Universidad de las Ciencias Informáticas
Facultad 3
Centro de Gobierno Electrónico (CEGEL)

Componente para generar reportes para el Sistema de digitalización de documentos con valor legal.

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

Autor:
Ernesto Carlos Pérez García

Tutor(es):
Ing. Doris Maza Oval
Ing. Yunior Duque Aguilar
Ing. Juan Carlos Moreira Lara

La Habana, junio de 2017
“Año 57 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro ser el 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 _____.

_____

Ernesto Carlos Pérez García

Autor

_____ _____

Ing. Doris Maza Oval **Ing. Yunior Duque Aguilar**

Tutor Tutor

AGRADECIMIENTOS

Es difícil ser hoy quien está agradeciéndole a la vida y a dios por haber logrado uno de mis sueños. Es una mescla entre alegría y llanto, es alivio y paz. Mi cabeza se llena de tantas preguntas y hasta ni quisiera contestarlas, solo la vida dirá. Sé que ahora todo cambiará de un momento a otro, nuevas responsabilidades y tareas a le que debo enfrentar como ingeniero, como hijo o quizás algún día como padre. Parece mentira que fuese ayer cuando era un niño y le preguntaba mi madre, - ¿Cómo es la uci? ¿Qué se hace en la UCI? – Mi madre como siempre me dio una respuesta, pero siempre le faltó estas cuatro palabras, excelencia, integralidad, madures, y amistad. Para llegar a conocerlas tuve que pasar por varias cosas, por lo que hoy les agradezco.

Primeramente, agradecer a mi madre, la que aún veo como una diosa, una guía, un puesto inalcanzable, es difícil lograr por mí mismo hacer el trabajo más difícil de todos los tiempos, y es la de criarme, la de enseñarme, la de mostrarme un mundo de muchas maneras y guiarme en cada paso que yo diera, tantos malos o bueno, pues yo sé que aún sigo siendo su primer hijito.

También agradezco a quien una vez dijo furioso y preocupado luego de verme en mis decadencias, - ¡¿Algún día habrá un ingeniero aquí en esta familia?! Hoy le agradezco, por enseñarme, por cuidarme, por apoyarme y ser capaz de comprenderme. Padre mío, hoy te respondo tu pregunta, hoy te quito esa preocupación. ¡Ya soy ingeniero!

Le agradezco a mis hermanos Carlos, Harassay, Makel, a mi prima Suilen y a mi tía (segunda madre) que siempre le pidió a Santo Tomás de Aquino para que yo saliera bien en cada prueba, a mis padrinos. A mis amigos del cerro, tierra donde nací, a mi profe Mercedes que tuvo mucho que ver en mi formación durante mi infancia. Agradecer a los socios del barrio allá en la Habana del Este, como a Eider, Leste, Jorgito, Javier, Alexander, Raidel el temba, en fin todo ese equipo de jóvenes ya trabajadores, les digo que ya soy uno más. Quiero también gradecer a la Escuela militar Camilo Cienfuegos de Guanabacoa, por ser la institución que me formó como lo que soy hoy y a un amigo que conocí allí, Ariel Navarro, gracias por ayudarme a seguir hacia adelante. Agradecer a mi piquete freaky Pazzo, allá en guanabo que juntos pasamos buenos momentos.

Durante mi estancia aquí en la UCI he logrado conocer a muchas personas, amigos, novia, en fin, a tal manera de que hoy me dicen la sombra, nesti o simplemente TUTTO. Le quiero agradecer a Rosmery mi hermanita (la tutta), Juan Carlos, el flaco, mi amigo, mi cotutor de mi tesis, Fauri, Erlys. A la gente del cuarto, Yairon, Gabi, Yoe, Yoandy (el teacher, el rubio o el dios) y Samuel. A Asyledis, Claudia, Yanet, Claudia Fernandez, a mi actriz favorita Leidis, Jessica, Selma, Dalili, Adolfo, Celso, Marcelo, David, Rafael, Luis Javier, Osbel, Ernesto Miro, Miguel Antonio, Norberto, Felix y Ruben y no menos

importante alguien que compartió conmigo casi completa mi vida universitaria, Kenia. Eternamente agradecido estoy con los profes, Rosalina, Zenel, Susana, Monica, Raynel, a mi oponente, al tribunal, mi tutora Doris, y a mi tutor Junior, a Elizabeth y Isis, chicas que pusieron a prueba mi aplicación. Agradezco a Silverio, Juan David y al equipo de producción de CEGEL, que siempre me aconsejaban darle clean and build cuando la cosa se ponía dura y a Carel Carmenate quien me enseñó las potencialidades del Jasper Report. A todos en general por ser parte de mí día a día en esta universidad. Gracias a la revolución, gracias CUBA.

DEDICATORIA

Este trabajo es dedicado a mi mamá, a mi papá, a mi hermano Carlos, mi tía Suni y a mi prima Suilen. A mi abuela Zenaida, y mis abuelos por parte de madre Alberto y Susana.

RESUMEN

El Centro de Gobierno Electrónico (CEGEL), perteneciente a la Universidad de las Ciencias Informáticas (UCI), ha desarrollado el proyecto “Sistema para la obtención de documentos digitales con valor legal”. Este sistema tiene como objetivo desarrollar una solución tecnológica que permita la generación de objetos digitales con valor legal y lograr automatizar los procesos de un Centro de Digitalización. En el proceso de digitalización de un fondo documental, se hacen reportes que brindan una gran información y ayuda en la toma de decisiones. Sin embargo, estos reportes se generan manualmente, trayendo consigo diversos problemas. Para mitigar estos problemas se toma como decisión desarrollar e incorporar al sistema un componente para generar reportes. En el presente trabajo se expone un estudio de varios sistemas generadores de reportes, se muestran las herramientas y tecnologías utilizadas, y se brinda una descripción detallada de la solución de software. Son expresadas las actividades realizadas por cada una de las fases del modelo de desarrollo seleccionado para obtener una solución consistente. Por último, se realiza la validación del módulo mediante métricas y pruebas de software.

Palabras claves: fondo documental, digitalización, objeto digital, reportes, generador de reporte, sistema.

ABSTRACT

The Centre of Electronic Government (CEGEL), belonging to the University of Informatics Sciences (UCI), has developed the project "System for obtaining digital documents with legal value". This system aims to develop a technological solution that allows the generation of digital objects with legal value and automate the processes of a Digitization Center. In the process of digitizing documents, reports are made to provide great information and aiding in decision making. However, these reports are generated manually, bringing some problems. To mitigate these ones, it is decided to develop and incorporate into the system a component to generate reports. In this paper it is presented a study of several reporting systems, it is shown the tools and technologies used, and a detailed description of the software solution. Also are stated the activities performed in each of the phases of the selected development methodology to obtain a consistent solution. Finally, it is performed the module's validation using metrics and software tests.

Keywords: *documentary fund*, *Digitalization*, *Digital object*, *Reports*, *Report generator*, *system*.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Desde el surgimiento de las antiguas civilizaciones, Egipto, Grecia y Roma, sus gobernantes crearon mecanismos para defender los intereses del estado como es el pago de los impuestos y control de la densidad poblacional. La recolección de estos datos se registraba en papel o en piedra y se preservaba año tras año, de esta manera surgieron los primeros archivos. La conservación de los archivos le ha permitido al hombre investigar el surgimiento de la humanidad y el estudio de los procesos históricos.

Los archivos forman parte del patrimonio cultural de los pueblos. Con el paso del tiempo los archivos se enfrentan a su desaparición paulatina, ya que la mayoría de ellos se encuentran en papel y su deterioro es inevitable. En la actualidad, los gobiernos de diferentes países del mundo buscan alternativas para la preservación de los archivos o fondo documental, como también se le conoce, que es el conjunto de documentos producidos o recibidos por una persona física o jurídica en el ejercicio de sus actividades. En Cuba se han implementado diversos mecanismos para el cuidado de los fondos documentales, uno de ellos es el uso de las Tecnologías de la Información y las Comunicaciones.

El Centro de Gobierno Electrónico (CEGEL), perteneciente a la Universidad de las Ciencias Informáticas (UCI) ha desarrollado, en el marco del Convenio Intergubernamental Cuba-Venezuela, sistemas para la obtención de objetos digitales con valor legal para determinados entes gubernamentales y cuyos resultados han sido satisfactorios. La construcción de estos sistemas forjó bases sólidas en cuanto al tema de digitalización en dicho centro. Es por eso que el centro se ve la necesidad de desarrollar un nuevo sistema que fuera adaptable a cualquier entorno, algo que no era posible con ninguno de los desarrollados anteriormente ya que eran sistemas a la medida. Surge entonces, el “Sistema para la obtención de documentos digitales con valor legal” (DIGILEX).

El sistema DIGILEX tiene como objetivo, desarrollar una solución tecnológica que permita la generación de objetos digitales con valor legal. Con el desarrollo de este sistema se pretende automatizar los procesos de un Centro de Digitalización entre los que se encuentran: Preparación, Digitalización, Metadatos, Calidad y Firma electrónica. Entre las ventajas que se pretende obtener con este sistema está la disminución del tiempo de digitalización y comercialización de servicio, la modernización e informatización de la gestión de objetos digitales con valor legal, así como la preservación de fondos documentales.

En el proceso de digitalización de un fondo documental se hacen reportes que permiten saber en tiempo real la cantidad de folios que se encuentran en cada área, el balance de carga de cada una, folios

pendientes en cada área y la cantidad de folios digitalizados, rechazados entre otros datos. Estos reportes le permiten saber a quién dirige el proceso, en qué estado se encuentra la digitalización de manera general o simplemente porque en muchas ocasiones los clientes solicitan saber en qué estado se encuentra la digitalización de sus fondos documentales. Actualmente, para obtener cualquier tipo de información sobre los procesos y realizar los reportes, los especialistas deben obtener los datos de forma manual realizando conteo, clasificación de los folios y los lotes.

Teniendo en cuenta el volumen de folios que posea un fondo documental, se pueden identificar las siguientes deficiencias de este proceso:

- La lentitud en la búsqueda y obtención de datos para la realización de los reportes,
- Datos erróneos
- Datos duplicados
- Perdida de la información por el tratamiento desordenado de la misma
- La incorrecta manipulación de los folios
- Pérdida de su ubicación en los lotes.

A partir de estos argumentos se hace necesario determinar un mecanismo que dé solución a las deficiencias planteadas.

La situación problemática antes descrita ha generado el siguiente **problema de investigación**: ¿Cómo facilitar la obtención de datos en el proceso de digitalización de los fondos documentales para la generación de reportes a través del Sistema para la obtención de documentos digitales con valor legal?

El problema de investigación se enmarca en el **objeto de estudio** de la generación de reportes, centrándose en el **campo de acción** de las herramientas para la generación de reporte en plataformas Java.

Como **objetivo general** de esta investigación se tiene desarrollar un componente con el fin de emitir reportes sobre el proceso de digitalización para el Sistema para la obtención de documentos digitales con valor legal, que permita facilitar la obtención de datos en el proceso de digitalización de los fondos documentales.

Con el propósito de asegurar el cumplimiento del objetivo general, se definen los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación para fundamentar las bases de la solución.

- Desarrollar la propuesta de solución que permitirá la disminución de la obtención de datos en el proceso de digitalización de los fondos documentales, para la generación de reportes a través del Sistema para la obtención de documentos digitales con valor legal.
- Validar la solución propuesta aplicando las pruebas internas para la implementación y las pruebas de aceptación para los requisitos del cliente y las métricas relación de clases y tamaño operacional para la validación de software en busca de una mejor calidad.

Se plantea como **idea a defender** que, si se desarrolla un componente para la generación de reportes, entonces se facilitará la obtención de la información referente a cada una de las áreas del proceso de digitalización de los fondos documentales de manera que se disminuya el tiempo para la generación de los reportes.

Para lograr los objetivos específicos propuestos se plantean las siguientes **tareas de investigación:**

- Realización de un estudio del estado del arte de los diferentes conceptos relacionados con el tema, así como de los sistemas existentes para la realización de reportes de procesos.
- Estudio de la metodología, lenguajes y herramientas para darle solución al problema planteado.
- Análisis y modelado de los procesos del negocio.
- Levantamiento de las funcionalidades de los procesos de negocio.
- Estudio de la Arquitectura y los patrones a utilizar
- Diseño de la estructura y comportamiento de los componentes de los procesos identificados.
- Diseño del modelo de datos.
- Diseño del diagrama de despliegue del sistema.
- Validación del diseño propuesto.
- Implementación de los componentes diseñados.
- Validación de la implementación realizada.
- Validación de las variables de la investigación.

Los métodos científicos utilizados en esta investigación para garantizar la realización de las tareas propuestas son:

Métodos teóricos:

- El método Analítico - Sintético fue puesto en práctica en el estudio y análisis de la bibliografía. Permitió hacer una correcta selección de los conceptos y definiciones relacionados con fondos documentales, digitalización de documentos; generación de reportes entre otros términos para comprender mejor el problema y darles cumplimiento a los objetivos trazados.

- El método de Modelación se utilizó en la creación de modelos que representan abstracciones, con el objetivo de modelar la estructura y funcionamiento de la implementación del componente que permitirá generar reportes.
- El método Inductivo - Deductivo permitió arribar a conclusiones particulares y decidir cada elemento a utilizar.
- El método Histórico - Lógico permitió realizar un estudio sobre las herramientas que se han creado con el fin de digitalizar fondos documentales que realizan reportes de sus procesos.

Métodos empíricos:

- El método de Simulación permitió ejecutar el producto desarrollado con datos artificiales y así poder medir la calidad de la implementación de las funcionalidades del componente.

- La Entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información (Hernández Sampier, 2008). Esta permitió obtener información sobre procesos de digitalización que se han llevado a cabo en la universidad, como los reportes que se realizaban en esos momentos y que podían formar parte de la solución.

El presente documento de tesis está estructurado en tres capítulos, quedado estructurados de la siguiente forma:

Capítulo 1 FUNDAMENTACIÓN TEÓRICA: En este capítulo se realiza un análisis de la literatura consultada, al tiempo que se presenta una taxonomía de los conceptos fundamentales asociados a la investigación, los cuales permitirán una mejor comprensión del proceso de digitalización de un fondo documental. Se muestra un análisis del estado del arte, donde son tratados los sistemas generadores de reportes. También se hace un estudio del proceso de desarrollo de software incluyendo metodologías, herramientas de desarrollo, patrones arquitectónicos y de diseño y, por último, métodos y técnicas de prueba de software. Constituye el basamento teórico-conceptual de la investigación.

Capítulo 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA: En este capítulo se realiza una descripción de la propuesta de solución y sus principales funcionalidades. Se argumenta el uso de los patrones de diseño, se detalla la modelación de los diagramas de clases del diseño y de secuencia.

Capítulo 3 ANÁLISIS DE RESULTADOS: En este capítulo se aplican las métricas predeterminadas para medir el diseño, se elaboran y ejecutan los casos de prueba para verificar el correcto funcionamiento de los componentes implementados y se hace un análisis de los resultados obtenidos.

INTRODUCCIÓN

Se pretende una vez concluida la presente investigación obtener un componente capaz de gestionar diferentes reportes sobre la información de las distintas áreas de un centro de digitalización de documentos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El presente capítulo tiene como objetivo abordar los elementos teóricos e investigativos para el desarrollo de la solución al problema planteado. Como resultado de la elaboración de este capítulo se encontrará un estudio realizado sobre los diferentes generadores de reportes existentes, así como un análisis de las metodologías, herramientas de desarrollo, patrones arquitectónicos y de diseño, y, por último, los métodos y técnicas de prueba de software que se emplearán para el desarrollo de la solución.

1.2. Marco Teórico

En este acápite se describen los principales conceptos asociados al dominio del problema con el objetivo de avalar la investigación, permitiendo de esta forma crear la base de conocimientos necesaria para su desarrollo.

1.2.1. Fondo documental

Los fondos constituyen la mayor agrupación documental existente en un archivo, y corresponden al "conjunto de documentos, de cualquier formato o soporte, producidos orgánicamente y/o reunidos y utilizados por una persona particular, familia u organismo en el ejercicio de sus actividades".(Archivo Nacional, 2017) Se conoce también como fondo documental, colecciones de materiales inéditos, de carácter histórico y científico, a menudos relacionados directamente con los fondos museográficos, o cuando menos con el área científica.

1.2.2. Digitalización de documentos

En el proceso de digitalización de documentos son los pasos mediante los que los registros físicos, como texto e imágenes se convierten en formatos digitales. Se conoce también como el proceso de la digitalización de un documento, es la representación de un documento por un conjunto de sus puntos o muestras y su resultado se denomina imagen digital del documento.(TBS-Telecon, 2017)

1.2.3. Objeto digital

En pedagogía un objeto digital es aquel recurso en formato digital que se pueden utilizar, reutilizar o referenciar durante un proceso de aprendizaje.(UPV, 2017). Se entiende por objeto digital el resultado de la conversión de cualquier documento en formato digital, en estado digital.

1.2.4. Generación de reportes

Se encarga de producir la información requerida y transmitirla a los puntos o centros de información que la soliciten. Esta transmisión de información se puede efectuar mediante el movimiento físico de los elementos de almacenamiento o mediante la comunicación de señales eléctricas digitales o analógicas a dispositivos receptores.

1.3. Estado del Arte

El proceso de generación de reportes, implica tomar un conjunto de decisiones técnicas y metodológicas complejas. ¿Cuáles son los distintos generadores de reportes? ¿Cuáles son las principales limitaciones que poseen estos generadores de reportes? ¿Cuáles son los pasos principales que se deben tener en cuenta a la hora de acometer el desarrollo de un proyecto con estas características? En este epígrafe se realiza un análisis crítico de las principales aproximaciones existentes a la generación de reportes y se analizará los distintos sistemas para la obtención de objetos digitales con valor legal creados en el marco intergubernamental Cuba-Venezuela por el Centro de Gobiernos Electrónico.

1.3.1. Sistemas para la obtención de objetos digitales con valor legal.

El Centro de Gobierno Electrónico (CEGEL), perteneciente a la Universidad de las Ciencias Informáticas (UCI) ha desarrollado, en el marco del Convenio Intergubernamental Cuba-Venezuela, sistemas para la obtención de objetos digitales con valor legal como: DigiPRO, DigiPyrus, DigiDAP, CDA entre otros, obteniendo resultados satisfactorios. Este sistema tiene como principal objetivo gestionar la información para las entidades a las que fueron creados. A continuación, se hace un estudio de los sistemas anteriormente mencionados.

1.3.1.1. DigiPyrus

El sistema de Digitalización DigiPyrus, desarrollado por el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI), permite digitalizar los Tomos de los Registros y Notarías de la República Bolivariana de Venezuela, contribuyendo a agilizar el acceso a la información que a diario se tramita en cada una de sus oficinas. Este sistema surge debido a la demanda de una gestión más eficiente y segura de los trámites que se realizan con los documentos archivados en las Oficinas de los Registros y Notarías Públicas. De ahí, que se haya propuesto presentar una solución que cumpla con el objetivo de dotar a esta entidad de un sistema que contenga las funciones, para realizar el proceso de digitalización de todos los documentos archivados y de esta forma obtener el fondo digital de cada una de estas(Pupo Acosta 2011). .

