vista del despliegue (distribución) de los artefactos del software en los destinos de despliegue; como se observa en la Figura 6 estos destinos de despliegue generalmente están

representado por un nodo que son los elementos de hardware y muestra cómo los elementos y artefactos del software se trazan en esos nodos.

Figura 6. Diagrama de despliegue del sistema.

A continuación se describen cada uno de los elementos e interfaces de comunicación:

PC Cliente: estará instalada el sistema, es decir el Sistema Diseñador de Plantillas.

Servidor de Bases de Datos: estará el Sistema Gestor de Base de Datos con la información del sistema.

Servidor de Plantillas: se almacenarán las plantillas previamente diseñadas.

TCP/IP: es un conjunto de protocolos y representan todas las reglas de comunicación para Internet y se basa en la noción de dirección IP. En ocasiones se le denomina 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. (28)

FTP: un protocolo para transferir archivos. Entre las operaciones más frecuentes se encuentra la copia de ficheros de una máquina a otra. La transferencia de datos entre cliente y servidor puede producirse en cualquier dirección. El cliente puede enviar o pedir un fichero al servidor. Este protocolo permite que los equipos remotos puedan compartir archivos y permite también la independencia entre los sistemas de archivo del equipo del cliente y del equipo del servidor. (29)

2.5.4. Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Entre las que se destaca es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, debe ser reusable, establecer un vocabulario de diseño común entre los desarrolladores y debe acotar la fase de diseño en un proceso de desarrollo de software. (30)

A continuación se verán los patrones de asignación de responsabilidades (GRASP[32]) y los patrones GoF[33] que se tuvieron en cuenta para el diseño del sistema:

[32] *por sus siglas en inglés, General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades).*

[33] *por sus siglas en inglés, Gang of Four (Pandilla de los cuatro).*

Patrones de asignación de responsabilidades (GRASP)

Los GRASP son patrones generales de software para asignación de responsabilidades. Describen los principios fundamentales para asignar responsabilidades a los objetos, este sistema orientado a objetos se compone de objetos que envían mensajes a otros objetos para que lleven a cabo las operaciones requeridas, es decir las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento. Para el diseño de la propuesta de solución de los patrones GRASP se tuvo en cuenta: Alta cohesión, Creador, Controlador, Experto y Bajo acoplamiento. A continuación se describen los patrones mencionados anteriormente:

- **Experto:** cada objeto es responsable por mantener su propia información, conoce y puede informar el valor de sus atributos, así como modificar el valor de sus atributos. El objetivo es asignar responsabilidades al experto en información, que es la clase que tiene la información necesaria para cumplir esta responsabilidad. Su uso se evidencia en las clases modelos que son las que tienen el acceso a datos.
- **Creador:** guía la asignación de responsabilidades al identificar quién debe ser el responsable de la creación de nuevos objetos o clases. Este patrón se utilizó para identificar qué clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B. Su uso se evidencia en la clase vista_adicionar_plantilla ya que en esta se crean los componentes.
- **Alta cohesión:** asignar una responsabilidad de modo que la cohesión siga siendo alta. Este patrón plantea que la información que almacena una clase debe ser coherente y debe estar relacionada con la clase. El uso de este patrón se evidencia en la clase miEncriptador que cuenta únicamente con las funcionalidades necesarias para encriptar palabras.
- **Bajo acoplamiento:** asignar una responsabilidad para mantener bajo acoplamiento. La idea de este patrón es tener las clases lo menos ligadas entre sí que se pueda. De tal forma, que se realiza una modificación en alguna de ellas, no repercuta en el resto de las clases o tenga la más mínima repercusión sobre estas. Su uso se evidencia en la clase ComponenteArrastable que no depende de otras para su funcionamiento.
- **Controlador:** es utilizado como intermediario entre cada una de las capas, de tal forma que recibe los datos del usuario y los envía a las diferentes clases según el método llamado. El uso de este patrón se evidencia en las clases controladoras que se encargan de obtener los datos y enviarlos a las vistas. (31)

Patrones GOF

Los patrones GOF describen las formas frecuentes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación y combinación entre clases y la formación de

estructuras de mayor complejidad. A continuación se describen los patrones utilizados en la propuesta de solución:

- **Patrón solitario (*Sigleton*):** este patrón se utilizó para garantizar la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su uso se evidencia en la clase miControladora que es la que garantiza que se cree una única instancia del fichero de configuración.
- **Patrón fachada (*Facade*):** se empleó para proveer de una interfaz unificada y sencilla, que sirva de intermediaria entre el cliente y la interfaz o a un grupo de interfaces más complejas, permitiendo una mayor flexibilidad en la implementación del sistema. El uso de este patrón se evidencia en la clase sesión ya que esta describe a un usuario autenticado en el sistema. (32)

2.5.5. Patrones de base de datos

Los patrones de diseño de una base de datos permiten al usuario crear una base de datos (BD) más fortalecida ya que constituyen una guía que especifica cómo debe ser la misma. El diseño y construcción de una BD 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. (33)

A continuación se describen los patrones de diseño de BD utilizados en el desarrollo de la propuesta de solución:

- **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[34] 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. (33)

  Mediante este patrón, fueron generados los identificadores de todas las tablas pertenecientes al modelo de datos que se muestra en el acápite 2.5.10.
- **Arboles fuertemente codificados*:*** a cada nivel del árbol se le asocia una entidad. Normalmente constituyen relaciones de 1 a muchos (n). Normalmente utilizado para representar jerarquías donde es bien conocida la estructura y es importante representar la correspondencia, por ejemplo las estructuras organizacionales. (33)

[34] *Global Uniquel Dentifier (identificador único global).*

El modelo de datos propuesto en el acápite 2.5.10 presenta dicho patrón entre las tablas del esquema diseñador: tb_dplantilla -> tb_dcomponente -> tb_rcomponente _propiedad.

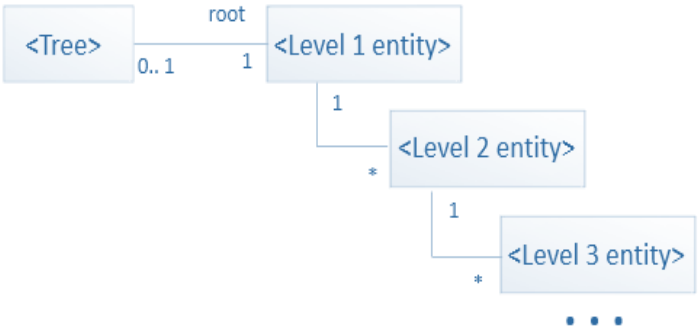


Figura 7. Árbol fuertemente codificado.

2.5.6. Vistas principales de la propuesta de solución

A continuación se describen el objetivo de cada una de las pantallas o vistas principales de la propuesta de solución por áreas.

2.5.6.1. Contenidos de la vista Presentación (Áreas)

Para acceder a esta vista es necesario ejecutar la aplicación.

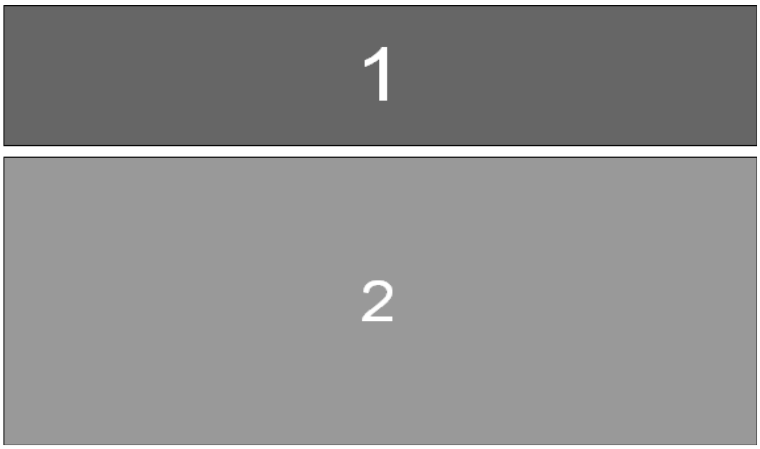


Figura 8. Áreas de la vista Presentación.

Área 1: incluye el logotipo de la UCI a la izquierda y a la derecha incluye el nombre del Sistema (Sistema de Gestión Universitaria) y debajo el nombre del software (Sistema Diseñador de Plantillas).

Área 2: incluye el formulario de autenticación del sistema.

2.5.6.2. Contenidos de la vista Gestión (Áreas)

Para acceder a esta vista es necesario ejecutar la aplicación, y autenticarse en el sistema. Una vez autenticado en el mismo se accede a la vista de gestión. Se selecciona un módulo del sistema y se accede en él. Posteriormente se accede a alguna de las funcionalidades del componente.

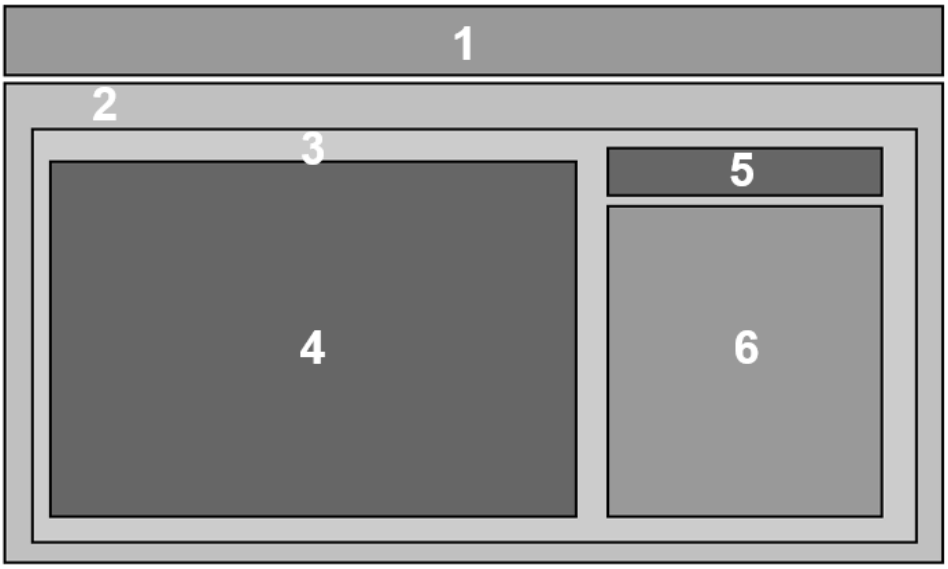


Figura 9. Áreas de la vista de Gestión.

Área 1: constituye el menú del sistema. Dicho menú incluye las configuraciones del sistema, la opción de cierre del sistema, la opción de cambiar de usuario, así como el manual de usuario del sistema.

Área 2: constituye el área de contenido. Incluye los accesos a los diferentes módulos que integran el Sistema Diseñador de Plantillas. Ejemplo: Seguridad, Diseño.

Área 3: constituye el área de contenido general. (Incluye el área de las acciones superiores y la vista de gestión para una determinada funcionalidad).

Área 4: constituye el área donde se realiza la gestión de una determinada funcionalidad.

Área 5: constituye el área de las acciones superiores.

Área 6: constituye el área donde se muestran los detalles de una determinada funcionalidad.

2.5.6.3. Contenidos de la vista Diseño (Áreas)

Para acceder a esta vista es necesario ejecutar la aplicación, y autenticarse en el sistema. Una vez autenticado en el mismo se accede a la vista de gestión. Se selecciona el módulo Diseño del sistema y se accede en él. Posteriormente se accede al componente Plantilla y a la funcionalidad crear plantilla de las acciones superiores.

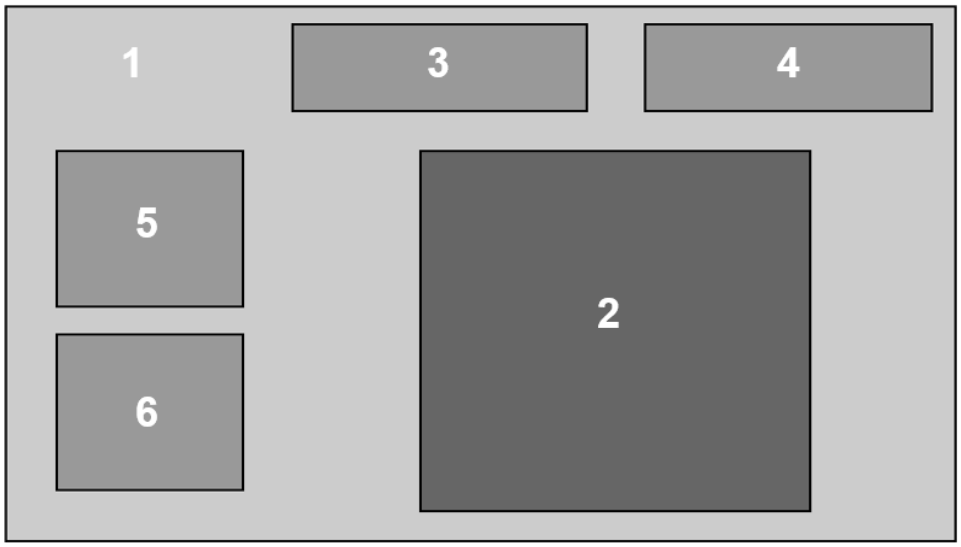


Figura 10. Áreas de la vista de Diseño.

Área 1: constituye el área de contenido general. (Incluye el área de diseño de la plantilla, vista, componentes, propiedades).

Área 2: constituye el área donde se realiza el diseño de la plantilla.

Área 3: constituye el área de diseño de la plantilla, donde se configura el tamaño, margen y orientación de la plantilla.

Área 4: constituye el área de vista, esta contiene la opción de mostrar la plantilla en tiempo de diseño.

Área 5: constituye el área que contiene los componentes que se insertarán en la plantilla.

Área 6: constituye el área que contiene las propiedades de los componentes que se insertarán en la plantilla.

2.5.7. Sistema de mensajes de la propuesta de solución

Dentro de los dos tipos de mensajes definidos se encuentran: Mensajes incrustados en los formularios y mensajes en ventanas emergentes. A continuación se describen cada uno de ellos:

- Mensajes incrustados en los formularios: Representan mensajes de error y generalmente se muestran sobre algún componente de la vista cuando falla una validación determinada.

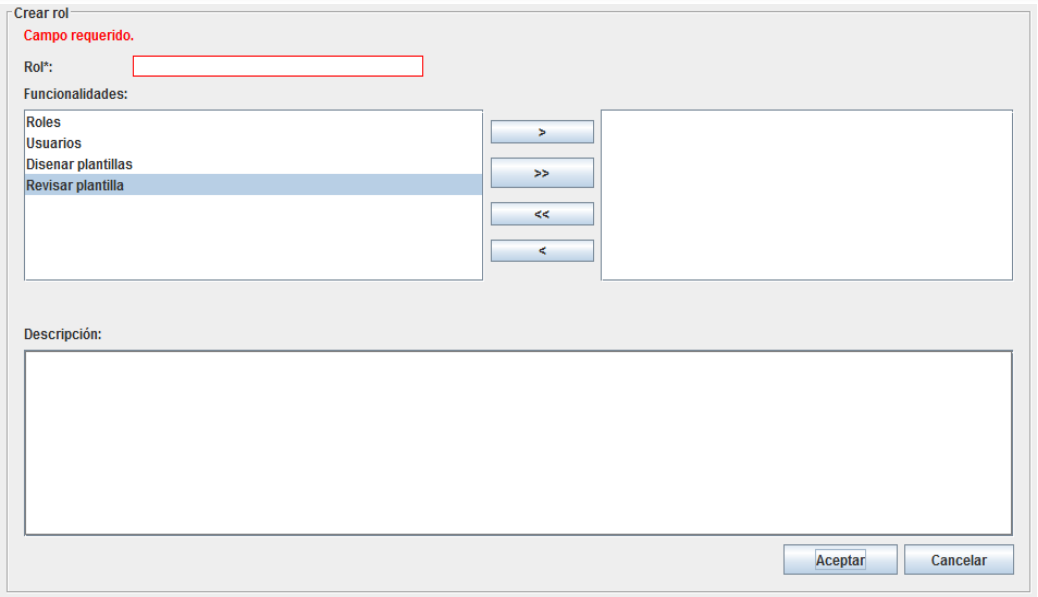


Figura 11. Ejemplo de mensaje incrustado en el formulario.

- Mensajes en ventanas emergentes: Pueden ser de advertencia (alerta), de notificación y de error.
  - **Notificación:** Se muestra como resultado de una acción realizada satisfactoriamente en el sistema.

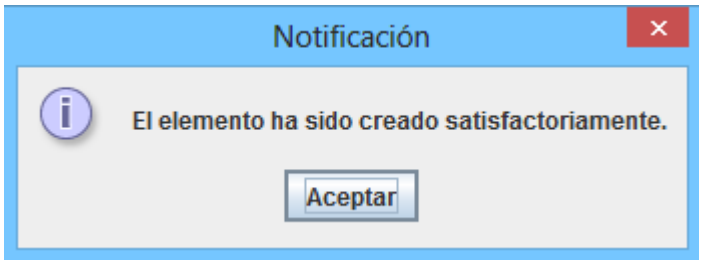


Figura 12. Ejemplo de mensaje en ventanas emergentes. Notificación.

  - **Error:** Se muestra como resultado de la captura de una excepción sobre una acción realizada.

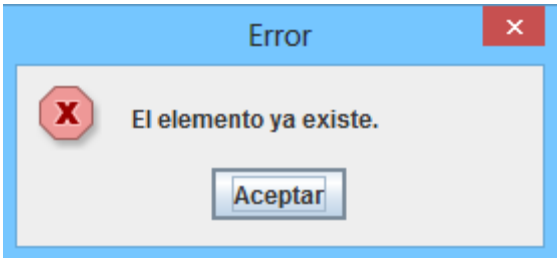


Figura 13. Ejemplo de mensaje en ventanas emergentes. Error.

  - **Advertencia o Alerta:** Se muestra para alertar al usuario que su acción puede influir significativamente en el resultado esperado tras realizar la acción.

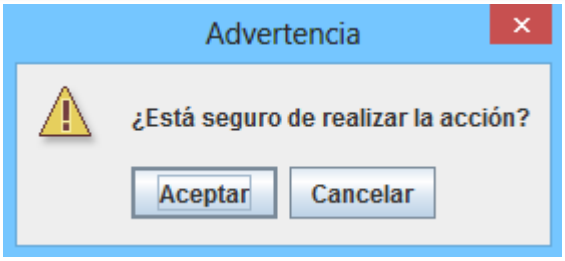


Figura 14. Ejemplo de mensaje en ventanas emergentes. Advertencia.

2.5.8. Diagrama de componentes

Los diagramas de componentes muestran los elementos de diseño de un sistema de software, permitiendo visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces.

