Editor de Plantillas de Documentos de Identificación para el Sistema de Personalización de Documentos de Identificación v2.0.

Autores: Yunior Martínez Suarez.

Ariel Ramos Figueredo.

Tutores: Msc. Adrian Machado Cento

Ing. Manuel Alejandro Quert Gómez.

Consultante: Ing. Yanet Silva Fernández.

La Habana, 12 de junio de 2014.

DECLARACION DE AUTORÍA

Declaramos ser los únicos autores del trabajo titulado: Editor de plantillas para el Sistema de Personalización de Documentos de Identificación v2.0 y autorizamos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Y para que así conste firmamos la presente a los ___ días del mes de _____ de 2014.

_____	_____
Firma del Autor	Firma del Autor
Ariel Ramos Figueredo	**Yunior Martínez Suarez**
_____	_____
Firma del Tutor	Firma del Tutor
Msc. Adrian Alberto Machado Cento	**Ing. Manuel Alejandro Quert Gómez.**

DATOS DE CONTACTO

Yunior Martínez Suarez.

Correo: ymartinez@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana Cuba.

Ariel Ramos Figueredo.

Correo: arfigueredo@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana Cuba.

Msc. Adrian Alberto Machado Cento.

Correo: amachado@.uci.cu

Universidad de las Ciencias Informáticas, La Habana Cuba.

Ing. Manuel Alejandro Quert Gómez.

Correo: maquert@.uci.cu

Universidad de las Ciencias Informáticas, La Habana Cuba.

AGRADECIMIENTOS

Yunior:

Parece mentira que han pasado cinco años de nuestras vidas así de rápido, debe ser que fue una etapa tan buena y bonita que como se dice a lo cubano “paso volando”. A todas las personas que me enseñaron cosas importantes para mi vida como profesional, y a las que pretendieron ser un obstáculo en mi meta van dedicadas estas líneas. Agradezco:

Primero que todo a los amigos que hice durante toda la carrera, los que supieron apoyarme en circunstancias de dificultad, que criticaron lo malo, y me hicieron madurar en muchos criterios de la vida.

A mis compañeros de aula, que aguantaron mis pesadeces tanto tiempo, pero sé que en muchas ocasiones serví de alivio y desconecte para tanto obstine estudiantil.

Al equipo de la salsa, posiblemente el grupo más completo que ha tenido esta escuela en todos los ámbitos de la vida en general, al equipo Voltus, como nos bautizó un día una profesora nuestra, y gran amiga Yohana Baró Montenegro, la profe de la controvertida Matemática lV. Mis hermanos Lele, Doime, Alejandro y el Yoe la amenaza los mejores por siempre.

A las niñas de mi aula: Nani, Esteliña, Adysmaris, y mi gran amiga del alma Laurita.

A amigos como Pablo David (La tripa), Gregorio (Memo), Angel Miguel y Angel Noel, por tantos momentos buenos y malos, chucho y más cucho que dimos juntos.

A la gente de mi zona: El Robert, Silva, Katty, Lisandra, Julio César, César por cada viaje que hicimos.

A los grandes profesores Yadier Perdomo Cuevas y Raúl de la Cruz, que son sin duda alguna los mejores educadores, profesores y amigos que puedan tener sus estudiantes.

A Yeneit, el grupo 1502 sabe porque.

A mis amigos de crianza, mis hermanos de siempre, los que hemos pasado buenos y malos momentos juntos, los que hemos metido las mejores fiestas del mundo, Radiel (El tiro) y Cristian (Piño).

A Dios y todos los santos por permitirme llegar con salud, fuerza y claridad a este día.

Al “barbero”, el único que me gusta cómo me pela, mi amigo de la cuadra, Cuchi, por darme aliento y buenos consejos siempre.

Y sobre todas las cosas a mi familia: mi madre querida “La china”, mi querido e incansable padre “Pablito”, a los mejores abuelos del mundo Agapito y Nena, a mi tío de oro Julio, y a la mitad de mi alma, mi querida y amada hermana Yurianne.

Ariel:

Ya terminan 5 años de una etapa que, sin duda, es una de las mejores que pudiera tener cualquier persona. Se acaba una época donde se aprende mucha de la vida, donde abundan tanto los buenos momentos como los malos, pero que siempre te aportan algo que te hará crecer. Agradezco a todas las personas que de una forma u otra me ayudaron a que se hiciera realidad este sueño.

A lo profesores que nos enseñaron como enfrentarnos a la vida y que no dieron los conocimientos para realizar este trabajo, especialmente a Yadier Perdomo Cuevas y a Yanet Silva Fernández.

A los viejos amigos por el apoyo que me brindaron cuando hacía falta.

A los nuevos amigos que durante la carrera me apoyaron y pasamos buenos momentos, Memo, Gago, Nani, Laura, y especialmente a la gente del equipo “Voltus”, Yunior, Yoelmy, Ariel y Alejandro, por todos los momentos de esta hermosa etapa de la vida.

Y especialmente a mi familia por el apoyo que me brindaron durante la carrera, a mis padres con sus consejos que me ayudaron en innumerables ocasiones, a mis abuelos con su sabiduría y esperanzadora alegría y a mi novia por todo el apoyo, amor y cariño que me brindó.

DEDICATORIA

Yunior:

Ya nos encontramos en el momento más esperado de la vida estudiantil, convertirnos en profesionales, pero seguro estoy que para la mayoría es un momento que de tanta alegría puede arrancar lágrimas de nuestros ojos.

Siempre pensé, soñé y hasta agonicé con la idea de que llegara este día, porque sé que sería motivo de alegría para muchos que me quieren. Cada uno de nosotros sabe el trabajo inmenso que pasa la familia para que podamos estudiar lejos y que no nos falte lo mínimo indispensable para poder estudiar. He podido contar con el apoyo, consejos, y regaños constructivos de mi madre y mi padre, con los consejos de mi abuelo y abuela, que siempre tenían pesetas guardadas para que pudiera coger guaguas en la capital, y mi tío que siempre me apoyó y me hizo reír tanto con sus cuentos, y mi flor, mi querida hermana, que poder decir de alguien que lleva mi misma sangre y me demostrado el significado que tiene ser hermanos.

A todos ellos les dedico lo que soy, porque a ellos me debo y sin ellos no sería nada.

Ariel: Luego de 5 años nos encontramos en la culminación de una época llena de momentos que marcaran nuestra vida para siempre, llego la hora y nos convertimos en profesionales luego de tantos años de sacrificio por parte de todas las personas que durante algún tiempo nos han apoyado y aconsejado.

Este logro se lo dedico a todas las personas que durante la carrera han creído en mí y me han apoyado o criticado según fuera la situación; especialmente a mi familia, que todo el tiempo estuvo a mi lado dándome el aliento para luchar contra todas las vicisitudes que se presentaron durante estos 5 años, a mi madre y a mi padre que siempre me ayudaron y si se presentaba algún problema, aunque no supieran como resolverlo buscaban bajo cada piedra a alguien que pudiera orientarme, a mis abuelos por alentarme a seguir adelante cuando se presentaba alguna situación adversa, a mi novia, que me ha aguantado y ha estado siempre para mí cuando lo necesitaba y me ha apoyado en mis decisiones y a mis amigos, los viejos y los nuevos, por tantos momentos que permitieron que despejara cuando me hacía falta.

Este título es para ustedes, ustedes también son ingenieros.

RESUMEN

El diseño de documentos de identificación se realiza mediante programas informáticos que brindan al usuario encargado de diseñar estos documentos la posibilidad de entregar un producto con calidad. El Sistema de Personalización de Documentos de Identificación para la República de Cuba es una aplicación que se encarga de proveer los datos y trámites necesarios para la creación de documentos de identidad de los ciudadanos cubanos. En su versión 2.0 la aplicación sufrió cambios en su arquitectura y modelo de datos, por lo que el módulo de administración de plantilla de la nueva versión no es compatible con el de la anterior. Carece de elementos básicos para diseñar como figuras geométricas, imágenes, textos y otras características. Debido a que las soluciones existentes no cumplen en su totalidad con las funcionalidades necesarias para el editor del Sistema de Personalización de Documentos de Identificación versión 2.0, el presente trabajo de diploma tiene como objetivo desarrollar un editor de plantillas que permita la personalización de documentos de identificación garantizando la reutilización y la calidad de las plantillas diseñadas, de forma genérica y sin que las mismas pierdan calidad al ser exportadas como imágenes, utilizando para esto herramientas y tecnologías como *Visual Studio 2012*, *Windows Communication Foundation* y otras. Posibilita además introducir los datos de los usuarios durante la personalización del documento mediante un servicio web encargado de la captura de datos.

Palabras claves:

Calidad, módulo de administración de plantillas, reutilización, Sistema de Personalización de Documentos de Identificación, *Windows Communication Foundation.*

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Una de las formas más utilizadas para identificar a las personas son los documentos de identificación. Estas credenciales son documentos públicos, personales e intransferibles, emitidos por el Ministerio del Interior, que acredita la identidad y los datos personales de su titular. Estos documentos, debido a su importancia, tienen diseños determinados que vienen regidos por estándares internacionales (1); no obstante los organismos encargados de confeccionar y distribuir estas credenciales pueden decidir la trasformación de los mismos de acuerdo a sus particularidades y exigencias.

Para el diseño de estos documentos se utilizan los editores de plantillas de documentos de identificación que se desarrollan teniendo en cuenta las especificidades del entorno donde se van a desplegar estas credenciales. Mediante estos editores, se pueden realizar diseños de documentos de una forma más rápida y confiable, ya que los mismos brindan un sistema de facilidades funcionales para que los usuarios realicen esta tarea.

El Sistema de Personalización de Documentos de Identificación (SPDI) para la República de Cuba es una solución informática que permite la personalización de documentos relativos a la identificación de los ciudadanos cubanos (2). Este sistema en su versión 1.0 cuenta con un módulo que permite la gestión de las plantillas de los documentos, específicamente del pasaporte corriente de Cuba.

Esto último planteado limita al diseñador de las plantillas, pues si se quisiera personalizar un nuevo tipo de documento se debe realizar cambios en el código del sistema para que acepte los atributos que este tipo de documento pueda tener. Para resolver esta problemática se desarrolla la versión 2.0 del SPDI, pero con el inconveniente de que al actualizar la arquitectura y el modelo de datos para que sea un sistema genérico que permita la personalización de cualquier tipo de documento, el módulo de administración de plantillas implementado en la versión 1.0 del SPDI no es compatible con la versión 2.0.

Otro inconveniente que tiene este módulo es que no incluye algunos elementos para realizar diseños que contengan separadores para los datos, bordes, márgenes como separadores de líneas, figuras geométricas elementales e imágenes predefinidas. Además cuando se crea una plantilla no se pueden ubicar los datos del usuario final durante la personalización, y la misma no se puede convertir en una imagen al introducir estos datos. Es necesario que la imagen obtenida no sea susceptible a la distorsión y a la pérdida de calidad durante su manipulación.

Dada esta situación se plantea el siguiente **problema de investigación:** ¿Cómo realizar el diseño en línea de plantillas para documentos de identificación que garantice la reutilización y calidad de las mismas, para la versión 2.0 del Sistema de Personalización de Documentos de Identificación?
De aquí que se defina como **objeto de estudio:** Los editores de plantillas para documentos de identificación.

Planteándose como **objetivo general:** Desarrollar un editor de plantillas para la personalización de documentos de identificación que garantice la reutilización y la calidad de las plantillas diseñadas para la versión 2.0 del Sistema de Personalización de Documentos de Identificación.

Como **campo de acción** se tiene**:** Los editores de plantillas en línea para documentos de identificación**.**

Durante el desarrollo de esta investigación se le darán cumplimiento a las siguientes **Tareas de investigación:**

- Caracterización de los principales editores de plantillas para documentos de identificación.
- Caracterización de la metodología de desarrollo *MSF for CMMI.*
- Descripción de las herramientas, lenguajes y tecnologías a utilizar para el desarrollo de la solución.
- Especificación de los requisitos funcionales y no funcionales.
- Definición del modelo de dominio, diagrama de clases, despliegue y componentes para la solución.
- Desarrollo del análisis y diseño de la solución.
- Implementación de los requisitos funcionales.
- Realización de las pruebas a la solución desarrollada.

Los métodos de investigación teóricos a utilizar en la realización de este trabajo son:

- **Histórico - Lógico:** Se analiza "la trayectoria completa del fenómeno, su condicionamiento a los diferentes períodos de la historia, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales; basado en el estudio histórico del fenómeno, ponen de manifiesto la lógica interna de su desarrollo, de su teoría y encuentra el conocimiento más profundo de su esencia" (3). Se realiza un estudio de la evolución de los documentos de identificación, y cómo han ido surgiendo nuevos diseños y formas de personalizarlos.
- **Analítico - Sintético:** Este método "permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio, permite establecer mentalmente la unión entre las partes previamente analizadas, y posibilita descubrir sus características generales y las relaciones esenciales entre ellas" (3).

El método de investigación empírico a utilizar es:

- **Entrevista:** "La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información. Su uso constituye un medio para el conocimiento cualitativo de los fenómenos o sobre características personales del entrevistado y puede influir en determinados aspectos de la conducta humana por lo que es importante una buena comunicación. La entrevista puede ser individual o colectiva, en ambos casos el entrevistador debe realizar una preparación previa, sobre el tema a tratar, y elaborar una guía para su desarrollo" (3).

Posibles Resultados:

- Desarrollo de un editor de plantillas para la personalización de documentos de identificación que garantiza la reutilización y calidad de las plantillas diseñadas para el Sistema de Personalización de Documentos de Identificación v2.0.
- Documentación que recoge de forma general todo lo referente al proceso de desarrollo del editor de plantillas.
- Manuales de usuario para el editor desarrollado.

Posibles Aportes:

- Reutilización del componente en otros proyectos desarrollados con ASP.NET.
- Edición de la plantilla diseñada en otros editores que permitan el formato SVG.
- Editor genérico que permite la edición de plantillas para distintos tipos de documentos.

El presente trabajo consta de tres capítulos:

Capítulo 1 - Fundamentación Teórica: En este capítulo se presentan algunas de las características de las aplicaciones existentes para resolver el problema en cuestión. Además se plantean los principales conceptos y aspectos relacionados con el dominio del problema a resolver y se exponen las normas y técnicas utilizadas actualmente para el diseño de documentos de identificación. También se caracterizan las herramientas, tecnologías, lenguajes y metodología que se utilizarán en este trabajo.