Este sistema permite la generación de reportes de los fondos documentales, pero eran creados con el fin de crear un reporte con el histórico de los Tomos y objetos digitales importados por cada uno de los estados de la República Bolivariana de Venezuela, en un período de tiempo determinado. Muestra un resumen de los Tomos y metadatos importados, los cuales se agrupan por el Estado de la nación, Tipo de oficina, Oficina a la que pertenecen los documentos, y por cada Tomo se muestra el Número de Tomo, Año, Trimestre, Cantidad de unidades documentales y las unidades documentales no incluidas(Pupo Acosta 2011).

1.3.1.2. DigiDaP

DigiDAP forma parte de la Solución Tecnológica Integral para la automatización y modernización de la División de Antecedentes Penales de la República Bolivariana de Venezuela, como el subsistema que informatiza los procesos del Centro de Digitalización para el fondo documental de dicha institución. Su objetivo fundamental es garantizar la obtención de objetos digitales con valor legal a partir de la digitalización del fondo para mejorar los servicios de inscripción y certificación de antecedentes penales a los ciudadanos de la nación venezolana (Gonzalez valdez 2011) .

1.3.1.3. CDA

CDA es la solución tecnológica desarrollada para el Centro de Digitalización de Alfabéticas para el Servicio Administrativo de Identificación, Migración y Extranjería. Para su desarrollo se utilizó como marco de trabajo el utilizado por la solución DigiPyrus por lo que las características arquitectónicas y tecnologías de desarrollo utilizadas son similares. Su objetivo fundamental es extraer tarjetas alfabéticas para convertirlas en metadatos. Esta solución está compuesta por varios módulos, entre ellos el módulo de reporte. En el módulo de reporte se lleva a cabo la gestión de información del centro mediante diferentes reportes que permiten conocer el estado de desempeño de los trabajadores y el estado de procesamiento de lotes, cajas y documentos dentro del Centro de Digitalización(Rodriguez Lopez, Martinez Garcia, y Hernandez Ciseros 2011).

1.3.1.4. DigiPRO

La Plataforma para la digitalización de documentos DigiPRO es una solución tecnológica con el objetivo de agilizar la construcción de sistemas para la generación de objetos digitales con valor legal. La automatización de los procesos de un Centro de Digitalización propicia la disminución del tiempo de desarrollo y comercialización de este tipo de soluciones, la modernización e informatización de la

gestión de objetos digitales con valor legal; así como, la preservación y protección de fondos documentales(Blanco Cala 2014).

A partir de la investigación de los distintos sistemas para la obtención de objetos digitales con valor legal, creados durante este confraternal y desinteresado convenio. Se puede concluir mediante la siguiente tabla los sistemas que son capaces de generar reportes.

Sistema	**Genera Reportes**
DigiPyrus	Sí
DigiDAP	No
CDA	Sí
DigiPRO	No

Tabla 1: Comparativa de los sistemas desarrollos por el CEGEL que generan reporte.

Teniendo en cuenta la comparativa anterior, permite inferir en esta investigación que solo dos generan reportes en interés de su negocio DigiPyrus y CDA. De estos dos, solo el CDA genera reportes con el objetivo de llevar un buen control del desempeño de sus trabajadores y el estado del procesamiento de lotes, cajas y documentos dentro del Centro de Digitalización. Es por eso que se es necesarios hacer un estudio de las distintas herramientas de generaciones de reportes.

1.3.2. Generadores de reportes.

Los generadores de reportes son herramientas complementarias de los sistemas de información. Utilizan un lenguaje transparente para el usuario por medio del cual realiza consultas y se obtiene la información en forma de reporte.

Para comprender el ámbito de la solución propuesta se analizaron varios generadores de reportes con el objetivo de evaluar el comportamiento de las características en cuanto a: arquitectura, tipo de software, multiplataforma, generación de reportes dinámicos, personalización al detalle, escalabilidad, formato de las salidas y generación de sub reportes. Se analizaron los sistemas generadores de reportes siguientes: Dynamic Report Generator, Jasper Report, PHP Reports Maker, Crystal Reports, Active Reports, Microsoft SQL. Estos sistemas son los más reconocidos en cuanto a la generación de

reportes y su utilización se extiende en diferentes tipos de aplicaciones (incluyendo aplicaciones del tipo registral). Pero existen otros no tan reconocidos mundialmente como por ejemplo el GEReport v1.0.

1.3.2.1. Jasper Report.

Jasper Reports es una librería de código abierto poderosa y flexible para la generación y gestión de reportes permitiendo la creación de informes en Java. Permite la creación de reportes que pueden ser presentados en la web o se pueden crear ficheros en diversos formatos, para que los datos puedan ser procesados con otras aplicaciones. Admite la creación de reportes, permite añadir enlaces en los reportes, lo que haría más fácil y rápida la navegación entre las diferentes secciones del informe cuando este es muy largo.(R. Heffelfinger 2011)

Las principales características de Jasper Reports son:

Permitir una diagramación flexible de los reportes: Los reportes se pueden dividir en secciones opcionales que son: título del reporte, el encabezado de página, una sección para los detalles del reporte, el pie de página y una sección de resumen que aparece al final del reporte.

Permite que los desarrolladores le surtan datos en varias formas, es decir los desarrolladores permitan el paso de los datos a los reportes por medios distintos parámetros. Estos parámetros de reportes pueden ser instancia de cualquier clase de Java. Los reportes son capaces de presentar los datos de manera textual o a través de gráficos: no sólo son capaces de mostrar los datos que le son pasados, sino que pueden generar o calcular con esos datos otros datos de forma dinámica y mostrarlos.

No es una herramienta por sí sola, por lo que no se puede instalar. Para utilizar Jasper Reports es necesario añadirlo a las aplicaciones Java por medio de la inclusión de su librería al classpath[1] (por su traducción al español, ruta de la clase) de la aplicación. En cuanto a las especificaciones de sistemas operativos se puede utilizar en cualquier entorno, siempre y cuando exista una implementación de la máquina virtual de Java. Su licencia se distribuye bajo los términos de la Licencia Pública para Librerías GNU[2], por lo que es software libre y está respaldado por una gran comunidad internacional de desarrollo. (R. Heffelfinger 2011)

1.3.2.2. Microsoft SQL Server 2005 Reporting Services (SSRS)

[1] *Classpath (Java) En el lenguaje de programación Java se entiende por Classpath una opción admitida en la línea de órdenes o mediante variable de entorno que indica a la Máquina Virtual de Java dónde buscar paquetes y clases definidas por el usuario a la hora de ejecutar programas.*

[2] *GNU (por su traducción al inglés GNU Library Public License)*

Reporting Services es un componente clave de SQL[3] Server 2005. Es una plataforma de elaboración de informes basada en servidor que se puede utilizar para crear y administrar informes tabulares, matriciales, de gráficos y de formato libre con datos extraídos de orígenes de datos relacionales y multidimensionales. Se pueden visualizar y administrar mediante una conexión basada en la web. (Robinson 2012)

Contiene un servidor de informes que aloja y procesa informes en diversos formatos. Los informes incluyen características interactivas basadas en la web API[4] que permite a los programadores integrar o extender procesamiento de datos e informes en aplicaciones personalizadas. En SSRS, los usuarios pueden enrutar directamente los trabajos de impresión, sin necesidad de exportarlos antes. Sistema Operativo: Microsoft Windows, preferiblemente en sus versiones para servidores. Licencia: se distribuye bajo licencia privativa perteneciente a Microsoft Corporation. (Robinson 2012)

Reporting Services permite que los informes incluyan características interactivas basadas en la web: informes de varios niveles de detalle, que permiten la navegación por distintas capas de datos; los informes con parámetros, que admiten el filtro de contenido en tiempo de ejecución; o los informes de formato libre, con diseños verticales, anidados o adyacentes.

1.3.2.3. Crystal Reports

Crystal Reports es un producto de alta tecnología, además de ser una herramienta potente, es fácil de usar para el diseño y generación de reportes a partir de datos almacenados en una base de datos u otra fuente de información. (Almansa Marín 2016)

Permite transformar rápidamente cualquier fuente de datos en contenido interactivo, integrar estrechamente capacidades de diseño, modificación y visualización en aplicaciones .NET[5], Java o COM[6], y además permite a los usuarios finales acceder e interactuar con los reportes a través de portales web, dispositivos móviles y documentos de Microsoft Office.(Almansa Marín 2016)

[3] *SQL (por sus siglas en inglés Structured Query Language; en español lenguaje de consulta estructurada) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.*

[4] *API (La interfaz de programación de aplicaciones, abreviada como API del inglés: Application Programming Interface, es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.)*

[5] *.NET (derivado del inglés "Network")*

[6] *COM (Component Object Model o por su traducción al español Modelo de Objeto Componente)*

Se puede utilizar en sistemas operativos como, Microsoft Windows XP con Service Pack (SP) 2, Windows Server 2003 con SP 1 o posterior y se distribuye bajo los términos de la EULA[7], por lo que es software privativo y de uso restringido mediante el pago de patente. Los visores web avanzados habilitan a los usuarios finales para realizar búsquedas dentro de los datos de un reporte y exportarlas posteriormente a Microsoft Excel, Word y páginas HTML[8] con el vínculo dinámico al reporte original. Adicionalmente, el reporte completo puede ser exportado a una variedad de formatos incluyendo XML[9], PDF[10], HTML y Microsoft Excel. (Almansa Marín 2016)

1.3.2.4. Active Reports

Active Reports es una herramienta de generación de reportes que brinda posibilidades de personalización, alto rendimiento e interfaz intuitiva. Se distribuye bajo licencia privativa perteneciente a Grape City. Es un componente de informes .NET, entre las características claves de este componente figuran la personalización, rendimiento alto, alta calidad y numerosas prestaciones. Admite exportaciones de datos a todos los formatos de archivo habituales dentro de las características principales se tiene que los informes se crean dentro de Visual Studio .NET y se compilan directamente en el ejecutable y dado que Active Reports es totalmente administrado, no presenta dependencias de aplicaciones de terceros. (GrapeCity 1997).

1.3.2.5. PHP Report Maker

Es una poderosa herramienta de informes que se pueden generar informes Web PHP[11] dinámicos de MySQL, PostgresSQL, Microsoft Acces, Microsoft SQL Server y base de datos Oracle. Las páginas Web generadas son PHP puro, está diseñado para la alta flexibilidad y permite generar reportes que se adapten a las necesidades. Los códigos generados son limpios y fáciles de personalizar. PHP Report Maker puede ahorrar tiempo y su uso es adecuado tanto para principiantes como para desarrolladores experimentados.

Con PHP Report Maker, los usuarios pueden exportar los informes a formatos como PDF, Word o Excel. Dentro de sus características cuenta con: informes de resumen y detalle, propiedades

[7] *EULA (End User Licensing Agreement o por su traducción al español Licencia de Usuario Final)*

[8] *HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web.*

[9] *XML (Extensible Markup Language o por su traducción al español lenguaje de marcas extensible)*

[10] *PDF (Portable Document Format o por su traducción al español Formato de documento Portátil)*

[11] *PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.*

avanzadas de gráfico, plantilla personalizable y sincroniza los ajustes de proyectos con los cambios en la base de datos.

1.3.2.6. Generador Dinámico de Reportes (GDR) v2.0

Constituye una de las herramientas que posibilitan la inteligencia de negocios ya que genera vistas agregadas de datos para mantener a la gerencia informada sobre el estado de su negocio. Proporciona soporte a todo el ciclo de vida de los reportes, provee accesibilidad a la información emitiendo reportes a través de un navegador web u otros formatos estándares y está orientada al usuario final, brindando herramientas como el diseñador de reportes que permite la confección de informes de forma interactiva en la web. (Abreu Medina et al. 2012).

Este diseñador permite personalizar al detalle la salida de los reportes y la forma en que la información será visualizada. Presenta una arquitectura modular y las posibilidades de escalabilidad por medio de extensiones y complementos que se integrarán a los procesadores principales que componen el servidor de reportes, (Abreu Medina et al. 2012).

1.3.2.7. GEReport v1.0

Es una herramienta capaz de construir reportes dinámicos y su utilización no se restringe debido a las licencias de software, se ejecuta sobre un entorno web, se minimiza las dependencias tecnológicas y es multiplataforma. Posibilita exportar como imagen y en los formatos HTML, PDF y Excel la información del reporte. (Gavio et al. 2014)

A partir del estudio de los generadores de reportes se realizó un análisis comparativo a través de los principales parámetros que deben cumplir estos reportadores, con el fin de buscar cual es el que más se acopla a esta investigación para llegar a la solución del problema.

Las herramientas tienen gran cantidad de funcionalidades, lo que permitió hacer un análisis teniendo en cuenta los siguientes aspectos: arquitectura, tipo de software, multiplataforma, generación de reportes dinámicos, personalización al detalle, escalabilidad y formato de las salidas. Donde en las columnas aparece las siglas de los generadores de reportes estudiados; Dynamic Report Generator (GRG), Jasper Report (JR), PHP Reports Marker (PHPRM), Crystal Reports (CR), Active Report (AR), Microsoft SQL Reprting Services (SSRS), Generador Dinámico de Reportes (GDR) y GEReport v1.0 (GR)

Parámetros	**GDR**	**JR**	**PHPRM**	**CR**	**AR**	**SSRS**	**GDR**	**GR**
Tipo de software.	Libre	libre	libre	privativo	privativo	privativo	Libre	Libre
Multiplataforma.	No	Sí	Sí	No	No	No	Sí	Sí
Reportes dinámicos.	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Personaliza al detalle.	Sí	Sí	Sí	No	Sí	Sí	Sí	Sí
Escalabilidad.	Sí	Sí	No	Sí	No	Sí	Sí	Sí
Salidas.	PDF	DOC,PDF	PDF	PDF	PDF	PDF	PDF	PDF
Arquitectura.	Modular	Librería	Extensión	Librería	Librería	Servidor	Modular	Web

Tabla 2: Comparativa de los distintos generadores de reportes estudiados. (Fuente: elaboración propia)

El análisis guiado por la anterior tabla, permitió conocer la existencia de generadores de reportes en el mundo y en Cuba desarrollados en distintos lenguajes, formas, y con el mismo propósito, además que permitió valorar la herramienta escogida en interés de esta investigación. Primeramente, se descarta los generadores de reporte privativos, ya que estos no se pueden utilizar si no se paga una licencia de software para su uso, lo que hace que la investigación se centre sobre los generadores de reportes de tipo de software libre. Al reducirse a cinco la cantidad de software, el análisis fue cada vez más detallado. Todos generan salidas de múltiples formatos, generaban reportes dinámicos y son

escalables, sin embargo, Dinamic Report Generator no es multiplataforma, mientras que los demás cumplían con estos parámetros. En cuanto a al generador de reportes dinámico en su versión 2.0, es eliminado de la competencia debido a que es una aplicación desarrollada en un entorno web al igual que el GeReport v1.1, lo que no permite una integración completa con el sistema DIGILEX, ya que fue implementada bajo el lenguaje de programación java. Por último, PHP Report Marker, presentaba interesantes características, pero a diferencia del Jasper Report decaía en dos parámetros importantes, su arquitectura y escalabilidad. Por tanto, Jasper Report es la herramienta idónea para su uso en interés de esta investigación. ¿Por qué?, porque Jasper Reports, se distribuye bajo licencia pública es multiplataforma, maneja fácilmente los reportes dinámicos, lo que permite un fácil manejo del diseño y configuración de tablas dinámicamente, permite la personalización de cada uno de los atributos, es escalable y además que es una librería que permite integrarse con el lenguaje de programación java y generar reporte en diferentes formatos (incluyendo extensión .pdf).

1.4. Metodología de Desarrollo de Software.

En el proceso de desarrollo de un software se tiene varios aspectos en cuenta, apoyado en un marco de trabajo, el cual permite describir un conjunto de actividades que transitan al logro de los objetivos propuestos. Este proceso se ve enmarcado en una serie de acciones de la ingeniería de software haciendo uso de metodologías y modelo para alcanzar las metas de desarrollo del software. Es entonces que de aquí se generan dos conceptos fundamentales modelos prescriptivos y metodologías ágiles.

1.2.1. Modelos prescriptivos

Para (R. S. Pressman 2010) los modelos prescriptivos son un conjunto de distintas actividades, acciones, tareas, fundamentos y productos de trabajos que se requieren para desarrollar software de alta calidad. También plantea que estos modelos de proceso no son perfectos, pero proporcionan una guía útil para el trabajo de la ingeniería del software. En otras palabras, los modelos prescriptivos son de manera general un marco de trabajo, una manera de desarrollar un software de alta calidad.

Estos modelos prescriptivos o tradicionales como bien se les conoce, comprenden de varias actividades que se organizan en un flujo de proceso, el cual puede ser lineal, incremental o evolutivos. Son varios las modelos tradicionales, las cuales se mencionan a continuación.

1.2.1.1. Modelo en cascada.

Modelo en cascada o ciclo de vida clásico según(R. S. Pressman 2010), sugiere un enfoque sistemático, secuencial hacia el desarrollo del software, que se inicia en la especificación de requisitos del cliente, continua con la planeación, el modelado, la construcción, y el despliegue. De manera más aterrizada este modelo se ejecuta de manera lineal pasando por distintas fases el desarrollo del software. La gran desventaja de este modelo es que conduce a un "estado de bloqueo", es decir, las actividades y tarea tienen una cierta dependencia, si no concluye la tarea anterior entonces la siguiente no puede ser realizada.

1.2.1.2. Modelo de proceso incremental

El modelo incremental combina elementos del modelo en cascada aplicado en forma iterativa. Este modelo aplica secuencias lineales de manera escalonada conforme avanza el tiempo en el calendario. A demás plantea que cada secuencia lineal produce incrementos del software. El modelo incrementar al igual que la construcción de prototipos y otros enfoques evolutivos, es iterativo por naturaleza, pero se diferencia de los demás en que el modelo incremental se enfoca en la entrega de un producto operacional con cada incremento, es decir hacen varias versiones del producto aun no terminado lo que proporciona al usuario la funcionalidad que necesita y una plataforma para evaluarlo.

1.2.1.3. Modelo de proceso evolutivos

Los modelos evolutivos son iterativos; los caracteriza la forma en que permiten que los ingenieros de software desarrollen versiones cada vez más completas del software. Este modelo tiene dos variantes; Construcción de prototipos y Modelo en espirar.

- Construcción de prototipos es un proceso evolutivo que puede ser utilizado como un modelo de proceso independiente. Ayuda a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos. Este modelo, inicia con la comunicación, luego pasa por las faces plan rápido, modelado diseño rápido, construcción del prototipo y desarrollo de entrega y retroalimentación, volviendo a la comunicación, permitiendo que el cliente/usuario redefine los requisitos del software que se desarrolla y al mismo tiempo el desarrollador entienda mejor lo que se debe hacer.

- Modelo en espirar es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de la construcción de prototipos con los aspectos controlados y sistemáticos del modelo en cascada. Proporciona el material para el desarrollo rápido de las versiones incrementales del software. Este modo al ser aplicado, el software se desarrolla en una serie de entregas

evolutivas. En las primeras iteraciones la entrega puede ser una documentación del modelo o un prototipo, mientras q en las últimas iteraciones se producen versiones cada vez más complejas del sistema desarrollado.