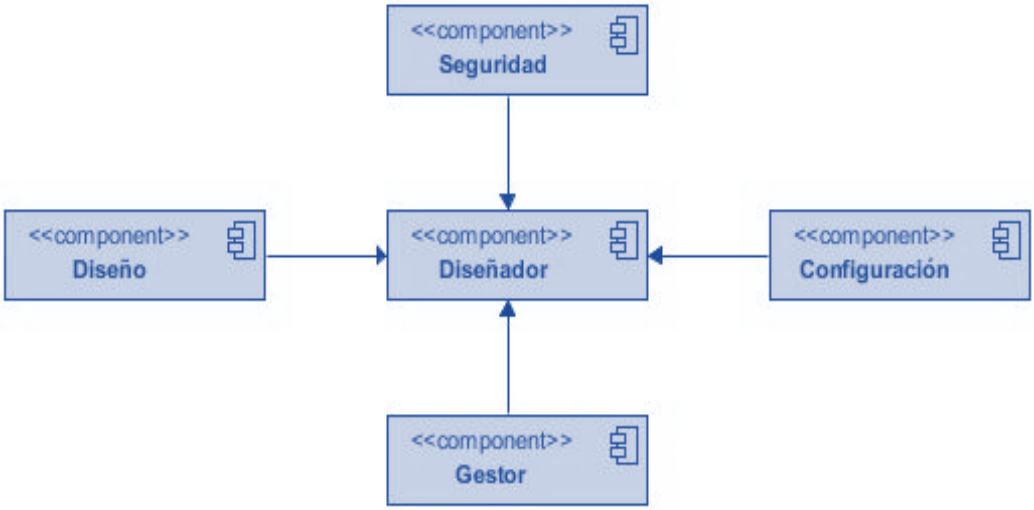


Figura 15. Diagrama de componentes del diseñador de plantillas.

2.5.9. Mapa de navegación

Los mapas de navegación proporcionan una representación esquemática de la estructura del sistema, indicando los principales conceptos incluidos en el espacio de la información y las interrelaciones que existen entre ellos. A continuación se muestran los mapas de navegación para cada uno de los componentes:

Componente Seguridad

El componente permite la centralización de la información relacionada con la seguridad en el sistema.

Está compuesto por dos agrupaciones funcionales: **Usuarios** permite gestionar los usuarios del sistema así como ver el listado y mostrar los detalles de los mismos. **Roles** permite gestionar los roles del sistema y también ver el listado y mostrar los detalles de los mismos.

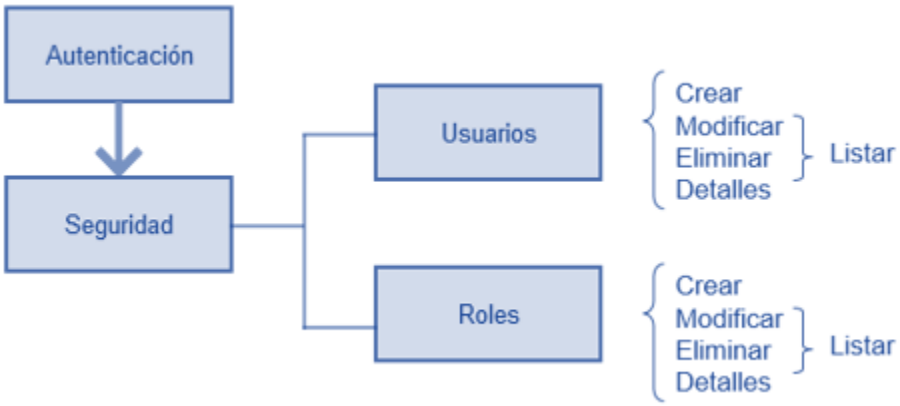


Figura 16. Mapa de navegación del componente Seguridad.

Componente Diseño

El componente permite la centralización de la información relacionada con la gestión de plantillas en el sistema. Está compuesto por una agrupación funcional: **Plantilla** permite gestionar plantillas y listar dichas plantillas. Esta agrupación funcional es la que permite al crear una plantilla gestionar los componentes que se insertarían en la plantilla para darle formato. También permite eliminar plantillas y cambiarles el estado, así como ir mostrando el diseño de la plantilla.

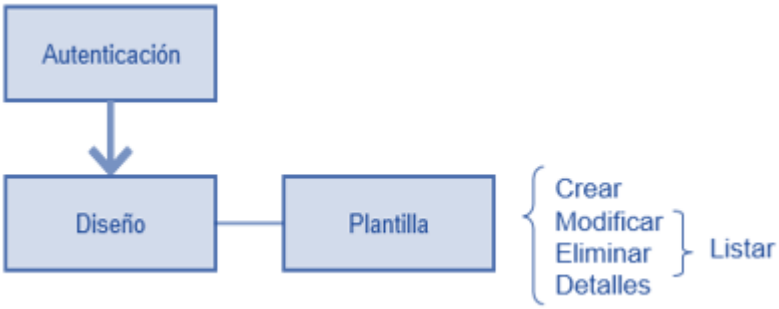


Figura 17. Mapa de navegación del componente Diseño.

Componente Gestor

El componente permite la centralización de la información relacionada con la gestión de plantillas en el sistema. Está compuesto por una agrupación funcional: **Revisar** permite realizarle comentarios a las plantillas así como ver el listado de las plantillas. También permite eliminar y cambiar el estado de las plantillas y exportar las mismas en los formatos XML y XSL a un espacio local.

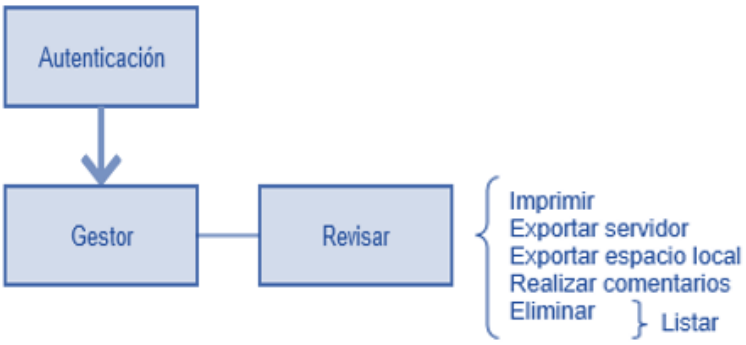


Figura 18. Mapa de navegación del componente Gestor.

Componente Configuración

El componente permite realizar las configuraciones en el sistema tales como: conectarse a un sistema externo y las configuraciones de Bases de Datos.

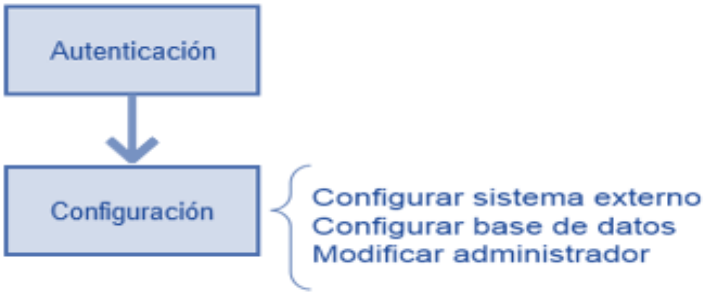


Figura 19. Mapa de navegación del componente Configuración.

2.5.10. Modelos de datos

Un modelo de datos es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de reglas de integridad, las cuales definen un conjunto de estados consistentes. El cual puede ser usado como una herramienta para especificar los tipos de datos y la organización de los mismos. Además para la manipulación de consultas y datos, así mismo es el elemento clave en el diseño de la arquitectura de un manejador de BD. (34) A continuación se muestra en la Figura 20 y en la Figura 21 el modelo físico de la propuesta de solución.

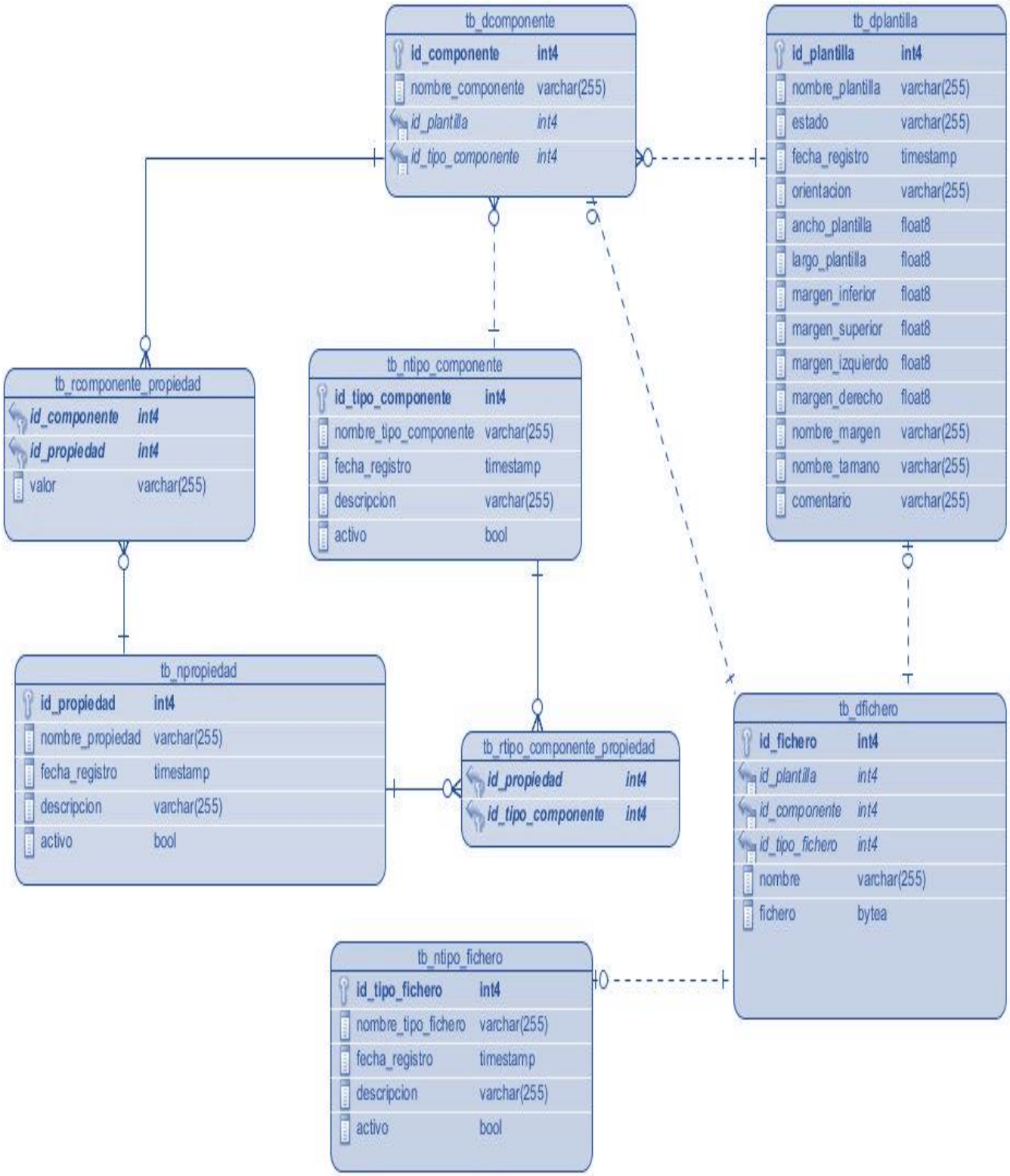


Figura 20. Modelo físico de datos del esquema diseñador de la propuesta de solución.

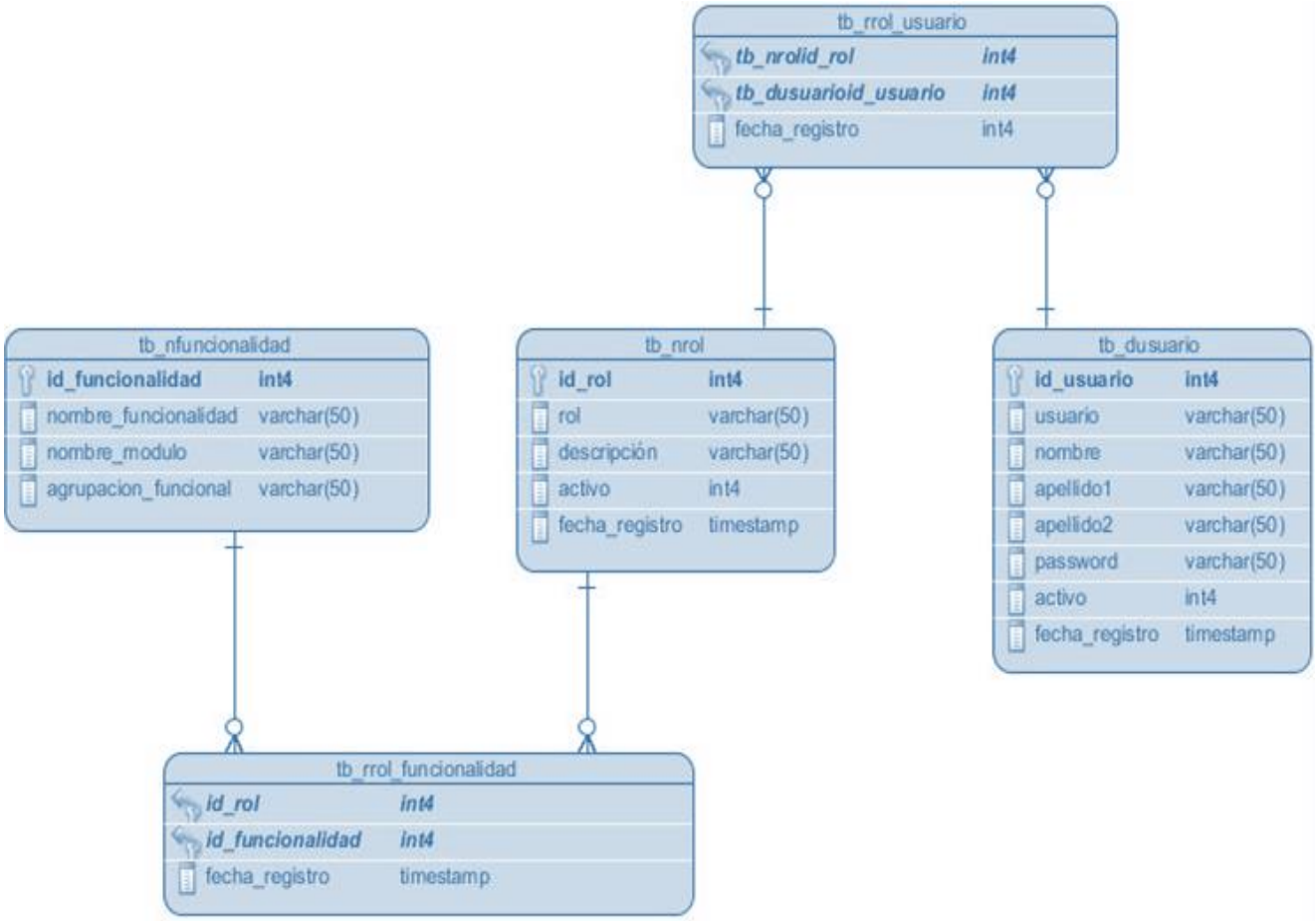


Figura 21. Modelo físico de datos del esquema seguridad de la propuesta de solución.

Conclusiones parciales

En este capítulo se ha caracterizado la propuesta de solución y presentado el modelo conceptual que sirvió para comprender el funcionamiento del proceso. También se describieron las técnicas de captura de requisitos utilizadas para la comprensión de las funcionalidades del sistema, lo que permitió realizar el levantamiento de los requisitos funcionales y no funcionales. La especificación de requisitos permitió documentar toda la información referente al dominio de la aplicación. También se caracterizó la arquitectura cliente-servidor, el patrón HMVC, los patrones de diseño y los de base de datos, así como la obtención de los modelos de datos y la distribución física de la propuesta de solución, en fin, se logró comprender toda la arquitectura necesaria para el desarrollo de la propuesta de solución. Además de mostrarse las principales vistas del sistema, se mostraron el diagrama de componentes del sistema así como el mapa de navegación, el cual proporciona una representación esquemática de la estructura del sistema.

Capítulo 3: Implementación y validación de la solución

Las pruebas de software son un elemento crítico para la garantía de la calidad del mismo y representan una revisión final de las restricciones del diseño y la codificación. En el presente capítulo se describe la implementación del sistema teniendo en cuenta las técnicas de programación y estándares de codificación, así como el tratamiento de errores. También se presentan las pruebas a realizar como son: de unidad, de validación y del sistema para validar dicha solución y así satisfacer las necesidades del cliente y cumplir con todas sus peticiones.

3.1. Técnicas de programación

Las técnicas de programación constituyen una parte fundamental en el proceso de desarrollo e ingeniería del software dentro del ámbito informático. Tienen como objetivo principal facilitar la comprensión del programa, y además permitir, de forma rápida, las ampliaciones y modificaciones que surjan en la fase de explotación del ciclo de vida del software.

3.1.1. Programación modular

En este tipo de programación el programa es dividido en módulos. Cada uno de las cuales realiza una tarea específica, codificándose independientemente de otros módulos. Cada uno de estos módulos es analizado, codificados y puestos a punto por separado. (35)

Los programas contienen un módulo denominado módulo principal, el cual supervisa todo lo que sucede, transfiriendo el control a submódulos, para que puedan realizar sus funciones. Sin embargo, cada submódulo devolverá el control al módulo principal una vez completada su tarea. Si las tareas asignadas a cada submódulo son demasiado complejas, se procederá a una nueva subdivisión en otros módulos más pequeños aún. (35)

En cuanto a la seguridad se puede decir que los módulos son independientes, de modo que ningún módulo puede tener acceso directo a cualquier otro módulo, excepto el módulo al que llama y sus submódulos correspondientes. Sin embargo, los resultados producidos por un módulo pueden ser utilizados por otro módulo cuando se transfiera a ellos el control. (35)

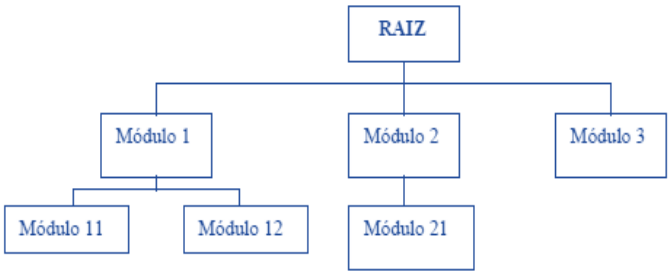


Figura 22. Programación modular.

3.1.2. Programación orientada a objetos (POO)

En la POO se debe tener tantos objetos-lista como sea necesario. Cada objeto implementa su propio módulo, permitiendo por ejemplo que coexistan muchas listas. Cada objeto es responsable de inicializarse y destruirse en forma correcta. No existe la necesidad de llamar explícitamente al procedimiento de creación o de terminación. (36)

La POO está basado en cuatro aspectos importantes, tales como: abstracción, herencia, encapsulamiento y polimorfismo.

En la POO la abstracción es una descripción o especificación simplificada de un sistema que hace énfasis en algunos detalles significativos y suprime los irrelevantes. La abstracción se debe enfocar más en qué es un objeto y qué hace antes, de pensar en la implementación. Al encapsular u ocultar información se separan los aspectos externos de un objeto (los accesibles para todos) de los detalles de implementación (los accesibles para nadie). Con esto se trata de lograr que al tener algún cambio en la implementación de un objeto no se tengan que modificar los programas que utilizan tal objeto. La herencia es la contribución más importante de la POO, pues mediante este mecanismo es posible lograr la principal meta de la POO que es la reutilización de código. Otro de los mecanismos aportados por la POO es el de polimorfismo, el cual brinda la posibilidad de tener métodos con el mismo nombre pero que su implementación sea diferente. (35)

3.2. Estándares de codificación

Un estándar de programación es una forma de "normalizar" la programación de forma tal que al trabajar en un proyecto cualquiera, las personas involucradas entiendan y pueda mantener la aplicación ya que en muy raras ocasiones una misma aplicación es mantenida por su autor original. Estos a su vez también mejoran la legibilidad del código, al mismo tiempo que permiten su compresión rápida.