Capítulo 2 – Análisis y diseño: En este capítulo se dará a conocer la solución propuesta para resolver el problema planteado, representando el modelo de dominio de la misma. Se realiza la descripción de los roles que interactúan con la aplicación y se definen los requisitos funcionales y no funcionales con que deberá contar la aplicación, así como la descripción de los mismos. Se definirá la arquitectura a utilizar para el desarrollo de la aplicación y se representará la forma en que estarán estructurados los distintos elementos de la solución. Se explica la función de cada una de las clases con que cuenta el editor además de las relaciones que existen entre ellas, apoyándose para esto en el diagrama de clases. Se ofrece una representación de la estructura de la base de datos que se utiliza para el desarrollo de la aplicación y la descripción de sus entidades.

Capítulo 3 – Implementación y pruebas: En este capítulo se muestran los estándares de codificación que se definieron para utilizar en el código del editor, así como de qué forma se realizará el tratamiento de errores. Se describe la implementación del editor a través del diagrama de componentes y la forma en que se deberá desplegar la aplicación una vez que se ponga en explotación. Se muestran todas las pruebas realizadas a la aplicación, buscando la calidad del producto, y se explica cuál es el beneficio del mismo.

Capítulo 1: Fundamentación Teórica.

En el presente capítulo se expondrá la fundamentación teórica que servirá de marco para la solución que se propone. Para un mejor entendimiento del problema se realiza un estudio del arte, donde se analizan las soluciones existentes, los inconvenientes que presentan su utilización y las normas o estándares internacionales que regulan el diseño de los documentos de identificación. Además se abordan las definiciones afines con la investigación y se caracterizan las herramientas, tecnologías, lenguajes y la metodología a usar.

Definiciones.

Documentos de identificación: "Los documentos de identificación, también llamados documentos nacionales de identidad (DNI) o cédula de identidad, son documentos emitidos por una autoridad administrativa competente para permitir la identificación personal de los ciudadanos residentes en el país, así como aquellos que puedan encontrarse en territorio nacional de manera temporal. Constituyen documentos únicos de identificación personal e intransferible extendido a todos los ciudadanos. Estos usualmente poseen la fotografía del portador, su firma, nombre, nacionalidad, fecha de nacimiento, así como otros elementos que contribuyen a la identificación." (4)

Editor Gráfico: "En informática, software o un paquete de estos dotados de herramientas que permiten la edición apoyada en computadores de imágenes digitales, en la mayoría de los casos fotos o documentos escaneados. Estas imágenes son modificadas para optimizarlas, manipularlas, retocarlas, etc., con el fin de alcanzar la meta deseada." (5). Algunas de las funcionalidades de estos editores son: oscurecer y aclarar, borrador, filtro, efectos, fotomontaje, pintar, etc. (5)

Editor de Texto: "A instancias de la informática**,** el término resulta muy común como consecuencia del llamado editor de textos**,** que es aquel programa que permitirá redactar, editar, modificar e imprimir documentos digitales." (6). "Entre sus principales y típicas funciones se cuentan las siguientes: marcar un renglón, buscar, reemplazar, cortar, copiar, pegar, deshacer, rehacer, importar fotografías o cualquier otro tipo de imagen**,** entre otras." (6)

Normas Internacionales.

La Organización Internacional para la Normalización (ISO por sus siglas en inglés) define la norma ISO/IEC 7810 para estandarizar el diseño físico de los documentos de identificación (7).

Esta norma tiene como objetivo especificar los requisitos que deben poseer las tarjetas y documentos para poder ser utilizados en intercambios internacionales, teniendo en consideración los aspectos humanos y tecnológicos (7).

La norma ISO/IEC 7810:2003 específica (7):

- Cuatro tamaños diferentes para la tarjetas de identificación con un espesor de 0,76 mm y dimensiones de:
    - Tipo 1: ID-000 25 milímetros x 15 milímetros con un ángulo ligeramente biselado (3 mm). Este formato es usado para las tarjetas SIM
    - Tipo 2: ID-1 85,60 milímetros x 53,98 milímetros. Es comúnmente usado para tarjetas de banco (tarjetas ATM, tarjetas de crédito, tarjetas de débito, etc.). Se aplica para licencias de conducción en muchos países y para tarjetas de identidad personal en algunos países. La tarjeta de pasaporte de Estados Unidos usa también el formato ID -1.
        - ISO/IEC 7813 define características adicionales para ID-1 en las tarjetas de banco plásticas, por ejemplo, un espesor de 0.76 milímetros y esquinas redondeadas con un radio de 3.18 milímetros.
        - ISO/IEC 7811 define técnicas tradicionales en ID -1 para grabar datos en las tarjetas de identificación, caracteres a saber y varios formatos de grabación magnética.
        - ISO/IEC 7816 ID -1 define que las tarjetas de identificación tendrán un chip integrado (tarjeta inteligente), superficies de contacto para la energía, reloj, señales de datos en serie y reseteo.
    - Tipo 3: ID-2 105 milímetros x 74 milímetros. Este tamaño es del formato A7. El formato ID -2 es usado para visas. Fue usado en Alemania en las tarjetas de identidad, fabricadas hasta Octubre del 2010, desde noviembre de ese mismo año las tarjetas de identificación alemanas son emitidas en el formato ID -1.
    - Tipo 4: ID-3 125 milímetros x 88 milímetros. Este tamaño es del formato B7. Este formato es usado en el mundo para pasaportes.
- Los materiales de construcción de las tarjetas de identificación. Estos materiales pueden ser variados, de acuerdo a las especificaciones de fabricación, como: Policloruro de vinilo, Tereftalato de polietileno, Acrilonitrilo Butadieno Estireno, Policarbonato, entre otros.
- Las características físicas como la rigidez a la flexión, inflamabilidad, toxicidad, resistencia a los productos químicos, estabilidad dimensional, la adhesión o el bloqueo, la deformación, resistencia al calor, las distorsiones de la superficie, y la contaminación.

La historia de la creación de los documentos de identidad viene acompañada de factores sociales que han ameritado a su surgimiento. En 1824, Fernando Vll crea la policía en España y le dio la potestad de adoptar un documento que incluía edad, sexo, estado, profesión, y la naturaleza del vecindario para identificar a los ciudadanos. (8). En 1871 el Estado aprueba una ley para tener registrados los datos de todos los ciudadanos independientemente de su religión. (9). Para el año 1944 se crea un nuevo documento, para mantener un mayor control sobre los españoles. La iniciativa parte de la Presidencia del Gobierno, y los primeros obligados a formalizarlo fueron los presos y los que se mantenían en libertad condicional. Desde un principio,

los números del documento de identidad se asignaron por lotes a los equipos de expedición, lo que aún se encuentra vigente. (8). Siete han sido los diseños que han servido para la confección del documento de identidad en España. El primero data de 1951 e incluía los datos de filiación, profesión, empleo o cargo. La segunda llega en 1962 e incorporaba el estado civil y grupo sanguíneo. En 1965 surge el tercer diseño y se expide hasta 1980, siendo unas de las tarjetas más longevas y populares, quitando la firma del director del equipo que expedía el carné. El siguiente diseño viene en 1981 hasta 1985, con las variantes que esta vez se incluyen el escudo constitucional, el sexo del ciudadano y se excluyen las categorías. Para el modelo de los años 1985 hasta 1991 se descartan la profesión, el estado civil y grupo sanguíneo, debido a que en ocasiones provocaba errores médicos. Los modelos de la década del 90 fueron ideados mediante tecnologías informáticas. El actual documento de identidad español cuenta con un chip que agiliza los trámites con la Administración e Internet, además de contener toda la información del individuo, incluyendo la huella dactilar. (8).

Otro ejemplo que refleja mediante su historia la evolución de los documentos de identidad es Chile. El primer documento de identidad data de 1924, junto con el Servicio de Identificación; se llamó Libreta de Identidad y la misma tenía datos como nombre, domicilio, huella dactilar y una foto del titular. Desde entonces ha sufrido una serie de cambios, hasta llegar al diseño actual. (10). En el año 1930 se comienza a otorgar un número a las cédulas con el Registro de Numeración Civil; en 1934 se fusionan el Servicio de Registro Civil, creado en 1884, y el Servicio de Identificación, continuando con la emisión del carné de identidad. En 1973 se implementó el Rol Único Nacional, de modo que la información de cada persona, natural o jurídica, podía ser procesada de forma electrónica sobre la base de un número de identificación, lo que para el año 1984 se comenzaría a realizar de forma computacional. Pero sin duda alguna, el año que más aportó cambios fue el 2002, cuando se convierte en una lámina de plástico polimérico e impresión láser, con una serie de características que no permitían que se dañara el documento. Este documento fue utilizado hasta el 2 de septiembre de 2013, cuando entró en vigencia un nuevo tipo de cédula con 25 medidas de seguridad, entre ellas un chip electrónico. (10)

Existen aplicaciones que cuentan con una serie de funcionalidades que facilitan el trabajo en el diseño de plantillas de documentos de identificación, tanto en el ámbito internacional como nacional, pero por razones que serán expuestas a continuación no representan la solución que se necesita.

En el ámbito nacional se tienen:

- **Editor de plantillas del SPDI 1.0.**

  El Editor de plantillas del SPDI en su versión 1.0, brinda la posibilidad de crear pasaportes, y en una actualización reciente los documentos de acceso a la Universidad de Ciencias Informáticas (UCI), lo que resultó de gran importancia debido a sus funcionalidades, pero no se puede incorporar a la

segunda versión puesto que no se ajusta a la arquitectura y tendría que ser modificado el código fuente del mismo. Además solo tiene en cuenta la definición de una plantilla para un tipo de documento en específico, en caso de querer definir una plantilla nueva se debe modificar el código del SPDI; no cuenta con elementos de diseño como figuras geométricas básicas, separadores e imágenes predefinidas (2).

- **Editor de plantillas para la confección de pasaportes corrientes de la República de Cuba (EMIPAS).**

  El EMIPAS cuenta con varias facilidades funcionales, como son: la personalización de pasaportes de lectura mecánica, el control de calidad y supervisión del pasaporte, garantiza fiabilidad y exactitud de la información contenida, control de inventarios de los pasaportes y el estado de cada documento personalizado. Permite la captación de la información en vivo o a través de documentos que la contengan y posibilita la verificación de cada persona que solicita pasaporte contra Listas de Impedimento (11). Estas características hacen que este sea un producto de calidad para la personalización de pasaportes, pero este sistema no se puede utilizar porque su arquitectura no se ajusta a la del SPDI v2.0, por lo que se debería modificar el código del mismo. Otro de los inconvenientes que presenta este editor es que las plantillas diseñadas no tienen un formato estándar, por lo que si se quiere modificar la plantilla en otro editor no sería posible, además de que tampoco cuenta con elementos de diseño como figuras geométricas básicas.

Por otra parte en el ámbito internacional se encuentran:

- ***Digital Identification***

  Uno de los ejemplos que existe actualmente es el software *Digital Identification* de tecnología alemana. Una de las ventajas que posee este software es que “permite la creación de documentos de identificación en 10 lenguajes diferentes, además de la codificación de la información que se utiliza en el proceso. Cuenta con un módulo de captura de imágenes” (12). Este software no se puede utilizar ya que es propietario y viene instalado en el Sistema Operativo que se encuentra en la computadora embebida de la impresora láser *Digital Identification solutions* y no se tiene el código fuente para interpretarlo. Cada una de estas impresoras tiene un costo de entre 2000 y 10000 euros y las plantillas son generadas en un formato específico para estas impresoras el cual no es estándar (12).

- ***CardFive***

  Otro ejemplo muy parecido al expuesto antes, es el software *CardFive,* incluida en las impresoras marca *Zebra*. Esta aplicación permite un desarrollo integral e intuitivo, permite realizar múltiples funciones en la misma ventana de manera sencilla. Cuenta con altas prestaciones para diseñar, ya que presenta una gran capacidad para integrar cualquier tipo de imagen. Contiene múltiples recursos profesionales, posee gran velocidad de impresión y control de errores, además de las rápidas

actualizaciones y fácil gestión de almacenaje de datos (13). Como principales desventajas presenta un lenguaje de impresión estándar, el DCL (*Direct Command Language*), cuando se crea se codifica de forma binaria, por lo que no permite que se pueda leer o editar en otra aplicación. Es propietaria por lo que no permite la visualización del código fuente y su precio es de alrededor de $2500 USD, lo que demuestra que su utilización solo propicia la dependencia tecnológica (13).

Luego de realizado el estudio de algunas herramientas de edición de plantillas de documentos de identificación a nivel nacional e internacional se concluye que, ninguna de las estudiadas resuelve en su totalidad las necesidades actuales. En el caso de la nación, no se pueden personalizar documentos de identificación de forma genérica, hay que transformar el código fuente de las dos aplicaciones, si se desea personalizar un nuevo tipo de documento. Internacionalmente, solo se propiciaría la dependencia tecnológica, y los costos de adquisición son altos, teniendo en cuenta que hay que comprar las impresoras donde vienen embebidas las aplicaciones de edición.

Metodologías de desarrollo de software.

La metodología que se utilizará es *MSF* (*Microsoft Solutions Framework*) *for CMMI* (*Capability Maturity Model Integration*) ya que es la utilizada en el proyecto. Esta metodología es un enfoque personalizable para entregar correcta y más rápidamente soluciones tecnológicas, con menos personas y menos riesgo, pero con resultados de más calidad. *MSF* ayuda a los equipos a resolver directamente las causas más comunes de error en el proyecto de tecnología, lo cual mejora los índices de buenos resultados, de calidad de la solución y de impacto comerciales. (14)

MSF se centra en: (14)

- Establecer objetivos, roles y responsabilidades claros para el proyecto.
- Implementar un proceso iterativo, basado en hitos/puntos de control.
- Respuestas efectivas a los cambios.

Además con las buenas prácticas y el conjunto completo e integrado de guías para desarrollar productos y servicios que proveen los modelos *CMMI* ayudan a las organizaciones a mejorar sus procesos. (15)

Tecnologías, Herramientas y Lenguajes de desarrollo

Gráficos Vectoriales Escalables (*Scalable Vector Graphics (SVG)*).