1.2.2. Desarrollo Ágil.

La ingeniería de software ágil o desarrollo ágil combina una filosofía y un conjunto de directrices de desarrollo. Esta filosofía tiene como objetivo la buena aceptación del cliente y la entrega temprana de software incremental; equipos de proyectos pequeños y con alta motivación; métodos informales; un mínimo de productos de trabajo de la ingeniería de software y una simplicidad general del desarrollo. Las directrices de desarrollo tratan conceptos como el análisis, diseño y sobre todo la comunicación entre el desarrollador y el cliente. (R. S. Pressman 2010)

Existen disimiles metodologías ágiles de proceso y muchos tienen una cierta similitud o siguen a misma filosofía, pero si es importante señalar que todas las metodologías ágiles se ajustan en mayor o menor grado en un equipo de trabajo para el desarrollo de software.

1.2.2.1. Programación extrema.

Plantea (R. S. Pressman 2010) que esta metodología ágil utiliza un enfoque orientado a objetos como su paradigma de desarrollo preferido. Abarca un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades del marco de trabajo: planeación, diseño, codificación y prueba. Estas actividades manejan una gran cantidad de elementos para el desarrollo del software. En la planeación se describe las características y la funcionalidad requeridas para el software que se construirá. El diseño sigue de manera rigurosa el principio de mantenimiento simple y ofrece una guía de implementación como debería ser el software a desarrollar.

1.2.2.2. Desarrollo adaptativo de software.

El desarrollo adaptativo de software es conocido como una técnica para construir el software y sistemas complejos. Su filosofía se enfoca en la colaboración humana y la organización propia del equipo. En este tipo de desarrollo ágil se define como un “ciclo de vida” e incluye tres fases: especulación, colaboración y aprendizaje. Donde se definen dentro de la especulación elementos como: planeación del ciclo adaptativo, enunciado de la misión, restricciones del proyecto, requisitos básicos, plan de lanzamiento en el tiempo. En cuanto a la colaboración se tratan temas como la recopilación de

requisitos JAD[12] y las especificaciones mínimas. Por último, esta tercera fase la cual se basa en el aprendizaje y en el progreso a través de un ciclo completo contiene varios elementos como: componentes implementados y probados, grupos de enfoques para retroalimentación y revisores técnicas formales.

1.2.2.3. Proceso Unificado Ágil

El Proceso Unificado Ágil o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles como; el Desarrollo Dirigido por Pruebas, Modelado ágil, Gestión de cambio ágil, Refactorización de Base de Datos para mejorar la productividad. Este tipo de desarrollo ágil al igual que el RUP tiene varias fases las cuales son: inicio, elaboración, construcción y transición. El proceso unificado ágil también comprende siete disciplinas, cuatro son ingenieriles y tres de gestión de software, las cuales son: Modelo, implementación, prueba, despliegue, gestión de configuración, gestión de proyecto y entorno.

1.2.2.4. Variación de AUP para la UCI

Este desarrollo ágil es una adaptación de la metodología AUP creada en la universidad teniendo en cuenta las características de los proyectos de software en la UCI. La característica principal del uso de esta metodología en la universidad es la capacidad de adaptación al ciclo de vida definido por las actividades productivas de la universidad. Propone aumentar la calidad del software que se produce apoyándose en el Modelo CMMI-DEV v1.3. (Rodríguez Sánchez 2015). Actualmente es usado en los proyectos productores de software dentro de la Universidad de las Ciencias Informáticas.

Al igual que el AUP original esta variación tiene 4 faces, lo que a diferencia del original sufren varios cambios. Para el ciclo de vida de los proyectos de la UCI se mantiene la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la cual tiene como nombre Ejecución y se agrega la fase de Cierre. Para una mayor comprensión se muestra la tabla siguiente:

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)

[12] *JDA: Empresa que se dedica a la planificación y programación de los productos.*

Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración Construcción Transición	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 3: Comparativa entre la metodología AUP y la variación AUP-UCI. (Rodríguez Sánchez 2015)

La versión AUP-UCI comprende también de siete disciplinas, pero son tratadas a un nivel aún más atómico que el AUP original. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en tres disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes tres disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMIDEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). (Rodríguez Sánchez 2015)

Es por eso que luego de hacer un estudio por las diferentes metodologías, tanto las tradicionales como las ágiles teniendo en cuenta las distintas características y habilidad o facilidad al adaptarse al proyecto, se toma como metodología a utilizar la versión AUP-UCI. Porque permite estandarizar cada proceso del desarrollo del software dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. A demás que logra hablar un lenguaje común en cuanto a las fases y disciplinas. Estas fases y disciplinas, permiten elaborar un software que cumpla las necesidades el cliente.

En la fase de inicio ya el cliente forma parte del equipo de desarrollo, con el objetivo de que exista una comprensión común cliente- equipo de desarrollo, esta fase permite que el componente a desarrollas converja con más rapidez a la solución y cumpla con los requisitos del cliente.

En la fase elaboración, el equipo de desarrollo tiene bien claro la arquitectura que se va a utilizar y conocen de los requisitos del sistema, lo que permite que el componente de reporte tenga definido su arquitectura y requisitos definidos. Esto propicia que se desarrolle con más exactitud, teniendo en cuenta que es lo que se va hacer y cómo se va hacer.

La próxima fase se adentra al trabajo duro, de suma importancia. En fase de construcción, el equipo de desarrollo implementa el producto y lo prueba. Es importante que el componente pase por esta fase porque permite conocer si se está cumpliendo con lo especificado por el cliente, además permite resolver problemas de manera inmediata a medida que se implemente y se pruebe.

Por último, en la fase de transición, el sistema creado es sometido a las pruebas que define esta metodología (Prueba internas, prueba de liberación, y prueba de aceptación). Al someter al componente de reportes bajo estas pruebas se verifica el buen funcionamiento, si está listo para usuarios finales.

1.5. Herramientas para el desarrollo del software.

Durante el proceso de desarrollo de un software es necesario tener definido cuáles son las herramientas a utilizar. Conocer que permite y las características fundamentales de cada una de ellas. En este acápite se hace una investigación de las aplicaciones que participarán en el desarrollo del software en aras de solucionar los problemas existentes a la hora de generar un reporte de en el centro de digitalización.

1.5.1. Visual Paradigm for UML 8.0 Enterprise Edition

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software. Permite representar diagramas de clases, generar código desde diagramas y generar documentación. Tiene soporte multiplataforma y proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Tiene apoyo adicional en cuanto a generación de artefactos automáticamente y generación de documentación, ya que brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas. (Visual Paradigm 2013).

Esta herramienta cuenta con varias características fundamentales como los diagramas de Procesos de Negocio y el modelado colaborativo con el repositorio subversión, generación de bases de datos, y transformación de diagramas de Entidad-Relación en código SQL, así como un generador de informes para generación de documentación.

Esta herramienta fue seleccionada debido a las especificidades del modelo de desarrollo donde se maneja cada aspecto durante el análisis y diseño. También porque es fácil de interactuar con esta herramienta además provee muchos mecanismos para hacer un buen análisis y diseño de un software en específico.

1.5.2. Sistema Gestor de Bases de Datos PostgreSQL 9.3

PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofrece control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo transacciones, y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación incluyendo Java.

Se caracteriza, además por ser un sistema estable, orientado a objeto, de alto rendimiento y gran flexibilidad. El lenguaje de consultas de PostgreSQL es una variación del lenguaje SQL estándar SQL, y son extensiones propias de PostgreSQL.

Además, la universidad cuenta con la información necesaria gracias a la existencia de una comunidad de PostgreSQL.

1.5.3. NetBeans 8.0

NetBeans es un entorno de desarrollo integrado, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. Es un producto libre y gratuito sin restricciones de uso. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos.

NetBeans IDE permite desarrollar aplicaciones Java en entorno de escritorio, móviles y aplicaciones web. Está escrito y pensado para Java; pero brinda soporte para los lenguajes de programación: C/C++, Ruby, Python, HTML, CSS[13] y PHP(Keegan et al. 2006).

Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. (Keegan et al. 2006).

[13] *CSS: Hoja de Estilo de Cascada*

1.5.4. Marco de trabajo.

Un marco de trabajo es un entorno, plataforma o estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Los marcos de trabajo facilitan el desarrollo de software, permiten evitar los detalles de bajo nivel, concentrando más esfuerzos y tiempo en identificar los requerimientos del software. Un marco de trabajo es conocido como una manera de crear algún producto o software de manera organizada, lo que permite trabajar con facilidad y que el producto sea creado con mayor rapidez (Alegsa 2016).

Existen múltiples marcos de trabajos y para disimiles entornos de desarrollos. Los más utilizados en el lenguaje de programación java son Spring[14], Struts 2[15],JSF[16] y Vaading[17] según una encuesta realizada por RebelLabs[18] . Existen otros no muy conocido mundialmente como es el caso de XEGFORT, creado por el Centro de gobierno electrónico (CEGEL) (Gómez Martínez y Durán Bolaño 2015).

1.5.5. XEGFORT

Este marco de trabajo es muy utilizado en distintos proyectos productivos de CEGEL, como es el caso de SIGFE[19], SIRECC[20] Y SIGESAP[21]. XEFORT tiene como objetivo implementar los sistemas registrales que se desarrollan en este centro. Incorpora una biblioteca de componentes basada en Swing[22] como una Interfaz de Programación de Aplicaciones (API) para el desarrollo de componentes visuales.(Gómez Martínez y Durán Bolaño 2015). Está diseñado para la creación de aplicaciones con arquitectura Cliente-Servidor utilizando la plataforma Edición Empresarial de Java (JEE por sus siglas en inglés). Consta de dos partes, una que se encuentra en el lado del cliente facilitando la creación de la capa de presentación y la comunicación con los objetos remotos desplegados en el servidor y una segunda que se encuentra en el lado del servidor donde se encuentra la lógica de negocio. XEGFORT provee un alto nivel de configuración y adaptabilidad, lo cual lo convierte en un software altamente reusable. Actualmente este marco de trabajo tiene grandes condiciones para la administración de las aplicaciones informáticas. (Gómez Martínez y Durán Bolaño 2015)

14 Spring: Utilizado para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

15 Struts 2: es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma Java EE (Java Enterprise Edition.)

16 JSF: es una tecnología para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

17*Vaadin: Es un marco web de código abierto para aplicaciones de Internet.*

18 *RebelLab: empresa social que se enfoca en el aprendizaje significativo de herramientas informáticas.*

20 Sistema de Informatización Registral de la Cámara de Comercio.

21Sistema de Gestión de Antecedentes Penales

22 Es una biblioteca gráfica para java.

Algunas de las funcionalidades y componentes que provee (Gómez Martínez y Durán Bolaño 2015):

- Una interfaz de usuario básica
- Un mecanismo para la internacionalización de las interfaces de usuario.
- Un mecanismo para el acceso a los procedimientos remotos publicados en un servidor de aplicaciones.
- Un mecanismo de mensajería.
- Un mecanismo para gestionar las excepciones lanzadas por la aplicación que lo utilice.

1.5.6. Servidor de aplicaciones.

Un servidor de aplicaciones es una implementación de la especificación J2EE[23]. Se define como J2EE un estándar que permite el desarrollo de aplicaciones de empresa de una manera sencilla y eficiente. Los servidores de aplicaciones proporcionan servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas, brindan soporte a una gran variedad de estándares, tales como HTML, XML, IIOP, JDBC, SSL, entre otros, que les permiten su funcionamiento en ambientes Web y la conexión a una gran variedad de fuentes de datos, sistemas y dispositivos.

1.5.6.1. GlassFish 4.0.

El servidor GlassFish proporciona un servidor ligero y modular para el desarrollo de aplicaciones Java Enterprise Edition (Java EE) 7 y Java Web Services. Ofrece rendimiento empresarial, escalabilidad y fiabilidad(Heffelfinger 2007). GlassFish Server 4.0 es compatible con Java EE 7, que proporciona la base para la entrega de aplicaciones HTML dinámica y escalable. Esta versión tiene como principales características: clusters [24] e instancias independientes, alta disponibilidad, Servidor incorporado(Heffelfinger 2007).

1.5.7. Lenguajes de programación.

Un lenguaje de programación es un "conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una

23 *J2EE: Java 2 Platform, Enterprise Edition*

24 Clusters: es un grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio.

computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias" (Mazón Olivo et al. 2015).

Los lenguajes de programación son herramientas o mecanismos que permiten crear programas y software. Es una forma de comunicarse con la computadora dándole instrucciones de manera tal que cumpla con cada una de ellas para que funciones un determinado programa. Se conoce además que un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.

En el mundo de hoy existen muchos lenguajes de programación diseñados para desarrollar aplicaciones web, móviles y crear software de computadora, es decir aplicaciones desktops. Actualmente se estima que hay más de 200 lenguajes de programación alrededor del mundo, pero para (TIOBE Index 2017) las más utilizadas son java, C, C++, C#, Python, PHP, Visual Basic .NET y JavaScript. En estos lenguajes mencionados anteriormente cuatro de ellos son utilizados para crear aplicaciones de escritorios como es el caso de java, C, C++, C#. Según las características del proyecto, tiempo de desarrollo y las exigencias del cliente se decide de estos tres lenguajes escoger a Java para implementar el sistema informático propuesto.

1.5.8. Lenguaje Java

Java es un lenguaje totalmente orientado a objeto, diseñado como una mejora de C++. Posee una curva de aprendizaje muy rápida. Proporciona una colección de clases para su uso en aplicaciones de red, que permite establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Es un lenguaje robusto ya que fue diseñado para crear software altamente fiable. Java está diseñado para soportar aplicaciones que serán ejecutados en los más variados entornos de red, desde Unix a Windows, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos(Mazón Olivo et al. 2015).

1.6. Arquitectura de software.

La Arquitectura de Software (AS) es la organización fundamental de un sistema, encargada de sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución (Reynoso 2004).

La arquitectura no es un software operativo. Es una representación que permite que el ingeniero de software: analice la efectividad del diseño para cumplir con los requisitos establecidos, considere

opciones arquitectónicas en una etapa en que aún resulta relativamente fácil hacer cambios al diseño y reduzca los riesgos asociados con la construcción del software (R. S. Pressman 2010).

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software. Permite crear un producto bajo las exigencias de distintos patrones de diseños con el fin de que el mismo desarrollo de dicho producto se guiado y soportado por una base arquitectónica y converja paulatinamente cumplir con los distintos requisitos del cliente.

Los patrones arquitectónicos se abocan a un problema de aplicación específica dentro de un contexto dado y sujeto a limitaciones y restricciones. El patrón propone una solución arquitectónica que sirve como base para el diseño de la arquitectura (R. S. Pressman 2010).

Para el sistema propuesto se utilizó una arquitectura Cliente-Servidor aplicando el patrón arquitectónico N-capa, específicamente se definieron 4 niveles o capas, las cuales se describen a continuación. En el Cliente se encuentran la Capa de Presentación, que contiene las vistas de la aplicación y es la encargada de la comunicación con el usuario y la Capa de Negocio de Cliente, la cual contiene las acciones y los gestores de negocio del cliente. Las acciones se encargan de manejar la información existente en los formularios, permitiendo que los gestores de negocio del cliente realicen las validaciones necesarias. En el Servidor se encuentra la Capa de Negocio Servidor, la Capa de Acceso a Datos y las Entidades Persistentes. La primera contiene los gestores de negocio del servidor que se encargan de realizar validaciones más cercanas al negocio delegando en los gestores de acceso a datos la persistencia de la información obtenida, a través de las entidades persistentes. Transversal a ellas se encuentran las Entidades de Dominio que son las responsables de transportar la información desde el servidor hasta el cliente.

1.7. Métricas para medir el diseño de software.

Las métricas del producto ayudan a conocer mejor el diseño y la construcción del software que se elaboran. Se centran en atributos específicos de los productos de trabajo de ingeniería de software (R. S. Pressman 2010).

Las métricas son usadas para medir un software lo cual es un elemento clave en cualquier proceso de la ingeniería. Esto permite comprender mejor los atributos de los modelos que se crean y evaluar la calidad de los productos o software que se construyen. Estos aspectos de manera breve explican la importancia de tener en cuenta las métricas para medir el diseño y calidad del software.

La serie de métricas LK (Kidd y Lorenz 1994) son las seleccionadas para la validación del diseño de las clases de la solución, de ellas específicamente la métrica Tamaño Operacional de las Clase (TOC) que consiste en medir el tamaño general de una clase tomando como valores el total de operaciones y el número de atributos (operaciones y atributos tanto heredados como privados de la instancia), encapsulados por la clase; y la métrica Relaciones entre Clases (RC) cuyo resultado viene dado por el número de relaciones de uso de una clase con otra u otras.

A través de estas métricas se puede medir el estado de los atributos de calidad que a continuación se mencionan:

- Responsabilidad: consiste en la responsabilidad asignada a una clase modelada de un dominio o concepto, de la problemática propuesta.
- Complejidad de implementación: consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento: consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras.
- Complejidad de mantenimiento: consiste en el grado de esfuerzo necesario a realizar para desarrollar una reparación, una mejora o una corrección de algún error de un diseño de software.
- Cantidad de pruebas: consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

1.7.1. Métrica Relaciones entre Clases (RC)

La métrica RC está dada por el número de relaciones de uso de una clase con otra y evalúa los atributos de calidad, acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas. A continuación, se explican los pasos para aplicar la métrica:

1. Determinar la cantidad de relaciones de uso (CRU) que poseen las clases a medir.

2. Calcular el promedio de las CRU.

3. Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases.

1.7.2. Métrica Tamaño Operacional de las Clase (TOC)

La métrica TOC se aplicada a cada una de las clases del diseño con el objetivo de medir la calidad de las mismas con respecto a su grado de responsabilidad, complejidad de implementación y reutilización. A continuación, se explican los pasos que se llevaron a cabo para aplicar la métrica:

1. Cálculo del umbral. El umbral se toma del tamaño general de una clase que se determina sumando todas las operaciones que posee.

2. Calcular el promedio de los umbrales.

3. Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases.

1.8. Pruebas de software.

El desarrollo de software requiere de un conjunto de actividades donde las posibilidades que aparezcan fallos de implementación o usabilidad son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, por lo que el desarrollo de software debe estar basado en garantizar la calidad desde el inicio. Las pruebas constituyen un elemento fundamental para medir el grado en que se encuentran o se cumplen los requisitos del cliente.

Las pruebas son un conjunto de actividades que se planean con anticipación y se realizan de manera sistemáticas. Permiten descubrir errores cometidos durante el diseño y construcción del software (R. S. Pressman 2010). La importancia de hacer pruebas de software es enorme, pues se le dedica más esfuerzo que otra actividad durante el desarrollo del producto, lo que permite garantizar que el software porte con la calidad suficiente y cumpla con las exigencias del cliente. Esto se logra debido el conjunto de pasos que se definen, que incluyen técnicas y métodos específicos del diseño de casos de pruebas.

Existen cuatro niveles de pruebas:

1. Pruebas unitarias: se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño, ya sea un componente o un módulo del software (R. S. Pressman 2010).

2. Pruebas de integración: es una técnica sistemática para confeccionar la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño (R. S. Pressman 2010).

3. Pruebas de sistema: abarca una serie de pruebas diferentes cuyo propósito principal es ejercitar profundamente el sistema de cómputo. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se hayan integrado adecuadamente todos los elementos del sistema y que realizan las funciones adecuadas (R. S. Pressman 2010).