3.2.1. Indentación, llaves de apertura y cierre y tamaño de líneas

Usar una indentación sin tabulaciones, con un equivalente a 4 espacios, para mantener integridad en las revisiones. 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:

```
public int ejemplo()
{
    int a=1;
    return a;
}
```

Figura 23. Indentación, llaves de apertura y cierre y tamaño de líneas.

3.2.2. Conversión de nomenclatura

- Variables: 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.

  Ejemplo:

```
int variable=1;
String variableNombreCompuesto="";
```

Figura 24. Conversión de nomenclatura. Variables.

- Constantes: Siempre debe ser todo en mayúsculas, con caracteres de subrayado "_" para separar palabras en caso de nombres compuestos.

  Ejemplo:

```
final int CONSTANTE=1;
final String CONSTANTE_COMPUESTA="CONSTANTE";
```

Figura 25. Conversión de nomenclatura. Constantes.

- Clases: Siempre comienzan con mayúscula, en caso de nombre compuesto las palabras se separan con el carácter subrayado "_" y el resto en minúscula.

  Ejemplo:

```
public clase()
{...}

public clase_nombre_compuesto()
{...}
```

Figura 26. Conversión de nomenclatura. Clases.

- Métodos: 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.

  Ejemplo:

```
public int metodo(int parametro1, String parametro2)
{
    return 1;
}
public int metodoNombreCompuesto(int parametro1, String parametro2)
{
    return 1;
}
```

Figura 27. Conversión de nomenclatura. Métodos.

- Ficheros: Todo siempre en minúscula y en caso de nombres compuestos se usa el carácter subrayado "_".
- Vistas: Intuitivo y relacionado con el formulario y/o vista que representa que incluye el prefijo "vista_".
- Modelos: Con el mismo nombre de la tabla que representa.
- Controladoras: Con el mismo nombre del proceso que representa que contiene el prefijo "cc_".

3.2.3. Estructuras de control

Las sentencias if, for, while, switch, try-catch se incluyen en estas estructuras. Entre las estructuras de control y los paréntesis debe existir un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

Ejemplo:

```
if (parametro1 == 1)
{

}
else if(parametro2.equals("2"))
{

}
```

Figura 28. Estructuras de control.

Si las condiciones son muy largas, que sobrepasen el tamaño de la línea, estas se dividen en varias líneas. En el mejor de los casos cuando la condición es muy extensa, se puede dividir esta en variables y compararlas dentro de la estructura de control.

3.2.4. 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:

- Clase:

```
/**
 * descripcion: Breve descripción de la clase.
 * @version jDK 1.6 SGIDA 1.0
 * category: categoría de la clase implementada(controladora, modelo, vista, auxiliar)
 * @author Nombre y apellidos del autor
 */
```

Figura 29. Documentación. Clases.

- Método:

```
/**
 * Breve descripción de la función
 * @param tipo y nombre del parámetro por cada parámetro que recibe el método
 * @return tipo que retorna
 * @throws Tipo de excepción que lanza
 * @autor Nombre y apellidos del autor
 */
```

Figura 30. Documentación. Métodos.

3.3. Tratamiento de errores

La presencia de errores durante el desarrollo de software es bastante común, pero lo importante no es no equivocarse sino poder detectar los errores a tiempo. Entre más tarde se descubran más costosos será corregirlos. El tratamiento de errores debe estar enfocado tanto a los errores que se producen a la hora de los usuarios introducir datos, como a los errores que puedan ser generados por el comportamiento incorrecto de los componentes internos.

En el Sistema Diseñador de Plantillas el seguimiento que se le da a los datos introducidos por los usuarios es mediante validaciones que se les asignan a los campos de entrada de datos. El tipo de error se les muestra a los usuarios sobre el campo donde se introdujeron datos incorrectos.

Una excepción es un evento que ocurre durante la ejecución de un programa y detiene el flujo normal de la secuencia de instrucciones de ese programa. Las excepciones en *Java* están destinadas, al igual que en el resto de los lenguajes que las soportan, para la detección y corrección de errores. En *Java* el mecanismo para la gestión de excepciones consiste en el uso de bloques try/catch. En el bloque try es en donde se coloca las instrucciones que se prevé que generen una excepción, mientras que se colocan uno o más bloques catch, de tal forma que si se produce una excepción se ejecuta dicho bloque. En caso de que no se produzca ningún error en el bloque try, nunca se ejecutaría ningún bloque catch. (37)

3.4. Estrategia de pruebas de software

Las pruebas de software es el instrumento adecuado para determinar la calidad de un producto de software en la fase de cierre de iteración a partir de la metodología DAC. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante métodos y técnicas de prueba. (38) El método de caja blanca es en la cual se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de prueba que ejerciten conjuntos específicos de condiciones, bucles o ambos. Este tipo de prueba se realizó mediante la herramienta JUnit de forma automática. El método de caja negra son pruebas que se realizan al software sin acceso al código fuente, donde se trabaja con entradas y salidas. Para este método se usaron diferentes técnicas como son las particiones de equivalencia y las listas de chequeo. En la Tabla 5 se muestra la estrategia de pruebas trazadas para realizar al Sistema Diseñador de Plantillas para comprobar que la solución implementada cumple con los requisitos previamente acordados.

Tabla 5. Estrategia de pruebas de software.

Nivel	Tipo de prueba	Método	Técnica
Unidad	Funcional	Caja blanca	Automática
Validación	Funcional	Caja negra	Particiones de equivalencia
Sistema	Aceptación	Caja negra	Alfa
	Seguridad	Caja negra	Listas de chequeo

3.4.1. Pruebas de unidad

Al desarrollar un nuevo software o sistema, la primera etapa de pruebas a considerar es la etapa de pruebas de unidad o también llamada pruebas de caja blanca, estás pruebas también son llamadas pruebas modulares ya que permiten determinar si un módulo del programa está listo y correctamente terminado. (39) El objetivo fundamental de las pruebas de unidad es asegurar el correcto funcionamiento de las interfaces, o flujo de datos entre componentes. No es un requisito indispensable la culminación de todos los módulos del sistema para iniciar las pruebas. La importancia de este tipo de pruebas es que se deben tener claros los siguientes aspectos:

- Los datos de entrada son conocidos por el *Tester* o Analista de Pruebas y estos deben ser preparados con minuciosidad, ya que el resultado de las pruebas dependen de estos.
- Se debe conocer que componentes interactúan en cada caso de prueba.
- Se debe conocer de antemano que resultados debe devolver el componente según los datos de entrada utilizados en la prueba.
- Finalmente se deben comparar los datos obtenidos en la prueba con los datos esperados, si son idénticos se puede decir que el módulo superó la prueba y se inicia con la siguiente. (39)

Resultado de las pruebas de unidad

Haciendo uso de la librería JUnit se aplicó el método de caja blanca en 4 iteraciones en las cuales se detectaron un total de 26 no conformidades. En la primera iteración se detectaron 15 no conformidades que fueron corregidas en su totalidad. En la segunda iteración se detectaron 9 no conformidades que igualmente fueron resueltas. En la tercera iteración se encontraron 2 no conformidades que fueron igualmente corregidas. Finalmente en la cuarta y última iteración no se detectaron no conformidades obteniéndose un resultado satisfactorio. A continuación se muestra en la Tabla 6 un ejemplo de una prueba realizada al software. Ejemplos de otros resultados se muestran en los Anexos, para consultar las restantes ir a los **Anexos (Tabla 26-30)**.

Tabla 6. Casos de prueba de caja blanca. RFDP29_ModificarPlantilla

Técnica Caja blanca	**Código caso de prueba:** DIN-SGU-DP-CPCB_ RFDP29_ModificarPlantilla
Probador: Alejandro Vázquez Chávez	
Procedimiento prueba automatizada	
Descripción: Los datos de entrada serán los atributos de la plantilla a modificar.	
Datos de entrada:	Objeto nulo, identificador igual a 0.
Tipo de dato esperado:	Vacío.
Función de evaluación:	

```
public void testModificarPlantilla() throws ConfiguracionBaseDatosException, EncriptacionException {
    System.out.println("modificarPlantilla");
    int id = 0;
    tb_dplantilla instance = new tb_dplantilla();
    int expResult = 0;
    int result = instance.modificarPlantilla(id);
    assertEquals(expResult, result);
}
```

Evaluación del caso de prueba: Satisfactorio.

3.4.2. Pruebas de validación

Las pruebas de validación comienzan cuando se ha terminado de ensamblar el software como paquete y se han descubierto y corregido los errores de interfaz. Las pruebas se concentran en las acciones visibles para los usuarios y en la salida del sistema que este puede reconocer. La validación se define de una forma simple en que se alcanza cuando el software funciona de tal manera que satisface las expectativas razonables del cliente (especificaciones de requisitos). (40)

Las pruebas de validación de datos se centra en probar todas las entradas de datos, así como lugares donde pueda existir información para ver si cumple con los requisitos funcionales.

Resultado de las pruebas de validación

Al realizar las pruebas de validación por el método de caja negra mediante la técnica de particiones de equivalencia en 4 iteraciones se obtuvieron los siguientes resultados en cada una de las iteraciones realizadas al software:

- En la primera iteración se detectaron 23 no conformidades entre las que se encuentran la no visualización de los mensajes de error encima de algunos componentes y la no indicación de las acciones de contexto cuando se posicionaba el cursor encima de la acción. Estas no conformidades fueron corregidas en su totalidad.
- Una segunda iteración arrojó un total de 11 no conformidades entre las que se encuentran la no validación de algunos campos de texto para la entrada de datos incorrectos. Estas no conformidades fueron resueltas totalmente.
- En la tercera iteración se detectaron 3 no conformidades, las cuales fueron errores ortográficos en los diferentes mensajes que muestra el software.
- En la última y cuarta iteración al sistema no le fueron detectadas no conformidades.

A continuación se muestra en la Tabla 7 un ejemplo de un diseño de casos de prueba correspondiente a la funcionalidad Guardar plantilla. Otros ejemplos de diseño de casos de prueba se muestran en los Anexos, para consultar las restantes ir a los **Anexos (Tabla 31-57)**.

Tabla 7. Diseño de casos de prueba. RFDP48_Guardar plantilla. Parte uno.

Escenario	Descripción	Variable 1: Nombre	Variable 2: Estado
EC 1.1 Guardar plantilla correctamente.	Mediante este escenario el usuario puede guardar una plantilla correctamente.	V	V
		Título	Seleccionado
		V	V
		Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos para una palabra es de 30.	No seleccionado
EC. 1.2 Guardar plantilla incorrectamente.	Mediante este escenario el usuario no puede guardar una plantilla correctamente.	I	V
		” Pureba%”	Seleccionado
		I	V
		El usuario entra al menos de 2 caracteres.	Seleccionado
EC 1.3 Guardar plantilla repetida.	Mediante este escenario el usuario no puede guardar una plantilla con un nombre igual a otra ya existente en el sistema.	I	V
		El usuario introduce un nombre de plantilla igual al de otra ya existente en el sistema.	Seleccionado

Tabla 8. Diseño de casos de prueba. RFDP48_Guardar plantilla. Parte dos.

Respuesta del sistema	Flujo central
El sistema guarda una plantilla correctamente.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores y luego se escribe el nombre de la plantilla y se actualiza su estado. Posteriormente se puede seleccionar la opción Guardar o Cancelar.

El sistema muestra un mensaje en rojo encima del componente: "No se admiten apóstrofes, ni porcientos sobre el campo requerido".	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores y luego se escribe el nombre de la plantilla y se actualiza su estado. Posteriormente se puede seleccionar la opción Guardar o Cancelar.
El sistema muestra un mensaje de error: “Entre más de 2 caracteres.”.	
El sistema muestra un mensaje de error: "El elemento ya existe.".	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores y luego se escribe el nombre de la plantilla y se actualiza su estado. Posteriormente se puede seleccionar la opción Guardar o Cancelar.

3.4.3. Pruebas del sistema

El proceso de prueba de un sistema tiene dos etapas que pueden estar muy separadas en el tiempo: la preparación de las pruebas y la aplicación de las mismas. La primera está muy ligada a la obtención de requerimientos, por lo que ocurre en las primeras etapas del proyecto, mientras que la segunda requiere del sistema completo o al menos una integración, como se denomina a un producto parcial, aún no liberado, para poder aplicar las pruebas, por lo que ocurre en etapas avanzadas del proyecto. La situación exacta de estas partes depende del modelo de ciclo de vida que se haya elegido. (41) A continuación se describen el tipo de prueba realizada a nivel de sistema.

3.4.3.1. Pruebas de seguridad

Las pruebas de seguridad tienen como objetivo verificar que un actor solo pueda acceder a las funciones y datos que su usuario tiene permitido y verificar que solo los actores con acceso al sistema y a la aplicación están habilitados para accederla. (42)

Con las pruebas de seguridad se garantiza que los usuarios están restringidos a funciones específicas o su acceso está limitado únicamente a los datos que están autorizado a acceder. El objetivo de esta prueba es evaluar el funcionamiento correcto de los controles de seguridad del sistema para asegurar la integridad y

confidencialidad de los datos. El foco principal es probar la vulnerabilidad del sistema frente a accesos o manipulaciones no autorizadas. (42)

La técnica aplicada para realizar las pruebas de seguridad es mediante las listas de chequeo. Esta tiene como objetivo evaluar la seguridad de las aplicaciones. A continuación se describe los campos que componen las listas de chequeo:

- Peso: Define si el indicador a evaluar es crítico o no.
- Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.
- N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.
- Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.
- Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Resultados de las pruebas de seguridad

Al hacer uso de la técnica de las listas de chequeo se prueba la seguridad del software. A continuación se muestra en la Tabla 9 un ejemplo de una prueba realizada al software. Ejemplos de otros resultados se muestran en los Anexos, para consultar los restantes ir a los **Anexos (Tabla 31- 32)**.

Tabla 9. Lista de Chequeo. Pruebas de Autorización.

Pruebas de Autorización					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	Un usuario estándar (no administrador) no debe poder modificar sus privilegios en la aplicación.	0		No se encontraron errores del indicador evaluado.	
Crítico	Un usuario estándar (no administrador) no debe poder modificar los privilegios de otro usuario.	0		No se encontraron errores del indicador evaluado.	

3.4.3.2. Pruebas de aceptación

Una prueba de aceptación del usuario es una evaluación dada a un programa de software para asegurarse de que cumple con los objetivos fijados para este último durante la fase de diseño y programación de su desarrollo. La realización de una prueba de aceptación del usuario es una parte importante del desarrollo de software, ya que asegura que el programa terminado satisface las necesidades de sus usuarios. Una prueba de aceptación del usuario se realiza típicamente durante la etapa final del desarrollo, cuando el software se prueba como un todo. (43)

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 mediante las técnicas alfa y beta. La técnica alfa es realizada por el usuario en un entorno controlado por el desarrollador como observador y la beta es realizada por el usuario en un ambiente no controlado por los desarrolladores. En fin el usuario comprueba en su propio entorno de explotación si acepta el software.

Resultados de las pruebas de aceptación

Después de haber presentado la propuesta de solución a la Jefa de I+D+i se obtuvo como resultado la aprobación para el uso y explotación de la herramienta, lo cual quedó contemplado en el Acta de Aceptación del Cliente. (Ver **Anexo 6**).

Conclusiones parciales

En el desarrollo de este capítulo se definieron las técnicas de programación y estándares de codificaciones empleadas en la implementación del diseñador de plantillas, permitiendo una mejor organización y comprensión del código. También se definió y aplicó la estrategia de pruebas a realizar al software para asegurar y garantizar la calidad del mismo, permitiendo obtener resultados satisfactorios y asegurando a su vez que el Sistema Diseñador de Plantillas desarrollado no contenga errores y cubra las expectativas de aceptación requerida.

Conclusiones generales

Una vez elaborada la fundamentación teórica que sustentó la investigación, así como definidas las características del Sistema Diseñador de Plantillas y efectuado su implementación y respectiva validación mediante las pruebas de software, se arrojaron resultados que permiten plantear que:

- Al analizar los diferentes diseñadores de plantillas tanto en el ámbito internacional como nacional, 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, aunque poseen características que sirvieron a tener en cuenta en la implementación de la propuesta de solución.
- Con la aplicación de la metodología Desarrollo Ágil con Calidad que combina las metas y prácticas de las áreas de procesos del nivel 2 de CMMI con las buenas prácticas de la dirección y desarrollo ágil de proyectos de software, se alcanzó especificar y describir los requisitos, así como elaborar los artefactos propuestos y el diseño de la solución.
- Con el desarrollo del Sistema Diseñador de Plantillas se dotó a la Universidad de Ciencias Informáticas de un sistema que permite a los usuarios diseñar plantillas personalizadas de documentos acreditativos ya sean títulos, certificados u otros documentos, a través de una interfaz de fácil uso, integrando al SGU como herramienta que complementa el trabajo del Sistema de Impresión de Documentos.
- El Sistema Diseñador de Plantillas es seguro, multiplataforma, desarrollado en software libre y cuenta con todas las funcionalidades necesarias para el diseño de plantillas de documentos acreditativos.

Recomendaciones

Al dar cumplimiento al objetivo general de esta investigación, resuelto el problema de investigación y teniendo en cuenta las experiencias adquiridas en el desarrollo del software para facilitar el diseño de plantillas se recomienda:

- Permitir duplicar plantillas en el Sistema Diseñador de Plantillas para disminuir el tiempo de diseño ya que en caso de que se desee diseñar una misma plantilla con algunos cambios particulares para que la nueva plantilla no se realice desde el principio; es decir se realiza la copia y se trabaja sobre esa nueva copia.
- Adicionar nuevas propiedades al componente de tipo tabla como: cambiar el color de fondo e incorporar imágenes a las tablas.
- Incorporar el sistema desarrollado como un componente al Sistema de Impresión de Documentos del Sistema de Gestión Universitaria.

Referencias bibliográficas

1. Rodríguez Cruz, Yunier y Galán Domínguez, Esther. Scielo. [En línea] Instituto Brasileño de Información Ciencia y Tecnología - IBICT, 16 de 05 de 2008. [Citado el: 22 de 11 de 2013.] http://www.scielo.br/scielo.php?pid=S0100-19652007000300006&script=sci_arttext.

2. Ngram, David I y Media, Demand. La voz. [En línea] [Citado el: 22 de 11 de 2013.] http://pyme.lavoztx.com/qu-es-un-sistema-de-gestin-de-la-informacin-7690.html.

3. Hechavarría Palacios, Ailin y Fernández Alvarez, Enrique. Repositorio Institucional. [En línea] 25 de 06 de 2012. [Citado el: 2013 de 12 de 08.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/ TD_05200_12/1/TD_05200_12.pdf.

4. Compart AG. Compart. [En línea] [Citado el: 05 de 04 de 2014.] http://www.compart.com/es/xsl-fo.

5. Microsoft Developer Network. [En línea] [Citado el: 08 de 12 de 2013.] http://msdn.microsoft.com/es-es/library/aa287920%28v=vs.71%29.aspx.

6. Component Source. [En línea] [Citado el: 2013 de 12 de 08.] http://www.componentsource.com/products/activereports-7/summary-es.html.

7. Herrera, Cristhian. Autentia. [En línea] 29 de 04 de 2005. [Citado el: 10 de 12 de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=ireport.

8. Brito Rodríguez, Julio César, y otros. Serie Científica UCI. [En línea] 15 de 09 de 2013. [Citado el: 2013 de 12 de 08.] http://publicaciones.uci.cu/index.php/SC/article/view/1258/711.

9. Martín Soler, Nayilet y Santiesteban Rojas, Yander . [En línea] 12 de 06 de 2013. [Citado el: 08 de 12 de 2013.] http://bibliodoc.uci.cu/RDigitales/2013/septiembre/24/TD_06564_13.pdf.