Es una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados, en formato XML y su desarrollo está a cargo del consorcio W3C *(World Wide Web Consortium*). Un dibujo vectorial es una imagen formada por una serie de elementos geométricos definidos por fórmulas

matemáticas. Las figuras geométricas utilizadas en el formato de imágenes vectoriales son: la línea, los polígonos, las formas circulares y los nodos.

Las imágenes vectoriales tienen como principal características que se utilizan fórmulas matemáticas de rectas, curvas y puntos para representar la imagen lo que hacen que ocupen menos espacio que los mapas de bits y la característica principal que define a estos gráficos es que al ampliar la imagen no se pierde la calidad (16).

Ventajas de utilizar SVG (17)**:**

- Renderizado.
- Patrones de relleno y gradientes.
- Filtros y efectos avanzados.
- Animaciones.
- No pierde calidad si se amplia.
- Puede escalarse.
- Ideal para ser impreso.
- Pueden mostrarse de forma progresiva (igual que los GIF), no teniendo que esperar a que todo el documento sea descargado.
- Pueden distribuirse en formato comprimido GZIP para la reducción del tiempo de descarga .SVGZ.
- Pueden ser indexados y buscados debido a que su contenido es XML y es textual.
- Pueden ser transformados por hojas de estilos (XSL o CSS).
- Integración con otras tecnologías XML del W3 y compatible con:
    - XML 1.0
    - CSS
    - XSL
    - SMIL 1.0. Sólo algunas de sus funcionalidades
    - HTML 4 y XHTML 1.0
    - UNICODE
- Accesibilidad a contenidos web (WAI)

XML (*eXtensible Markup Language*)

"XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Las tecnologías XML son un conjunto de módulos que

ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información" (18).

ASP.NET

ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. ASP.NET forma parte de .NET *Framework* y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET *Framework*. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el *Common Language Runtime* (CLR), entre ellos *Microsoft Visual Basic* y C#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del *Common Language Runtime*, seguridad de tipos, herencia, etc (19).

C# 4

C# es un lenguaje de programación que se ha diseñado para generar diversas aplicaciones que se ejecutan en .NET *Framework*. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C (20).

Ventajas (21)**:**

- Declaraciones en el espacio de nombres: al empezar a programar algo, se puede definir una o más clases dentro de un mismo espacio de nombres.
- Tipos de datos: en C# existe un rango más amplio y definido de tipos de datos que los que se encuentran en C, C++ o Java.
- Atributos: cada miembro de una clase tiene un atributo de acceso del tipo público, protegido, interno, interno protegido y privado.
- Pase de parámetros: aquí se puede declarar a los métodos para que acepten un número variable de parámetros.
- Métodos virtuales y redefiniciones: antes de que un método pueda ser redefinido en una clase base, debe declararse como virtual.
- Propiedades: un objeto tiene intrínsecamente propiedades, y debido a que las clases en C# pueden ser utilizadas como objetos, C# permite la declaración de propiedades dentro de cualquier clase.
- Inicializador: un inicializador es como una propiedad, con la diferencia de que en lugar de un nombre de propiedad, un valor de índice entre corchetes se utiliza en forma anónima para hacer referencia al miembro de una clase.
- Control de versiones: C# permite mantener múltiples versiones de clases en forma binaria, colocándolas en diferentes espacios de nombres

JQUERY 1.7

jQuery es una librería JavaScript de código abierto, pequeña, rápida y con muchas funcionalidades ya implementadas que hacen que el desarrollador realice sus tareas en menos tiempo ya que algunas de ellas resuelven de manera eficiente alguno de los problemas más comunes a la hora de implementar. *jQuery* a través de su versatilidad y extensibilidad ha cambiado la forma en que muchas personas desarrollan aplicaciones web (22).

Ventajas

La ventaja principal de jQuery es que es mucho más fácil que sus competidores. Se pueden agregar *plugins* fácilmente, traduciéndose esto en un ahorro substancial de tiempo y esfuerzo. La licencia *open source* de jQuery permite que la librería siempre cuente con soporte constante y rápido. Otra ventaja de jQuery sobre sus competidores como Flash y puro CSS es su excelente integración con AJAX (23).

Visual Studio 2010

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y hace más sencilla la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET *Framework*, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML (24).

WCF (*Windows Communication Foundation*)

"Es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro. Un extremo de servicio puede formar parte de un servicio disponible continuamente hospedado por IIS, o puede ser un servicio hospedado en una aplicación. Un extremo puede ser un cliente de un servicio que solicita datos de un extremo de servicio. Los mensajes pueden ser tan simples como un carácter o una palabra que se envía como XML, o tan complejos como una secuencia de datos binarios" (25).

Características (25)**:**

- **Orientación a servicios**: Como consecuencia del uso de los estándares de WS, WCF le permite crear aplicaciones orientadas a servicios. SOA, la arquitectura orientada a servicios es el uso de servicios web para enviar y recibir datos. Los servicios tienen la ventaja general de estar débilmente acoplados entre una aplicación y otra en lugar de incluidos en el código.
- **Interoperabilidad:** WCF implementa los estándares del sector modernos para la interoperabilidad de servicios web.
- **Varios modelos de mensajes:** Los mensajes se intercambian mediante uno de los distintos modelos. El más común es el de solicitud/respuesta, en que un extremo solicita datos de otro

extremo y el otro extremo responde. Existen otros modelos, como un mensaje unidireccional, en que un único extremo envía un mensaje sin esperar ninguna respuesta. Un modelo más complejo es el modelo de intercambio dúplex donde dos extremos establecen una conexión y envían datos hacia delante y hacia atrás, similar a un programa de mensajería instantánea.

- **Metadatos de servicios:** WCF admite la publicación de metadatos de servicios utilizando los formatos especificados en los estándares de la industria, como WSDL, Esquemas XML y *WS-Policy*. Estos metadatos pueden utilizarse para generar y configurar automáticamente clientes para el acceso a los servicios de WCF.
- **Contratos de datos:** Dado que WCF se basa en .NET *Framework*, también incluye métodos con código sencillo para proporcionar los contratos que desea aplicar. Uno de los tipos de contrato universales es el contrato de datos. Básicamente, mientras se escribe el código del servicio usando Visual C# o Visual Basic, la forma más sencilla de controlar los datos consiste en crear clases que representan una entidad de datos con propiedades que pertenecen a la misma.
- **Seguridad:** Es posible cifrar los mensajes para proteger la privacidad, así como obligar a los usuarios a que se autentiquen antes de permitirles recibir mensajes. La seguridad puede implementarse utilizando estándares conocidos como SSL o WS-*SecureConversation*.

WCF se ha diseñado para ofrecer un enfoque manejable para la creación de servicios web y clientes de servicios web. Con la utilización de este marco de trabajo se pueden desarrollar aplicaciones las cuales tengan un mayor nivel de confiabilidad ya que se puede implementar un servicio para procesar datos que garantizarían la seguridad de una compañía, por ejemplos transacciones, entre otros. Además con el desarrollo de los servicios web muchos procesos empresariales han alcanzado un alto nivel organizacional ya que las funcionalidades que pueden brindar los servicios son muy diversas y están orientadas a agilizar el funcionamiento de un sistema.

Altova UModel 2010

UModel es una potente y sencilla herramienta UML para diseñar software de forma visual. Este programa permite diseñar modelos de aplicaciones con UML de forma visual y genera código Java, C# o Visual Basic .NET, así como documentación de proyecto. Además brinda la posibilidad de convertir programas en diagramas UML 2 mediante ingeniería inversa, mejorarlos y terminar el proceso regenerando el código de programa (26).

Características (26)**:**

- Herramientas de modelado visual
- Generación de código UML
- Transformación de código en modelos UML mediante ingeniería inversa

- Ingeniería de ida y vuelta
- Esquemas XML en UML
- Diagramas de base de datos UML
- Documentación de proyectos UML
- Modelado de procesos de negocio
- Importación y exportación en XMI
- Compatibilidad con control de versiones
- Integración con Eclipse y Visual Studio
- Editor de scripts y API
- Arquitectura dirigida por modelos UML
- Funciones para trabajo en equipo

Diagramas de estructura UML (26)**:**

- Diagramas de clases y diagramas de objetos
- Diagramas de estructura de un compuesto
- Diagramas de componentes
- Diagramas de implementación
- Diagramas de paquetes
- Diagramas de perfil

Diagramas de comportamiento UML (26)**:**

- Diagramas de secuencia
- Diagramas de actividades
- Diagramas de máquina de estados
- Diagramas globales de interacción
- Diagramas de ciclo de vida
- Diagramas de comunicación

Oracle

El Sistema de Gestión de Bases de Datos Oracle, fabricado por *Oracle Corporation*, utiliza la arquitectura cliente/servidor. Ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Así ofrece un servidor de bases de datos híbrido. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad. Los objetos

de **Oracle** son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos (27).

Ventajas

- Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia.
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente.
- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos.
- Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix.
- Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal.
- Las copias de la Base de Datos productiva pueden estar en modo de lectura solamente.

LINQ (*Language-Integrated Query*)

Es una innovación introducida en *Visual Studio 2008* y *.NET 3.5* que persigue el objetivo de eliminar la distancia entre los objetos y los datos asociados a estos, agregando la capacidad de consultas eficientes a las sintaxis de los leguajes *Visual Basic y C#*. En su contenido LINQ incluye patrones estándares y de sencillo aprendizaje para realizar consultas y actualizar los datos y a su vez permite extenderse para poder utilizar cualquier almacén de datos (28).

Ventajas de LINQ

- Sintaxis familiar para escribir consultas.
- Comprobación en tiempo de compilación de errores de sintaxis y seguridad de tipos.
- Compatibilidad mejorada con el depurador.
- Compatibilidad con *IntelliSense*.
- Capacidad para trabajar directamente con elementos XML en lugar de crear un documento XML contenedor, que es lo que se requiere en W3C DOM.
- Modificación de documentos XML en memoria de gran eficacia, aún más fácil de usar que *XPath* o *XQuery.*
- Funciones de filtrado, ordenación y agrupación eficaces.
- Modelo coherente para trabajar con datos en varios tipos de formatos y orígenes de datos.

PL/SQL

Oracle PL/SQL es un lenguaje de programación creado por Oracle como una extensión de SQL. Este lenguaje puede combinar las consultas SQL y las instrucciones de procedimientos para crear un tratamiento complejo y ser almacenados en la base de datos (29).

Ventajas PL/SQL (30)**:**

- Estructura de bloques: PL/SQL consiste en bloques de código. Cada bloque forma una unidad de tarea o un módulo lógico. Los bloques de PL/SQL pueden ser almacenados en la base de datos y ser reutilizados.
- Capacidad de lenguaje de procedimientos: PL/SQL contiene lenguaje de procedimientos que permite construir estructuras como sentencias condicionales y ciclos.
- Mejor rendimiento: El motor de PL/SQL procesa varias sentencias SQL al mismo tiempo como un solo bloque lo que permite reducir el tráfico de la red.
- Manejo de errores: PL/SQL maneja los errores y las excepciones efectivamente durante la ejecución de un programa PL/SQL. Una vez que la excepción es capturada, se pueden tomar acciones específicas dependiendo del tipo de excepción o se le puede mostrar al usuario mediante un mensaje el motivo del error.

Conclusiones Parciales.

- Al realizarse el estudio de los editores de documentos de identificación a nivel nacional e internacional se concluye que en conjunto no contienen las funcionalidades y características que debe tener el editor de plantillas para SPDI v2.0.
- Las herramientas, tecnologías y lenguajes seleccionados permiten que se pueda llevar a cabo la integración de la aplicación al Sistema de Personalización de Documentos de Identificación.

CAPÍTULO 2: Análisis y Diseño

En este capítulo se presenta la solución con la que se propone resolver el problema existente, además se define el modelo del dominio en el cual se enmarca dicha situación. Se realiza el levantamiento y especificación de los requisitos funcionales y no funcionales y se exponen los roles que interactuarán con el sistema. Se define la arquitectura y la estructura de la solución, los patrones utilizados además del modelo de datos del sistema, el diagrama de clases y otros artefactos que ayudaran al mejor entendimiento de la solución.

Propuesta de Solución.

Se propone la implementación del editor de plantillas de documentos de identificación para SPDI v2.0, basándose en la reutilización del componente *svgEdit*, extendiendo las funcionalidades de este para la corrección de algunas deficiencias que presenta la versión 1.0, que dificultan el trabajo en el mismo, y no tener que transformar el código fuente cada vez que se necesite la personalización de un documento de identificación. La nueva versión del editor permitirá diseñar diferentes tipos de documentos de identificación como pasaportes y licencia de conducción y disminuirá considerablemente el tiempo empleado en la confección de las plantillas de dichos documentos. Permitirá insertar los datos del usuario final en la plantilla, durante el proceso de personalización, y podrán ser creadas y guardadas en los formatos de salida JPG y SVG, persiguiendo con este último que se pueda cargar en otros editores de plantilla que admitan esta extensión y que la calidad de las plantillas no se vea afectada, ya que crea las imágenes vectorialmente, evitando la pérdida de resolución al aumentar su tamaño. Contará con la existencia de figuras geométricas elementales, separadores, bordes, e imágenes predefinidas que se podrán incluir en el diseño de cualquier plantilla. Dispondrá de un servicio web llamado *SPDIService*, el cual permitirá la conexión del editor con la base de datos, para la administración de la información asociada a cada plantilla, además de poder ser utilizado por otro editor distinto al implementado, u otra aplicación que necesite manejar datos.

Modelo de dominio

“Un modelo de dominio muestra (a los modeladores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés” (31). Para el desarrollo de la solución se realiza el modelo de dominio ya que este permite ver de una forma más clara la interacción que existe entre las entidades involucradas y permite tener un mejor entendimiento de la lógica del negocio. A continuación se muestra el modelo de dominio definido para la aplicación.