4. Pruebas de aceptación: una vez culminado el proceso de pruebas por parte del equipo de desarrollo, es indispensable, que el cliente verifique que el producto ha sido desarrollado con las normas y criterios establecidos, y cumple con todos los requisitos especificados por el cliente (Zapata 2013).

Para validar la solución propuesta se decidió aplicar pruebas de unitarias, utilizando las técnicas de caja negra y caja blanca, así como pruebas de aceptación con el cliente.

1.8.1. Técnica de Caja Negra

La prueba de caja negra son las que se aplican a la interfaz del software. Esta examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software (R. S. Pressman 2010).

Esta técnica se lleva a cabo sobre la interfaz del software, utilizando casos de prueba que quien al probador durante el desarrollo de la actividad. Esta técnica examina algunos aspectos del funcionamiento del software, sin tener en cuenta la estructura interna del mismo (Sommerville 2011)

Estas pruebas permiten encontrar:

1. Funciones incorrectas o ausentes.

2. Errores de interfaz

3. Errores en estructuras de datos o en accesos de las Bases de Datos externas.

4. Errores de rendimiento

5. Errores de inicialización y terminación.

Existen varias técnicas para desarrollar la técnica de caja negra:

Técnica de la Partición de Equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Técnica del Análisis de Valores Límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Técnica de Grafos de Causa- Efecto: permite al encargado de la prueba, validar complejos conjuntos de acciones y condiciones.

1.8.2. Técnica de Caja Blanca

Las pruebas de caja blanca del software se basan en un examen cercano al detalle procedimental. Se prueba la lógica del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejercen conjuntos específicos de condiciones, bucles o ambos (R. S. Pressman 2010).

La técnica de caja blanca se basa en el minucioso examen de los procedimientos. Se comprueban los caminos lógicos del software por casos de prueba que ejerciten conjuntos específicos de condiciones y se puede examinar el estado del programa en varios puntos, para determinar si el estado real coincide con el esperado o mencionado.

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

Para ejecutar caja blanca se utiliza la técnica de camino básico. La misma permite al diseñador de casos de prueba, obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (R. S. Pressman 2010).

1.9. Conclusiones parciales.

Se realizó un análisis de las herramientas generadoras de reportes a nivel internacional y nacional, lo que permitió tomar como decisión la utilización de la librería Jasper Report, debido a que es una librería de código abierto poderosa y flexible, permite la creación de reportes que pueden ser presentados en la web o se pueden crear ficheros en diversos formatos y los reportes son capaces de presentar los datos de manera textual o a través de gráficos. Se analizaron herramientas, tecnologías y lenguajes propuestos por políticas del Centro de Gobierno Electrónico (CEGEL) de la universidad, utilizando para este trabajo el modelo de desarrollo Programa de Mejora y el Marco de Trabajo XEFORT, como herramienta CASE Visual Paradigm 8.0 para el modelado y como lenguaje de modelado UML. Como servidor de bases de datos PostgreSQL en su versión 9.3, el entorno de desarrollo integrado NetBeans 8.0 y el servidor de aplicaciones GlassFish 4.0.

CAPÍTULO 2. DESCRIPCIÓN DE LA PROPUESTA

2.1. Introducción

Debido a las dificultades existentes en el instante en que se obtiene la información sobre los procesos y se realizan los reportes, se toma como decisión desarrollar un componente para generar reportes que permita solucionar los inconvenientes a la hora de obtener y consultar información en cada una de las áreas del proceso de la digitalización. Este componente será capaz de generar reportes minimizando los problemas existentes en el momento en que se obtiene información de una determinada área. La generación de reportes, es una operación que surge como consecuencia directa de la necesidad de interrelación y operacionalización de los procesos negocio entre las áreas del sistema de digitalización de los documentos con valor legal. En este capítulo se describe la propuesta de solución del módulo según las fases propuestas por el Programa de Mejora y las caracterizaciones de su utilización en el desarrollo del componente.

2.2. Levantamiento de requisitos.

En concordancia con lo planteado por el Plan de Mejora en el Libro de Procesos para la Administración de Requisitos, se generaron los artefactos correspondientes al modelo de desarrollo.

Se realiza además el levantamiento de requisitos, tanto funcionales como no funcionales, generándose el artefacto Especificación de Requisitos del Software para el componente para generar reportes, lo cual constituye un elemento clave para el diseño y la posterior implementación. Este artefacto se caracteriza por contener para cada uno de los requerimientos el nombre, una descripción, establece la complejidad y prioridad para el cliente, facilitando el orden y criticidad de implementación. En esta plantilla además se establecen los campos necesarios, tipo de datos y regla o restricciones.

Luego de realizada la Especificación de Requisitos se dio paso a la Descripción de Requisitos del Software: Componente para generar reportes para el sistema de digitalización de documentos con valor legal; Caracterizado fundamentalmente por presentar los requisitos de forma completa, definiéndose todas las responsabilidades del componente. Este artefacto presenta una adecuada organización y documentación donde los términos, las tablas y los prototipos están correctamente descritos y referenciados. Cada requisito, tiene una única interpretación evitando la ambigüedad en las definiciones y funcionalidades.

Según lo referido anteriormente, la descripción de requisitos puede ser tomada como artefacto de entrada al diseño para facilitar la comprensión del desarrollo del componente. A continuación, se presentan los requisitos funcionales y no funcionales del componente para generar reportes.

2.2.1. Requisitos funcionales.

Para realizar el proceso de captura de los requisitos, se hizo necesaria la utilización de algunas técnicas que sirvieron para verificar la veracidad de los objetivos propuestos para el desarrollo de este componente, permitiendo que dieran satisfacción a las necesidades de los clientes. Dentro de las técnicas utilizadas se encuentran: las entrevistas, aplicada a los especialistas de Centro de Gobierno Electrónico que fueron considerados proveedores válidos para los cuales va a ser desarrollado el componente.

Listado de los requisitos funcionales (RF):

- RF1: Visualizar documentos pendientes en digitalización.
- RF2: Visualizar documentos pendientes en Metadatos.
- RF3: Visualizar documentos pendientes en Firma.
- RF4: Visualizar documentos rechazados en digitalización.
- RF5: Visualizar documentos rechazados en Metadatos.
- RF6: Visualizar documentos rechazados en Firma.
- RF7: Visualizar documentos finalizados en digitalización.
- RF8: Visualizar documentos finalizados en Metadatos.
- RF9: Visualizar documentos finalizados en Firma.
- RF10: Visualizar documentos terminados.
- RF11: Visualizar cantidad de documentos por lotes.
- RF12: Visualizar cantidad de folios por documentos.
- RF13: Visualizar cantidad de folios de un fondo documental.
- RF14: Visualizar cantidad documentos digitalizados por un usuario determinado.
- RF15: Visualizar documentos digitalizados por usuario.
- RF16: Visualizar documentos con metadatos por usuario.
- RF17: Visualizar documentos con firma por usuario.
- RF18: Visualizar documentos por tipo documental.
- RF19: Exportar reporte a formato PDF.
- RF20: Generar reportes.

2.2.2. Requisitos no funcionales.

Los requerimientos no funcionales (NF) cumplen con las especificaciones establecidas en el documento Especificación de Requisitos de Software NF del Componente para generar reportes. Este documento puede ser consultado en (González Hernández, y otros, 2014). Seguidamente, listado de los requisitos NF:

Usabilidad.

RnF-01 El sistema podrá ser utilizado por los usuarios que se describen a continuación: Director de Formación y Membrecía, Director Jurídico, Asesor Jurídico, Especialista en Desarrollo Empresarial, Junta Ejecutiva, Abogado.

RnF-02 El sistema deberá presentar una interfaz de usuario fácil de entender y usar.

RnF-03 Agrupar vínculos y botones por grupos funcionales. La consistencia de la interacción entre usuario y sistema estará determinada por el diseño de la interfaz de usuario que mantendrá los elementos como menús, banners y zona de trabajo, en posiciones fijas, además de la mayor uniformidad posible entre cuadros de texto y botones.

RnF-04 El tiempo de respuesta brindado por el sistema será menor de 3 segundos. Teniendo en cuenta el nivel de concurrencia que pueda existir, debe ser capaz de prestar servicio sin que se deterioren los tiempos de respuestas en función de los requisitos de hardware necesarios.

Disponibilidad.

RnF-05 El sistema deberá estar disponible las 24 horas de los días laborales para su funcionamiento y realización de las salvas que se deben ejecutar después de la jornada laboral (8 horas).

Restricción de diseño

RnF-06 Para el montaje del sistema se requerirá del sistema gestor de bases de datos PostgreSQL 9.3 y del servidor de aplicaciones Glassfish 4.0.

RnF-07 Diseñar el sistema a través subsistemas: el sistema debe garantizar que cada subsistema tenga fronteras claramente definidas y funciones relacionadas.

Interfaz.

RnF-08 El sistema presentará una interfaz legible, simple de usar e interactiva.

RnF-09 El sistema deberá tener un diseño de interfaz de usuario en funciones de los estándares definidos en la UCI para las Interfaces de usuario.

Estándares aplicables.

RnF-10 Se aplicarán los estándares de java en función de facilitar el mantenimiento de la aplicación.

Seguridad

RnF-11 El sistema podrá ser utilizado solamente por usuarios autenticados en el sistema.

RnF-12 El sistema brindará la posibilidad de establecer permisos sobre acciones, garantizando que solo acceda a la información quien esté autorizado.

RnF-13 El sistema mostrará las funcionalidades de acuerdo a quien esté autenticado en el sistema.

RnF-14 El sistema debe asegurar el almacenamiento de las credenciales de los usuarios utilizando algoritmos criptográficos que oculten la identidad verdadera de los usuarios.

RnF-15 El sistema debe permitir almacenar todas las acciones de los usuarios sobre el sistema como constancia de las acciones realizadas.

Al concluir el levantamiento de los requisitos, tanto funcionales como no funcionales, se aplicaron técnicas para validar los requisitos recolectados. Estas técnicas serán analizadas posteriormente en el capítulo 3.

2.3. Historias de usuarios.

Las historias de usuarios son herramientas utilizadas para la creación del sistema. Permiten conocer los requerimientos del sistema al equipo de desarrollo y brinda una mejor comprensión a los desarrolladores del proyecto. Son varios textos que describe el funcionamiento del sistema según el cliente de una forma clara, sencilla y con poca profundidad de detalle de los requerimientos. También son utilizadas para estimar el tiempo que el equipo de desarrollo tomará para realizar las entregas. En una entrega se puede desarrollar una o varias historias de usuario, esto depende del tiempo que demore la implementación de cada una de las mismas (Carabia y Píriz 2003).

A continuación, se muestra un ejemplo de historia de usuario del requisito funcional: Visualizar documentos pendientes en digitalización.

Número: 1	**Nombre del requisito:** Visualizar documentos pendientes en digitalización
Programador: 1	**Iteración Asignada:**1
Prioridad: Alta	**Tiempo Estimado:** 2d
Riesgo en Desarrollo: media	**Tiempo Real:** 4d
Descripción: Muestra en una tabla los documentos que están pendientes a pasar por cada proceso en el área de digitalización. Dicha tabla contiene código de barra del documento, el fondo al que pertenece documento y el lote del documento.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

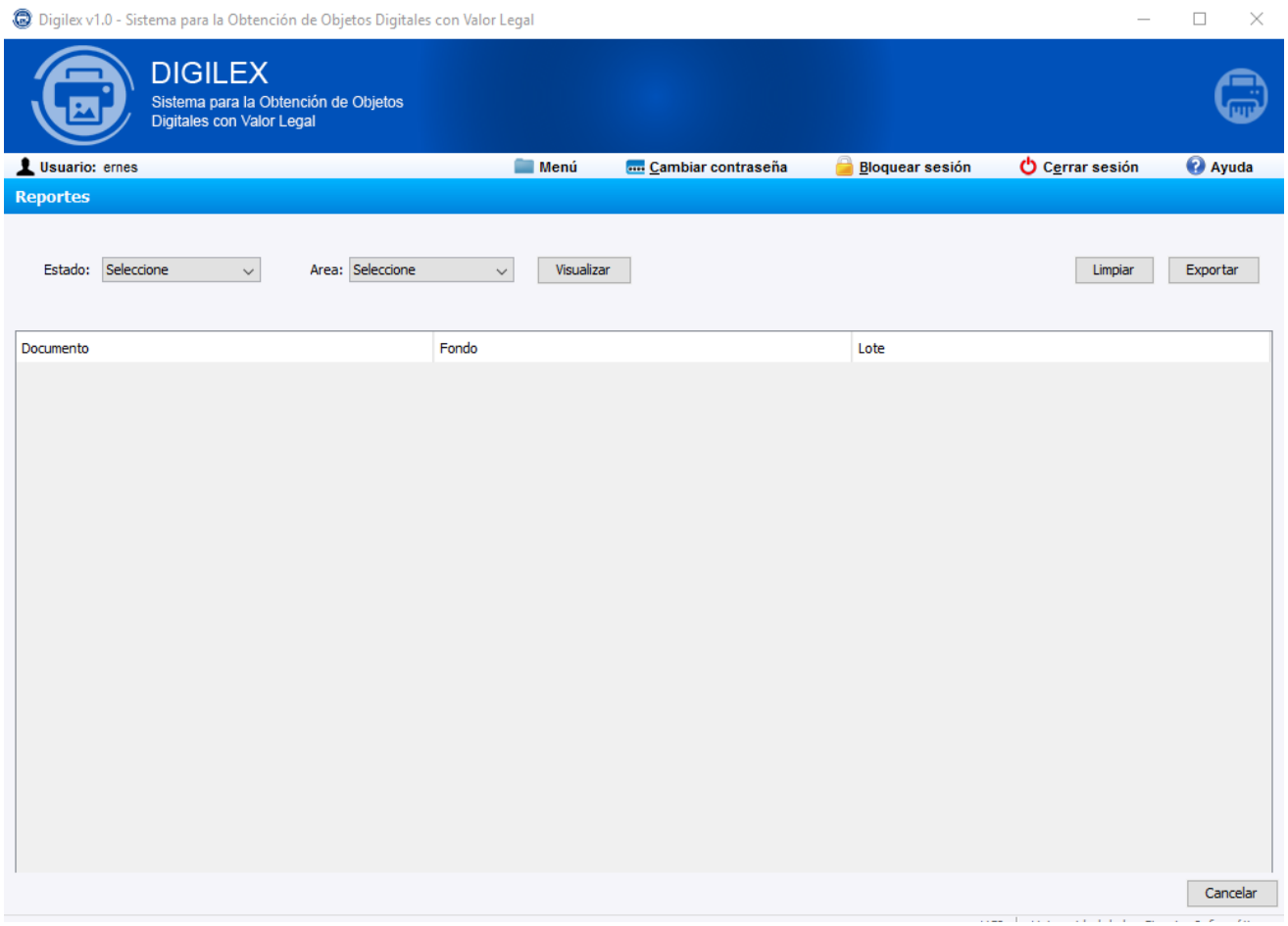


Tabla 4: Historia de usuario Visualizar documentos pendientes en digitalización.

2.4. Análisis y diseño

En esta fase se modela el sistema y su forma, es decir su arquitectura, cumpliendo con los requisitos funcionales detectados en el epígrafe anterior y los no funcionales también visto anteriormente. Esto contribuye a tener una arquitectura sólida y estable que se convierte en un plano para la implementación. También en esta fase se desarrollan: documento de arquitectura, diagramas de clases, diagramas de entidad relación, diagrama de despliegue entre otros.

2.4.1. Diseño arquitectónico.

La arquitectura del componente que se utiliza según lo investigado en el capítulo anterior, es Cliente-Servidor con el objetivo de proporcionar un nivel de mantenibilidad adecuado para la solución del sistema. Este estilo permitirá la comunicación contractual entre la aplicación cliente y la aplicación servidor que permitirán dar solución a los requerimientos de los usuarios. Esta arquitectura se le incluirá la arquitectura n-capas, la cual permite organizar mejor el diseño teniendo en cuenta que establecer una organización jerárquica entre cada una de las capas, garantizando que cada capa proporcione servicios a la capa inmediatamente superior. Las capas para el desarrollo del componte son: Presentación, Negocio del cliente, Negocio del servidor y Acceso de datos.

Capa de Presentación: Esta capa contendrá los componentes de interfaz de usuario. Muestra los datos, que obtiene en la capa negocio cliente. Controla los eventos que se generan en esta capa.

Capa de Negocio del Cliente: Esta capa tendrá como responsabilidad fundamental llevar a cabo procesos de validación y gestión de la información entre la aplicación cliente y la aplicación servidor. Gestiona la lógica de negocio relacionada en la parte del cliente, y el acceso a las funcionalidades de los componentes del servidor.

Capa de Negocio del Servidor: Esta capa gestionará los componentes que implementarán la lógica de negocio del sistema. Es responsable de presentar una fachada a todos los componentes que se encuentran en el servidor a modo de gestores de negocio, físicamente alojada en un servidor de aplicaciones al cual se accede de forma remota.
Capa de Acceso de Datos: Esta capa será la encargada de manejar la información entre la aplicación servidor y la base de datos. Es la que recibe los datos de la capa de negocio servidor y los puede consultar, persistir, actualizar y eliminar en la base de datos, mediante mecanismos que ofrecen las especificaciones utilizadas, que se realizan en conjunto con las entidades persistentes.

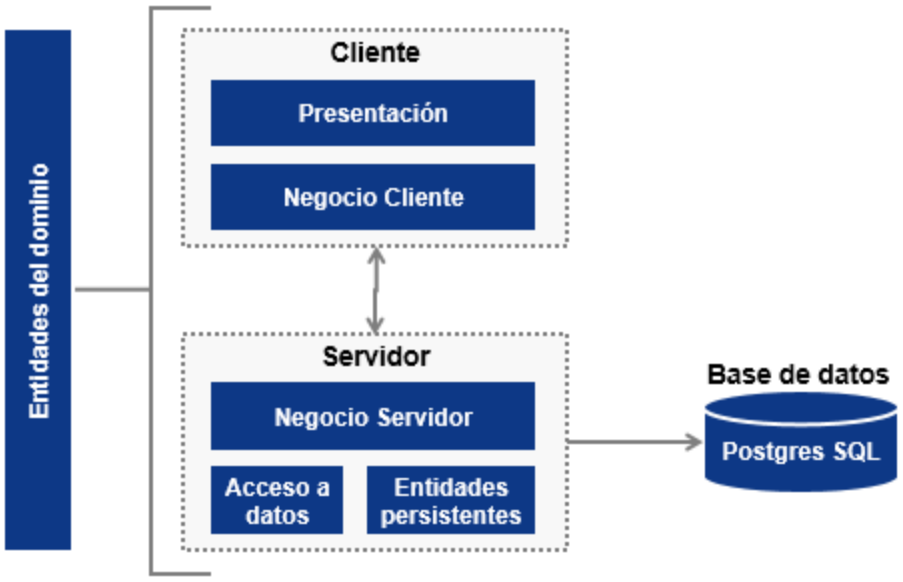


Figura 1: Arquitectura del componente (Fuente de elaboración propia).