10. Taller Web. [En línea] 2011. [Citado el: 10 de 12 de 2013.] http://www.webtaller.com/manual-java/caracteristicas-java.php.

11. Profesorado, Instituto Nacional de Tecnologías Educativas y de Formación. Observatorio Tecnológico. [En línea] 12 de 02 de 2009. [Citado el: 11 de 12 de 2013.] http://recursostic.educacion.es/observatorio/web/es/software/programacion/675-xml.

12. Buenas Tareas. [En línea] 05 de 2011. [Citado el: 11 de 12 de 2013.] http://www.buenastareas.com/ensayos/Caracteristicas-Principales-De-Uml/2234495.html.

13. Alvarez, Rubén. Desarrollo Web. [En línea] 01 de 01 de 2001. [Citado el: 11 de 12 de 2013.] http://www.desarrolloweb.com/articulos/262.php.

14. Shores, Redwood. Oracle. [En línea] 23 de 01 de 2012. [Citado el: 11 de 12 de 2013.] http://www.oracle.com/lad/corporate/press/pr-lad-23-jan-2012-1498968-esa.html.

15. Targetware. Software. [En línea] [Citado el: 11 de 12 de 2013.] http://www.software.com.ar/visual-paradigm-para-uml.html.

16. Company Evolus . Pencil Project. [En línea] 2012. [Citado el: 11 de 12 de 2013.] http://pencil.evolus.vn/.

17. Martínez, Rafael. PostgresSQL-es. [En línea] 02 de 10 de 2010. [Citado el: 11 de 12 de 2013.] http://www.postgresql.org.es/sobre_postgresql.

18. Noriega, J. NORIEGA. [En línea] 22 de 03 de 2010. [Citado el: 11 de 12 de 2013.] http://noriegatic.blogspot.com/2010/03/postgre-sql.html.

19. Gómez, Rubén. Slideshare. [En línea] 20 de 01 de 2010. [Citado el: 11 de 12 de 2013.] http://www.slideshare.net/siticfje/pruebas-software-con-junit.

20. Gómez Giménez, Martín. INFORMACIÓN BÁSICA SOBRE GNU/LINUX. [En línea] 01 de 2004. [Citado el: 11 de 12 de 2013.] http://www.i-nis.com.ar/linux/cuerpo2.html.

21. Departamento de Desarrollo de la Dirección de Informatización. *DIN-DD-Proceso_de_desarrollo_de_software_DACv1.1.* LaHabana : s.n., 2013. pág. 87.

22. Buenas Tareas. [En línea] octubre de 2009. [Citado el: 24 de 01 de 2014.] http://www.buenastareas.com/ensayos/Modelos-Conceptuales/23455.html.

23. Sommerville, Ian. Entorno Virtual de Aprendizaje. [En línea] 2005. [Citado el: 24 de 01 de 2014.] eva2.uci.cu/pluginfile.php/1834/mod_folder/content/0/Sommerville%207ma%20Edicion/Sommerville_Parte_II_Requerimientos.pdf?forcedownload=1.

24. A project of the IEEE Computer Society. Entorno Virtual de Aprendizaje. [En línea] 2004. [Citado el: 27 de 01 de 2014.] eva2.uci.cu/pluginfile.php/1834/mod_folder/content/0/SWEBOK/Swebok_ ESPANOL_Capitulo_2_Requerimientos_de_Software.pdf?forcedownload=1.

25. Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. Entorno Virtual de Aprendizaje. [En línea] abril de 2004. [Citado el: 23 de enero de 2014.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Arquitectura_de_Software/Arquitecturas_de_software._Guias_de_estudio.pdf.

26. Intituto Tecnológico de Colima. [En línea] Departamento de Sistemas y Computación. [Citado el: 27 de 01 de 2014.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u1_6.htm.

27. Alvarez, Miguel Angel . Desarrollo web. [En línea] 02 de 01 de 2014. [Citado el: 27 de 01 de 2014.] http://www.desarrolloweb.com/articulos/que-es-mvc.html.

28. Barajas, Saulo. Curso de protocolos TCP/IP. [En línea] 08 de 12 de 2001. [Citado el: 27 de 01 de 2014.] http://www.saulo.net/pub/tcpip/index.html#2-8.

29. Kioskea. Kioskea .net. [En línea] 02 de 05 de 2014. [Citado el: 20 de 05 de 2014.] http://es.kioskea.net/contents/263-protocolo-ftp-protocolo-de-transferencia-de-archivos.

30. Olivares Rojas, Juan Carlos. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 27 de enero de 2014.] ttp://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/

Diseno_de_software/Patrones_de_Diseno_Art._2.pdf.

31. Visconti , Marcello y Astudillo, Hernán. Departamento de Informática. [En línea] [Citado el: 27 de 01 de 2014.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

32. Wordpress. Mundo Infomático. [En línea] 11 de 02 de 2013. [Citado el: 27 de 01 de 2014.]

33. EVA. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 28 de enero de 2014.] http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf.

34. Departamento de Sistemas y Computación. Instituto Tecnológico de Colima. [En línea] [Citado el: 28 de enero de 2014.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm.

35. Instituto Nacional de Estadística e Informática. Servidor de Apoyo al Sistema Escolarizado --Blearning--. [En línea] [Citado el: 07 de 04 de 2014.] http://blearning.itmina.edu.mx/dep/sada/carreras/Ingenieria%20Electronica/3er%20Semestre/Programacion%201/Programacion1_IE/unidad1.pdf.

36. Lenguajes de Programación. [En línea] 2009. [Citado el: 07 de 04 de 2014.] http://www.lenguajes-de-programacion.com/programacion-orientada-a-objetos.shtml.

37. MOROS VALLE, BEGOÑA . PROGRAMACIÓN ORIENTADA A OBJETOS. [En línea] [Citado el: 07 de 04 de 2014.] http://dis.um.es/~bmoros/Tutorial/parte9/cap9-3.html.

38. PRUEBAS DE SOFTWARE. pruebasdesoftware. [En línea] 2005. [Citado el: 08 de 04 de 2014.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

39. Oré B, Alexander. Calidad y Software. [En línea] 2009. [Citado el: 08 de 04 de 2014.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

40. Mansilla, Ricardo . SlideShare. [En línea] 04 de 11 de 2009. [Citado el: 10 de 04 de 2014.] http://www.slideshare.net/cliceduca/pruebas-de-software-2420588.

41. Fernández Peña , Juan Manuel. Universidad Veracruzana. [En línea] 2010. [Citado el: 08 de 04 de 2014.] http://www.uv.mx/personal/jfernandez/files/2010/07/Pruebas-de-Sistema.pdf.

42. Abad Londoño, Jorge Hernan. Ingeniería de Software. [En línea] 06 de 04 de 2005. [Citado el: 10 de 04 de 2014.] http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html.

43. Casteele, John . eHow en Español. [En línea] [Citado el: 08 de 04 de 2014.] http://www.ehowenespanol.com/llevar-cabo-prueba-aceptacion-del-usuario-pau-como_131118/.

Bibliografía

1. Hernández León, Rolando Alfredo y Coello González, Sayda. *EL PROCESO DE INVESTIGACIÓN CIENTÍFICA.* La Habana: Editorial Universitaria del Ministerio de Educación Superior, 2011. 978-959-16-1307-3.

2. Larman, Craig. UML Y PATRONES. Introducción al análisis y diseño orientado a objetos. [En línea] 1999. http://eva.uci.cu/file.php/158/Documentos/Bibliografia_general/Textos_Basicos/UML_y_Patrones/00_Presentacion_y_contenido.pdf. 970-17-0261-1.

3. Pressman, Roger S. *Ingeniería de Software 7ma edición.* 2010. 978-0-07-337597-7.

4. Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *El Lenguaje Unificado de Modelado. Manual de Referencia.* California: s.n., 1998.

5. Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. Entorno Virtual de Aprendizaje. [En línea] abril de 2004. http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Arquitectura_de_Software/Arquitecturas_de_software._Guias_de_estudio.pdf.

6. Schmuller, Joseph. Entorno Virtual de Aprendizaje. [En línea] 2000. http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Diseno_de_software/Aprendiendo_UML_en_24_Horas/00_Presentacion_y_contenido.pdf. 968-444-463-X.

7. Álvarez, Sara. Desarrollo web.com. [En línea] 30 de agosto de 2007. http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

8. EVA. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 17 de marzo de 2014.] http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf.

9. López, Ángel. *JAVA, la programación del futuro.* Buenos Aires: MP Ediciones S.A, 1997. 987-9131-38-X.

10. Manual for iReport. [En línea] 2013. [Citado el: 18 de febrero de 2013.] http://ireport.sourceforge.net/manual0.2.0.html#1.0.

11. Somerville, Ian. Ingeniería de Software 7ma edición. [En línea] 2005. [Citado el: 15 de febrero de 2013.] 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=true. 84-7829-074-5.

12. Brito Rodríguez, Julio César, Abreu Medina, Aldis Joan y Bedoya Rusenko, Jorge. Módulo diseñador de modelos para el generador dinámico de reportes. [En línea] 15 de septiembre de 2013. http://publicaciones.uci.cu/index.php/SC/article/view/1258/711. 2306-2495.

13. Becerril C., Francisco. *Java a su alcance.* México: Litográfica Ingramex, 1998. 736.

14. Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar. *Metodología de la investigación.* México: Interamericana editores S.A, 2006. 970-10-5753-8.

Glosario de términos

CASE: *Computer Aided Software Engineering*, son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software.

Componente: elemento de un software que ofrece un conjunto de servicios, o funcionalidades, a través de interfaces específicas.

Gestión: hace referencia a la acción y al efecto de administrar o gestionar un negocio. A través de una gestión se llevarán a cabo diversas diligencias, trámites, las cuales conducirán al logro de un objetivo determinado.

Información: conocimiento emitido o recibido relativo a un hecho o circunstancia en particular, que se genera por una parte en mente de las personas y por otro lado se expresa o transmite en algún tipo de soporte como puede ser la televisión, radio, prensa, ordenador. Por tanto la información es la forma de comunicar el conocimiento que origina el pensamiento humano.

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 acciones o actividades sistematizadas que se realizan o tienen lugar con un fin.

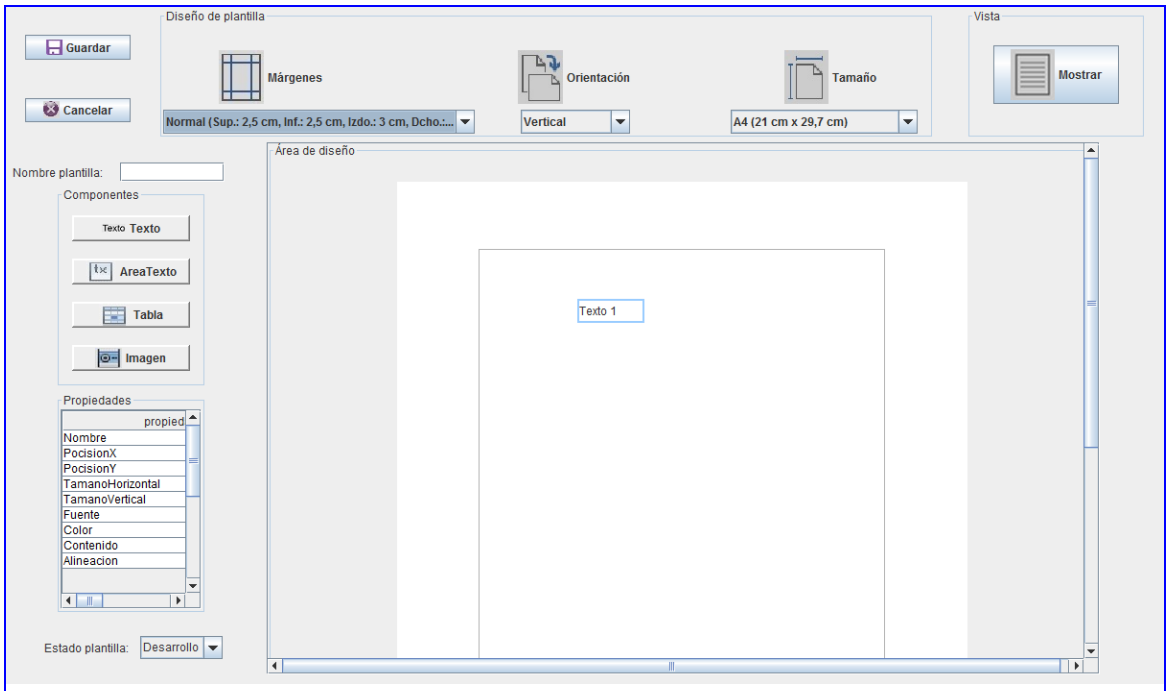
Tecnologías libres: son aquellas que no precisan de autorización o licencia para su uso. Más bien, pertenecen a la sabiduría y cultura popular, propias de la ciudadanía, que es quien las utiliza y explota en su propio beneficio.

Anexos

Anexo 1: Descripción textual y prototipo de los requisitos funcionales.

Tabla 10. Descripción del requisito funcional Insertar componente de tipo texto.

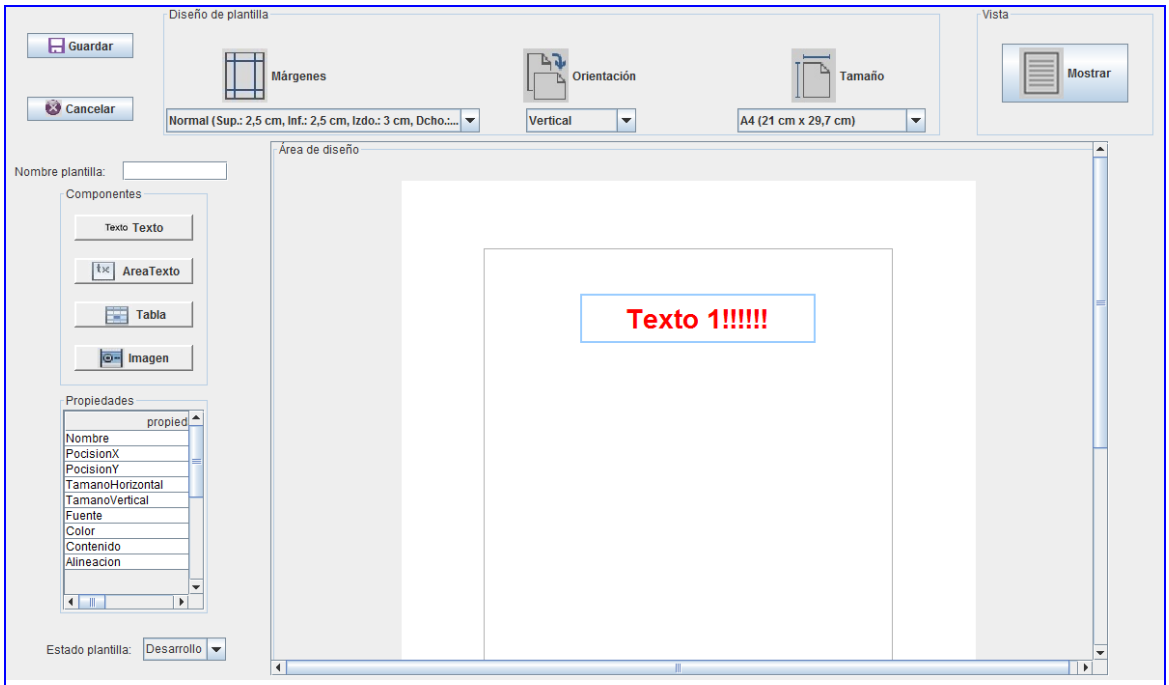
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_16	Insertar componente de tipo texto.	• El requisito permite insertar un componente de tipo texto en el área de diseño de la plantilla. • El Diseñador después de crear la plantilla procede a insertar un componente de tipo texto en el área de diseño de la plantilla al dar clic izquierdo encima del componente y luego en el área de diseño en cualquiera región. • Una vez insertado el componente este toma por defecto los valores de las propiedades del texto (*Nombre, Posición X, Posición Y, Contenido, Fuente, Tamaño Vertical, Tamaño Horizontal, Color, Alineación*).	Alta	Alta
	• **Prototipo**			



Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.
• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Contenido	• character varying	• Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
• Fuente	• character varying	• Campo de selección.
• Tamaño Vertical	• integer	• Solo lectura.
• Tamaño Horizontal	• integer	• Solo lectura.
• Color	• character varying	• Campo de selección.
• Alineación	• character varying	• Campo de selección.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema.	

Tabla 11. Descripción del requisito funcional Modificar componente de tipo texto.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_17	Modificar componente de tipo texto.	• El requisito permite modificar las propiedades de un componente de tipo texto en el área de diseño de la plantilla. • El Diseñador después de tener el componente seleccionado procede a modificar las propiedades (*Nombre, Posición X, Posición Y, Contenido, Fuente, Tamaño Vertical, Tamaño Horizontal, Color, Alineación*) del componente de tipo texto en el área de diseño donde fue creado.	Alta	Alta
	• **Prototipo**			

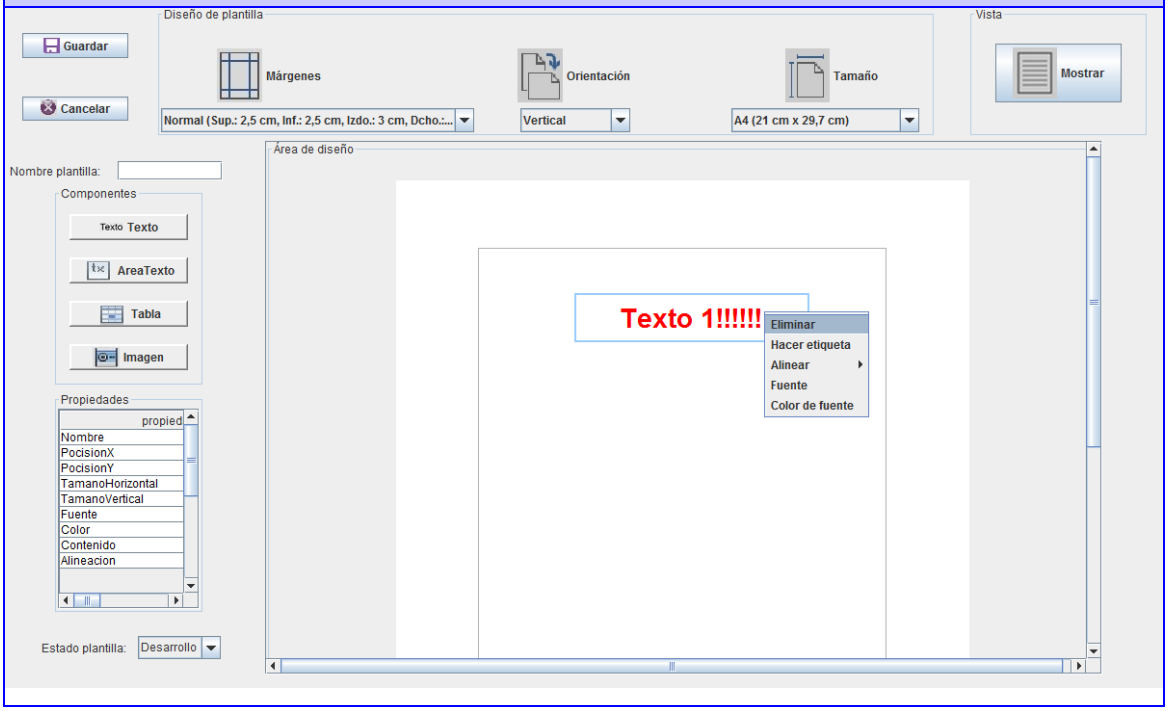


Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.
• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Contenido	• character varying	• Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
• Fuente	• character varying	• Campo de selección.
• Tamaño Vertical	• integer	• De solo lectura.
• Tamaño Horizontal	• integer	• De solo lectura.
• Color	• character varying	• Campo de selección.
• Alineación	• character varying	• Campo de selección.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. Si se introduce caracteres no permitidos el sistema muestra el	

	mensaje de error: “No se admite % y comillas simples”. 3. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.