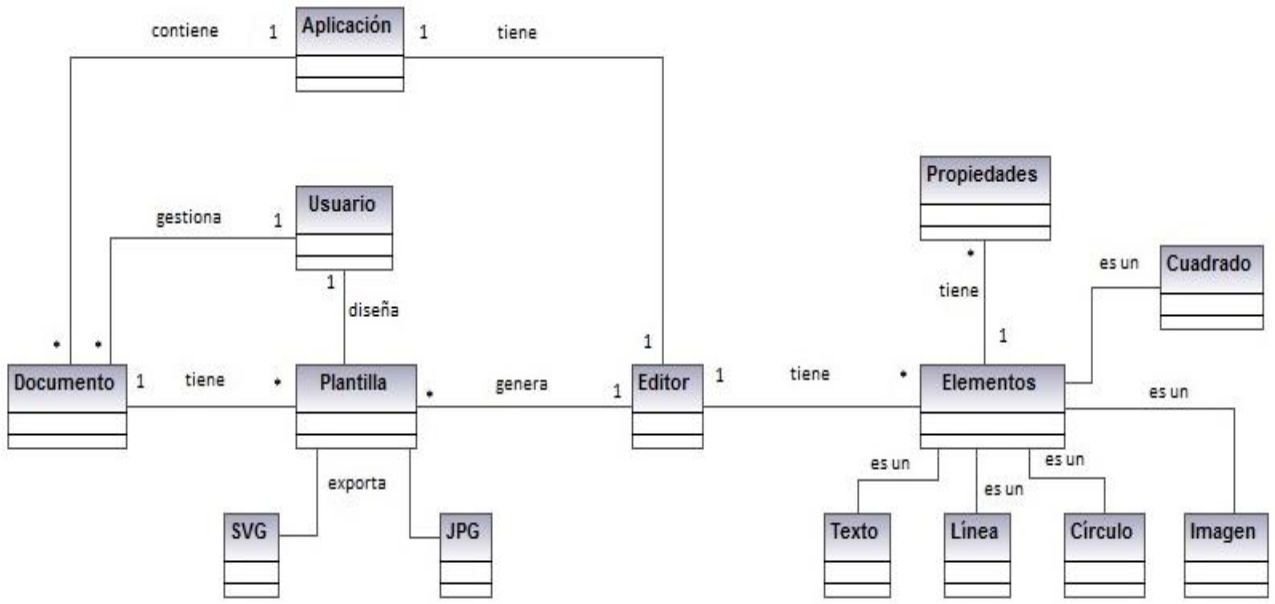


Figura 1: Modelo de Dominio.

Aplicación: es el Sistema de Personalización de Documentos de Identificación (SPDI).

Documento: archivo que contiene algún tipo de información atendiendo al tipo de documento que sea seleccionado.

Plantilla: archivo en el cual se realizará el diseño de un determinado tipo de documento previamente seleccionado.

Editor: herramienta que se utiliza para el trabajo de personalización de los documentos de identificación.

SVG: extensión en la que es exportado el documento personalizado, la misma permite que sea cargado en cualquier otro editor que pueda trabajar con esta tecnología.

JPG: extensión en la que es exportado el documento personalizado.

Usuario: es quien trabaja con el editor atendiendo al rol que desempeñe el mismo.

Elementos de Diseño: medios que facilitan el trabajo en la personalización de documentos.

Propiedades: accesorios de configuración para los elementos que componen el documento que se personalice.

Texto: elemento de diseño que permite introducir escritos en el documento que se esté personalizando.

Línea: elemento de diseño que permite separar o delimitar las áreas del documento que se esté personalizando.

Imagen: elemento de diseño que permite añadir una o varias imágenes al documento que se esté personalizando, además de la imagen predeterminada.

Cuadrado: figura geométrica que puede ser utilizada en la personalización de un documento.

Círculo: figura geométrica que puede ser utilizada en la personalización de un documento.

Requerimientos del sistema

Los requerimientos son la descripción de los servicios que brinda el sistema y las restricciones con que cuenta el mismo a la hora de su ejecución, además reflejan las necesidades que tiene el cliente sobre el sistema que desea para resolver la situación existente (32). Es decir, los requerimientos son las características específicas con que debe cumplir un producto para que resuelva de manera eficiente los problemas en determinado escenario.

Descripción de los roles

El sistema contará con un rol, el cual tendrán todos los usuarios que interactúen con el sistema.

Tabla 1: Roles del sistema.

Rol	Descripción
Usuario	✓ Gestiona los tipos de documentos, atributos de los tipos de documentos, y tipos de datos que serán utilizados en el diseño de las plantillas. ✓ Crea, diseña y modifica las plantillas.

Definición de los requisitos

Requisitos funcionales

"Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer" (32). Los requisitos con que debe cumplir el editor de plantillas que se quiere desarrollar son:

RF1 - Gestionar Tipo de documento.

- ✓ **RF1.1** - Adicionar tipo de documento.
- ✓ **RF1.2** - Modificar tipo de documento.
- ✓ **RF1.3** - Eliminar tipo de documento.
- ✓ **RF1.4** - Mostrar tipo de documento.

RF2 - Editar plantillas.

- ✓ **RF2.1**- Crear plantilla.
    - RF2.1.1- Insertar elementos.
    - RF2.1.2- Modificar propiedades de los elementos insertados.
    - RF2.1.3- Insertar atributos.
    - RF2.1.4- Guardar la plantilla creada.
- ✓ **RF2.2** - Modificar plantilla.
    - RF2.2.1- Modificar elementos.
    - RF2.2.2- Modificar propiedades de los elementos insertados.
    - RF2.2.3- Insertar atributos.
    - RF2.2.4- Guardar la plantilla modificada.

RF3 - Gestionar atributos de documentos.

- ✓ **RF3.1** - Adicionar atributos de documentos.
- ✓ **RF3.2** - Modificar atributos de documentos.
- ✓ **RF3.3** - Eliminar atributos de documentos.
- ✓ **RF3.4** - Mostrar atributos de documentos.

RF4 - Gestionar tipo de dato.

- ✓ **RF4.1** - Adicionar tipo de dato.
- ✓ **RF4.2** - Modificar tipo de dato.
- ✓ **RF4.3** - Eliminar tipo de dato.
- ✓ **RF4.4** - Mostrar tipo de dato.

RF5 - Buscar.

- ✓ **RF5.1** - Buscar tipo de documento.
- ✓ **RF5.2** - Buscar atributos de documentos.
- ✓ **RF5.3** - Buscar tipo de dato.

El servicio web *SPDIService* encargado de la conexión a la base de datos para la obtención de la información solicitada deberá:

RF6 - Gestionar Tipo de documento.

- ✓ **RF6.1** - Adicionar tipo de documento.
- ✓ **RF6.2** - Modificar tipo de documento.
- ✓ **RF6.3** - Eliminar tipo de documento.

RF7 - Gestionar tipo de dato.

- ✓ **RF7.1** - Adicionar tipo de dato.
- ✓ **RF7.2** - Modificar tipo de dato.
- ✓ **RF7.3** - Eliminar tipo de dato.

RF7 - Gestionar atributos de documentos.

- ✓ **RF8.1** - Adicionar atributos de documentos.
- ✓ **RF8.2** - Modificar atributos de documentos.
- ✓ **RF8.3** - Eliminar atributos de documentos.

RF9 - Buscar.

- ✓ **RF9.1** - Buscar tipo de documento.
- ✓ **RF9.2** - Buscar atributos de documentos.
- ✓ **RF9.3** - Buscar tipo de dato.

Especificación de los requisitos

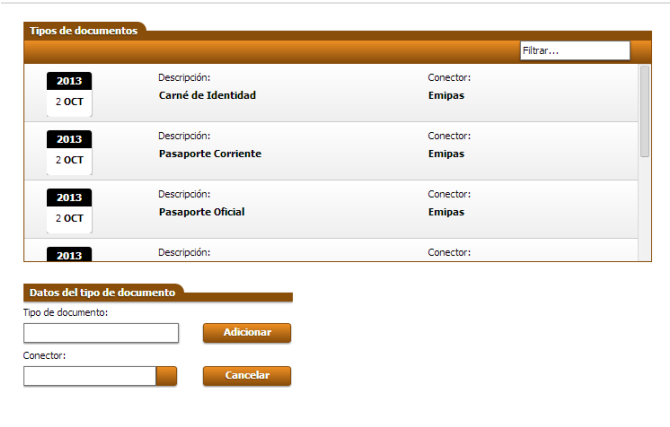
"La especificación es el producto de trabajo final que genera la ingeniería de requisitos. Sirve como base para las actividades de ingeniería de software subsecuentes. Describe la función y el desempeño de un sistema basado en computadora y las restricciones que regirán su desarrollo" (33).

Requisitos funcionales

RF1.1 - Adicionar tipo de documento

Tabla 2: Especificación del requisito funcional: Adicionar tipo de documento.

Propósito	Insertar un tipo de documento para la creación de una plantilla.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El documento no debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Seleccionar la opción "Gestionar Tipo de Documento".* 2. *Seleccionar la opción "Adicionar".*	

	3. *Introducir el nombre del tipo de documento.* 4. *Seleccionar el conector.* 5. *Seleccionar la opción "Adicionar".*
Validaciones	1. El campo "Tipo de documento" no puede estar vacío.
Post-condiciones	Se adiciona el tipo de documento.
Prototipo	

Requisitos no funcionales

"Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente se aplican a características o servicios individuales del sistema" (32). "Los requisitos no funcionales rara vez se asocian con características particulares del sistema. Por lo tanto, pueden especificar el rendimiento del sistema, la protección, la disponibilidad, y otras propiedades emergentes." (34)

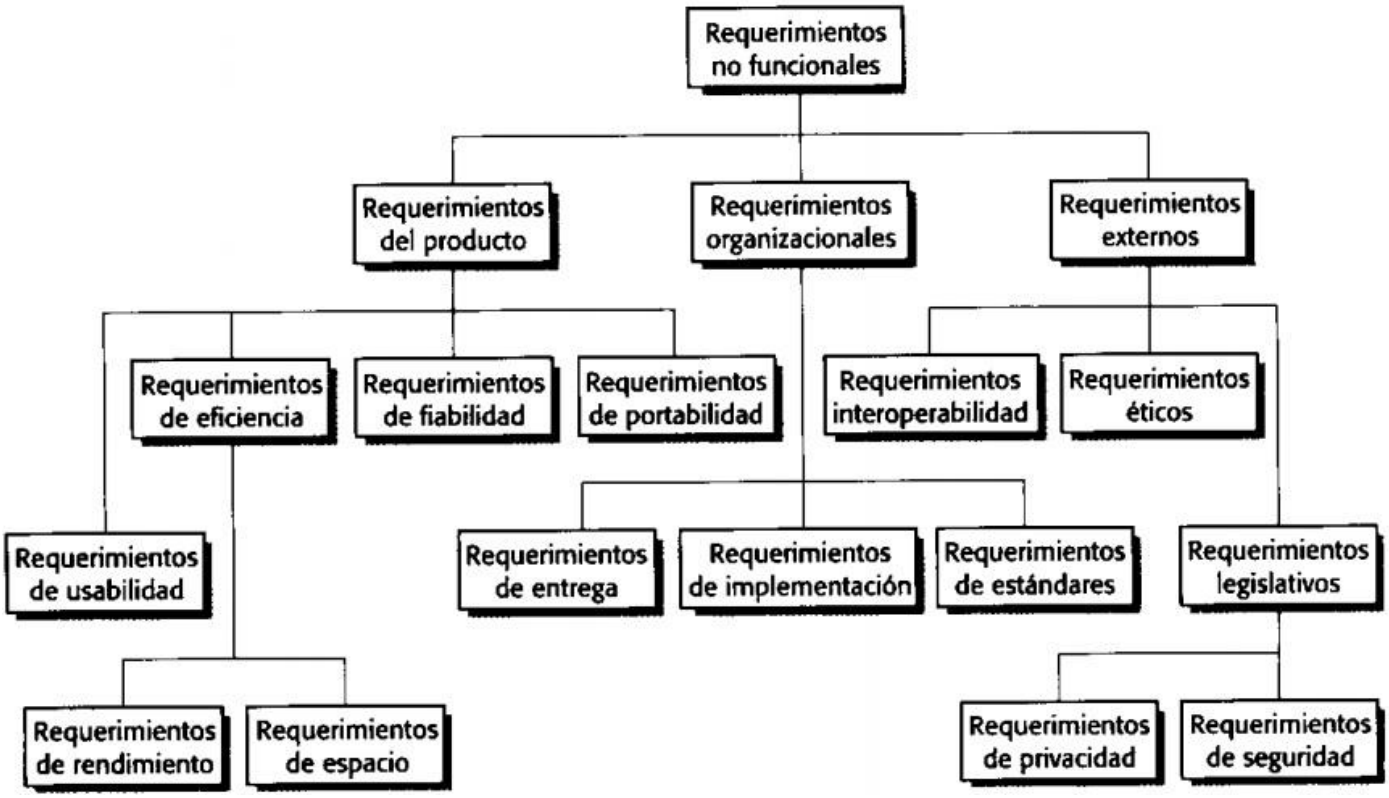


Figura 2: Clasificaciones de los requisitos no funcionales *(34)*.

RNF1 - Requerimientos no funcionales de apariencia o interfaz externa

- ✓ El sistema debe poseer un diseño lo más sencillo e intuitivo posible.
- ✓ El idioma que se utilizará será el español

RNF2 - Requerimientos no funcionales de usabilidad

- ✓ Los usuarios deberán poseer un conocimiento previo del manejo de una computadora personal.

RNF3 - Requerimientos no funcionales de portabilidad

- ✓ La aplicación no requiere ser instalada

RNF4 - Requerimientos no funcionales de software del cliente

- ✓ Los requerimientos mínimos de software necesarios son una computadora personal con plataforma del sistema operativo *Microsoft Windows XP* o superior.
- ✓ La computadora debe contar con alguno de los navegadores más utilizados como *Firefox* o *Chrome*.

RNF5 - Requerimientos no funcionales de software del servidor

- ✓ El servidor debe contar con el *framework* .NET 4.0.

RNF6 - Requerimientos no funcionales de hardware para el cliente

- ✓ Procesador *Intel Pentium* 4 de 1.2 GHz (o equivalente) y versiones posteriores y 512 MB de RAM o mayor.

Arquitectura de la solución

El documento de IEEE Std 1471-2000, declara que: “La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución” (35).

La arquitectura N-Capas es la que se utiliza para el desarrollo de la aplicación, “esta organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios” (32). Con esta arquitectura se busca la separación de la lógica del negocio con la lógica de diseño y su principal ventaja es que en caso de que surja algún cambio no hay que modificar toda la aplicación, solo el nivel afectado.