2.4.2. Diseño de clases

Los diseños de clases permiten obtener de forma estática la representación de los requisitos que se lleva a cabo a través de las clases del sistema y sus relaciones. Su principal utilidad radica en mostrar a través de sus atributos y métodos la estructura de las clases que después serían traducidas al lenguaje de programación Java.

Clase de diseño del Componente de Reporte y sus funciones.	
Nombre	**Descripción**
Documento	Clase entidad persistente que hace referencia a la tabla Documento en la base de datos.
PDocumentoDAO	Clase que permite la comunicación con la base de datos. Permite además trabajar con las consultas SQL para extraes la información de la BD.
EDDocumento	Clase entidad de dominio, la cual tiene toda la información de los documentos.
GestorNSReporte	Es la clase que permite la comunicación entre el servidor y el cliente
EDDocumentoComverte	Clase que transforma de entidad de domino a clase persistente y viceversa.

GestorNCReporte	Clase que permite la comunicación entre el cliente y el servidor.
AcciónMostrarDocumentos	Contiene toda la implementación que describe cada funcionalidad con el fin de mostrar los documentos.
MostrarDocumento	Interfaz o formulario que muestra los datos de los documentos extraídos de la Base de Datos.

A continuación, se muestra un fragmento de la descripción de las clases del diseño.

Tabla 5: Clase de diseño del Componente de Reporte y sus funciones

Visto la tabla anterior seguidamente se muestra un fragmento del diagrama de clases del diseño:

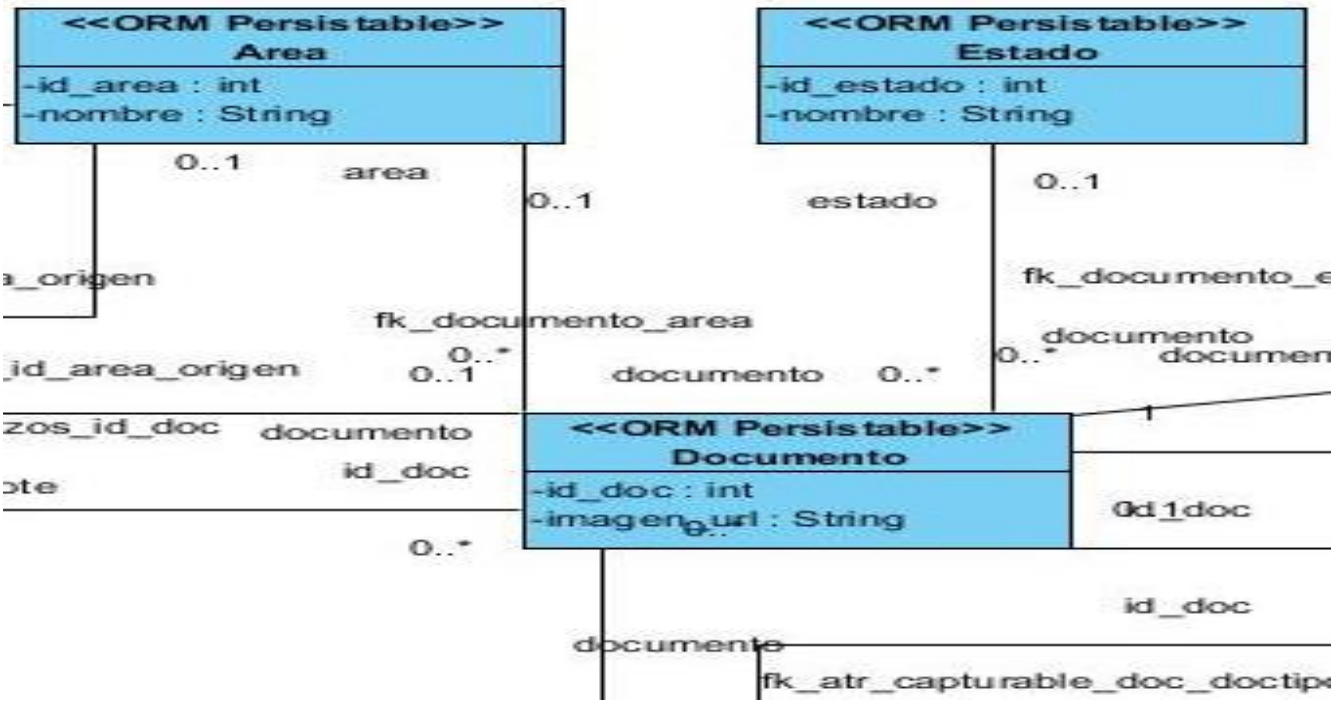


Figura 2: Fragmento del Diagrama de clases

2.4.3. Patrones de diseños

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Durante el desarrollo de la solución se hace uso de varios patrones de diseños. A continuación, se describen los patrones empleados:

Patrones DAO.

Los patrones Objeto de Acceso a Datos son las interfaces que se encuentran entre la aplicación y uno o más dispositivos de almacenamiento de datos. Básicamente son las clases que interactúan con la base de datos. Este patrón se utiliza en la clase DocumentoDAO, la cual permite el trabajo con la base de datos a través de las consultas SQL.

Patrones DTO.

Son utilizados por las clases DAO, para transportar los datos desde la base de datos hasta los gestores de negocio del servidor y viceversa. Se utiliza este patrón en la clase EDDocumento.

Patrones de diseño GRASP.

Experto: otorga la responsabilidad de adicionar un tipo documental a la clase GestorMarcoTrabajo, que es la experta de instanciar y mostrar los formularios y las acciones en el área de trabajo.

Controlador: La clase AccionMostrarDocumento le asigna la responsabilidad de controlar y coordinar el esfuerzo de las demás clases, para responder a la petición del usuario.

Creador: Es el patrón que asigna responsabilidad de crear una instancia de una clase, es decir asigna la responsabilidad de crear un objeto de una clase. En la implementación se evidencia este patrón cuando en la clase PDocumentoDAO se crea un objeto de la clase EDDocumentoConverter, para transportar datos desde el cliente al servidor.

Alta Cohesión: la principal característica de este patrón es asignar responsabilidades de modo que la cohesión siga siendo alta. La información que almacena una clase debe de ser coherente y debe estar en la medida de lo posible relacionada con la clase. El patrón se evidencia en cada de una de las clases del componente, de tal forma que se elimina la sobrecarga de responsabilidades.

Bajo Acoplamiento: su principal característica es mantener las clases más independientes entre sí y con la menor cantidad de relaciones; la cual posibilita que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. El patrón se evidencia en cada una de las clases diseñadas.

2.4.4. Modelo de la Base de Datos

La estructura de la base de datos del componente se muestra mediante el modelo de datos. Este modelo fue elaborado utilizando la técnica de modelado de datos Entidad-Relación, mostrando las

entidades de datos, sus atributos asociados y las relaciones entre las entidades. La imagen que se muestra a continuación, refleja un fragmento del modelo de datos del componente.

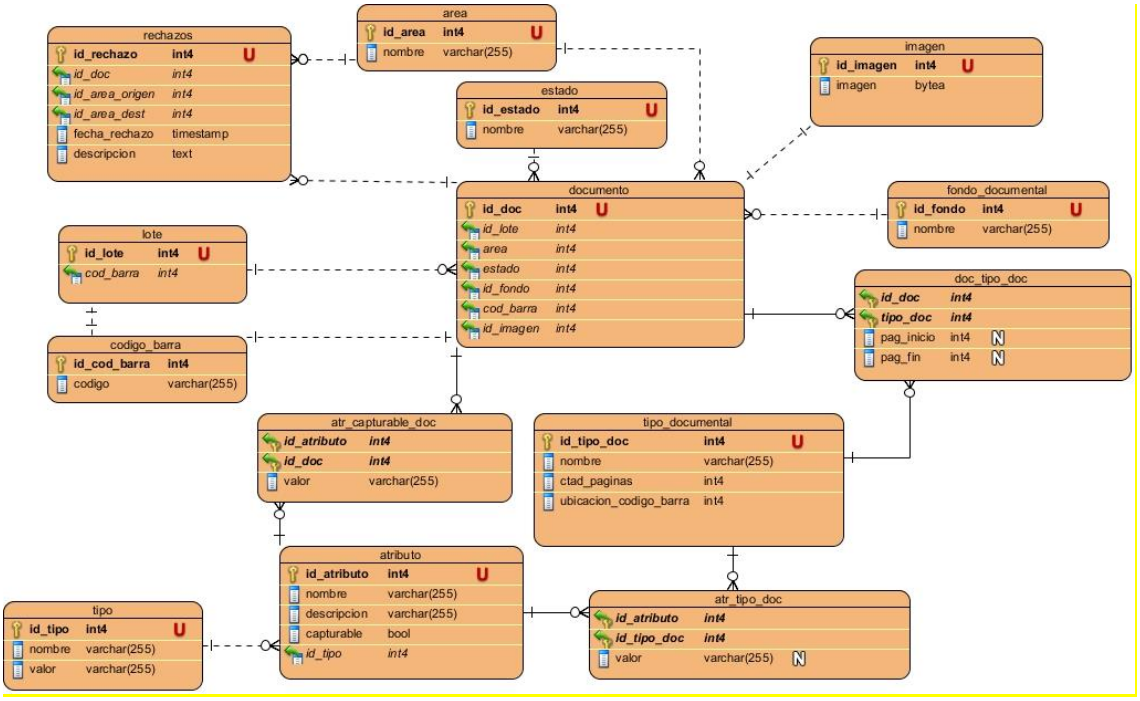


Figura 3: Modelo de datos.

Las actividades de Análisis y Diseño fueron esenciales en el proceso de desarrollo de software. Se realizaron con el objetivo de permitir la implementación del sistema de manera precisa. La descripción de las actividades realizadas durante la implementación se presenta en el epígrafe que aparece a continuación.

2.5. Implementación

2.5.1. Diseño Componentes

El diseño de componentes permite traducir el modelo de diseño en un software operacional. En este trabajo de tesis el componente reporte se encarga de informatizar todo el proceso de generación de informes, documentos y tablas dinámicas. El modelo de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan e interactúa. Para dar solución arquitectónica al componente se definen elementos siguiendo la lógica del marco de trabajo XEGFORT, creando nuevos paquetes con el nombre reporte en cada nivel, es decir se crean en capa de negocio del cliente y en la capa de negocio del servidor.

Dentro del cliente este paquete contiene las acciones, formularios y los gestores del negocio del cliente de este componente. Cada uno de estos elementos se comunica permitiendo mostrar los distintos reportes. En cuanto a la capa de negocio del servidor, se encuentran las entidades persistentes, los gestores del negocio del servidor, los convertidores y clases persistentes DAO. Estos elementos permiten enviar la información consultada en las bases de datos hacia el cliente.

A continuación, se muestra un fragmento del diagrama de componente del RF1: Visualizar documentos pendientes en digitalización. Puede encontrar la imagen completa en los anexos figura 12.

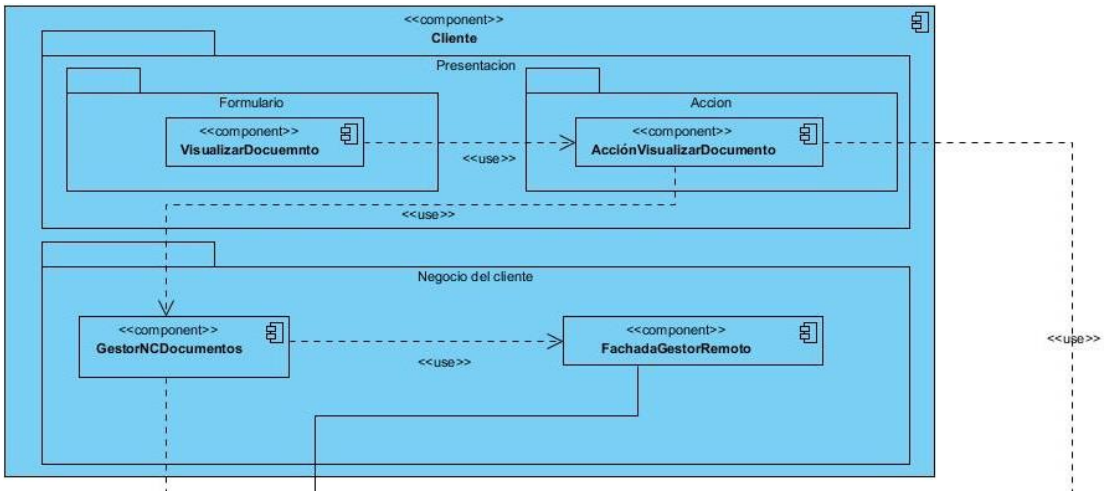


Figura 4: Fragmento del diagrama de componente del requisito funcional número 1.

2.5.2. Estándares de codificación.

El estándar de codificación empleado en el desarrollo del componente fue definido por el equipo de desarrollo. A continuación, se muestran algunas pautas de dicho estándar.

- Todas las nomenclaturas a utilizar se definirán en idioma español.
- Los nombres de los paquetes y las clases serán con mayúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán de igual forma.
- Los nombres de los métodos serán con minúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán con mayúscula.
- Las clases gestoras de negocio comienza con el prefijo Gestor y luego el nombre de la clase (GestorNSReporte.java).

Las clases de acceso a datos comienzan con el prefijo P, el nombre de la clase y luego con el prefijo DAO (PDocumentoDAO.java).

2.5.3. Implementación del componente.

La implementación del componente se rige por la arquitectura del marco de trabajo XEGFORT permitiendo una mejor organización del código, mejor comprensión y lógica del negocio. En esta sección se explicará el contenido de las carpetas utilizadas o creadas que complementan la solución propuesta.

Anteriormente se mencionó que en cada capa se encuentra un paquete con el nombre Reporte el cual contiene las clases y sus características. Siguiendo el ejemplo del requisito funcional número uno se describirá a continuación cómo está estructurado la implementación del componente.

En la parte del cliente existen dos paquetes: vista y gestores. Dentro de los gestores existe un paquete Reporte, el cual contiene la clase Gestores del negocio del cliente reporte, mientras que en la vista existen dos paquetes más: formulario y acciones. Formulario tiene un paquete reporte, este almacena lo formularios de los reportes, es decir las interfaces de usuario. Y en el paquete Acciones, se encuentran las acciones que van a realizar los formularios q están dentro del paquete reporte.

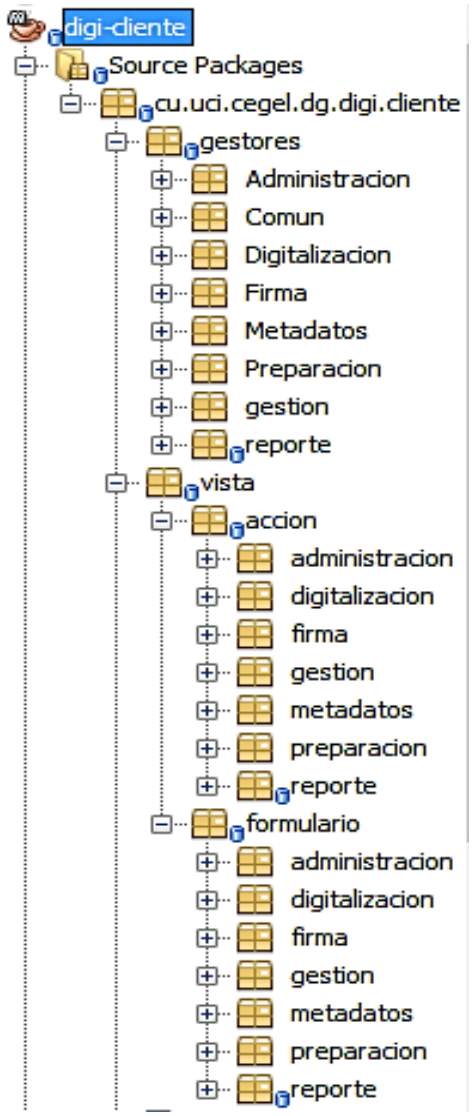


Ilustración 1: Estructura de paquetes para la implementación del componente.

La imagen anterior ilustra la estructura y organización del código. Dentro de cada paquete reporte se tiene distintas tareas y funcionalidades que permiten el buen funcionamiento del componente. Seguidamente se explica de forma detallada el interior de uno de los paquetes reporte.

cu.uci.cegel.dg.digi.cliente.Accion.Reporte: Describe la dirección que define la herramienta NetBeans para identificar en el paquete que se está trabajando. Dentro de este paquete están las acciones que describen mediante el código que van hacer los formularios creados.

2.5.4. Seguridad del sistema.

Un aspecto importante de un software es la integración de la seguridad, en la que diferentes programas desarrollados utilizando diversas tecnologías compartirán la misma infraestructura de seguridad, incluyendo autenticación, administración de acceso, aprovisionamiento de usuarios y SSO[25]. El modelo de seguridad Java EE proporciona los conceptos básicos para diseñar e implementar un modelo de seguridad completamente configurable, una amplia gama de interacciones con otros socios y sistemas de software y un modelo complejo de acceso basado en reglas, proporcionados por una línea de software denominada Identity Management (IDM por su sigla en inglés, Sistemas de Gestión de Identidades) o Sistemas de Gestión de Acceso.

Los sistemas de gestión de identidades cubren todos los requisitos relacionados con la seguridad que comienzan con la definición del usuario, la autenticación, la definición de reglas de acceso complejas, la audición, la seguridad de los Servicios Web, entre otros. Para ello existen diferentes tipos de agentes para proteger diferentes tipos de servidores pues son los encargados de hacer de un sistema más seguro. Estos agentes trabajan con una amplia gama de modos para integrarse con diferentes tipos de servidores y proteger diferentes tipos de recursos en esos servidores. Uno de los utilizados es el Glassfish, el cual puede operar en tres modos para satisfacer diversos requisitos.

En conjunto con el Glassfish se hace uso de varias técnicas de seguridad como: la autenticación y autorización y el bloqueo de sección con el uso del marco de trabajo XEGFORT. El sistema DIGILEX es desarrollado a través de este marco de trabajo, por lo que el generador de reporte al ser un componente que se integra en el sistema se rige por las mismas políticas y técnicas de seguridad de dicho sistema.

En general la seguridad del componente fue establecida en profundidad y por niveles siguiendo un conjunto de buenas prácticas para el desarrollo de aplicaciones seguras como definir las metas de seguridad del producto desarrollado y considerar la seguridad como una funcionalidad del producto.

2.5.5. Pruebas

Las pruebas de software realizadas estuvieron enmarcadas en detectar la mayor cantidad de defectos de software posibles para su debida corrección. Se realizaron una serie de actividades para

[25] *SSO: Single Sign-On es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.*

encontrar posibles fallos de implementación, calidad o usabilidad del sistema, probando el comportamiento del componente.

Las pruebas que se le realizaron al componente de Reportes fueron las establecidas por el programa de mejora: Pruebas Internas y Pruebas de Liberación. Las Pruebas Internas se realizaron para probar desde funcionalidades específicas hasta la versión final y se desarrollaron artefactos de pruebas como los Casos de Pruebas (CP), se hicieron las pruebas de liberación. Para consultar los resultados alcanzados en las pruebas de software ver Capítulo 3.