Tabla 12. Descripción del requisito funcional Eliminar componente de tipo texto.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_18	Eliminar componente de tipo texto.	• El requisito permite eliminar un componente de tipo texto en el área de diseño de la plantilla. • El Diseñador selecciona del área de diseño el componente de tipo texto a eliminar dando clic derecho sobre el texto y seleccionando la opción Eliminar.	Media	Alta
	• **Prototipo**			

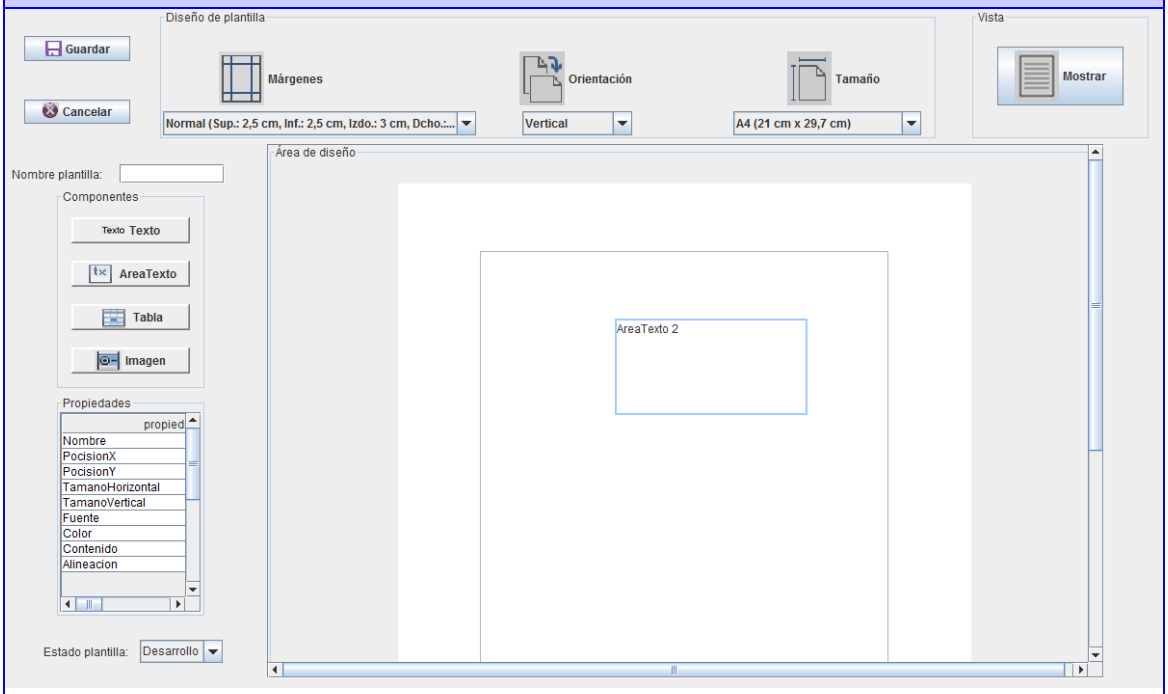


Campos	Tipos de Datos	Reglas o Restricciones
•	•	•
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. En caso de seleccionar la acción se muestra un mensaje de advertencia "¿Está seguro de realizar la acción? ".	

Tabla 13. Descripción del requisito funcional Insertar componente de tipo área de texto.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_19	Insertar componente de tipo área de texto.	• El requisito permite insertar un componente de tipo área de texto en el área de diseño de la plantilla. • El Diseñador después de crear la plantilla procede a insertar un componente de tipo área de texto en el área de diseño de la plantilla al dar clic izquierdo encima del componente y luego en el área de diseño en cualquiera región. • Una vez insertado el componente este toma por defecto los valores de las propiedades del área de texto (*Nombre, Posición X, Posición Y, Contenido, Fuente, Tamaño Vertical, Tamaño Horizontal, Color, Alineación*).	Alta	Alta

- **Prototipo**

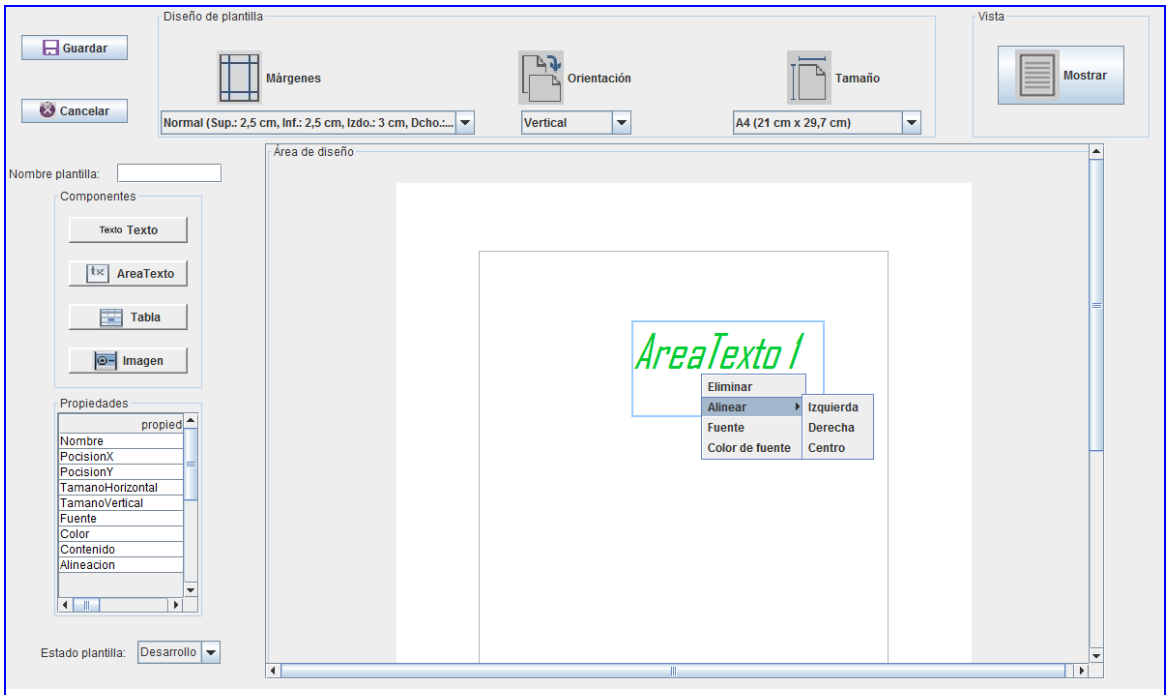


Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.
• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Contenido	• character varying	• Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
• Fuente	• character varying	• Campo de selección.
• Tamaño Vertical	• integer	• De solo lectura.
• Tamaño Horizontal	• integer	• De solo lectura.
• Color	• character varying	• Campo de selección.
• Alineación	• character varying	• Campo de selección.

Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. Si se introduce caracteres no permitidos el sistema muestra el mensaje de error: “No se admite % y comillas simples”. 3. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.

Tabla 14. Descripción del requisito funcional Modificar componente de tipo área de texto.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_20	Modificar componente de tipo área de texto.	• El requisito permite modificar las propiedades de un componente de tipo área de texto en el área de diseño de la plantilla. • El Diseñador después de tener el componente seleccionado procede a modificar las propiedades (*Nombre, Posición X, Posición Y, Contenido, Fuente, Tamaño Vertical, Tamaño Horizontal, Color, Alineación*) del componente de tipo área de texto en el área de diseño donde fue creado.	Alta	Alta
	• **Prototipo**			

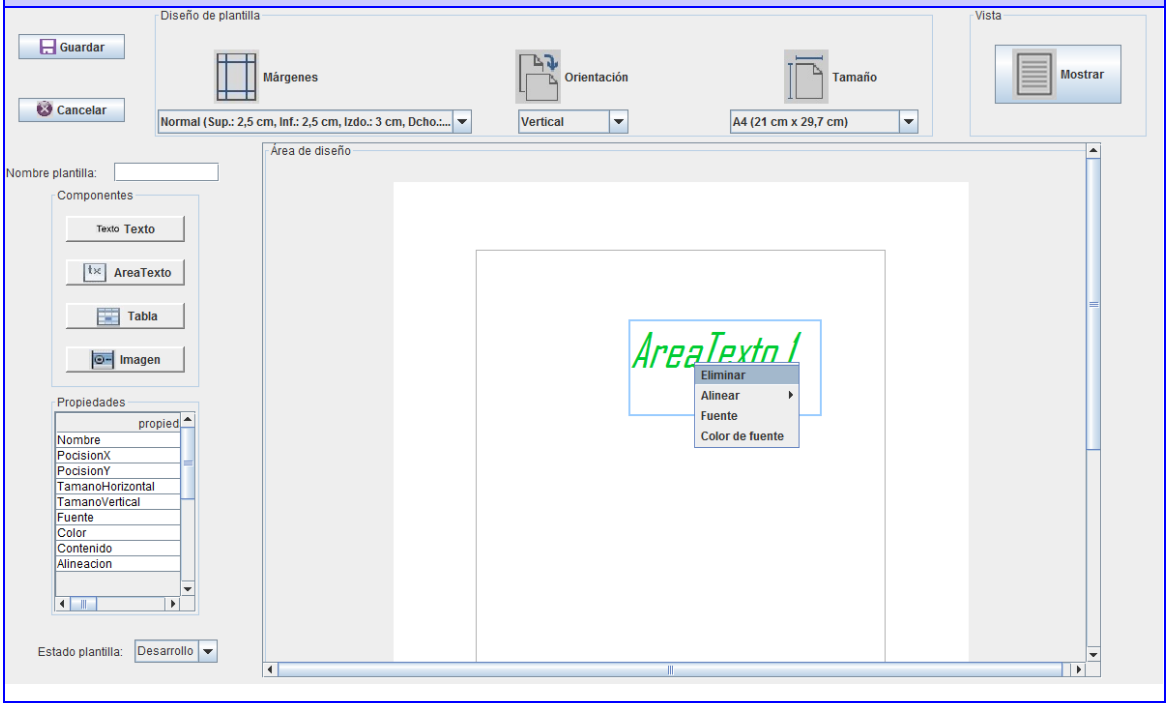


Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.
• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Contenido	• character varying	• Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
• Fuente	• character varying	• Campo de selección.
• Tamaño Vertical	• integer	• De solo lectura.
• Tamaño Horizontal	• integer	• De solo lectura.
• Color	• character varying	• Campo de selección.
• Alineación	• character varying	• Campo de selección.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. Si se introduce caracteres no permitidos el sistema muestra el	

	mensaje de error: “No se admite % y comillas simples”. 3. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.

Tabla 15. Descripción del requisito funcional Eliminar componente de tipo área de texto.

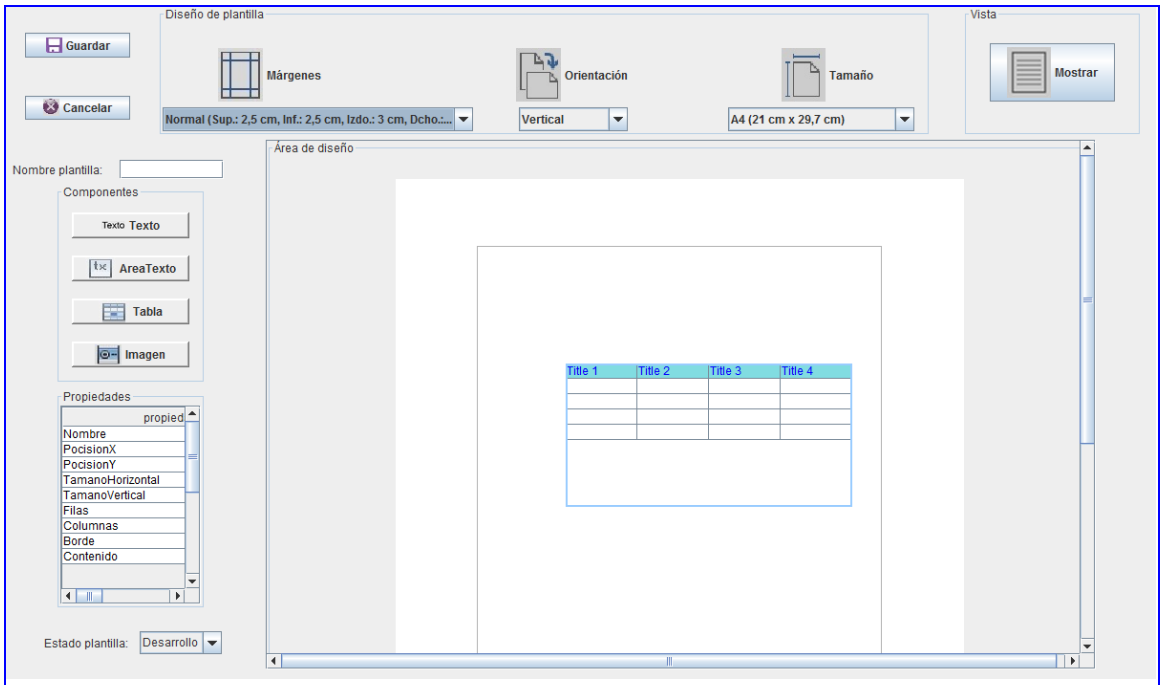
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_21	Eliminar componente de tipo área de texto.	• El requisito permite eliminar un componente de tipo texto en el área de diseño de la plantilla. • El Diseñador selecciona del área de diseño el componente de tipo área de texto a eliminar dando clic derecho sobre el área de texto y seleccionando la opción Eliminar.	Media	Alta
	• **Prototipo**			



Campos	Tipos de Datos	Reglas o Restricciones
•	•	•
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. En caso de seleccionar la acción se muestra un mensaje de advertencia "¿Está seguro de realizar la acción? ".	

Tabla 16. Descripción del requisito funcional Insertar componente de tipo tabla.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_22	Insertar componente de tipo tabla.	• El requisito permite insertar un componente de tipo tabla en el área de diseño de la plantilla. • El Diseñador después de crear la plantilla procede a insertar un componente de tipo tabla en el área de diseño de la plantilla al dar clic izquierdo encima del componente y luego en el área de diseño en cualquiera región. • Una vez insertado el componente este toma por defecto los valores de las propiedades de la tabla (*Nombre, Posición X, Posición Y, Tamaño Vertical, Tamaño Horizontal, Filas, Columnas, Borde, Contenido).*	Alta	Alta
	• **Prototipo**			

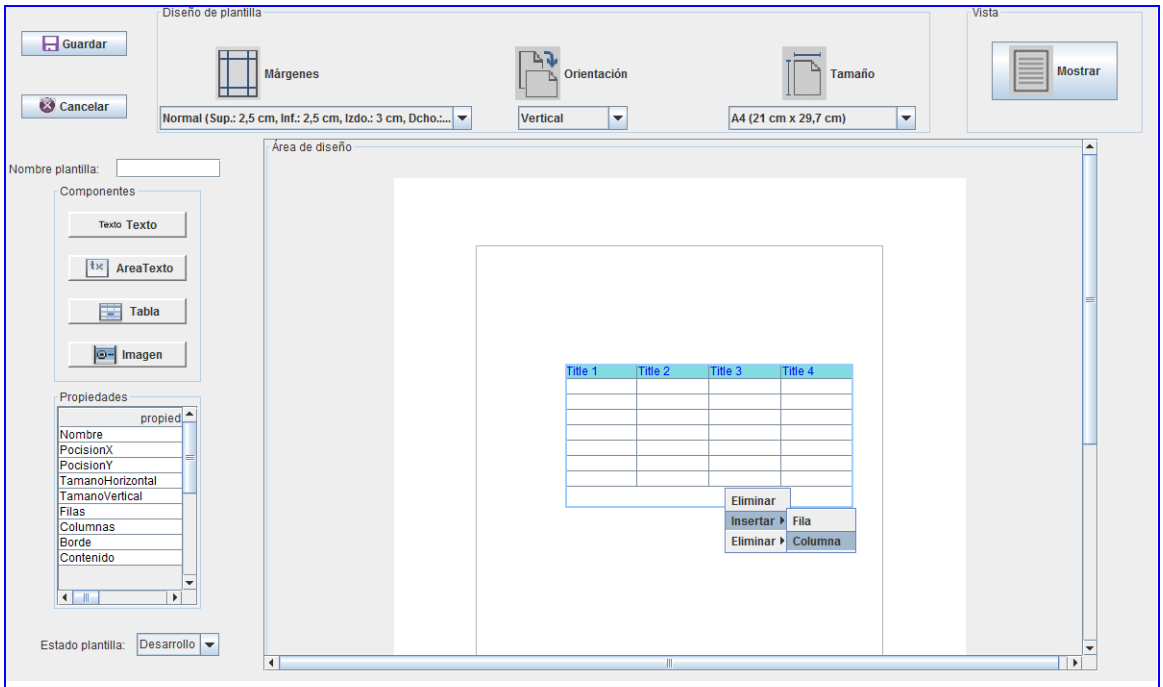


Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.
• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Tamaño Vertical	• integer	• De solo lectura.
• Tamaño Horizontal	• integer	• De solo lectura.
• Filas	• integer	• De solo lectura.
• Columnas	• integer	• De solo lectura.
• Borde	• integer	• De solo lectura.
• Contenido	• character varying	• Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. Si se introduce caracteres no permitidos el sistema muestra el mensaje	

		de error: “No se admite % y comillas simples”. 3. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.

Tabla 17. Descripción del requisito funcional Modificar componente de tipo tabla.