La arquitectura definida para la aplicación cuenta con 4 capas.

- Capa de “Presentación”: se encarga de ser el puente entre el usuario y el sistema brindando las interfaces mediante las cuales el cliente interactúa con el producto.
- Capa de “Negocio”: se define un servicio web encargado de realizar la conexión para obtener de la base toda la información vital para el correcto funcionamiento del proceso.
- Capa de “Acceso a datos”: tiene la responsabilidad de acceder a la base de datos para obtener los elementos que les sean solicitados de las capas superiores.
- Capa de “Datos”: se encuentran almacenados los datos de las plantillas diseñadas, ya sea en una base de datos o en formato SVG.

A continuación se muestra la arquitectura utilizada en la aplicación.

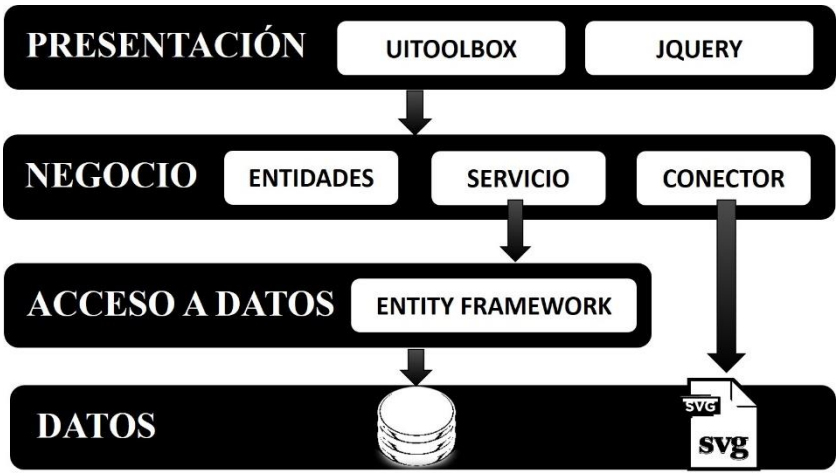


Figura 3: Arquitectura del sistema.

Estructura de la solución

La estructura de la solución se ha desarrollado encapsulando en paquetes los servicios, librerías y entidades con que cuenta el editor para dar cumplimiento a cada uno de sus requerimientos, de forma que sea los más concreta y funcional esta estructura. En la siguiente figura se representa la estructura de la solución.

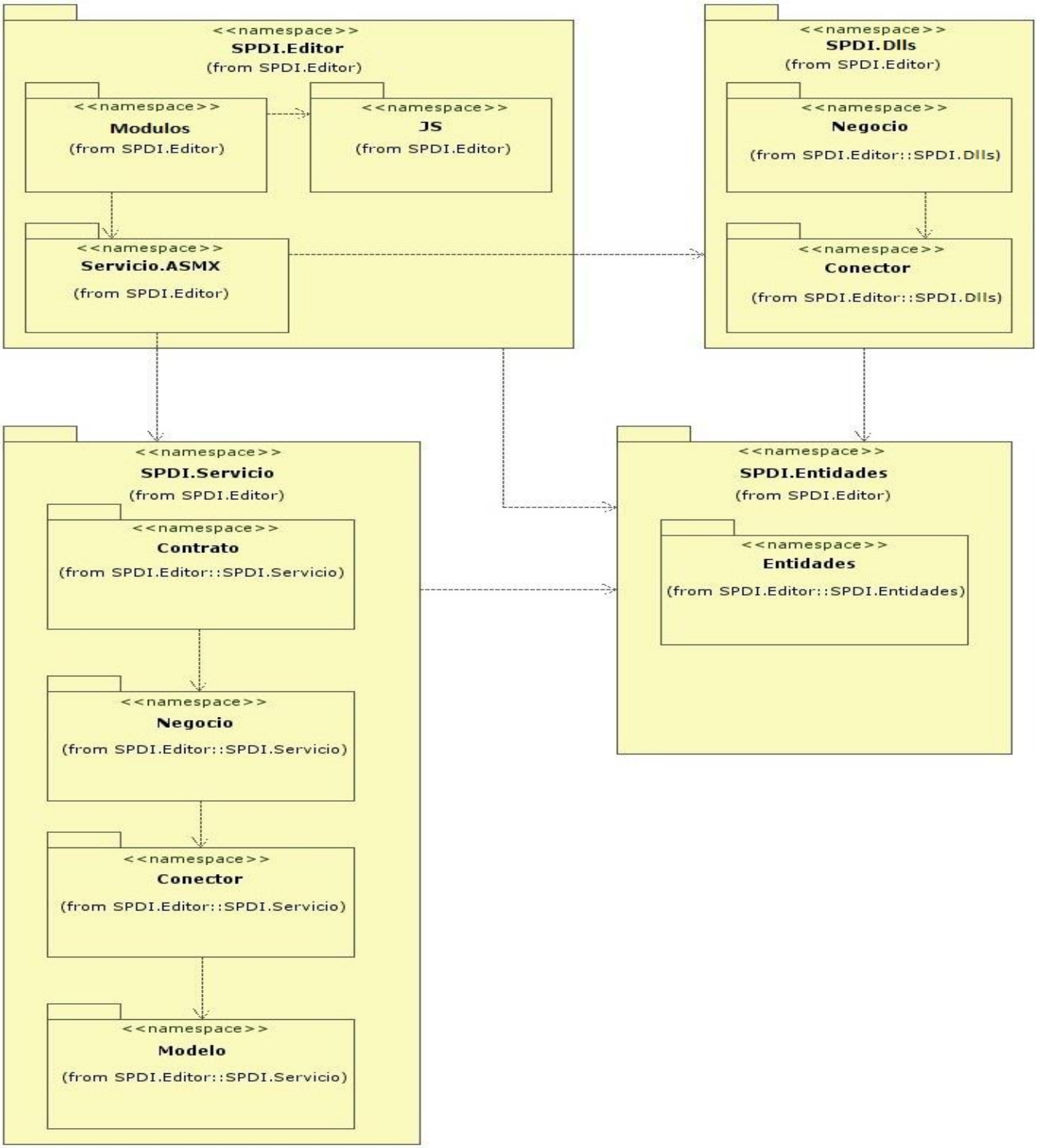


Figura 4: Estructura de la solución.

A continuación se hace una descripción del diagrama de paquetes que corresponde a la estructura del editor de plantillas del SPDI v2.0. La solución se encuentra estructurada por varios paquetes que contienen funcionalidades y relaciones con otros paquetes atendiendo a la función que tengan dentro de la solución. El paquete *SPDI.Editor* agrupa tres paquetes fundamentales para la edición de plantillas; las entidades se encuentran agrupadas en el paquete Modulos donde se tiene el diseño del editor, que mediante el paquete *Servicio.ASMX* se conecta al servicio de WCF o el componente de las Dlls, y el paquete JS es donde se encuentran todas las funcionalidades necesarias que se utilizan por el paquete Modulos.

Para el consumo del servicio de acceso a datos el paquete *Servicio.ASMX* se comunica con el paquete *SDPI.Servicio* que contiene el paquete Contrato, que es donde se especifican las funcionalidades del servicio; el paquete Negocio donde se realizan todas las validaciones antes de acceder a la base de datos; el paquete Conector que es el encargado de realizar como tal las funcionalidades en la base de datos, es donde se realizan las consultas; y el paquete Modelo que mapea la base de datos Oracle a objeto relacional. El paquete *SPDI.Servicio* se utiliza solamente para el acceso a la base de datos. El paquete *Servicio.ASMX* se comunica además con el paquete *SPDI.Dlls* para la obtención de los archivos XML o SVG.

El paquete *SPDI.Dlls* tiene un archivo de configuración que especifica la ruta donde se guardará la información referente a los XML; este paquete está conformado por los paquetes Negocio y Conector. En el paquete Negocio es donde se define si se tiene que tener una estructura específica para guardar todas las plantillas que se editen, como mecanismo de organización; y el paquete Conector es el encargado de guardarlo en lugar definido por el paquete Negocio.

Los paquetes *SPDI.Editor*, *SPDI.Dlls* y *SPDI.Servicio* se comunican con el paquete *SPDI.Entidades*, que es donde se encuentra el paquete Entidades que contiene todos los documentos.

Diagrama de clases

Un diagrama de clases proporciona una vista estructural o estática de un sistema. Muestra la naturaleza dinámica de las comunicaciones entre los objetos de las clases en el diagrama. Los principales elementos de un diagrama de clases son las cajas, las cuales son los iconos usados para representar las clases e interfaces. Cada caja es dividida en partes horizontales. La parte superior contiene el nombre de la clase y la sección media lista los atributos de la clase. La tercera sección del diagrama de clases contiene las operaciones o comportamientos de la clase. (36)

En la **figura 5** se muestra el diagrama de clases del editor de plantillas para el SPDI v2.0, donde se representan las clases que conforman la aplicación.

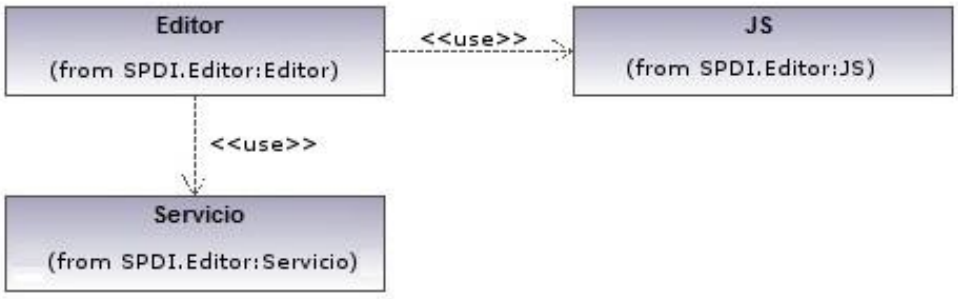


Figura 5: Diagrama de clases del editor.

Descripción de las principales clases.

Editor: Clase donde se implementa las principales funcionalidades para la carga de datos mediante la clase Servicio para la edición de una plantilla de un tipo de documento; se utiliza la clase JS para la gestión de los elementos que componen el editor.

JS: Contiene todas las clases JavaScript que se utilizan en el editor para implementar sus funcionalidades, se trabaja principalmente con las clases *jquery.js*, *embedapi.js* y *functions.js* por contener en ellas muchas de las acciones funcionales utilizadas por el editor.

Servicio: Clase encargada del acceso a datos, cuando la clase editor necesite datos para la edición de plantillas, esta clase se conecta con la base de datos para obtener los mismos.

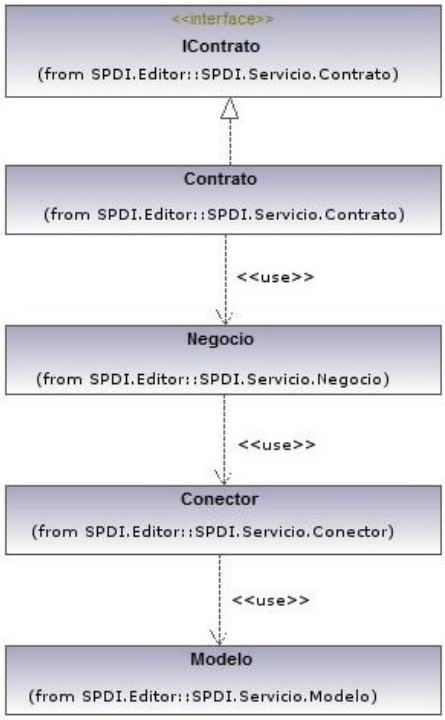


Figura 6: Diagrama de clases del servicio web.

IContrato: Clase interfaz del servicio web.

Contrato: Clase que hereda de IContrato, contiene todas las funcionalidades para que se pueda utilizar el servicio.

Negocio: Clase usada por Contrato, donde se valida cualquier funcionalidad que tenga relación con el acceso a datos.

Conector: Clase usada por Negocio para las consultas a la base de datos.

Modelo: Clase usada por Conector para el mapeo de la base de datos Oracle a objeto relacional. Estas clases son las que permiten el acceso a la base de datos.

Estas clases son las que permiten el acceso a datos para el trabajo con los tipos de documentos, los atributos y los tipos de datos.

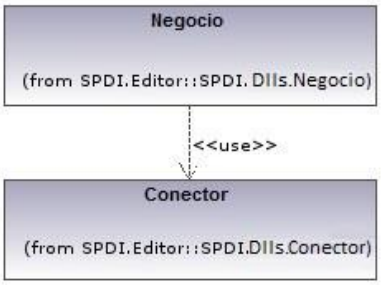


Figura 7: Diagrama de clases de las DLLs.

Negocio: Clase usada para validar cualquier funcionalidad que tenga relación con el acceso a datos, para el trabajo con las plantillas.

Conector: Clase usada por Negocio para las consultas a la base de datos.

Modelo de datos

Un modelo de datos es una representación que se emplea para describir la estructura de una base de datos. Esta representación muestra las entidades, sus atributos y las relaciones que se establecen entre estas. En la siguiente figura se muestra el modelo de datos asociado al desarrollo de la aplicación mostrando las relaciones existentes entre las entidades involucradas en el proceso.

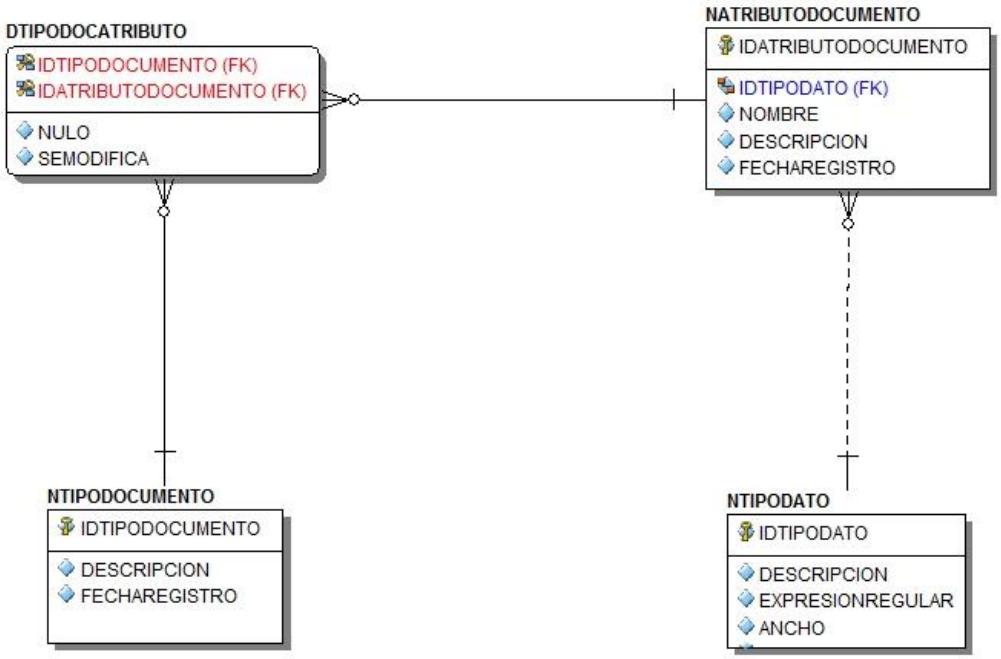


Figura 8: Modelo de datos del sistema.