2.6. Conclusiones parciales

- Se realizó el levantamiento de los requisitos del componente que permitió cumplir con todas las funcionalidades requeridas para el proceso de generación de Reportes de Información.
- La utilización de los patrones de diseño y arquitectura durante el desarrollo del módulo, proporcionaron una mayor calidad del producto
- El diseño del modelo de datos permitió conocer las relaciones existentes entre las diferentes tablas de la base de datos.
- El diagrama de componentes facilitó la fase de implementación.
- La utilización de los estándares de implementación a utilizar, facilitaron el entendimiento del código por los programadores y el futuro mantenimiento del componente.
- La seguridad del sistema se estableció como funcionalidad del producto.

CAPÍTULO 3. VALIDACIÓN DE LA PROPUESTA.

3.1. Introducción

En el presente capítulo se realizarán las validaciones correspondientes a las fases propuestas por el modelo de desarrollo como buenas prácticas del desarrollo de software Se validarán las variables dependiente e independiente a través de métricas Se cuantificarán los resultados obtenidos y se graficarán con el objetivo de su apreciación visual.

3.2. Validación de fases de desarrollo

Con el objetivo de obtener un producto de elevada calidad, mantener un proceso estable y continuo de desarrollo, para cometer el menor número no conformidades posibles y con el propósito de lograr que las salidas de cada etapa de desarrollo que se conviertan en entradas a la siguiente etapa se realizaron las validaciones de cada una de las fases del ciclo de desarrollo. A continuación, validación de la modelación del negocio y los requisitos de software.

3.2.1. Validación de la modelación del negocio y los requisitos de software

La revisión técnica formal es una de las técnicas que se utiliza. Esta técnica se encarga de hacer las correcciones a cada descripción de los requisitos una vez terminada. Durante la revisión se detectan errores, los cuales eran corregidos para comenzar una nueva revisión. Se logra con este mecanismo validar que la interpretación de cada una de las descripciones no fuera ambigua, ni tuviese omisiones o errores y además que cada uno de los requisitos cumplía con las necesidades del cliente.

Otra técnica que se utilizada es los prototipos de interfaz de usuario. Estos permiten hacer simulaciones del componente a implementar y les permite a los especialistas tener una idea de cómo serán las interfaces del componente. Los prototipos se realizan de forma no funcional con las herramientas Visual Paradigm for UML y en ocasiones se utiliza el NetBeans debido que este último provee una herramienta la cual se puede construir de forma visual los prototipos con el objetivo de lograr la aprobación del cliente.

3.3. Validación de las variables de la investigación.

Para la validación del Componente de Reportes (Variable independiente) se realizó un estudio de las principales técnicas utilizadas para dar solución al problema inicial, se hizo uso de métricas para medir

la reutilización, limpieza y buenas prácticas de diseño. Estas métricas sirvieron de instrumentos para cuantificar criterios sobre la calidad del producto.

Las métricas estudiadas abarcan atributos de calidad que permiten medir la calidad del diseño propuesto. Estas métricas concebidas para evaluar la calidad del diseño se seleccionaron teniendo en cuenta las características propias del componente. Estas son: Alcance y Calidad que permitirán la validación de la variable independiente.

El alcance del producto a desarrollar fue determinado por el cumplimiento de los requisitos. Para ello se calculó del Índice de requisitos implementados. A continuación, se desarrollan los cálculos pertinentes.

- Cantidad de requisitos implementados (CRI) = 20
- Cantidad total de requisitos (CTR) = 20
- Índice de requisitos implementados(IRI) = CRI/CTR=20/20=1

A continuación, se presenta el análisis gráfico correspondiente a los cálculos realizados.

Figura 5: Gráfica de índice de requisitos implementados.

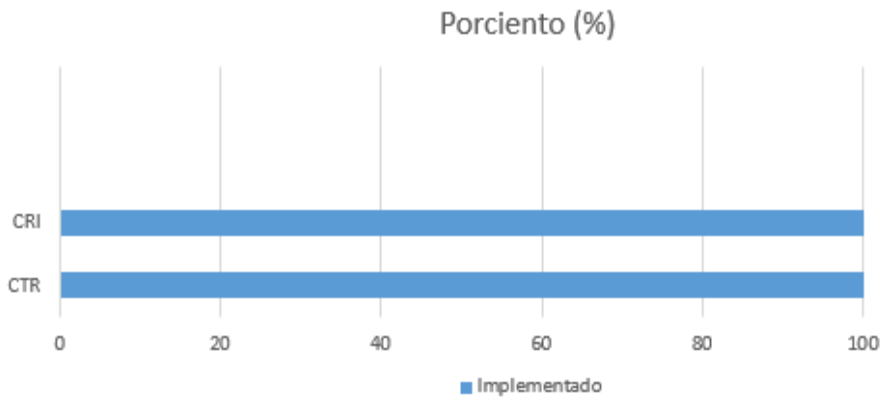


Figura 6: Grafica de porciento de Requisito implementados.

Para evaluar la Calidad del producto desarrollado se tomaron como indicadores la Calidad Interna y Externa. La Calidad Interna será cuantificada con el Índice de no conformidades detectadas en las Pruebas Internas y por los compañeros de calidad durante las pruebas de Liberación. En el siguiente epígrafe se hacen las pruebas de caja blanca y caja negra las cuales permiten medir la calidad interna y externa.

3.4. Validación del diseño.

La validación del diseño e implementación se realizó mediante la Métrica: Complejidad de Implementación, donde se cuantificó mediante el uso de la técnica: Tamaño Operacional de Clase (TOC) propuesta por Lorenz y Kidd. El tamaño general de las clases se determinó mediante las siguientes medidas:

- El total de operaciones (TO), tanto heredadas como privadas de la instancia, que se encapsulan dentro de la clase.
- El número de atributos (NA), tanto heredados como privados de la instancia, encapsulados por la clase.

Medidas para las principales clases de la solución.

La métrica de TOC fue aplicada a un total de 10 clases para un total de 58 atributos y un promedio (P) de atributos de 5.8. Para un total de 102 operaciones y un promedio operacional de 10.2.

Responsabilidad y Complejidad de Implementación (Cálculo del umbral)

(NA) Baja: P <= 5.8 5.8 < Media <= 10.8 Alta > 10.8 (TO) Baja: P <= 10 10< Media <= 20 Alta > 20

Atributo	Clases	NA	TO	Tamaño (NA)	Tamaño(TO)

	AccionMostrarDocumentos	6	10	media	media
RCI	AccionMostrarCantDocAreasxUsuario	6	10	media	media
	AccionMostrarCantDocDigixUsuarioDeterminado	6	10	media	media
	AccionMostrarCantidadFoliosFondoDocu	6	9	media	media
	AccionMostrarCantidadFolioxDocumento	6	9	media	media
	AccionMostrarDocumentosporLotes	6	9	media	media
	AccionMostrarReporteDinamico	11	12	alta	alta
	AccionVisualizarDocumentosporTipoDocumental	6	9	media	media
	AccionFiltrarComponentes	5	7	media	media
	GestorNCReporte	0	17	media	media

Tabla 6: Calculo del TOC para los atributos Responsabilidad y complejidad de implementación.

Reutilización (Cálculo del umbral)

(NA) Baja: P <= 5.8 5.8 < Media <= 10.8 Alta > 10.8 (TO) Baja: P <= 10 10< Media <= 20 Alta > 20

Atributo	**Clases**	**NA**	**TO**	**Tamaño (NA)**	**Tamaño(TO)**
	AccionMostrarDocumentos	6	10	media	media
Reutilización	AccionMostrarCantDocAreasxUsuario	6	10	media	media
	AccionMostrarCantDocDigixUsuarioDeterminado	6	10	media	media
	AccionMostrarCantidadFoliosFondoDocu	6	9	media	media
	AccionMostrarCantidadFolioxDocumento	6	9	media	media
	AccionMostrarDocumentosporLotes	6	9	media	media
	AccionMostrarReporteDinamico	11	12	alta	alta
	AccionVisualizarDocumentosporTipoDocumental	6	9	media	media
	AccionFiltrarComponentes	5	7	media	media
	GestorNCReporte	0	17	media	media

Tabla 7: Calculo del TOC para el atributo Reutilización.

Resultados obtenidos de la aplicación de la métrica al componente

Con la representación de los resultados obtenidos en un gráfico de por ciento se obtuvo que: del total de clases analizadas en cuanto a los atributos Responsabilidad y Complejidad de Implementación, el 90% presenta un TOC (NA) mediano y el 90% un TOC (TO) en igual calificación. De tamaño bajo se encuentran no se encuentra ninguna. En cuanto a calificación como alta, del total, el TOC (NA) cuenta con 10% y 10% el TOC (TO). Sobre el atributo Reutilización se tiene que la calificación media se comporta de igual manera e los atributos anteriores y la clasificación alta también, la baja sigue con valor 0%.

El análisis de los datos anteriormente planteados permite concluir que el componente presenta una reutilización alta, mientras que la Responsabilidad de las clases y Complejidad de Implementación se comportan de manera media. Estos datos favorecen el buen uso del diseño empleado y el resultado de la implementación.

3.5. Prueba

La prueba es un conjunto de actividades que pueden ser planeadas con antelación y realizarse de manera sistemática y durante la implementación o finalizar el desarrollo del proyecto. Por esta razón, durante el proceso de software, se debe definir las prueba del software: un conjunto de pasos que incluyen métodos de prueba y técnicas de diseño de casos de prueba específico (R. S. Pressman 2010). Al aplicar las pruebas de software el producto tiende a tener una mayor calidad porque pasa por un conjunto de pruebas que permiten validar y verificar el cumplimiento de los objetivos trazados y el buen funcionamiento de la aplicación.

3.4.1. Prueba unitaria

Las pruebas unitarias, se aplicaron al componente por el método de caja blanca. Los métodos de pruebas de caja blanca garantizan que se ejerciten todos los caminos independientes de cada módulo, así como la ejecución de todos los bucles y las estructuras de datos internas (R. S. Pressman 2010).

2.4.1.1. Prueba de caja blanca

Se aplica al método iniciarFormulario () de la clase AcciónMostrarDocumento la prueba de caja blanca mediante la técnica camino básico. El método llenartabla () permite iniciar un formulario con todas las funcionalidades activas. A continuación, el método anteriormente mencionado:

```
private void llenarTabla() {
    try {
        if (((EDArea) getFormulario().cmbArea.getSelectedItem()).getIdArea()!= -1
                ||((EDEstado)getFormulario().cmbEstado.getSelectedItem()).getIdEstado()!= -1 ) {//1

        List<EDDocumento> listdocu = new GestorNCReporte().mostrarDocumentosAreaEstado(((EDArea) getFormulario().cmbArea.getSelectedItem()).getIdArea(),
                ((EDEstado)getFormulario().cmbEstado.getSelectedItem()).getIdEstado());//2
        if (!listdocu.isEmpty()) {//3
            getFormulario().documentos.setFuenteDatos(listdocu);//4
        } else {//5
            GestorMensajes.MensajeAviso("La búsqueda no arrojó ningún resultado.");//6
        }
       }else {//7
            GestorMensajes.MensajeAviso("Debe seleccionar el area y el estado para realizar la busqueda.");//8
        }
    } catch (Exception ex) {//9
        Logger.getLogger(AccionMostrarDocumentos.class.getName()).log(Level.SEVERE, null, ex);//10
    }
}
```

Figura 7: Método Llenar tabla.

A continuación, se muestra los pasos para realizar la técnica de camino básico:

- **Confeccionar el grafo de flujo:** usando el método de la figura anterior, se realiza la representación del grafo del flujo.
  - Nodos: son círculos que representan una o más sentencias procedimentales.
  - Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
  - Regiones: son las áreas delimitadas por aristas y nodos.

En la figura siguiente se muestra el grafo obtenido:

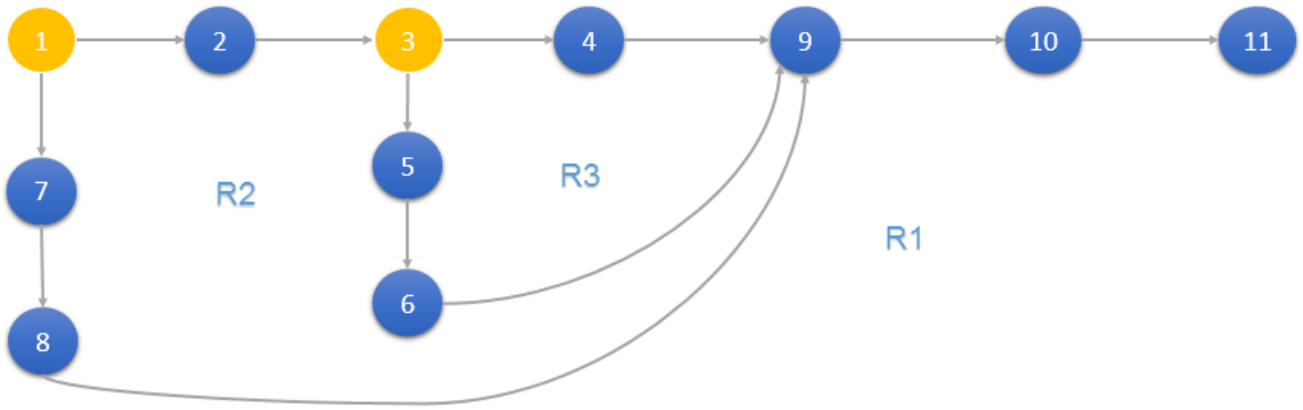


Figura 8: Grafo de flujo dl método llenar tabla.

En esta ilustración se muestran las regiones que se forman representados por la letra R seguido de un número (R1) y lo nodos q son predicados que están representado de color anaranjado.

Después de haber realizado el grafo, se calcula la complejidad ciclomática por tres fórmulas distintas, las cuales deben dar el mismo resultado para comprobar que el cálculo sea correcto.

- **Calcular la complejidad ciclomática:** proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, la complejidad ciclomática se calculó con las siguientes fórmulas:
  - ✓ V (G) = A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo. V (G) = 12 – 11 + 2 = 3.
  - ✓ V (G) = Regiones = 3.
  - ✓ V (G) = P + 1, donde P son los nodos predicados. V (G) = 2 + 1 = 3.

  V (G) es el valor que da el número de caminos linealmente independientes de la estructura de control del programa por lo que en el siguiente paso se determina los caminos básicos siguientes.
- **Determinar un conjunto básico de caminos linealmente independientes.**

No.	**Camino básico**
1	1-2-3-4-9-10-11
2	1-7-8-9-10-11
3	1-2-3-5-6-9-10-11

Tabla 8: Tabla de caminos básicos detectados a través del grafo de flujo.

Luego de haber aplicado el método anterior se comprobó que cada sentencia es ejecutada al menos una vez. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico (Pressman, 2002), lo que facilitó conocer el número de pruebas que se deben realizar. En el paso siguiente se aplicará los casos de pruebas realizados.

- **Obtención de casos de prueba (CP):**

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo. Para definir los casos de prueba es necesario tener en cuenta: descripción, condición de ejecución, entrada y resultados esperados.

Código: CPCB-01	
Descripción:	Obtiene la lista de documentos y la muestra mediante una la tabla.
Condición de evaluación:	Si la lista de documento no está vacía
Entrada:	El método recoge una lista de documentos extraída de una consulta SQL desde la clase EDDocumentoDAO.
Resultados esperados:	Muestra la lista de documentos en una tabla.
Evaluación de prueba: Satisfactoria.	

Tabla 9: Caso de prueba de camino básico número 1.

Código: CPCB-02	
Descripción:	Obtiene la lista de documentos y la muestra mediante una la tabla.
Condición de evaluación:	Si está vacía la lista lanza la excepción: No se encontraron resultados.
Entradas:	El método recoge una lista de documentos extraída de una consulta SQL desde la clase EDDocumentoDAO.
Resultados:	No se debe llenar la tabla y mostrar un diálogo con un mensaje
Evaluación de prueba: Satisfactoria.	

Tabla 10: Caso de prueba del camino básico número 2.

Resultado:

En una primera iteración se realizaron un total de 44 casos de pruebas de caja blanca, de los cuales resultaron satisfactorios 40, lo que representa el 95% del total de los casos de prueba.

3.4.2. Pruebas funcionales.

Las pruebas funcionales se encargan de probar la funcionalidad completa, donde pueden estar implicadas una o varias clases y hasta la propia interfaz de usuario.

3.4.2.1. Prueba de caja negra

Los métodos de pruebas de caja negra son pruebas funcionales que se realizan al software permiten derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Además, permiten detectar funcionamiento incorrecto o incompleto, errores en la interfaz, errores en la estructura de datos externa, problemas de rendimientos y errores de inicialización y terminación (Pressman, 2002).

Para la realización de las pruebas de caja negra se empleó la técnica partición de equivalencia. Esta técnica permite examinar los valores válidos e inválidos de las entradas existentes en el software. A continuación, es sometida a pruebas de aceptación HU-1 definidas en el capítulo anterior. A continuación, se muestra la tabla 7.

CPCB-01	Historia de usuario: HU-1	
Nombre: Visualizar documentos pendientes en digitalización		
Descripción**:** Muestra en una tabla los documentos que están pendientes a pasar por cada proceso en el área de digitalización. Dicha tabla contiene código de barra del documento, el fondo al que pertenece documento y el lote del documento.		
Acción Probar	Datos de entrada	Resultados esperados
Seleccionar un área y un estado.	Estado: pendiente. Área: digitalización	Muestra en una tabla todos los documentos que están pendientes en el área de digitalización
Evaluación de las pruebas: Satisfactoria		

*Tabla 11: Caso de prueba 01**:** Visualizar documentos pendientes en digitalización*

CPCB-02	Historia de usuario: HU-1
Nombre: Visualizar documentos pendientes en digitalización	
Descripción**:** Muestra en una tabla los documentos que están pendientes a pasar por cada proceso en el área de digitalización. Dicha tabla contiene código de barra del documento, el fondo al que pertenece documento y el lote del documento.	

Acción Probar	Datos de entrada	Resultados esperados
No selecciona el área ni el estado.	Estado: No seleccionado. Área: No seleccionado	Muestra una excepción: Debe seleccionar el estado y el área para realizar la búsqueda.
Selecciona un área o un documento.	Estado: pendiente Área: no seleccionado	Muestra una expresión: Debe seleccionar el estado y el área para realizar la búsqueda.
Evaluación de las pruebas: Satisfactoria		

Tabla 12: Caso de prueba 02: Visualizar documentos pendientes en digitalización.

Resultados.

Se realizaron un total de 44 casos de pruebas de caja negra, siendo encontrado en la primera iteración 24 no conformidades, lo cual representa el 54% del total de casos de prueba de caja negra realizados, mientras los 20 casos de prueba restantes resultaron satisfactorios para un 46% del total.

Luego para la segunda iteración se encontraron 12 no conformidades, lo cual representa el 27% de los casos de pruebas realizados, mientras que los 32 casos de pruebas restantes resultaron satisfactorios para un 73% del total.

La realizarse la 3era iteración se encontraron 2 no conformidades, lo cual representa el 4% del total de casos de pruebas analizados, siendo así 42 casos de pruebas con resultados satisfactorios lo cual representa el 96% del total. Ya para la última iteración no se obtuvieron no conformidades.

De esta manera quedan aplicadas las pruebas de caja negra y caja blanca, lo que garantiza que el software está en óptimas condiciones para ser usado por usuarios finales.