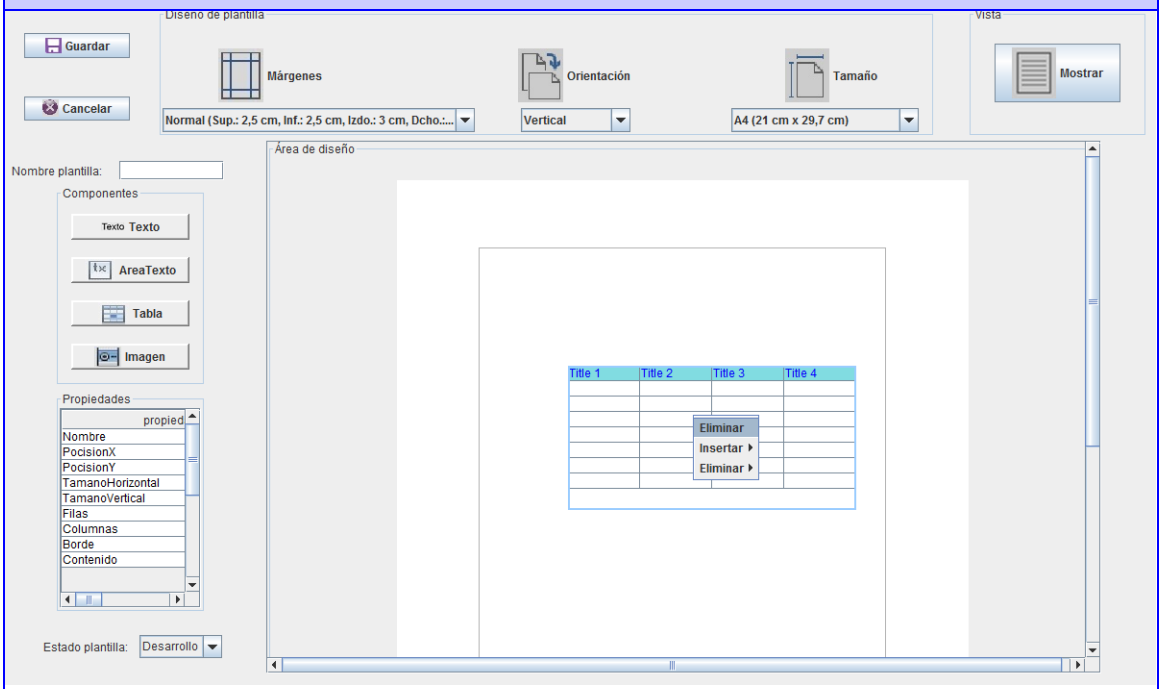
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_23	Modificar componente de tipo tabla.	• El requisito permite modificar las propiedades de un componente de tipo tabla en el área de diseño de la plantilla. • El Diseñador después de tener el componente seleccionado procede a modificar las propiedades (*Nombre, Posición X, Posición Y, Tamaño Vertical, Tamaño Horizontal, Filas, Columnas, Borde, Contenido)* del componente de tipo tabla en el área de diseño donde fue creado.	Alta	Alta
	• **Prototipo**			



Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.
• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Tamaño Vertical	• integer	• De solo lectura.
• Tamaño Horizontal	• integer	• De solo lectura.
• Filas	• integer	• De solo lectura.
• Columnas	• integer	• De solo lectura.
• Borde	• integer	• De solo lectura.
• Contenido	• character varying	• Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. Si se introduce caracteres no permitidos el sistema muestra el mensaje	

	de error: “No se admite % y comillas simples”. 3. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.

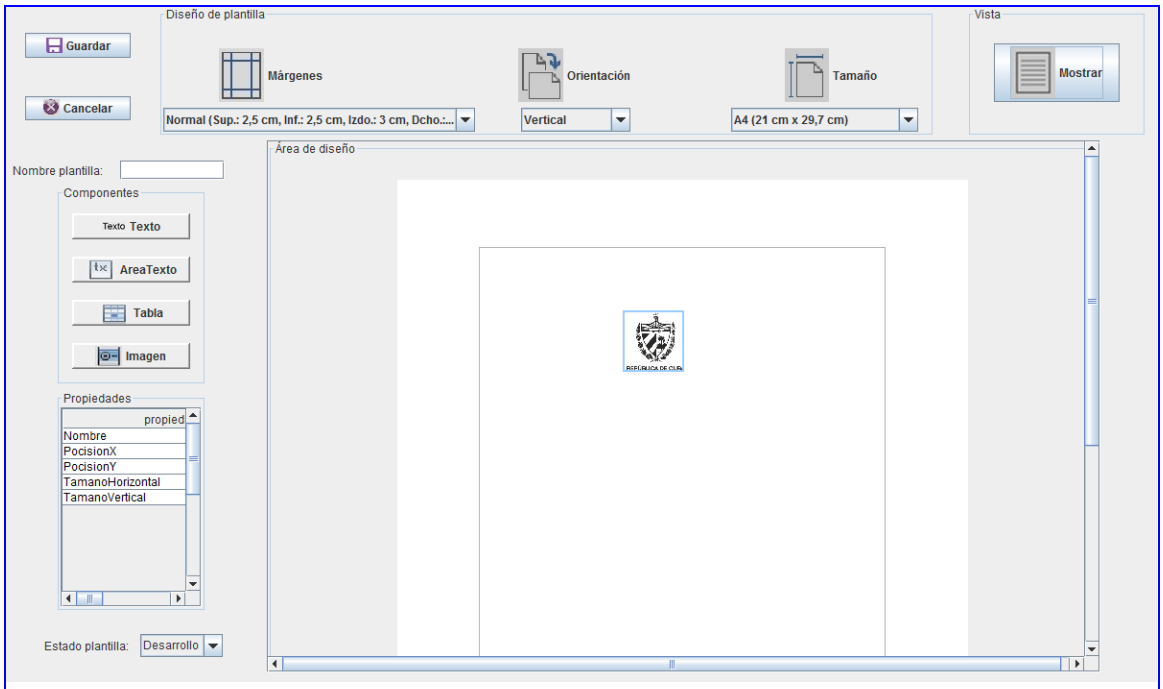
Tabla 18. Descripción del requisito funcional Eliminar componente de tipo tabla.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_24	Eliminar componente de tipo tabla.	• El requisito permite eliminar un componente de tipo tabla en el área de diseño de la plantilla. • El Diseñador selecciona del área de diseño el componente de tipo tabla a eliminar dando clic derecho sobre la tabla y seleccionando la opción Eliminar.	Media	Alta
	• **Prototipo**			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	

•	•	•
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. En caso de seleccionar la acción se muestra un mensaje de advertencia "¿Está seguro de realizar la acción? ".	

Tabla 19. Descripción del requisito funcional Insertar componente de tipo imagen.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_25	Insertar componente de tipo imagen.	• El requisito permite insertar un componente de tipo imagen en el área de diseño de la plantilla. • El Diseñador después de crear la plantilla procede a insertar un componente de tipo imagen en el área de diseño de la plantilla al dar clic izquierdo encima del componente y luego en el área de diseño en cualquiera región. • Una vez insertado el componente este toma por defecto los valores de las propiedades de la imagen (*Nombre, Posición X, Posición Y, Tamaño Vertical, Tamaño Horizontal).*	Alta	Alta
	• **Prototipo**			



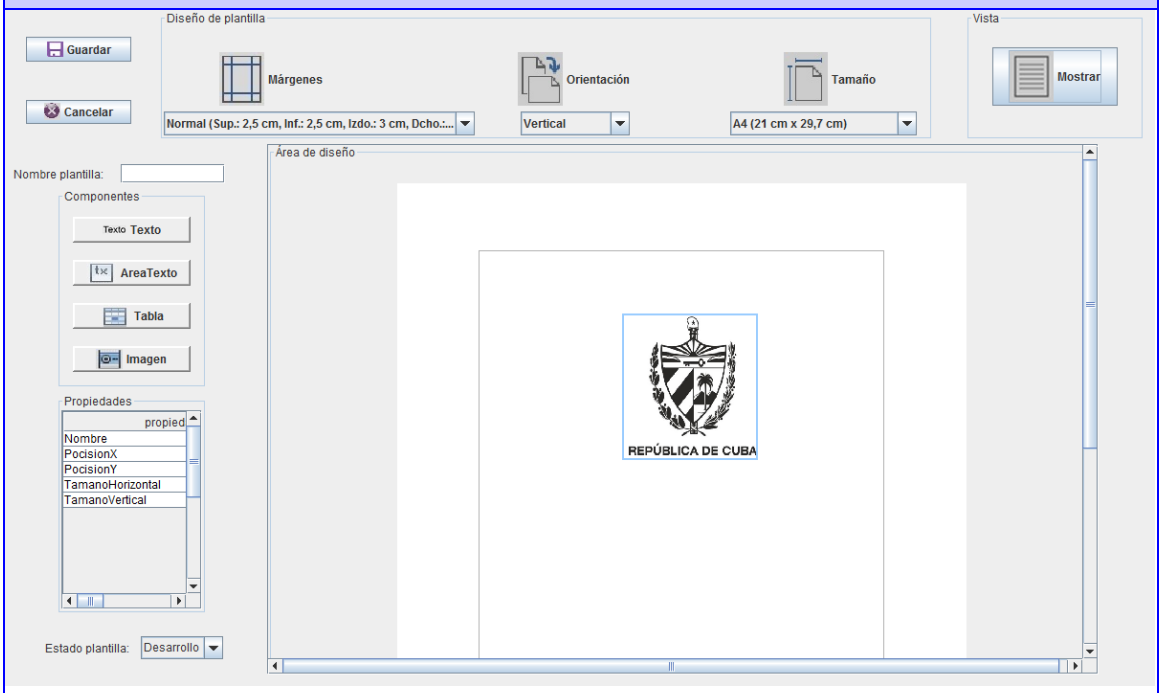
Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.
• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Tamaño Vertical	• integer	• De solo lectura.
• Tamaño Horizontal	• integer	• De solo lectura.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema.	

Tabla 20. Descripción del requisito funcional Modificar componente de tipo imagen.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_26	Modificar componente de tipo imagen.	• El requisito permite modificar las propiedades de un componente de tipo imagen en el área de	Alta	Alta

	diseño de la plantilla. • El Diseñador después de tener el componente seleccionado procede a modificar las propiedades (*Nombre, Posición X, Posición Y, Tamaño Vertical, Tamaño Horizontal)* del componente de tipo imagen en el área de diseño de la plantilla donde fue creado.		

- **Prototipo**

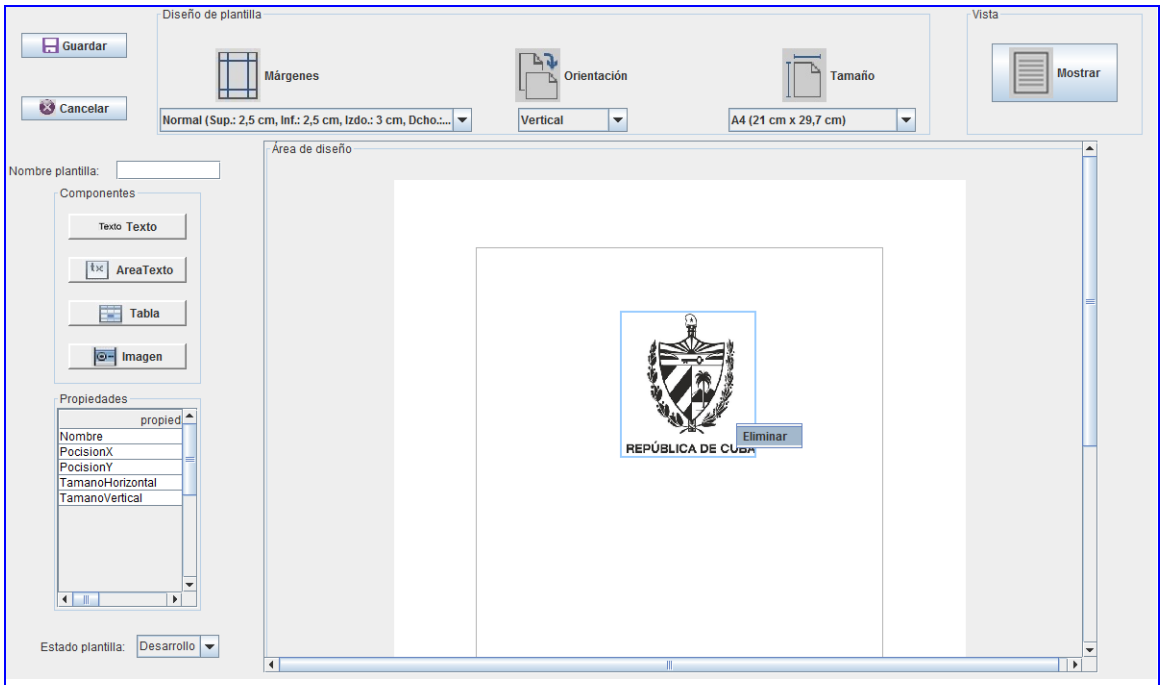


Campos	Tipos de Datos	Reglas o Restricciones
• Nombre	• character varying	• De solo lectura.

• Posición X	• integer	• De solo lectura.
• Posición Y	• integer	• De solo lectura.
• Tamaño Vertical	• integer	• De solo lectura.
• Tamaño Horizontal	• integer	• De solo lectura.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema.	

Tabla 21. Descripción del requisito funcional Eliminar componente de tipo imagen.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RFDP_27	Eliminar componente de tipo imagen.	• El requisito permite eliminar un componente de tipo imagen en el área de diseño de la plantilla. • El Diseñador selecciona del área de diseño el componente de tipo imagen a eliminar dando clic derecho sobre la imagen y seleccionando la opción Eliminar.	Media	Alta
	• **Prototipo**			



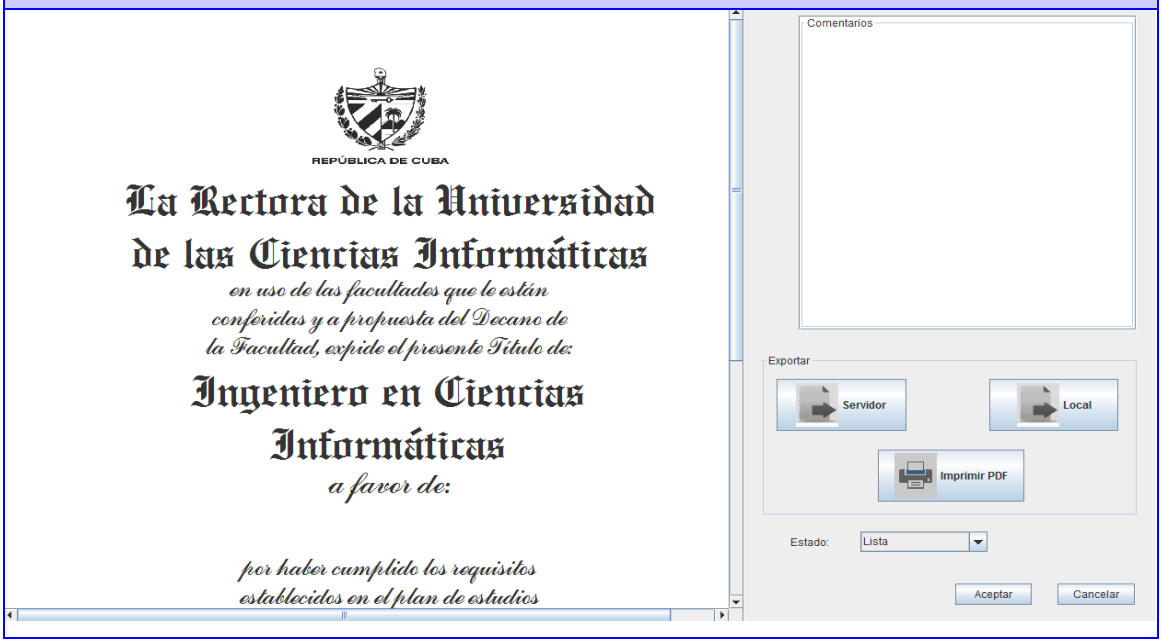
Campos	Tipos de Datos	Reglas o Restricciones
•	•	•
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. En caso de seleccionar la acción se muestra un mensaje de advertencia "¿Está seguro de realizar la acción? ".	

Tabla 22. Descripción del requisito funcional Revisar plantilla.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_41	Revisar plantilla.	• El requisito permite revisar la plantilla previamente diseñada por el diseñador. • El Gestor selecciona la agrupación funcional Gestor en el menú interior, luego la funcionalidad Revisar, y se le muestra el listado de plantillas.	Media	Media

	• El Gestor selecciona la plantilla a revisar del listado de plantillas mostrado. Posteriormente se muestra la plantilla a revisar y un área de texto para escribir los comentarios. • Si el Gestor le realiza algún comentario a la platilla le cambia el estado de Lista a Rechazada, en caso de que no se le haga ningún comentario simplemente le cambia el estado de Lista a Terminada.		

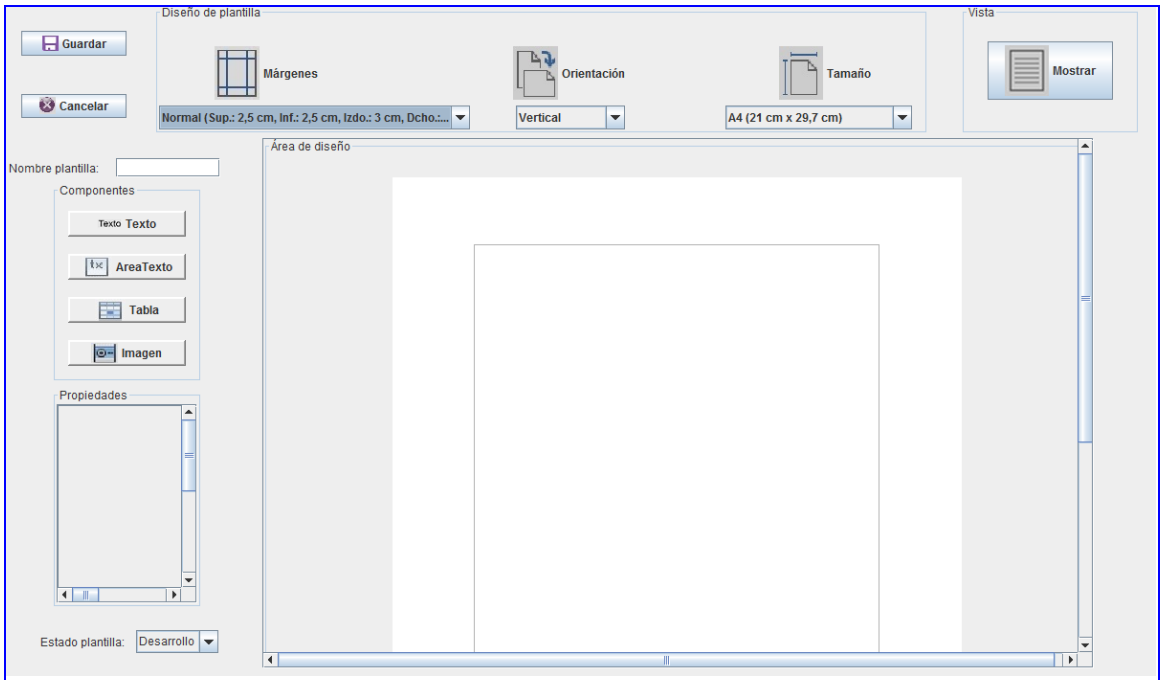
- **Prototipo**



Campos	Tipos de Datos	Reglas o Restricciones
• Comentarios	• character varying	• Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
• Estado	• character varying	• Campo de Selección.
Observaciones	1. El gestor debe estar debidamente autenticado en el sistema. 2. Si se introduce caracteres no permitidos el sistema muestra el mensaje de error: “No se admite % y comillas simples”. 3. Si se trata de introducir más de los caracteres permitidos para los campos el sistema no permite seguir introduciéndolos.	