Descripción de las entidades

En la **tabla 3 y 4** se hace una descripción de las entidades NATRIBUTODOCUMENTO y NTIPODOCUMENTO respectivamente, las cuales pertenecen al modelo de datos que se definió para el editor de plantillas de documentos de identificación para el SPDI v2.0, nombrando de los mismos sus atributos, con una especificación del tipo de dato, si puede o no tener valor nulo, y una pequeña descripción.

Tabla 3: Descripción de la entidad NATRIBUTODOCUMENTO.

Nombre de la Entidad: NATRIBUTODOCUMENTO.			
Descripción de la Entidad: Nomenclador de atributos de un tipo de documento.			
Servicio:			
Atributo	**Tipo de Dato**	**Nulo**	**Descripción**
IDATRIBUTODOCUMENTO	*Number(4,0)*	No	Identificador de atributos de documentos.
IDTIPODATO	*Number(4,0)*	No	Identificador del tipo de dato.
DESCRIPCION	*Varchar(5000)*	No	Descripción del nomenclador.
FECHAREGISTRO	*Date*	No	Fecha de registro del atributo del tipo de documento

NOMBRE	*Varchar(500)*	No	Nombre del atributo del tipo de documento.

Tabla 4: Descripción de la entidad NTIPODOCUMENTO.

Nombre de la Entidad: NTIPODOCUMENTO.			
Descripción de la Entidad: Nomenclador de tipos de documentos			
Servicio:			
Atributo	**Tipo de Dato**	**Nulo**	**Descripción**
IDTIPODOCUMENTO	*Number(4,0)*	No	Identificador del tipo de documento.
DESCRIPCION	*Varchar(5000)*	No	Descripción del tipo de documento.
FECHAREGISTRO	*Date*	No	Fecha en que se registra el tipo de documento.

Patrones de Diseño utilizados.

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares; se deben tener presente de ellos, su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). (37)

Cuando se participa en el desarrollo de un software, a menudo son encontrados varios problemas que ya son conocidos, por ser reiterados, o que simplemente son análogos, pero que todos responden a un determinado patrón. Se tiene en cuenta esto para conformar una selección de dichos patrones con sus soluciones para cada caso a modo de buenas prácticas de programación.

A continuación se brindan los patrones de diseño utilizados en la implementación del editor de plantillas del Sistema de Personalización de Documentos de Identificación v2.0:

Patrones GRASP.

GRASP es un acrónimo que significa *General Responsibility Asignment Software Patterns.* (33)

Cada uno de los patrones GRASP utilizados afecta como atributo de calidad a la mantenibilidad que según Barbacci “es la capacidad de someter a un sistema a reparaciones y evolución” (38), y según Booch es la “capacidad de modificar el sistema de manera rápida y a bajo costo” (39).

A continuación explicaremos los patrones *GRASP* utilizados:

Experto: “Es un patrón que se usa más que cualquier otro al asignar responsabilidades, expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender y de mantener” (33). Significa esto que las clases encargadas de crear un objeto o implementar un método son las que contienen toda la información necesaria para hacerlo, algunos ejemplos de clases donde fue usado este patrón son: *DBDocumentType, SPDIServiceBusiness, ConnectorEmipas,* etc*.*

Creador: “Guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización” (33). Mediante este patrón se garantiza que las nuevas instancias sean creadas por la clase que tiene la información necesaria para crear el objeto, algunos ejemplos de uso de este patrón son las clases: *ManagerSPDI, SPDIServiceBusiness, ManagerDocumentType,* y otras*.*

Alta Cohesión: “Es un principio que debemos tener presente en todas las decisiones de diseño: es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño” (33). Plantea esencialmente que la información contenida por una clase debe estar relacionada con la misma y ser coherente, algunas evidencias del uso de este patrón son las clases: *ManagerSPDI, DBOracleConnection, DBRequest, SPDIServiceBusiness,* y otras.

A continuación se mencionan algunos escenarios que ejemplifican los diversos grados de la cohesión funcional: (33)

> Muy baja cohesión. Una clase es la única responsable de muchas cosas en áreas funcionales muy heterogéneas.
>
> Baja cohesión. Una clase tiene la responsabilidad exclusiva de una tarea compleja dentro de un área funcional.
>
> Alta cohesión. Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.
>
> Cohesión moderada. Una clase tiene un peso ligero y responsabilidades exclusivas en unas cuantas áreas que están relacionadas lógicamente con el concepto de clase, pero no entre ellas.

Entre los beneficios con que cuenta este patrón se tiene que mejora la claridad y facilidad con que se entiende el diseño; se simplifican el mantenimiento y las mejoras en funcionalidad; a menudo se genera un bajo acoplamiento y la ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico. (33)

Bajo Acoplamiento: “El Bajo Acoplamiento estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. El Bajo Acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad” (33). Persigue mantener las clases lo menos ligadas entre sí, buscando que una transformación en una, no afecte demasiado a las demás. Ejemplo de la aplicación de este patrón son las clases: *DocPersonalizated, JobParameter, ValuableShape,* y otras más.

Controlador: “Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan” (33).

“Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión” (40). Sugiere que la lógica de negocios debe encontrarse separada de la capa de presentación para poder aumentar la reutilización de código y tener mayor control. Para aumentar la cohesión y disminuir el acoplamiento se recomienda dividir los eventos del sistema en el mayor número de controladores. Ejemplos de su uso lo evidencian las clases: *DBJob, ManagerSPDI, DBJobType, DBOracleConnection* y otras.

Patrón Polimorfismo:” Cuando por el tipo varían las alternativas o comportamientos afines, las responsabilidades del comportamiento se asignarán mediante operaciones polimórficas a los tipos en que el comportamiento presenta variantes. Es un principio fundamental en que se fundan las estrategias globales, o planes de ataque, al diseñar como organizar un sistema para que se encargue del trabajo. Un diseño basado en la asignación de responsabilidades mediante el polimorfismo puede ser extendido fácilmente para que realice nuevas variantes. Es fácil agregar las futuras extensiones que requieren las variaciones imprevistas” (33).

“Cuando identificamos variaciones en un comportamiento, asignar la clase (interfaz) al comportamiento y utilizar polimorfismo para implementar los comportamientos alternativos” (33). Su uso se ejemplifica en clases como: *ConnectorEmipas, DBJobType,* entre otras.

Conclusiones Parciales

- La propuesta de solución argumentó los beneficios que brinda la implementación del editor de plantillas para el SPDI v2.0.

- El modelo de dominio brindó una representación gráfica de las entidades que participan en la creación del editor de plantillas, así como una breve descripción de las mismas.
- La definición del rol que interactuará con el sistema, permitió conocer cuáles son las actividades que van ligadas al mismo.
- A partir del modelo de dominio se determinaron los requisitos funcionales y no funcionales con los que contará el editor, los cuáles en conjunto deben cumplir el objetivo propuesto.
- La arquitectura y los patrones de diseño demostraron las ventajas que proporcionan los mismos en la flexibilidad de la ejecución de la propuesta de solución, permitiendo la independencia entre las capas definidas.
- La estructura de la solución se definió por paquetes, teniendo en cuenta las entidades comunes para cada paquete y la función que cumplen cada uno por individual.
- El diagrama de clases proporcionó una vista estructural del editor de plantillas y las funcionalidades que lo conforman.
- El modelo de datos representó la estructura de la base de datos, permitiendo conocer como son las relaciones entre sus entidades.

CAPÍTULO 3: Implementación y pruebas.

En el presente capítulo se abordan los estándares de codificación que se utilizaron para la implementación del sistema, teniendo en cuenta que son los utilizados en la aplicación donde se integrará la solución. También se realizan los diagramas de componente y despliegue, se definen las pruebas que se le realizarán a la aplicación y se muestran los resultados de las mismas para constatar el cumplimiento de los requisitos solicitados. También se muestran los beneficios que aporta el sistema implementado.

Estándares de codificación.

El entendimiento del código fuente de un software depende en gran medida de la forma en que el programador organice el mismo. Cuando se desarrolla un sistema, el código del mismo debe escribirse de la forma más organizada y armoniosa posible. Se debe tener presente al comenzar un proyecto de software que es necesario definir un estándar de codificación para que todos los programadores trabajen de forma coordinada, y si se trabaja con un código añadido o en caso de mantenimiento de un sistema de software ya creado, se debe trabajar rigiéndose por lo que el estándar de codificación del mismo plantee. Aunque el entendimiento y mantenimiento de un sistema de software depende de varios factores, sin duda en la fase de desarrollo, en la que todos los programadores influyen es en la técnica de codificación.

El uso de técnicas de codificación y la realización de buenas prácticas de programación con el objetivo de generar un código fuente de alta calidad es de vital importancia para la calidad del producto a desarrollar y el buen rendimiento del mismo; además que todo esto posibilitará a que el software sea fácil de comprender y mantener.

Se recomienda que solo se utilice un estándar de codificación hasta que se finalice el proyecto, debido a que no es práctico, ni prudente usar varios, porque cambiarlo solo complicaría el entendimiento del código y el mantenimiento de la aplicación.

Estilos para la capitalización.

Se utilizaron los siguientes estilos para la capitalización de los identificadores.

1. Pascal: Se capitalizan la primera letra del identificador y la primera letra de cada subsiguiente palabra concatenada. Se pueden utilizar los identificadores de *Pascal case* en caso de tres o más caracteres. Uno de los ejemplos de utilización de este estilo es en el método *InicializaPagina()* en el archivo *functions.js.*
2. Camello: Se utiliza minúscula para la primera letra en el identificador y mayúscula para la primera letra de cada subsiguiente palabra concatenada. Uno de los ejemplos de utilización de este estilo es en el método *GetOrderTypeDescription().*
3. Mayúscula: Se capitalizan todas las letras en el identificador cuando el mismo consta de dos o menos letras.

4. Sensibilidad a mayúsculas: Para evitar conflictos y garantizar la interoperabilidad entre lenguajes, se siguieron las siguientes reglas sobre el uso de mayúsculas y minúsculas:
    - No utilizar espacios de nombres que solo se diferencien con el uso de las mayúsculas.
    - No utilizar espacios de nombres con nombres de clases que se diferencien solo en el uso de las mayúsculas.
    - No crear clases con propiedades que se diferencien solo en el uso de las mayúsculas.
    - No crear clases con métodos que se diferencien solo en el uso de las mayúsculas.

No se recomienda usar los nombres de los espacios de nombres y otras clases comúnmente usadas para nombrar las clases, por ejemplo *System*, *Collections* o *Forms*.

5. Evitando confusión de nombre y tipo: Varios lenguajes de programación emplean diferentes términos para la declaración de los principales tipos de identificadores. Se utilizarán nombres que describan a sus identificadores en vez del tipo de identificador.

Se utilizaron comentarios en las declaraciones de clases y funciones más complejas con el objetivo de incrementar el entendimiento del mismo, y se organizó el código de forma estructurada, en bloques de código.

Tratamiento de errores.

El tratamiento de errores en la implementación del editor, se realizó de una forma sencilla, de manera que les permita a los usuarios entender de forma fácil cualquier excepción que pueda producirse durante la utilización de la herramienta de diseño. Para el trato de las excepciones se utilizó la instrucción *try-catch* que permite controlar algunos o todos los errores que puedan producirse en un bloque de código.

```
public IList<DocumentType> Load()
{
    IList<object[]> list = new List<object[]>();
    IList<DocumentType> documentTypes = new List<DocumentType>();
    try
    {
        string querry = "select * from nordertype";
        list = connection.ExecuteQueryAsArray(querry);

        foreach (object[] item in list)
        {
            DocumentType documentType = new DocumentType(int.Parse(item[0].ToString()),
                item[1].ToString(), item[2].ToString());
            documentTypes.Add(documentType);
        }
    }
    catch (Exception ex)
    {
        ErrorConnecter connector = new ErrorConnecter();
        connector.PropagateException("Error cargando los tipos de documentos ", ex);
    }
    return documentTypes;
}
```

Figura 9: Ejemplo de tratamiento de errores.

Diagrama de componentes.

Los diagramas de componentes son los encargados de mostrar cómo se organizan y relacionan lógicamente los componentes (código fuente, binario o ejecutables) del software. Las relaciones entre sus componentes

agrupan los elementos del modelado. Los Diagramas de Componentes representan la estructura física del código. Los diagramas de componentes pueden ayudarle a establecer relaciones entre cada diagrama de clases y los archivos de código fuente. (41)

A continuación en la **figura 10** se muestra el diagrama de componentes que se confeccionó para la aplicación.

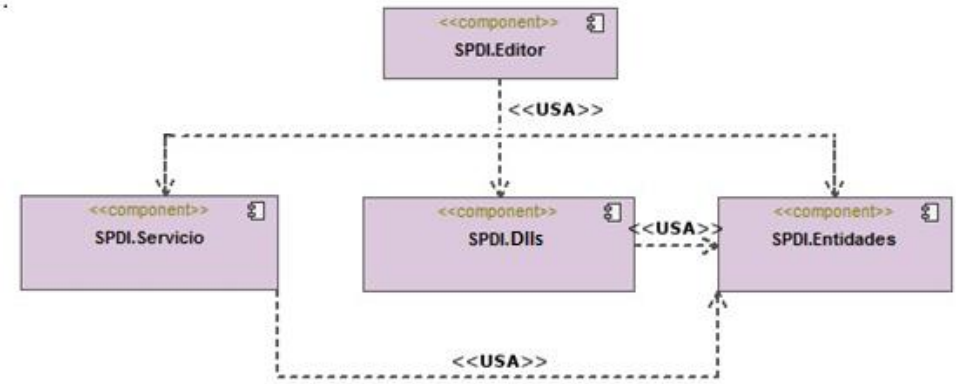


Figura 10: Diagrama de componentes.