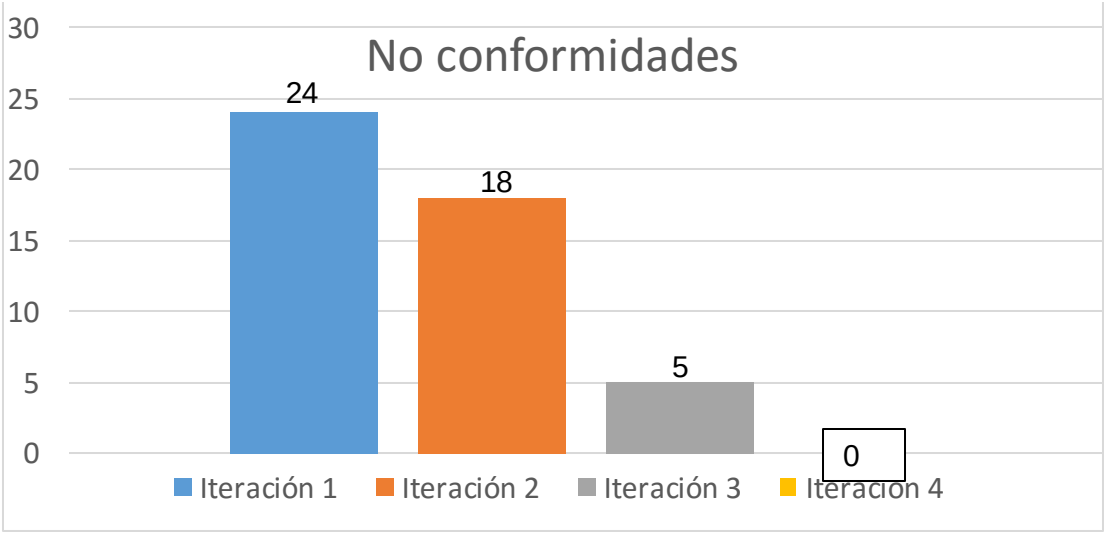


Figura 9: Grafica de no conformidades por iteraciones. (Elaboración propia)

3.6. Facilitar la obtención de datos

Para la validación de la variable facilitar la obtención de datos (variable dependiente) se les presentó el componente ya desarrollado a los especialistas del centro de digitalización, obteniéndose muy buena aceptación por parte de los mismos. Se pudo comprobar esto luego de hacer una pequeña entrevista a cada especialista del centro, para conocer sus impresiones, opiniones y sugerencias para reflejarlas en futuras versiones. La mayoría de los especialistas afirmaban que se sentían satisfechos con la herramienta, ya que permitía ahorrase el trabajo de cada vez que se hacía una acción sobre un lote o folio debía ser notificado a una persona encargada de armar el reporte auxiliándose de un Excel. Otros comentaban que la herramienta creada propiciaba mantener un control sobre los fondos documentales que eran procesados garantizando la organización a través de los reportes que se generaban. De esta manera se pudo evidenciar que el componente resolvía los principales problemas en el centro, como lo es la perdida de la información, el duplicado de la misma y la mala organización en lotes y folios.

3.7. Tiempo

Para darle cumplimiento al requisito no funcional del tiempo de respuesta del componente se pobló la base de datos hasta un valor de 1000 elementos en la tabla. De esta manera se mide los valores de los tiempos de ejecución de las consultas en la Base de Datos, donde se pudo apreciar que todas las consultas realizadas tardaban menos de 1 segundo en ejecutarse, comportándose como un tiempo favorable. Se midieron los tiempos de respuesta finales de los reportes incluyendo las consultas pertinentes a cada uno, en estas pruebas realizadas se constató que al igual que los anteriores, los tiempos de respuesta obtenidos fueron menores que 1 segundo. En cuanto a la generación de reportes a formato pdf, el tiempo que demora es menor o igual a 1 segundo, siendo así un tiempo satisfactorio aunque cuando se ejecuta por primera vez luego de haber sido iniciado el sistema, el tiempo de demora es menor de 2 segundos. Estos tiempos de respuestas dan la medida de cuán rápida es la obtención de la información de la base de datos.

3.8. Conclusiones parciales

En este capítulo se realizó la validación del componente, arrojando resultados importantes como:

- Se validaron las fases de desarrollo abarcadas durante el desarrollo del producto como buenas prácticas permitiendo obtener un producto de calidad.

- Se realizaron las pruebas de software, arrojando resultados satisfactorios en el desarrollo del componente de reporte.
- Se realizó la validación de la variable independiente mediante las métricas de Alcance y Calidad.
- Se realizó la validación de la variable dependiente mediante la evaluación de los tiempos de respuesta de los reportes.

CONCLUSIONES GENERALES

Con la culminación del presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir con el objetivo propuesto, por lo que se logra:

- Realizar el marco teórico referente a la investigación donde se establecieron las tendencias marcadas en cuanto al desarrollo de reportes de información y permitió fundamentar la necesidad de desarrollar el módulo.
- Se desarrolló el Componente para el proyecto DIGILEX abarcando las etapas establecidas por la metodología AUP versión UCI. La implementación del Componente de Reportes se realizó utilizando las herramientas, lenguajes y tecnologías definidas por el modelo utilizado.
- Las pruebas realizadas arrojaron resultados satisfactorios, demostrando la calidad del componente desarrollado, mientras que el módulo fue validado mediante el uso de métricas y técnicas aplicables al desarrollo de la solución.

RECOMENDACIONES

Se recomienda para este trabajo de diploma, para una nueva versión crear reportes que sean capaz de mostrar gráficas. Esto posibilitará una mejor representación del estado de la digitalización de un fondo documental, además permitirá una mejor compresión de la información obtenida y ayudará en la toma de decisiones, dentro del centro de digitalización. Para ello se recomienda hacer un estudio de la librería Jfreechart la cual puede fusionarse con la librería utilizada en este trabajo Jasper Report.

REFERENCIAS BIBLIOGRÁFICAS

Abreu Medina, Aldis Joan, Yasmany Hernández Hernández, Miguel Lezcano Ramos, y Aurelio Rodríguez Durán. 2012. «Generador Dinámico de Reportes».

Alegsa, Leandro. 2016. «Definición de Framework de desarrollo (informática)». http://www.alegsa.com.ar/Dic/framework.php.

Almansa Marín, y Antoni. 2016. «Selección de herramientas de análisis de datos para medir indicadores de rendimiento clave en un proyecto de una empresa desarrolladora de software .»

Blanco Cala, Alberto. 2014. «Diseño e implementación de la Base de datos para el Subsistema Base del Centro de Digitalización de la plataforma DigiPRO».

Carabia, Luis, y Pablo Píriz. 2003. «Metodología XP». Universidad ORT.

Gavio, Bomate, Rafael Felipe, Yenia Román Bu, Cinthya Rodríguez Hernández, Carlos Manuel Delgado Rivero, y Manuel Cortés Cortés. 2014. «GeReport: Sistema de Gestión de Reportes Dinámicos.» *Revista Cubana de Ciencias Informáticas* 8 (4). http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=19941536&AN=99861649&h=W6zkXJ4OoyG7DgKZ5g0j3a4y6Z9DQOMxjvbmH94DO%2B7yFNQiUqaOfeaQ0Ud0ozCBilHanavt0sI1v%2BEeNfa%2FjA%3D%3D&crl=c.

Gómez Martínez, Greter, y Anchel Durán Bolaño. 2015. «Biblioteca de componentes de interfaz de usuario para aplicaciones de escritorio desarrolladas en Java». http://repositorio.uci.cu/jspui/handle/123456789/7249.

Gonzalez valdez, Sandra. 2011. «Diseño e implementción de los módulos de Presentación del Centro de Digitalización para la división de antecedentes Penales». la habana.

GrapeCity. 1997. «ActiveReports and ActiveReports Server Documentation | GrapeCity ActiveReports - Documentation». http://activereports.grapecity.com/documentation/.

Heffelfinger, David R. 2007. *Java EE 5 Development Using GlassFish Application Server: The Complete Guide to Installing and Configuring the GlassFish Application Server and Developing Java EE 5 Applications to be Deployed to this Server*. Packt Publishing Ltd. https://books.google.es/books?hl=es&lr=&id=8GWHF6f40yYC&oi=fnd&pg=PT9&dq=GlassFish+Server+Open+Source+Edition&ots=SdSvP-f7us&sig=fjkMeBwFmSMZrK-3_oq2Fp8knuI.

Keegan, Patrick, Ludovic Champenois, Gregory Crawley, Charlie Hunt, y Christopher Webster. 2006. *NetBeans (TM) IDE Field Guide: Developing Desktop, Web, Enterprise, and Mobile Applications*. Prentice Hall PTR. http://dl.acm.org/citation.cfm?id=1207971.

Kidd, Jeff, y Mark Lorenz. 1994. *Object-oriented software metrics: a practical guide*.

REFERENCIAS BIBLIOGRÁFICAS

Mazón Olivo, Bertha Eugenia, Cartuche Calva, J. Joffre, Chimarro Chipantiza, L. Víctor, y Wilmer B. Rivas Asanza. 2015. *Fundamentos de programación orientada a objetos en JAVA.* Machala: Ecuador. http://repositorio.utmachala.edu.ec/handle/48000/6746.

«Objeto digital : Vicerrectorado de las Tecnologías de la Información y de las Comunicaciones : UPV». 2017. Accedido junio 17. http://www.upv.es/entidades/VTIC/info/524234normalc.html.

Pressman, Roger S. 2010. *Software Engineering. A PractiTiones Approach*. Seven Edition. Nueva York.

Pressman, Rogers. 2007. *Ingenieria de Software .Un enfoque práctico*.

Pupo Acosta, José Carlos. 2011. «Diseño de implementación del módulo de Impkementación de Metadatos del Sistema DigiPyrus de los Registros y Notarías de la República Bolivariana de Venezuela».

«Que es digitalizar un documento. Proceso de digitalización de documentos. | TBS-Telecon». 2017. Accedido junio 11. http://www.tbs-telecon.es/que-es-digitalizar-un-documento-proceso-digitalizacion-documentos.

«¿Qué es un Fondo Documental? - Archivo Nacional». 2017. Accedido junio 11. http://www.archivonacional.cl/616/w3-article-10983.html.

R. Heffelfinger. 2011. «JasperReports for Java Developers».

Reynoso, Carlos Billy. 2004. «Introducción a la Arquitectura de Software». *Recuperado de: http://carlosreynoso. com. ar/archivos/carlos-reynoso-introduccion-a-la-arquitectura-de-software. pdf*. http://carlosreynoso.com.ar/archivos/arquitectura/Arquitectura-software.pdf.

Robinson, Mark. 2012. «Microsoft SQL Server 2005 Reporting Services».

Rodriguez Lopez, Alien, José Angel Martinez Garcia, y Sergio Hernandez Ciseros. 2011. «Manual de usuarios: Sistema para la Digitalización de Alfabética 1.0».

Rodríguez Sánchez, 9. Tamara DE MEJORA. 2015. «Metodología de desarrollo para la Actividad productiva de la UCI.»

Sommerville, Ian. 2011. *Ingeniería del software*. Pearson Educación. https://books.google.es/books?hl=es&lr=&id=gQWd49zSut4C&oi=fnd&pg=PA1&dq=ingenieria+de+software+.+Un+enfoque+pr%C3%A1ctico&ots=s689qoxxrd&sig=9LmBpneNlKxdt6BbBODh9ww2Y5U.

TIOBE Index. 2017. «TIOBE Index | TIOBE - The Software Quality Company». https://www.tiobe.com/tiobe-index/.

Visual Paradigm. 2013. «Visual paradigm for uml». *Visual Paradigm for UML-UML tool for software application development*, 72.

Zapata, Javier. 2013. «Niveles de pruebas». *PRUEBAS DE SOFTWARE*. http://ingenieriadesoftwareaplicada.wordpress.com.

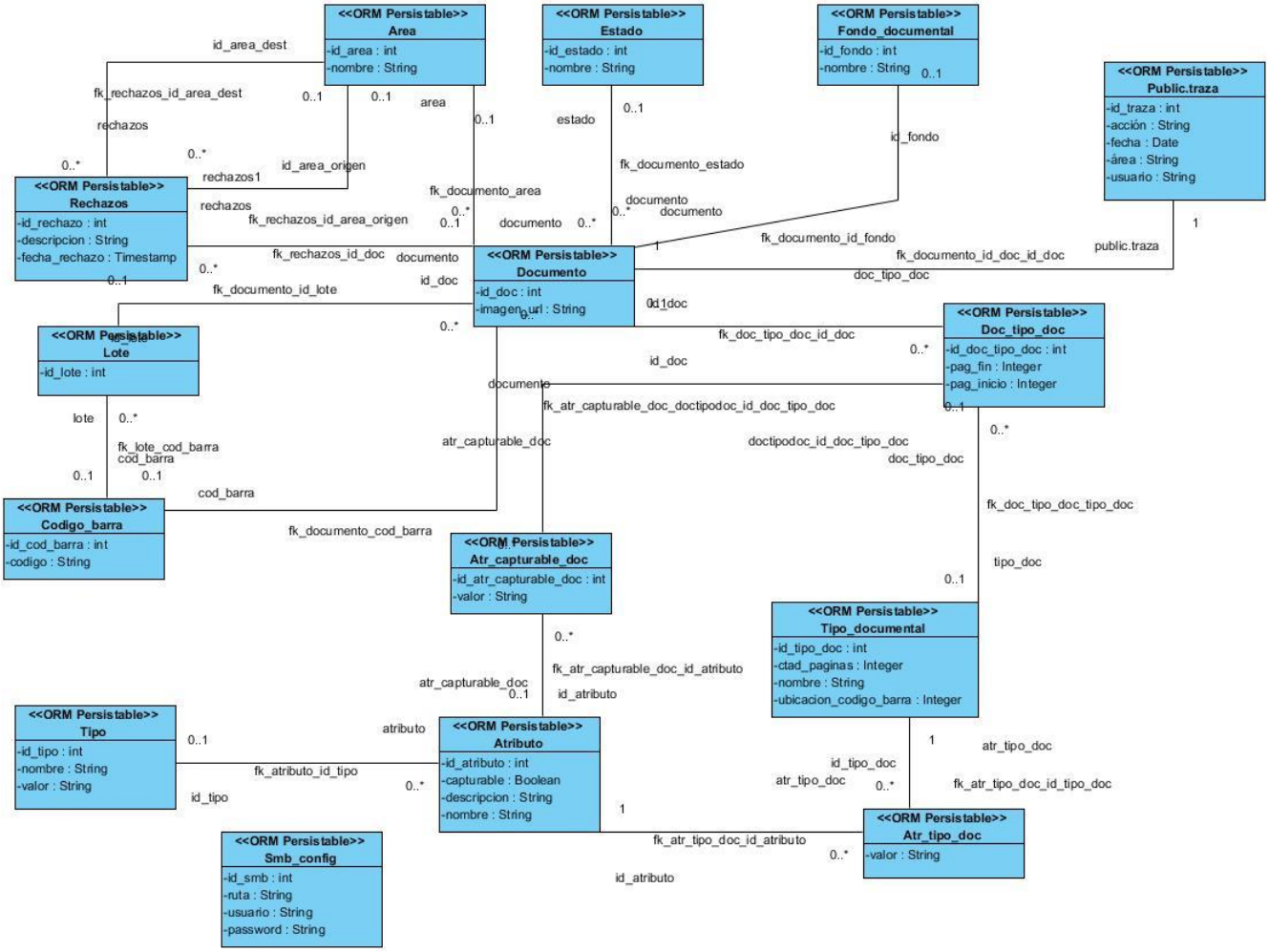


Figura 10: Diagrama de Clase del diseño.

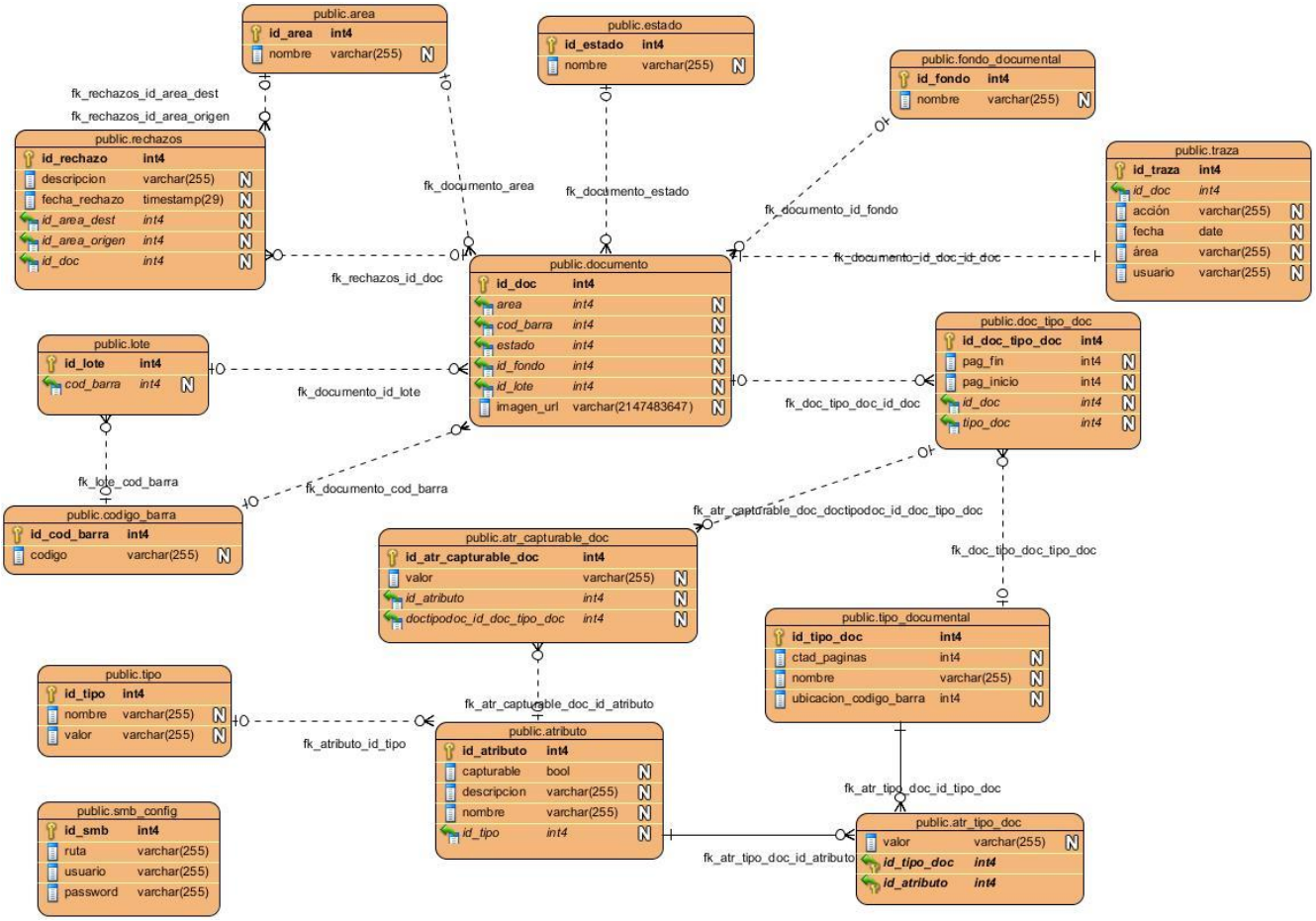


Figura 11: Diagrama de modelo de bases de datos.