Tabla 23. Descripción del requisito funcional Crear plantilla.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_28	Crear plantilla.	• El requisito permite crear una plantilla. • El Diseñador selecciona la agrupación funcional Diseño, luego la funcionalidad Plantilla y la acción Crear plantilla del grupo de acciones superiores.	Alta	Alta
	• **Prototipo**			



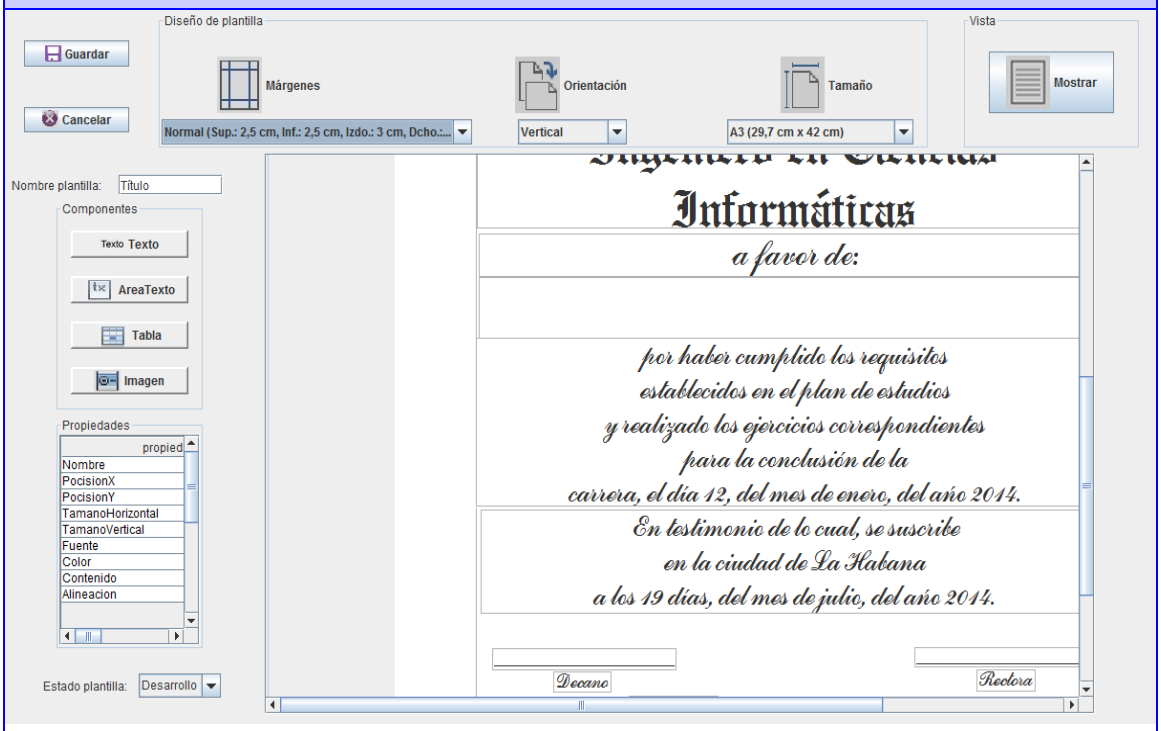
Campos	Tipos de Datos	Reglas o Restricciones
•	•	•
Observaciones	1. El diseñador debe estar debidamente autenticado en el sistema.	

Tabla 24. Descripción del requisito funcional Modificar plantilla.

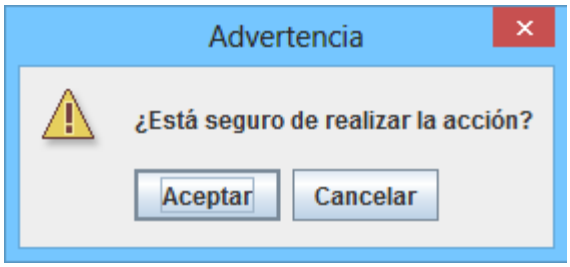
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_29	Modificar plantilla.	• El requisito permite modificar una plantilla. • El Diseñador selecciona la agrupación funcional Diseño, luego la funcionalidad Plantilla, posteriormente selecciona la plantilla de la lista de plantillas y la acción Modificar plantilla del grupo de acciones superiores.	Alta	Alta

	• El Diseñador modifica la plantilla y luego se guarda las modificaciones realizadas a la plantilla.		

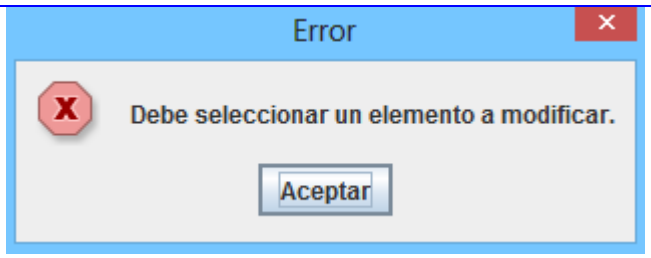
- **Prototipo**



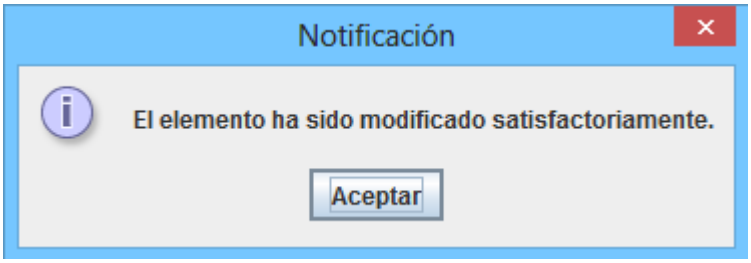
Prototipo 1



Prototipo 2



Prototipo 3



Prototipo 4

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 es de 30.
• Estado	• character varying	• Campo de selección. Campo requerido.
Observaciones	1. El Diseñador debe estar debidamente autenticado en el sistema. 2. Si ocurre un error durante la operación se muestra un mensaje de error con el título: “Error” indicando el error que produjo el problema. 3. En caso de que se escriba apóstrofos y porcientos se muestra un mensaje de error de color rojo "No se admite apóstrofos ni porcientos.". 4. Para todos los campos cuando llegue a la cantidad máxima de caracteres el sistema no permitirá continuar escribiendo. 5. En caso de cancelar la acción se muestra un mensaje de confirmación “¿Está seguro de realizar la acción?” 6. Si selecciona la acción modificar y no ha seleccionado un elemento se muestra el mensaje de error “Debe seleccionar un elemento a modificar.” 7. Si se modifica satisfactoriamente el elemento se mostrará un mensaje de información “El elemento ha sido modificado satisfactoriamente”.	

Tabla 25. Descripción del requisito Eliminar plantilla.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFDP_30	Eliminar plantilla.	• El requisito permite eliminar una plantilla. • El Diseñador selecciona la agrupación funcional Diseño, luego la funcionalidad Plantilla, posteriormente selecciona la plantilla que desea eliminar de la lista de plantillas y después la acción Eliminar plantilla del grupo de acciones superiores. • En caso de que se encuentre autenticado el Gestor, este selecciona la agrupación funcional Gestor, luego la funcionalidad Revisar, posteriormente selecciona la plantilla que desea eliminar de la lista de plantilla y después la acción Eliminar plantilla del grupo de acciones superiores. • El Diseñador o el	Baja	Alta

	Gestor elimina la plantilla y luego se muestra la lista de plantilla.		

- **Prototipo**

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

Observaciones	1. El Diseñador o el Gestor debe estar debidamente autenticado en el sistema. 2. Si selecciona la acción eliminar plantilla se mostrará el mensaje de confirmación: "¿Está seguro de realizar la acción?" 3. Si se elimina satisfactoriamente la plantilla el sistema mostrará un mensaje de información "El elemento ha sido eliminado satisfactoriamente". 4. Si ocurre un error durante la operación se muestra un mensaje de error con el título: "Error" indicando el error que produjo el problema. 5. Si selecciona la acción eliminar plantilla y no ha seleccionado un elemento se muestra el mensaje "Debe seleccionar un elemento a eliminar.".

Anexo 2: Modelo lógico de datos.

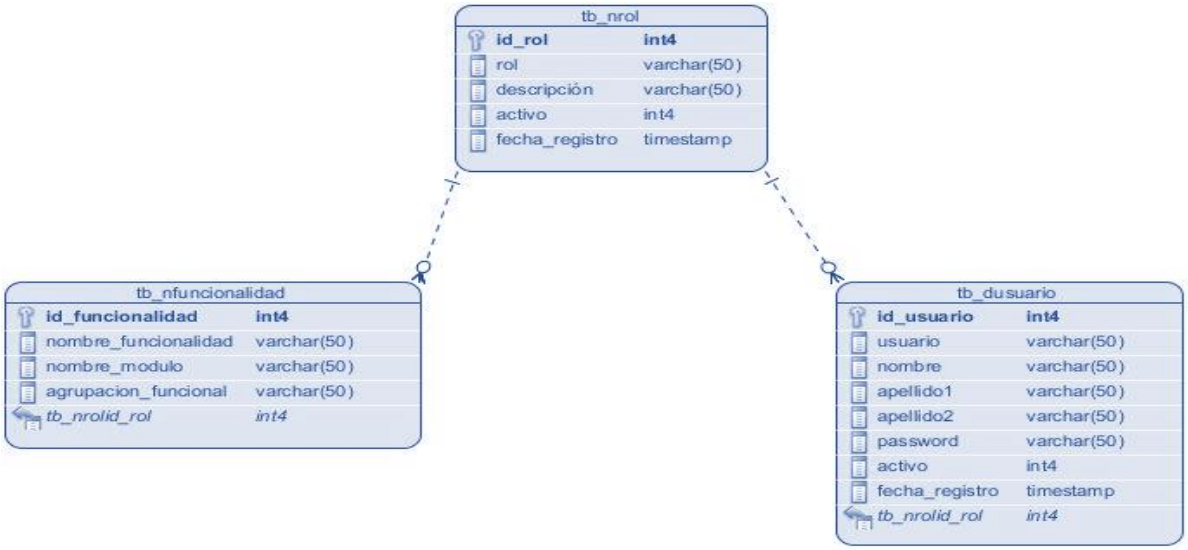


Figura 31. Modelo lógico de datos del esquema seguridad de la propuesta de solución.

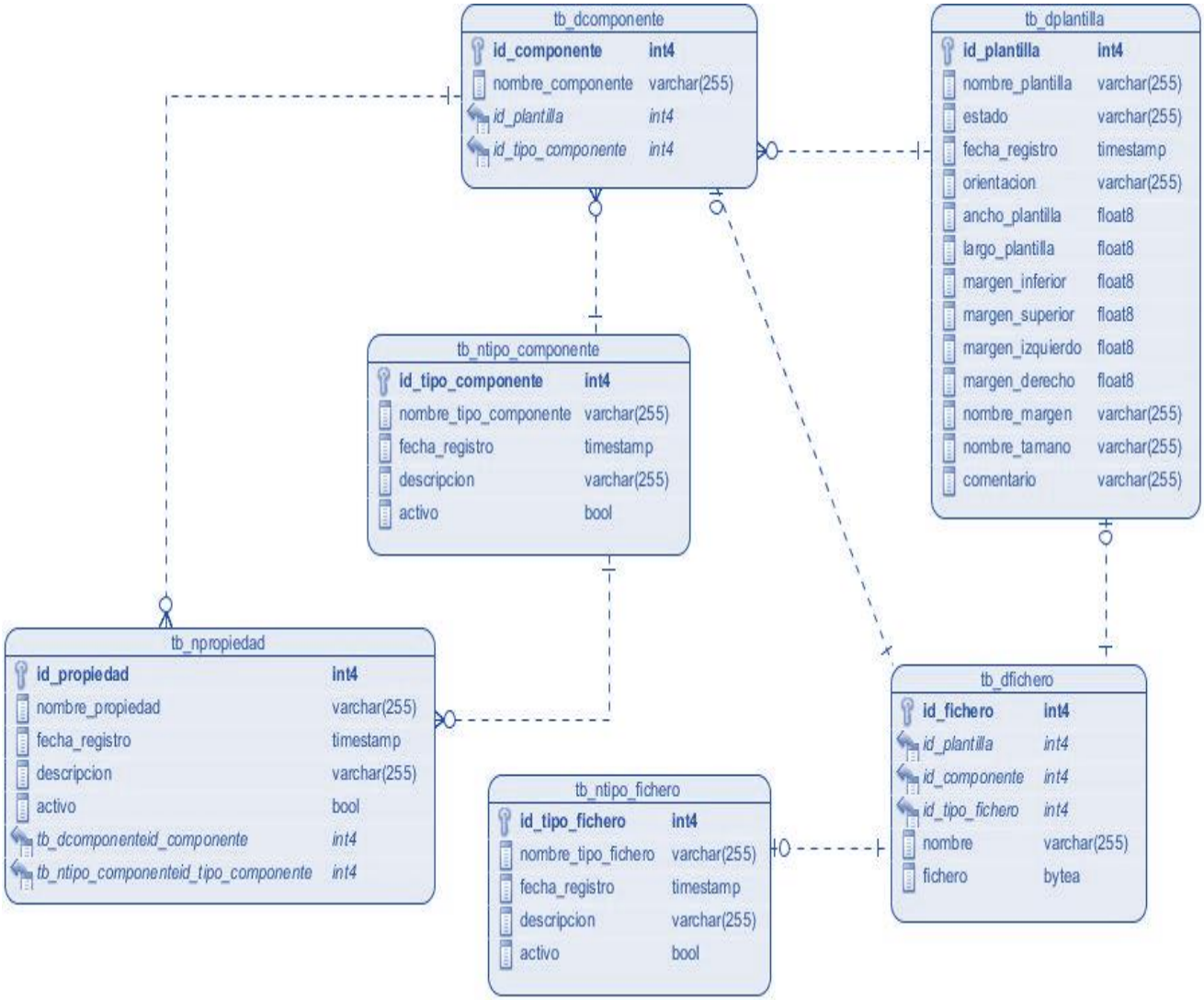


Figura 32. Modelo lógico de datos del esquema diseñador de la propuesta de solución.

Anexo 3: Casos de pruebas unitarias. Técnica de caja blanca.

Tabla 26. Casos de prueba de caja blanca. RFDP30_EliminarPlantilla.

Técnica Caja blanca	**Código caso de prueba:** DIN-SGU-DP-CPCB_ RFDP30_EliminarPlantilla
Probador: Alejandro Vázquez Chávez	
Procedimiento prueba automatizada	
Descripción: El dato de entrada será el identificador de la plantilla a eliminar.	
Datos de entrada:	Identificador igual a 0.
Tipo de dato esperado:	Vacío.
Función de evaluación:	

```
public void testEliminarPlantilla() throws Exception {
    System.out.println("eliminarPlantilla");
    int id = 0;
    tb_dplantilla instance = new tb_dplantilla();
    instance.eliminarPlantilla(id);

}
```

Evaluación del caso de prueba: Satisfactorio.

Tabla 27. Casos de prueba de caja blanca. RFDP31_ListarPlantillaDiseñador.

Técnica Caja blanca	**Código caso de prueba:** DIN-SGU-DP-CPCB_ RFDP31_ListarPlantillaDiseñador
Probador: Alejandro Vázquez Chávez	
Procedimiento prueba automatizada	
Descripción: No tiene entrada, se espera el listado de plantillas del diseñador.	
Datos de entrada:	No tiene.
Tipo de dato esperado:	Object [][]
Función de evaluación:	

```
public void testListarPlantillaDisenar() throws Exception {
    System.out.println("listarPlantillaDisenar");
    tb_dplantilla instance = new tb_dplantilla();
    Object[][] expResult = null;
    Object[][] result = instance.listarPlantillaDisenar();
    assertEquals(expResult, result);

}
```

Evaluación del caso de prueba: Satisfactorio.

Tabla 28. Casos de prueba de caja blanca. RFDP32_VerDetallesPlantilla.

Técnica Caja blanca	**Código caso de prueba:** DIN-SGU-DP-CPCB_ RFDP32_VerDetallesPlantilla
Probador: Alejandro Vázquez Chávez	
Procedimiento prueba automatizada	
Descripción: Los datos de entrada serán los atributos y el identificador de la plantilla a consultar.	
Datos de entrada:	Objeto nulo, identificador igual a 0.
Tipo de dato esperado:	Vacío.
Función de evaluación:	

```
public void testConsultarPlantilla() {
    System.out.println("consultarPlantilla");
    int fila = 0;
    vista_diseñar instance = null;
    instance.consultarPlantilla(fila);

}
```

Evaluación del caso de prueba: Satisfactorio.

Tabla 29. Casos de prueba de caja blanca. RFDP38_InsertarFuentes.

Técnica Caja blanca	**Código caso de prueba:** DIN-SGU-DP-CPCB_ RFDP38_InsertarFuentes
Probador: Alejandro Vázquez Chávez	
Procedimiento prueba automatizada	
Descripción: Los datos de entrada será el archivo TTF.	
Datos de entrada:	Objeto nulo.
Tipo de dato esperado:	Vacío.
Función de evaluación:	

```
public void testRegistrarFuenteTTF() throws Exception {
    System.out.println("registrarFuenteTTF");
    File dir = null;
    Font expResult = null;
    Font result = cc_gestionar_fuentes.registrarFuenteTTF(dir);
    assertEquals(expResult, result);

}
```

Evaluación del caso de prueba: Satisfactorio.

Tabla 30. Casos de prueba de caja blanca. RFDP48_GuardarPlantilla.

Técnica Caja blanca	**Código caso de prueba:** DIN-SGU-DP-CPCB_ RFDP48_GuardarPlantilla
Probador: Alejandro Vázquez Chávez	
Procedimiento prueba automatizada	
Descripción: Los datos de entrada serán los atributos de la plantilla a guardar.	
Datos de entrada:	Objeto nulo.
Tipo de dato esperado:	Vacío.

Función de evaluación:

```
public void testAdicionarPlantilla() throws Exception {
    System.out.println("AdicionarPlantilla");
    Plantilla plantilla = null;
    tb_dplantilla instance = new tb_dplantilla();
    int expResult = 0;
    int result = instance.AdicionarPlantilla(plantilla);
    assertEquals(expResult, result);

}
```

Evaluación del caso de prueba: Satisfactorio.

Anexo 4: Diseños de casos de pruebas. Técnica de caja negra.

Tabla 31. Diseño de casos de prueba.RFDP16_Insertar componente de tipo texto.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.2 Insertar elemento correctamente.	Mediante este escenario el usuario puede insertar un componente texto correctamente.	El sistema inserta el componente texto en el área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores y luego se selecciona en la paleta de componentes el componente texto, 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 texto correctamente en el área de diseño de la plantilla.	El sistema no inserta el componente texto en el área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción

			Crear plantilla en el área de acciones superiores y luego se selecciona en la paleta de componentes el componente texto, dando clic encima del componente y luego en el área de diseño.

Tabla 32. Diseño de casos de prueba. RFDP17_Modificar componente de tipo texto. Parte uno.

Escenario	Descripción	Variable 1: Nombre	Variable 2: Posición X	Variable 3: Posición Y
EC 1.1 Modificar propiedades correctamente.	Mediante este escenario el usuario puede modificar propiedades de un componente texto correctamente.	V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.
EC. 1.2 Insertar datos incorrectos.	Mediante este escenario se introducen datos incorrectos para modificar las propiedades del componente texto.	V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.

Tabla 33. Diseño de casos de prueba. RFDP17_Modificar componente de tipo texto. Parte dos.

Variable 4: Tamaño Horizontal	Variable 5: Tamaño Vertical	Variable 6: Contenido	Variable 7: Fuente	Variable 8:Alinear	Variable 9:Tamaño
V	V	V	V	V	V
Solo lectura.	Solo lectura.	Prueba	Arial	Derecha	18
V	V	V	V	V	V
Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.	Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.	Times New Roman	Centro	Solo dígitos del 8 al 32
V	V	I	V	V	V
Solo lectura.	Solo lectura.	´Pureba%¨	Tahoma	Solo lectura.	24
V	V	V	V	V	I
Solo lectura.	Solo lectura.	Estudiantes	Arial	Solo lectura.	100y
V	V	I	V	V	I
Solo lectura.	Solo lectura.	El usuario intenta entrar más de 100 caracteres.	Impact	Solo lectura.	El usuario intenta entrar dígitos anteriores al 8 y posteriores al 32.
V	V	I	V	V	V
Solo lectura.	Solo lectura.	El usuario entra al menos de 2 caracteres.	Times New Roman	Solo lectura.	32

Tabla 34. Diseño de casos de prueba. RFDP17_Modificar componente de tipo texto. Parte tres.