El *SPDI.Editor* representa el editor de plantilla que usa tres componentes fundamentales: *SPDI.Servicio* encierra todo lo referente al acceso a datos en la base de datos; el *SPDI.Dlls* engloba las funcionalidades que permiten el manejo con las plantillas que se diseñen en el editor. *SPDI.Entidades* contiene las entidades del negocio que son utilizadas en los componentes anteriormente mencionados.

Diagrama de despliegue.

Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del sistema la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP. (42)

En la presente figura se brinda una representación del Diagrama de Despliegue para la solución. Se muestra el Servidor de Aplicaciones en donde se encuentra el Editor de Plantillas para el SPDI v2.0, el cual se nutre de Servicios Comunes mediante el protocolo de transporte HTTP con el Servidor de Servicios, el que interacciona con el Servidor de Base de Datos mediante el protocolo de *Oracle "TNS Oracle"* para todos los procesos de acceso a datos.

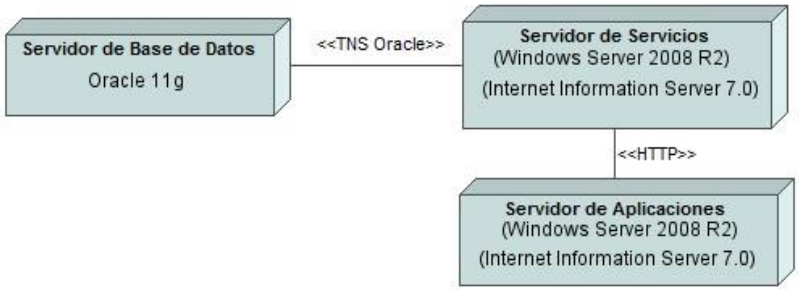


Figura 11: Diagrama de despliegue del sistema.

Pruebas.

En el proceso de desarrollo de un software cada una de las etapas por la cual pasa el mismo, tienen un peso importante que acreditará la calidad y triunfo del producto, sin duda una de estas fases es la de prueba. Mediante las pruebas que se le realicen al software se garantiza que cuente con la calidad necesaria para ser utilizado por los usuarios, atendiendo a los elementos de calidad del software que según Pressman "es el cumplimiento de los requisitos funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente" (33). Cuando se realiza una prueba, el software es probado bajo condiciones específicas, para observar o almacenar los resultados y a partir de aquí evaluar un determinado aspecto del software. Para los efectos de esta investigación se define una estrategia basada en las técnicas y métodos propuestos por Roger S. Pressman en la cual se propone realizar (33):

1. Pruebas unitarias.
2. Pruebas de caja negra.
3. Pruebas de integración.

A continuación se describen y realizan cada una de estas pruebas.

Pruebas unitarias.

Cuando se habla de prueba unitaria se entiende como un examen a una unidad determinada, cuando de programación se trata se refiere a las pruebas que se le realizan a un fragmento de código para comprobar su correcto funcionamiento, debido a la importancia que este tenga para el equipo de desarrolladores y pruebas.

EL explorador de prueba de Visual Studio está diseñado para permitir a los desarrolladores y equipos de trabajo que incorporen las pruebas unitarias en sus prácticas de desarrollo de software. Las pruebas unitarias les ayudarán a garantizar la exactitud de sus programas, comprueban que el código de la aplicación haga lo que esperaban. Las pruebas unitarias tiene el mayor efecto cuando es una parte integral de su flujo de trabajo de desarrollo de software. (43)

Diseño de casos de pruebas.

En la Ingeniería de Software los casos de pruebas estipulan si un determinado requisito en una aplicación, es parcial o completamente correcto para el analista, atendiendo a un conjunto de variables o condiciones. Se debe tener en cuenta que si quiere probar una aplicación entera, al menos debe haber un caso de prueba por cada requisito que exista. Una característica fundamental de los casos de pruebas es que tienen una entrada conocida (prueba una precondición) y una salida esperada (prueba una postcondición) las cuales deben ser formuladas antes de que se lleve a cabo la prueba.

A continuación se presenta el diseño de caso de prueba para la funcionalidad *GetDocumentType.* Los demás casos de pruebas se encuentran en los **Anexos.**

Tabla 5: Diseño de caso de prueba de la funcionalidad: *GetDocumentType.*

Escenario	Descripción	Parámetros	Respuesta del Sistema	Flujo Prueba
GetDocumentType	Devuelve los tipo de documentos	Tipo de Documento.	Devuelva una lista con los tipos de documentos.	Crear un objeto para tener acceso a la base de datos y obtener todos los tipos de documentos.

Resultados de las pruebas.

Para el desarrollo de la aplicación fue necesario realizar una sola iteración, en la misma se le realizaron pruebas unitarias a las principales funcionalidades de SPDI v2.0. A continuación se muestra el código de uno de los métodos al que se le realizó la prueba unitaria para comprobar su correcto funcionamiento. Para ver los resultados de las pruebas de las demás funcionalidades ver **Anexo 4.**

```
[TestMethod]
public void GetDocumentTypeTest()
{
    var target = new SPDIServiceWcf();
    var actual = target.GetDocumentType();
    Assert.AreNotEqual(null, actual);
}
```

Figura 12: Prueba Unitaria a *GetDocumentType*.

Figura 13: Resultado de la prueba unitaria a *GetDocumentType*.

Pruebas de caja negra.

"Las pruebas de caja negra son las que se aplican a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software. Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías: funciones incorrectas o faltantes, errores de interfaz, errores en estructuras de datos o en acceso a bases de datos externas, errores de comportamiento o desempeño, y errores de inicialización y término. (33)"

“Las pruebas de caja negra, también denominadas prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (44)*”.*

“La prueba de caja negra intenta encontrar errores de las siguientes categorías: (44)”

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

Caso de prueba: RF 2.1.1 - Insertar elementos.

Tabla 6: Caso de prueba del requisito funcional: insertar elementos.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Insertar Elementos.	E.C 1.1 Insertar figuras geométricas.	Debe mostrarse en el área de diseño dependiendo de la opción seleccionada un rectángulo, una elipse o una línea.	1. Seleccionar la opción de “Rectángulo”, “Elipse” o “Línea” en la barra de herramientas. 2. Seleccionar el lugar donde se quiere insertar en el área de diseño.
	E.C 1.2 Insertar imágenes.	Debe mostrarse en el área de diseño una imagen	1. Ir al menú principal.

		importada desde el ordenador.	2. Seleccionar la opción “Importar Imagen”.
	E.C 1.3 Insertar texto.	Debe mostrarse en el área de diseño un texto.	1. Seleccionar la opción de “Texto” en la barra de herramientas. 2. Seleccionar el lugar donde se quiere insertar en el área de diseño.

Resultados de las pruebas.

Durante el proceso de desarrollo de esta aplicación se vinieron detectando no conformidades las cuales se fueron eliminando con el transcurso del proceso. Se realizaron tres iteraciones de pruebas las cuales arrojaron respectivamente 15, 8 y ningún error lo cual demuestra que se fue trabajando en base a perfeccionar la aplicación para que el resultado final fuera el más cercano a lo esperado. Ver **figura 14.**

Figura 14: Resultado de las pruebas de caja negra.

Pruebas de integración.

“La prueba de integración es una técnica sistemática para construir 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. (33)”

Para esto fue utilizada la integración incremental. "El programa se construye y prueba en pequeños incrementos, en los cuales resulta más fácil aislar y corregir errores, es más probable que se prueben por completo las interfaces y se vuelve factible la aplicación de un enfoque de prueba sistemática. (33)"

Existen varias formas de integración incremental, como la integración descendente y la ascendente.

"Integración descendente: La prueba de integración descendente es un enfoque incremental para la construcción de la arquitectura de software. Los módulos se integran al descender por la jerarquía de control, empezando con el módulo de control principal (programa principal). Los módulos subordinados al módulo de control principal se incorporan en la estructura de una de dos maneras: primero-en-profundidad o primero-en-anchura. (33)"

"Integración ascendente: La prueba de integración ascendente, como su nombre lo indica, empieza la construcción y la prueba con módulos atómicos (es decir, componentes de los niveles más bajos de la estructura del programa). Debido a que los componentes se integran de abajo hacia arriba, siempre está disponible el procesamiento requerido para los componentes subordinados a un determinado nivel y se elimina la necesidad de resguardos. (33)"

A continuación se muestra la integración del módulo SPDIEditor con el módulo SPDIDlls.

Tabla 7: **CPI-SPDIEditor y SPDIDlls.**

Caso de Prueba: INT-EDlls
Módulo al que se integra: SPDIEditor y SPDIDlls.
Condiciones de ejecución: El módulo de SPDIEditor haya utilizado al módulo SPDIDlls para definir la ruta en donde guardar o cargar las plantillas editadas.
Descripción de la prueba: Comprobar que el módulo SDPIDlls permite definir una ruta para guardar o cargar las plantillas editadas en el módulo SPDIEditor.
Entradas/Pasos de ejecución: El módulo SPDIEditor utiliza al módulo SPDIDlls para guardar o cargar las plantillas editadas, en la ruta especificada por SPDIDlls.
Resultado esperado: Se guardan o cargan las plantillas editadas.
Evaluación: Prueba satisfactoria.

Beneficios de la solución.

- Permite la creación de cualquier tipo de documento de identificación, por ser un sistema genérico.
- Aumento de la comodidad de trabajo con las nuevas funcionalidades incorporadas.
- Cuenta con elementos básicos para el diseño como figuras geométricas, bordes, líneas, etc.
- Permite cargar imágenes de forma dinámica.
- Brinda la posibilidad de que las plantillas que sean exportadas en un formato estándar, por lo que pueden ser cargadas y editadas en otros editores de plantillas, y viceversa.
- Las imágenes de las plantillas creadas no pierden calidad al alterar su tamaño original debido a que son creadas mediante vectores.
- El servicio web puede ser utilizado en cualquier aplicación cuya finalidad sea la captura de datos.

A continuación se brinda una comparación entre algunos editores estudiados y la solución brindada, atendiendo a varios parámetros de funcionamiento.

Tabla 8: Comparación entre algunos editores y la solución brindada.

Editores / Parámetros	Editor SPDI v1.0	Editor Digital Identification (Matica)	Editor EMIPAS	Editor SPDI v2.0
Figuras Geométricas	X	X	X	X
Imágenes Predefinidas	-	X	X	X
Imágenes Cargadas De Forma Dinámica	X	X	X	X
Formato Estándar	-	-	-	X
Pérdida de Calidad	X (Píxeles)	X (Píxeles)	X (Píxeles)	- (Vectores, Píxeles)

Conclusiones parciales.

- Los estándares de codificación y el tratamiento de errores utilizados permitió realizar un trabajo de forma organizada.
- Las pruebas realizadas al SPDI v2.0 a partir de los requisitos funcionales demostró la calidad de producto final.
- El diagrama de componentes brindó una representación de cómo se organizan y relacionan los componentes del editor de plantillas, lo que posibilitó concluir como quedan complementados los mismos mediante el diagrama de despliegue.

CONCLUSIONES

Luego de realizada la fundamentación teórica que sustentó la presente investigación, definidas las características del módulo de administración de plantillas para el SPDI v2.0 y efectuado su desarrollo y validación se concluye que:

- La mejor opción para el acceso a datos en el editor de plantillas del SPDI v2.0 es la implementación de una arquitectura definida por capas bien estructuradas, utilizando la tecnología WCF de .NET ya que garantiza el bajo acoplamiento.
- Al exportar las plantillas en formato SVG quedó garantizado que no exista pérdida de calidad en las mismas y puedan ser editadas en cualquier herramienta de diseño.
- La estructura por paquetes que fue definida, permitió que sea un sistema genérico, por lo que no es necesario transformar el código fuente si se desea personalizar un nuevo tipo de documento.
- El modelo de dominio facilitó la comprensión de los conceptos que conforman el editor y cuales son utilizados para el intercambio de datos entre las capas de la arquitectura.
- A partir del modelo de dominio se determinaron los requisitos funcionales con los que debe contar el editor.
- Se demostró mediante las pruebas realizadas a la aplicación a partir de los requisitos funcionales, la calidad de la solución, evidenciado con las pruebas unitarias, de caja negra e integración.

RECOMENDACIONES

Se recomienda:

- Realizar un estudio acerca de estándares y editores para agregar mejoras en las funcionalidades de diseño del editor de plantillas para el SPDI v2.0.
- El estudio de librerías JavaScript para exportar las plantillas diseñadas en formato de imagen JPG o PNG.
- Incluir funcionalidades tales como líneas curvas, figuras personalizadas y una herramienta para cortar, que faciliten el trabajo en la edición de plantillas.

GLOSARIO DE TÉRMINOS

.NET: .*NET Framework* es una tecnología que admite la compilación y ejecución de la siguiente generación de aplicaciones y servicios Web *XML* (45).

EMIPAS: Emisión de Pasaportes.

Escalable: En informática, propiedad deseable en un sistema, red o proceso que indica su habilidad para poder hacerse más grande sin perder calidad en sus servicios.

Gradiente: Medida de la inclinación de una curva (con frecuencia una línea recta). Se define como la relación del cambio vertical (elevación) con respecto al cambio horizontal (recorrido) para una línea no vertical. En coordenadas Cartesianas rectangulares, el gradiente es la razón a la cual cambia la coordenada y con respecto a la coordenada x.

GRASP: *General Responsibility Assignment Software Patterns* (Patrones generales de software para asignación de responsabilidades).

IEEE: *Institute of Electrical and Electronics Engineers* (Asociación técnico-profesional mundial dedicada a la estandarización).

MININT: Ministerio del Interior de la República de Cuba.

MSF *for* CMMI: *Microsoft Solutions Framework for Capability Maturity Model Integration* (Metodología formal para la Ingeniería de *Software*).

Renderizado: (Del inglés *rendering*, renderizar, renderizado, renderización o interpretación en español). La renderización es el proceso de generar una imagen (imagen en 3D o una animación en 3D) a partir de un modelo, usando una aplicación de computadora.