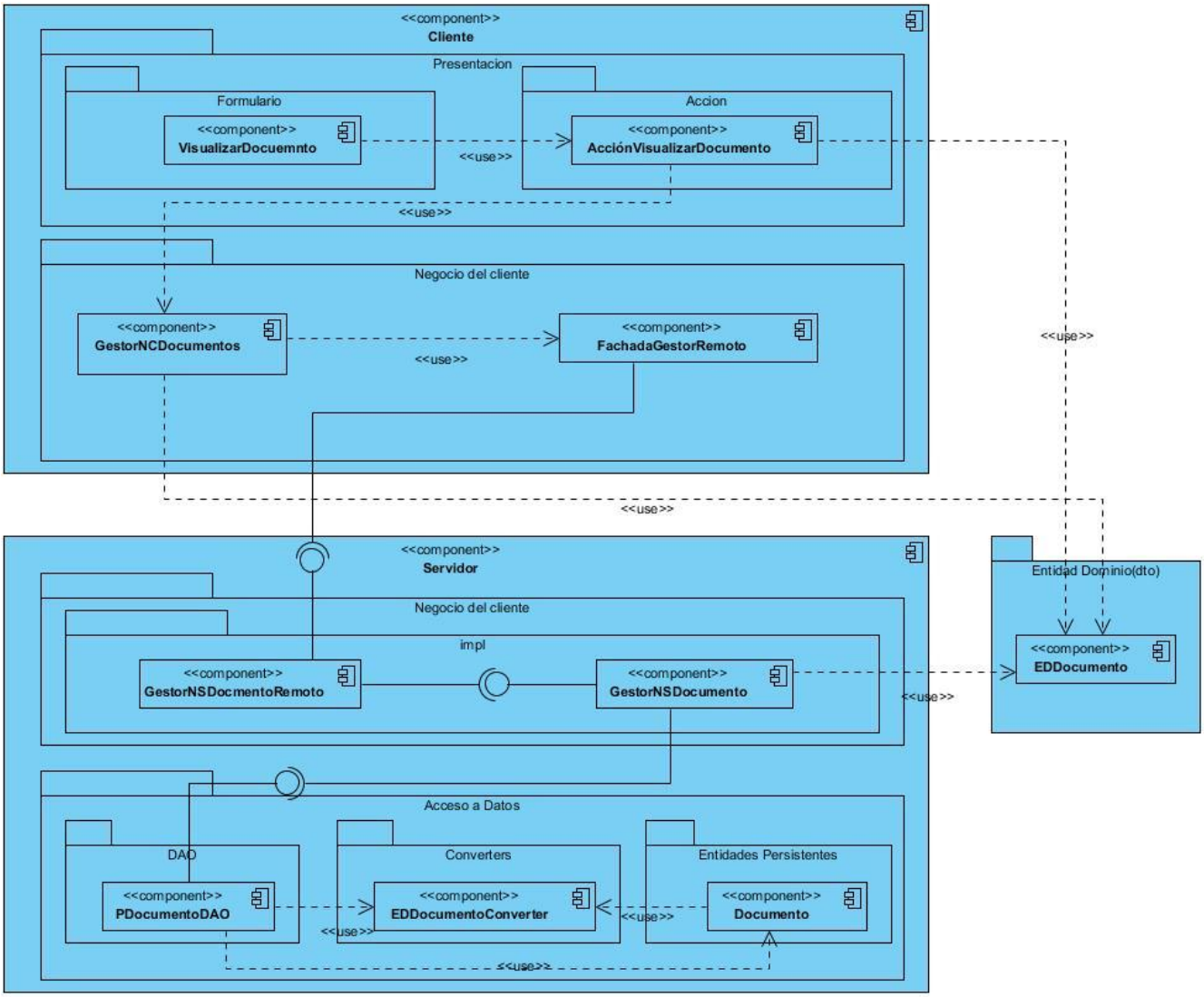


Figura 12: Diagrama de Componente.

Número: 11	**Nombre del requisito:** Visualizar cantidad de documentos por lotes
Programador: 1	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2h
Riesgo en Desarrollo: alta	**Tiempo Real:** 4h
Descripción: Muestra en una tabla los documentos según el lote especificado. Esta tabla contiene, además, código de barra del documento, estado y el área.	

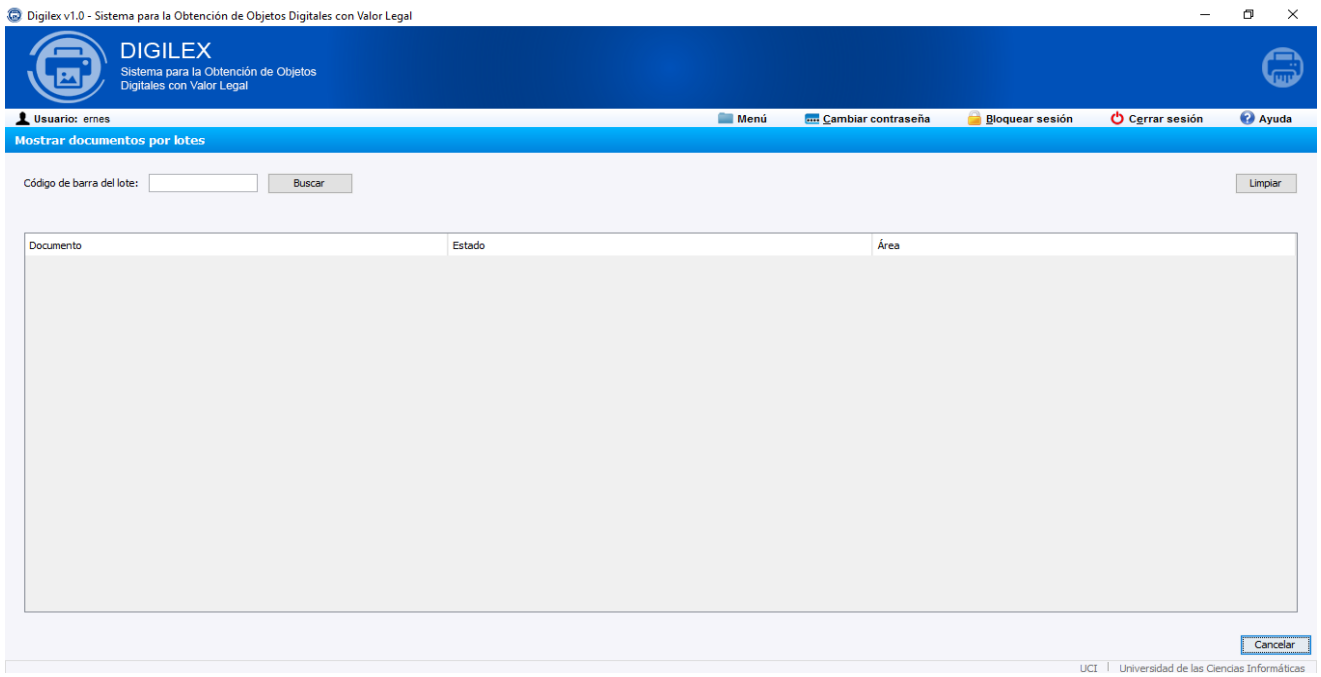
Observaciones: Debe insertar el código de barra del lote.
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 13: HU Visualizar cantidad de documento por lotes

Número: 12	**Nombre del requisito:** Visualizar cantidad de folios por documentos
Programador: 1	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2h
Riesgo en Desarrollo: alta	**Tiempo Real:** 4h
Descripción: Muestra la cantidad de folios por documentos en una tabla. La tabla proporciona información la cual es, código de barra del documento, la cantidad de folios y el código de barra del lote al que pertenece dicho documento.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

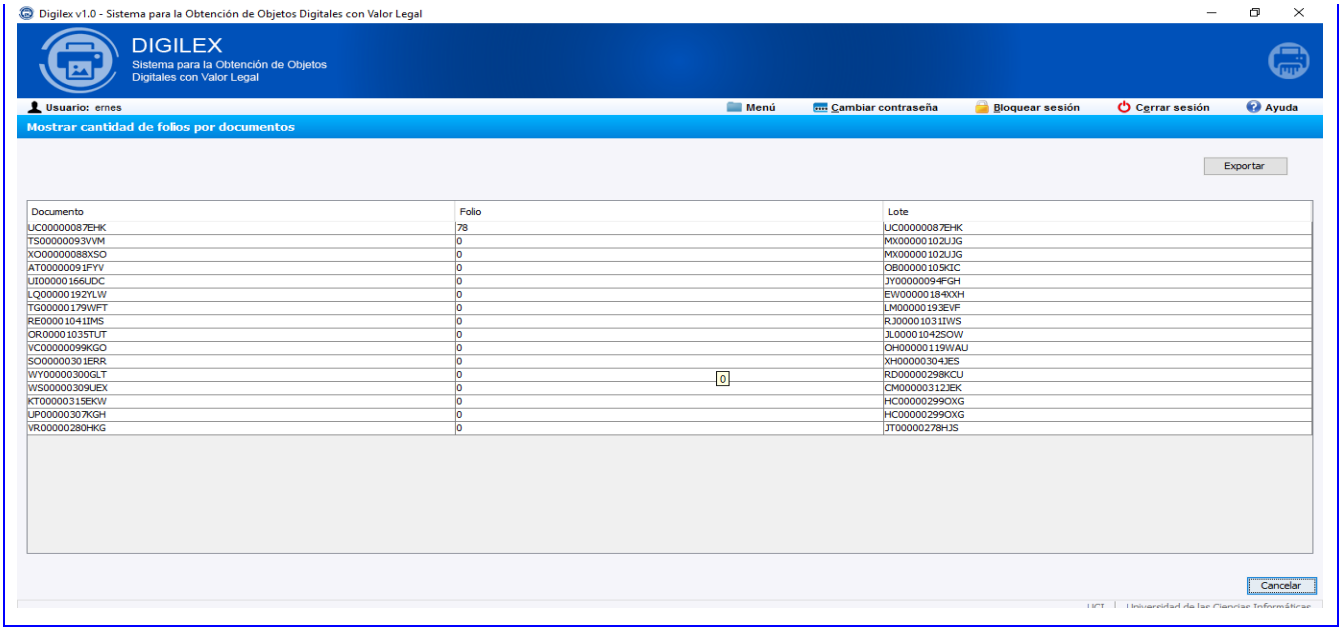


Tabla 14: HU Visualizar cantidad de folios

Número: 14	**Nombre del requisito:** Visualizar cantidad documentos digitalizados por un usuario determinado
Programador: 1	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2h
Riesgo en Desarrollo: alta	**Tiempo Real:** 4h
Descripción: Muestra en una tabla la cantidad de documentos que han sido digitalizados por un usuario específico. Dicha tabla contiene código de barra del documento, el estado del documento y el área del documento.	
Observaciones: Debe inserta un usuario para realizar la búsqueda	
Prototipo elemental de interfaz gráfica de usuario:	

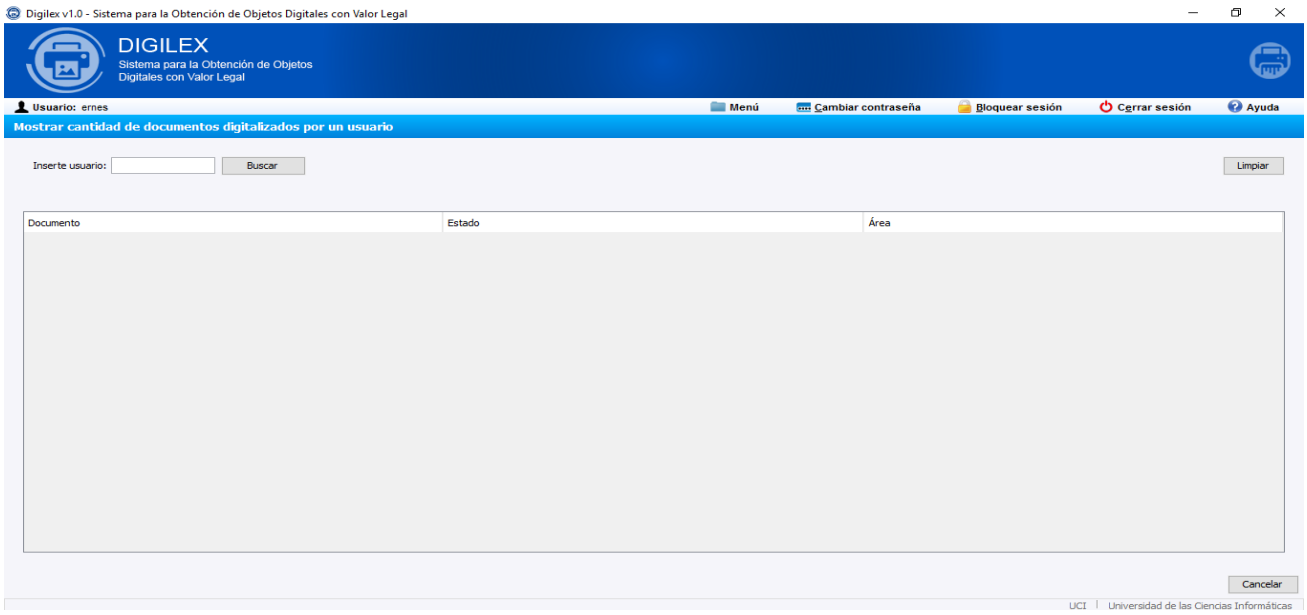


Tabla 15: Visualizar cantidad de documentos digitalizados por un usurario determinado

Número: 20	**Nombre del requisito:** Generar reportes
Programador: 1	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2h
Riesgo en Desarrollo: alta	**Tiempo Real:** 4h
Descripción: Permite generar los distintos reportes por cada criterio que se le especifique. En la tabla muestra toda la información de los documentos.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

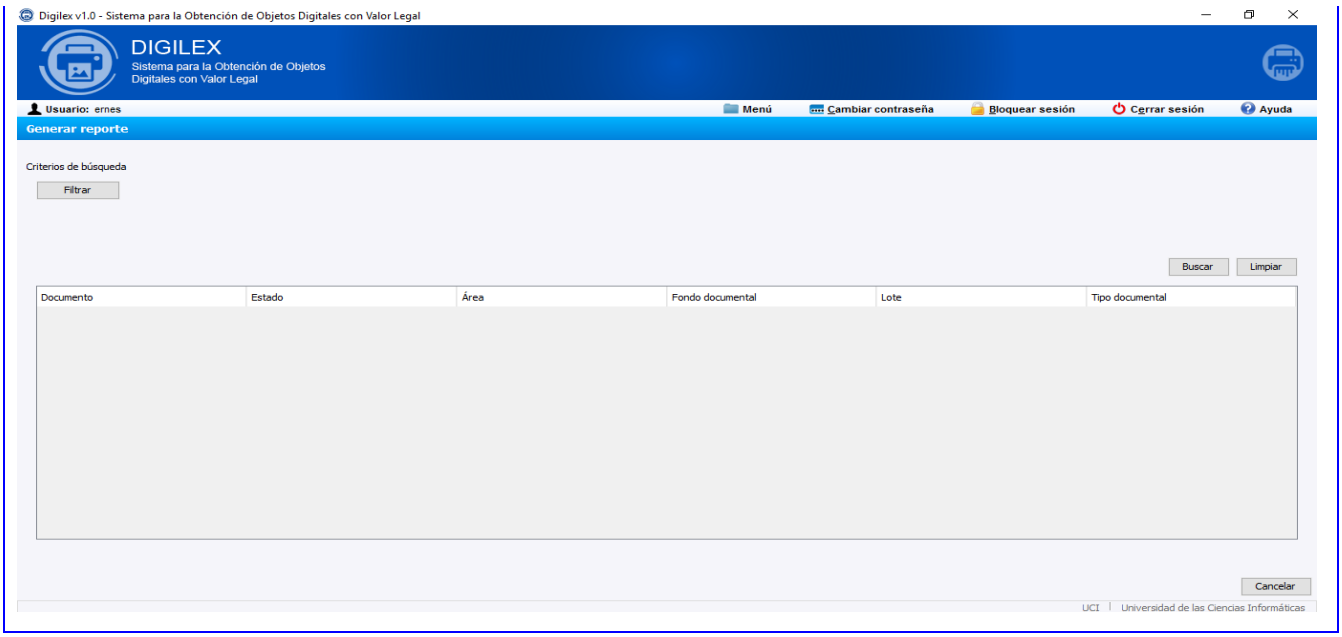
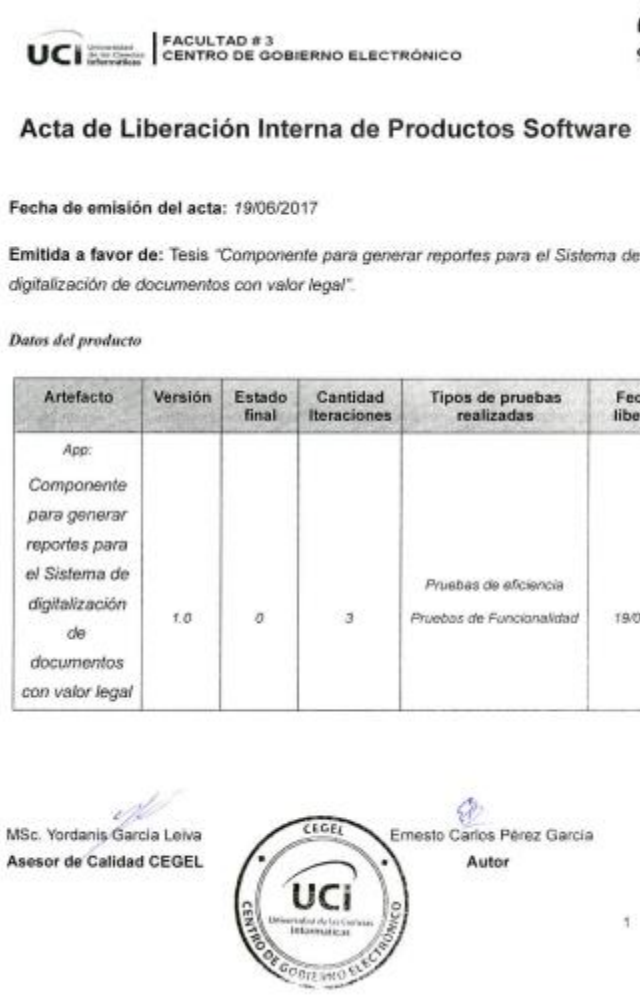


Tabla 16: HU Generar reporte.



UCI | FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

CEGEL

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: 19/06/2017

Emitida a favor de: Tesis *"Componente para generar reportes para el Sistema de digitalización de documentos con valor legal"*.

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
App: *Componente para generar reportes para el Sistema de digitalización de documentos con valor legal*	1.0	0	3	*Pruebas de eficiencia* *Pruebas de Funcionalidad*	19/05/2017

MSc. Yordanis García Leiva
Asesor de Calidad CEGEL

Ernesto Carlos Pérez García
Autor

CEGEL UCI CENTRO DE GOBIERNO ELECTRÓNICO

1

Figura 13: Carta de liberación interna de productos software