Variable 10: Color						
R	**G**	**B**	**H**	**S**	**B**	**#**
V	V	V	V	V	V	V

213	30	230	295	87	90	d51ee6
V	V	V	V	V	V	V
Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 6 caracteres.
V	V	V	V	V	V	V
345	555	33	67	6	23	re45t6
V	V	V	V	V	V	V
21	233	465	23	67	45	e45r5
El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 6 caracteres.
V	V	V	V	V	V	V
213	345	33d	43	55f	2we	se43r5

Tabla 35. Diseño de casos de prueba. RFDP17_Modificar componente de tipo texto. Parte cuatro.

Respuesta del sistema	Flujo central
El sistema modifica las propiedades del componente texto correctamente.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores. Después de insertado el componente texto se procede a modificar los valores de las propiedades al dar clic derecho sobre el componente y modificar las propiedades que se muestran.
El sistema muestra un mensaje: "No se admiten apóstrofes, ni porcientos sobre el campo requerido".	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores. Después

El sistema muestra un mensaje en rojo encima del componente: "Solo se admiten dígitos del 8 al 32."	de insertado el componente texto se procede a modificar los valores de las propiedades al dar clic derecho sobre el componente y modificar las propiedades incorrectamente que se muestran.
El sistema no deja seguir entrando caracteres.	
El sistema muestra un mensaje: "Entre al menos 2 caracteres".	

Tabla 36. Diseño de casos de prueba. RFDP18_Eliminar componente de tipo texto.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar componente correctamente.	Mediante este escenario el usuario puede eliminar un componente texto correctamente.	El sistema elimina el componente texto del área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores. Una vez insertado el componente texto para eliminarlo se selecciona y se da clic derecho encima de este y se selecciona la opción Eliminar.

Tabla 37. Diseño de casos de prueba. RFDP19_Insertar componente de tipo área de texto.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.2 Insertar elemento correctamente.	Mediante este escenario el usuario puede insertar un componente área de texto correctamente.	El sistema inserta el componente área de texto en el área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Una vez creada la plantilla se selecciona en la paleta de componentes el componente área de texto, 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 área de texto en el área de diseño de la plantilla.	El sistema no inserta el componente texto en el área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Una vez creada la plantilla se selecciona en la paleta de componentes el componente área de texto, dando clic encima del componente y luego en el área de diseño.

Tabla 38. Diseño de casos de prueba. RFDP20_Modificar componente de tipo área de texto. Parte uno.

Escenario	Descripción	Variable 1: Nombre	Variable 2: Posición X	Variable 3: Posición Y
EC 1.1 Modificar propiedades correctamente.	Mediante este escenario el usuario puede modificar propiedades de un componente área de texto correctamente.	V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.
EC. 1.2 Insertar datos incorrectos.	Mediante este escenario se introducen datos incorrectos para modificar las propiedades del componente área de texto.	V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.

Tabla 39. Diseño de casos de prueba. RFDP20_Modificar componente de tipo área de texto. Parte dos.

Variable 4: Tamaño Horizontal	Variable 5: Tamaño Vertical	Variable 6: Contenido	Variable 7: Fuente	Variable 8:Alinear	Variable 9:Tamaño
V	V	V	V	V	V
Solo lectura.	Solo lectura.	Prueba	Arial	Derecha	18
V	V	V	V	V	V
Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.	Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.	Times New Roman	Centro	Solo dígitos del 8 al 32
V	V	I	V	V	V
Solo lectura.	Solo lectura.	´Pureba%¨	Tahoma	Solo lectura.	24
V	V	V	V	V	I
Solo lectura.	Solo lectura.	Estudiantes	Arial	Solo lectura.	100y
V	V	I	V	V	I
Solo lectura.	Solo lectura.	El usuario intenta entrar más de 100 caracteres.	Impact	Solo lectura.	El usuario intenta entrar dígitos anteriores al 8 y posteriores al 32.
V	V	I	V	V	V
Solo lectura.	Solo lectura.	El usuario entra al menos de 2 caracteres.	Times New Roman	Solo lectura.	32

Tabla 40. Diseño de casos de prueba. RFDP20_Modificar componente de tipo área de texto. Parte tres.

Variable 10: Color						
R	**G**	**B**	**H**	**S**	**B**	**#**
V	V	V	V	V	V	V

213	30	230	295	87	90	d51ee6
V	V	V	V	V	V	V
Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 3 caracteres.	Hasta 6 caracteres.
V	V	V	V	V	V	V
345	555	33	67	6	23	re45t6
V	V	V	V	V	V	V
21	233	465	23	67	45	e45r5
I	I	I	I	I	I	I
El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 3 caracteres.	El usuario intenta introducir más de 6 caracteres.
V	V	V	V	V	V	V
213	345	33d	43	55f	2we	se43r5

Tabla 41. Diseño de casos de prueba. RFDP20_Modificar componente de tipo área de texto. Parte cuatro.

Respuesta del sistema	Flujo central
El sistema modifica las propiedades del componente área de texto correctamente.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de creada la plantilla e insertado el componente área de texto se procede a modificar los valores de las propiedades al dar clic derecho sobre el componente y modificar las propiedades que se muestran.
El sistema muestra un mensaje en rojo encima del componente: "No	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra

se admiten apóstrofes, ni porcientos sobre el campo requerido".	el listado de plantillas y el área de acciones superiores. Después de creada la plantilla e insertado el componente área de texto se procede a modificar los valores de las propiedades al dar clic derecho sobre el componente y modificar las propiedades que se muestran.
El sistema muestra un mensaje en rojo encima del componente: "Solo se admiten dígitos del 8 al 32.".	
El sistema no deja seguir entrando caracteres.	
El sistema muestra un mensaje en rojo encima del componente: "Entre al menos 2 caracteres.".	

Tabla 42. Diseño de casos de prueba. RFDP21_Eliminar componente de tipo área de texto.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar componente correctamente.	Mediante este escenario el usuario puede eliminar un componente área de texto correctamente.	El sistema elimina el componente área de texto del área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Una vez creada la plantilla e insertado el componente área de texto para eliminarlo se selecciona y se da clic derecho encima de este y se selecciona la opción Eliminar.

Tabla 43. Diseño de casos de prueba. RFDP22_Insertar componente de tipo tabla.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.2 Insertar elemento correctamente.	Mediante este escenario el usuario puede insertar	El sistema inserta el componente tabla en el área	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad

	un componente de tipo tabla correctamente.	de diseño de la plantilla.	Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de creada la plantilla se selecciona en la paleta de componentes el componente tabla, dando clic encima del componente y luego en cualquier región del área de diseño.
EC 1.2 Insertar elemento incorrectamente.	Mediante este escenario el usuario no puede insertar un componente de tipo tabla en el área de diseño de la plantilla.	El sistema no inserta el componente tabla en el área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de creada la plantilla se selecciona en la paleta de componentes el componente tabla, dando clic encima del componente y luego en cualquier región del área de diseño.

Tabla 44. Diseño de casos de prueba. RFDP23_Modificar componente de tipo tabla. Parte uno.

Escenario	Descripción	Variable 1: Nombre	Variable 2: Posición X	Variable 3: Posición Y
EC 1.1 Modificar propiedades correctamente.	Mediante este escenario el usuario puede modificar las propiedades de un componente de tipo tabla correctamente.	V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.
EC 1.2 Modificar propiedades incorrectamente	Mediante este escenario el usuario no puede modificar las propiedades de un componente de tipo tabla correctamente.	V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.

Tabla 45. Diseño de casos de prueba. RFDP23_Modificar componente de tipo tabla. Parte dos.

Variable 4: Tamaño Horizontal	Variable 5: Tamaño Vertical	Variable 6: Filas	Variable 7: Columnas	Variable 8: Borde
V	V	V	V	V
Solo lectura.	Solo lectura.	5	4	Solo lectura.
V	V	V	V	V
Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.	Solo lectura.	Solo lectura.	Solo lectura.
V	V	V	V	V
Solo lectura.	Solo lectura.	Solo lectura.	Solo lectura.	Solo lectura.
V	V	V	V	V
Solo lectura.	Solo lectura.	Solo lectura.	Solo lectura.	Solo lectura.
V	V	V	V	V
Solo lectura.	Solo lectura.	Solo lectura.	Solo lectura.	Solo lectura.

Tabla 46. Diseño de casos de prueba. RFDP23_Modificar componente de tipo tabla. Parte tres.

Variable 9: Contenido	Respuesta del sistema	Flujo central
V Prueba V Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.	El sistema modifica las propiedades del componente tabla correctamente.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de crear la plantilla se procede a seleccionar el componente tabla modificando los valores de las propiedades al dar clic derecho sobre el componente y modificar las propiedades que se muestran.
I	El sistema muestra un mensaje: “Entre un número válido”.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y

Vacío		el área de acciones superiores. Después de crear la plantilla se procede a seleccionar el componente tabla modificando los valores de las propiedades al dar clic derecho sobre el componente y modificar las propiedades que se muestran.
I		
El usuario intenta entrar más de 100 caracteres.		
I		
El usuario entra al menos de 2 caracteres.		

Tabla 47. Diseño de casos de prueba. RFDP24_Eliminar componente de tipo tabla.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar componente correctamente.	Mediante este escenario el usuario puede eliminar un componente de tipo tabla correctamente.	El sistema elimina el componente de tipo tabla del área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de tener creada la plantilla y tener seleccionado el componente tabla se le da clic derecho encima de este y se selecciona la opción Eliminar.

Tabla 48. Diseño de casos de prueba. RFDP25_Insertar componente de tipo imagen.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Insertar elemento correctamente.	Mediante este escenario el usuario puede insertar un	El sistema inserta el componente imagen en el área	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad

	componente de tipo imagen correctamente.	de diseño de la plantilla.	Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de crear la plantilla 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 de tipo 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 la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de crear la plantilla se selecciona en la paleta de componentes el componente imagen, dando clic encima del componente y luego en el área de diseño.

Tabla 49. Diseño de casos de prueba. RFDP26_Modificar componente de tipo imagen. Parte uno.

Escenario	**Descripción**	**Variable 1: Nombre**	**Variable 2: Posición X**	**Variable 3: Posición Y**	**Variable 4: Tamaño Horizontal**
EC 1.1 Modificar propiedades correctamente.	Mediante este escenario el usuario puede modificar las propiedades de un componente imagen correctamente.	V	V	V	V
		Solo lectura.	Solo lectura.	Solo lectura.	Solo lectura.
		V	V	V	V
		Solo lectura	Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.	Se modifica al arrastrar el componente por el área de diseño.

Tabla 50. Diseño de casos de prueba. RFDP26_Modificar componente de tipo imagen. Parte dos.

Variable 5: Tamaño Vertical	**Respuesta del sistema**	**Flujo central**
V		

Solo lectura. V Se modifica al arrastrar el componente por el área de diseño.	El sistema modifica las propiedades del componente imagen correctamente.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de tener creada la plantilla y el componente se procede a modifica los valores de las propiedades del componente imagen al arrastrar el componente por el área de diseño.

Tabla 51. Diseño de casos de prueba. RFDP27_Eliminar componente de tipo imagen.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar componente correctamente.	Mediante este escenario el usuario puede eliminar un componente de tipo imagen correctamente.	El sistema elimina el componente imagen del área de diseño de la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Después de tener creada la plantilla y el componente imagen se le da clic derecho encima de este y se selecciona la opción Eliminar.

Tabla 52. Diseño de casos de prueba. RFDP28_Crear plantilla.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Crear plantilla.	Mediante este escenario el usuario puede crear una plantilla correctamente.	El sistema crea la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores.

Tabla 53. Diseño de casos de prueba. RFDP30_Eliminar plantilla.

Escenario	Descripción	Respuesta del sistema	Flujo central

EC 1.1 Eliminar plantilla correctamente.	Mediante este escenario el usuario puede eliminar una plantilla correctamente.	El sistema elimina la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la plantilla que se desea eliminar y seguidamente la opción Eliminar plantilla en el área de acciones superiores.
EC 1.2 Eliminar plantilla incorrectamente.	Mediante este escenario el usuario no puede eliminar una plantilla correctamente.	El sistema no elimina la plantilla.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Eliminar plantilla en el área de acciones superiores y no se selecciona la plantilla a eliminar.

Tabla 54. Diseño de casos de prueba. RFDP41_Revisar plantilla. Parte uno.

Escenario	Descripción	Estado plantilla	Comentarios
EC 1.1 Revisar plantilla correctamente.	Mediante este escenario se revisa la plantilla correctamente.	V	V
		Seleccionado	Prueba
		Campo de selección	Campo de texto
		Seleccionado	Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.
EC 1.1 Revisar plantilla incorrectamente.	Mediante este escenario no se revisa la plantilla correctamente.	V	I
		Seleccionado	%"Prueba"%
	Mediante este escenario no se revisa la plantilla correctamente.	V	I
		Seleccionado	El usuario intenta entrar más de 100 caracteres.
		V	I

		Seleccionado	El usuario entra al menos de 2 caracteres.

Tabla 55. Diseño de casos de prueba. RFDP41_Revisar plantilla. Parte dos.

Respuesta del sistema	Flujo central
El sistema mostrará el mensaje: "El elemento ha sido modificado satisfactoriamente".	El usuario selecciona la agrupación funcional Gestor en el menú interior, luego la funcionalidad Revisar. Después selecciona la plantilla a revisar del listado de plantillas y la acción Revisar plantilla de las acciones superiores.
El sistema muestra un mensaje: "No se admiten apóstrofes, ni porcientos sobre el campo requerido".	El usuario selecciona la agrupación funcional Gestor en el menú interior, luego la funcionalidad Revisar. Después selecciona la plantilla a revisar del listado de plantillas y la acción Revisar plantilla de las acciones superiores.
El sistema no deja seguir entrando caracteres.	
El sistema muestra un mensaje: "Entre al menos 2 caracteres".	

Tabla 56. Diseño de casos de prueba. RFDP48_Guardar plantilla. Parte uno.

Escenario	Descripción	Variable 1: Nombre	Variable 2: Estado
EC 1.1 Guardar plantilla correctamente.	Mediante este escenario el usuario puede guardar una plantilla correctamente.	V	V
		Título	Seleccionado.
		V	V
		Admite a partir de 2 caracteres excepto % y comillas simples. La cantidad de caracteres permitidos es 100.	Seleccionado.
EC. 1.2 Guardar plantilla incorrectamente.	Mediante este escenario el usuario no puede guardar una plantilla correctamente.	I	V
		´Pureba%¨	Seleccionado.
		I	V
		El usuario intenta entrar más de 100 caracteres.	Seleccionado.
		I	V

		El usuario entra al menos de 2 caracteres.	Seleccionado.
EC 1.3 Guardar plantilla repetida.	Mediante este escenario el usuario puede guardar una plantilla con un nombre igual a otra ya existente en el sistema.	I	V
		El usuario introduce un nombre de plantilla igual al de otra ya existente en el sistema.	Seleccionado.

Tabla 57. Diseño de casos de prueba. RFDP48_Guardar plantilla. Parte dos.

Respuesta del sistema	Flujo central
El sistema guarda una plantilla correctamente.	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores y luego se escribe el nombre de la plantilla y se actualiza su estado. Posteriormente se puede seleccionar la opción Guardar.
El sistema muestra un mensaje en rojo encima del componente: "No se admiten apóstrofes, ni porcientos sobre el campo requerido.".	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores y luego se escribe el nombre de la plantilla y se actualiza su estado. Posteriormente se puede seleccionar la opción Guardar.
El sistema no deja seguir entrando caracteres.	
El sistema muestra un mensaje en rojo encima del componente: "Entre al menos 2 caracteres.".	
El sistema muestra un mensaje de error: "El elemento ya existe.".	El usuario una vez autenticado en el sistema selecciona la agrupación funcional Diseño en el menú interior, luego la funcionalidad Plantilla. El sistema muestra el listado de plantillas y el área de acciones superiores. Seguidamente se selecciona la opción Crear plantilla en el área de acciones superiores y luego se escribe el nombre de la plantilla y se actualiza su estado. Posteriormente se puede seleccionar la opción Guardar.

Anexo 5: Pruebas de seguridad. Técnica de caja negra.

Tabla 58. Lista de Chequeo. Pruebas de Comprobación del Sistema de Autenticación.

Comprobación del Sistema de Autenticación					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	El sistema no debe mostrar diferentes mensajes de error al usuario “intentar autenticarse con un usuario correcto y una contraseña incorrecta” y “al autenticarse con un usuario incorrecto y la misma contraseña del intento anterior”.	0		No se encontraron errores del indicador evaluado.	
Crítico	El tiempo de respuesta entre los 2 casos del indicador anterior debe ser el mismo, pues en ocasiones puede apreciarse una media ligeramente diferente en el tiempo de respuesta, que igualmente puede usarse esta información para descubrir cuentas válidas.	0		No se encontraron errores del indicador evaluado.	
Crítico	El campo usuario de la autenticación al sistema no debe tener el auto completamiento activado (guarda los usuarios que se autentican). Para ver esto se debe autenticar al sistema, luego salir y poner la primera letra del usuario autenticado	0		No se encontraron errores del indicador evaluado.	

	para ver si muestra el nombre de usuario completo.				
Crítico	El sistema debe proteger el envío de los datos mediante protocolo seguro (verificar si usan https).	0		No se encontraron errores del indicador evaluado.	
Crítico	El sistema debe usar algún certificado.	0		No se encontraron errores del indicador evaluado.	

Tabla 59. Lista de Chequeo. Pruebas de Validación de Datos.

Validación de Datos					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	El sistema debe enmascarar datos confiables cuando se visualicen en la aplicación (Por ejemplo: Contraseñas).	0		No se encontraron errores del indicador evaluado.	
Crítico	La aplicación solamente debe permitir contraseñas alfanuméricas, que incluyan caracteres especiales y que tengan seis caracteres mínimos de longitud.	0		No se encontraron errores del indicador evaluado.	
Crítico	La aplicación debe permitir la funcionalidad de cambio de contraseña únicamente a usuarios autenticados validando la antigua contraseña, la nueva contraseña y la	0		No se encontraron errores del indicador evaluado.	

	respuesta a la pregunta de seguridad (esta última es opcional en dependencia del proyecto).				
Crítico	El sistema no debe mostrar ningún mensaje indebido, al colocar en la barra de dirección o en campos de entrada los caracteres: comillas simples (´´), signos de ampersand (&), signos: + - /.	0		No se encontraron errores del indicador evaluado.	

Anexo 6: Acta de aceptación.

UCi Universidad de las Ciencias Informáticas

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento del **Convenio de colaboración** en la Dirección de Informatización y en función de la ejecución del Proyecto: Sistema de Gestión Universitaria se hace entrega del producto que se relaciona a continuación.

Lista de productos que serán aceptados:

- *Sistema Diseñador de Plantillas*
- *Estudio_preliminar*
- *Arquitectura_vista_de_sistema_DAC*
- *Arquitectura_vista_de_presentación_DAC*
- *Evaluación_de_requisitos*
- *Reglas_de_negocio*
- *Modelo_conceptual*
- *Especificación_de_requisitos_de_software*
- *Modelo_de_diseño_DAC*
- *Descripción_de_requisito_DAC*
- *Estándar_de_codificación_GUUD*
- *Diseño_de_casos_de_prueba*
- *Manual de Usuario*

Entrega	**Recibe**
Nombre y Apellidos: Alejandro Vázquez Chávez	**Nombre y Apellidos:** Alelí Sánchez Méndez
Cargo: Estudiante	**Cargo:** Jefa de I+D+i
Firma:	**Firma:**
Comentarios:	

Fecha: 19/05/2014

1