UCI: Universidad de las Ciencias Informáticas.

WCF: *Windows Communication Foundation* (Marco de trabajo para la creación de aplicaciones orientadas a servicios).

BIBLIOGRAFÍA REFERENCIADA

1. Identification cards -- Physical characteristics ISO/IEC 7810:2003. *ISO/IEC 7810:2003 Identification cards -- Physical characteristics.* [Online] 05 03, 2010. [Cited: 05 20, 2014.] http://www.iso.org/iso/catalogue_detail?csnumber=31432.

2. Fernández, Yanet Silva. Entrevista con el cliente. *Editor de plantillas de documentos de identificación.* La Habana, 11 16, 2013.

3. León, Rolando Alfredo Hernández and Coello, Sayda. *El proceso de investigación científica.* Ciudad de La Habana : Editorial Universitaria, 2011. ISBN 978-959-16-1307-3.

4. Mastrapa, Yudenia Ramírez. *Los documentos de viaje en la lucha contra el terrorismo.* La Habana : s.n., 2010.

5. Ecured. *Ecured.* [Online] 08 02, 2011. [Cited: 05 22, 2014.] http://www.ecured.cu/index.php/Editor_gr%C3%A1fico_%28Inform%C3%A1tica%29.

6. Concepto en Definición ABC. *Concepto en Definición ABC.* [Online] 05 24, 2010. [Cited: 05 22, 2014.] http://www.definicionabc.com/comunicacion/editor.php.

7. ISO/IEC 7810:2003 - Identification cards -- Physical characteristics. [Online] 05 03, 2010. [Cited: 11 16, 2013.] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=31432.

8. Historia del DNI. *Historia del DNI.* [Online] 04 19, 2010. [Cited: 05 22, 2014.] http://historiageneral.com/2014/04/10/breve-historia-del-dni-en-espana.

9. Gamero, Alejandro. El primer DNI y otras curiosidades del Documento Nacional de Identidad. *El primer DNI y otras curiosidades del Documento Nacional de Identidad .* [Online] 04 16, 2013. [Cited: 05 22, 2014.] http://www.lapiedradesisifo.com/2013/04/16/el-primer-dni-y-otras-curiosidades-del-documento-nacional-de-identidad/.

10. Sierralta, María Paz Nuñez y Pía. La evolución de la cédula de identidad: De una simple libreta a una tarjeta con microchip. *La evolución de la cédula de identidad: De una simple libreta a una tarjeta con microchip.* [Online] 05 06, 2013. [Cited: 05 22, 2014.] http://www.latercera.com/noticia/nacional/2013/05/680-522046-9-la-evolucion-de-la-cedula-de-identidad-de-una-simple-libreta-a-una-tarjeta-con.shtml.

11. Datys. Datys. [Online] [Cited: 01 20, 2014.] http://www.datys.cu/wpinfproducto.aspx?5.

12. Mr. Gerd Schaefer. Security. [Online] 09 14, 2010. [Cited: 05 20, 2014.] http://www.securitystockwatch.com/Interviews/in_Boardroom_DIS.html.

13. CardFive. [Online] [Cited: 01 20, 2013.] http://www.impresorasdetarjetaszebra.com/zebra/card-five.

14. Turner, Michael S. V. *Microsoft Solutions Framework Essentials : Building Successful Technology Solutions.* s.l. : Microsoft Press, 2006. 9780735623538.

15. Mary Beth Chrissis, Mike Konrad y Sandy Shrum. *CMMI: Guidelines for Process Integration and Product Improvement (2ª edición).* s.l. : Addison-Wesley Professional, 2011. 0321711505.

16. Velazquez, Cristina Suárez. Representacion de imágenes. [Online] 11 04, 2012. [Cited: 10 17, 2013.] http://informaticaaplicadaalabiología.blogspot.com/2012/11/representacion-de-imagenes.html.

17. Desarrollo y Diseño Web. [Online] [Cited: 10 17, 2013.] http://www.islavisual.com/articulos/desarrollo_web/svg-introduccion-a-los-graficos-..

18. Guia Breve de Tecnologias XML. [Online] [Cited: 10 17, 2013.] http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML..

19. Microsoft Development Network. [Online] [Cited: 11 26, 2013.] http://msdn.microsoft.com/es-es/library/4w3ex9c2%28v=vs.100%29.aspx.

20. Microsoft Development Network. [Online] [Cited: 11 26, 2013.] http://msdn.microsoft.com/es-es/library/kx37x362%28v=vs.90%29.aspx.

21. Programación 1. [Online] 09 08, 2009. [Cited: 10 17, 2013.] http://programacion1abundiz.blogspot.com/2009/09/ventajas-del-c-..

22. JQUERY. [Online] [Cited: 10 17, 2013.] http://jquery.com/..

23. Capacity. [Online] [Cited: 10 17, 2013.] http://blog.capacityacademy.com/2013/03/16/jquery-que-es-..

24. Microsoft Development Network. [Online] [Cited: 11 26, 2013.] http://msdn.microsoft.com/es-es/library/fx6bk1f4%28v=vs.90%29.aspx.

25. msdn. [Online] [Cited: 12 10, 2013.] http://msdn.microsoft.com/es-es/library/ms731082.aspx.

26. Altova. [Online] [Cited: 01 15, 2014.] http://www.altova.com/es/umodel.html.

27. Ecured. [Online] [Cited: 01 22, 2014.] http://www.ecured.cu/index.php/Oracle.

28. msdn. msdn. [Online] [Cited: 01 22, 2014.] http://msdn.microsoft.com/es-es/library/bb397926.aspx.

29. Tú Informática F@cil. [Online] 05 01, 2010. [Cited: 04 28, 2014.] http://www.tuinformaticafacil.com/oracle-pl-sql/introduccion-a-la-programacion-con-oracle-pl-sql.

30. pl/sql tutorial. [Online] 10 15, 2009. [Cited: 04 28, 2014.] http://plsql-tutorial.com/plsql-advantages.htm.

31. Larman, Craig. *UML y patrones.* s.l. : PEARSON ALHAMBRA, 2003. 9788420534381.

32. Sommerville, Ian. *Ingenieria del Software.* Madrid : Pearson education, 2005. 84-7829-074-5.

33. Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico .* s.l. : MCGRAW-HILL. 9701054733.

34. Sommerville. *INGENIERÍA DE SOFTWARE. Séptima edición.* Madrid : PEARSON EDUCACIÓN, 2005. 84-7829-074-5.

35. Escárate, Rodrigo. Gerencia. [Online] [Cited: 01 17, 2014.] http://www.emb.cl/gerencia/articulo.mvc?xid=1701.

36. Pressman. *An Introduction to UML.*

37. Tedeschi, Nicolás. Developer Network. *Developer Network.* [Online] 10 23, 2003. [Cited: 03 13, 2014.] http://msdn.microsoft.com/es-es/library/bb/972240.aspx.

38. Barbacci, M., Klein, M., Longstaff, T., & Weinstock, C. Quality Attributes. *Quality Attributes.* [Online] Carnegie Mellon University, 06 27, 1995. [Cited: 05 20, 2014.] http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.021.html .

39. Booch, G., Rumbaugh, J., & Jacobson, I. *The UML Modeling Language .* 1999.

40. Mora, Roberto Canales. *Patrones GRASP.* Madrid : Autentia, 2003.

41. Umodel. *Umodel.* [Online] [Cited: 03 14, 2014.] http://www.altova.com/es/umodel/uml-component-diagrams.html.

42. Ecured. *Ecured.* [Online] [Cited: 03 14, 2014.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

43. Developer . *Developer .* [Online] [Cited: 03 14, 2014.] http://msdn.microsoft.com/es-es/library/hh694602.aspx.

44. Tenorio, Roberto Ruiz. *Las Pruebas de Software y su Importancia en las Organizaciones.* VeraCruz : s.n., 2010.

45. Microsoft. Microsoft Developer Network. *Microsoft Developer Network.* [Online] [Cited: 05 07, 2014.] http://msdn.microsoft.com/es-es/library/zw4w595w%28v=vs.110%29.aspx.

46. Microsoft Development Network. [Online] [Cited: 11 26, 2013.] http://msdn.microsoft.com/es-es/library/cc425392%28v=vs.71%29.aspx.

47. Manual Web. [Online] 05 29, 2019. [Cited: 11 26, 2013.] http://www.manualweb.net/xslt/introduccion-a-xslt/.

ANEXOS

Anexo 1. Catálogo de Requisitos Funcionales.

RF1 - Gestionar Tipo de documento.

- ✓ **RF1.2** - Modificar tipo de documento.
- ✓ **RF1.3** - Eliminar tipo de documento.
- ✓ **RF1.4** - Mostrar tipo de documento.

RF2 - Editar plantillas.

- ✓ **RF2.1**- Crear plantilla.
    - RF2.1.1- Insertar elementos.
    - RF2.1.2- Modificar propiedades de los elementos insertados.
    - RF2.1.3- Insertar atributos.
    - RF2.1.4- Guardar la plantilla creada.
- ✓ **RF2.2** - Modificar plantilla.
    - RF2.2.1- Modificar elementos.
    - RF2.2.2- Modificar propiedades de los elementos insertados.
    - RF2.2.3- Insertar atributos.
    - RF2.2.4- Guardar la plantilla modificada.

RF3 - Gestionar atributos de documentos.

- ✓ **RF3.1** - Adicionar atributos de documentos.
- ✓ **RF3.2** - Modificar atributos de documentos.
- ✓ **RF3.3** - Eliminar atributos de documentos.
- ✓ **RF1.4** - Mostrar atributos de documentos.

RF4 - Gestionar tipo de dato.

- ✓ **RF4.1** - Adicionar tipo de dato.
- ✓ **RF4.2** - Modificar tipo de dato.
- ✓ **RF4.3** - Eliminar tipo de dato.
- ✓ **RF1.4** - Mostrar tipo de dato.

RF5 - Buscar.

- ✓ **RF5.1** - Buscar tipo de documento.
- ✓ **RF5.2** - Buscar atributos de documentos.
- ✓ **RF5.3** - Buscar tipo de dato.

Requisitos del servicio web *SPDIService*:

RF6 - Gestionar Tipo de documento.

- ✓ **RF6.1** - Adicionar tipo de documento.
- ✓ **RF6.2** - Modificar tipo de documento.
- ✓ **RF6.3** - Eliminar tipo de documento.

RF7 - Gestionar tipo de dato.

- ✓ **RF7.1** - Adicionar tipo de dato.
- ✓ **RF7.2** - Modificar tipo de dato.
- ✓ **RF7.3** - Eliminar tipo de dato.

RF8 - Gestionar atributos de documentos.

- ✓ **RF8.1** - Adicionar atributos de documentos.
- ✓ **RF8.2** - Modificar atributos de documentos.
- ✓ **RF8.3** - Eliminar atributos de documentos.

RF9 - Buscar.

- ✓ **RF9.1** - Buscar tipo de documento.
- ✓ **RF9.2** - Buscar atributos de documentos.
- ✓ **RF9.3** - Buscar tipo de dato.

Anexo 2. Descripción de Requisitos Funcionales.
RF1 Gestionar Tipo de documento.

Tabla 9: RF1.2 Modificar tipo de documento.

Propósito	Modificar un tipo de documento atendiendo a uno o varios parámetros.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Seleccionar la opción "Gestionar Tipo de Documento".* 2. *Seleccionar el tipo de documento que quiere modificar.* 3. *Introducir el nombre.* 4. *Seleccionar el conector.* 5. *Seleccionar la opción "Guardar".*	
Validaciones	1. El campo "Tipo de documento" no puede estar vacío.	
Post-condiciones	Se modifica el tipo de documento.	
Prototipo		

Tabla 10: RF1.3 Eliminar tipo de documento.

Propósito	Eliminar un tipo de documento	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Seleccionar la opción "Gestionar Tipo de Documento".* 2. *Seleccionar el tipo de documento que se quiere eliminar.* 3. *Seleccionar la opción "Eliminar".*	
Validaciones		
Post-condiciones	Se elimina el tipo de documento.	
Prototipo		

Tabla 11: RF1.4 Mostrar tipo de documento.

Propósito	Mostrar los tipos de documento.
Roles	Usuario
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento debe existir.

Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	6. *Seleccionar la opción "Gestionar Tipo de Documento".* 7. *Se muestran los tipos de documentos.*	
Validaciones		
Post-condiciones	Se muestran los tipos de documentos.	
Prototipo		

RF2 Editar plantillas.

Tabla 12: RF2.1 Crear plantilla.

Propósito	Crea una plantilla para el diseño de un documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento de la plantilla a crear debe existir en la base de datos.	
Entidades tratadas	Entidad	Atributos

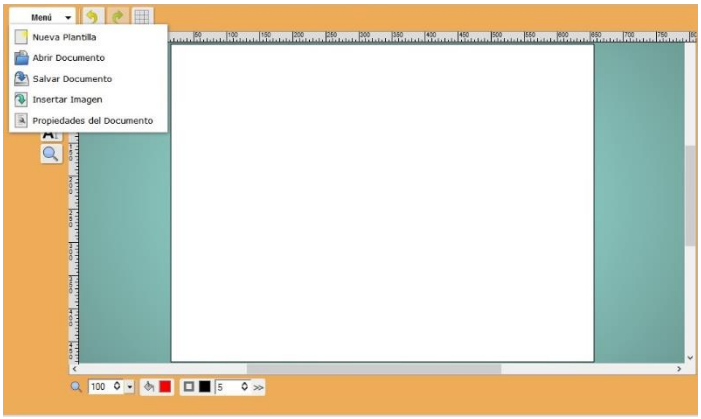
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	*1. Seleccionar la opción "Nueva plantilla".* *2. Seleccionar la opción "Aceptar".*	
Validaciones		
Post-condiciones	Se crea la plantilla seleccionada.	
Prototipo		

Tabla 13: RF2.1.1. Insertar elementos.

Propósito	Insertar elementos a la plantilla previamente creada para realizar el diseño de la misma.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento de la plantilla a crear debe existir en la base de datos. La plantilla debe estar creada.	
Elementos tratados	Elemento	Propiedades
	Figuras Geométricas (rectángulo, elipse, línea)	Ancho. Largo. Posición. Ancho de línea. Tipo de línea.

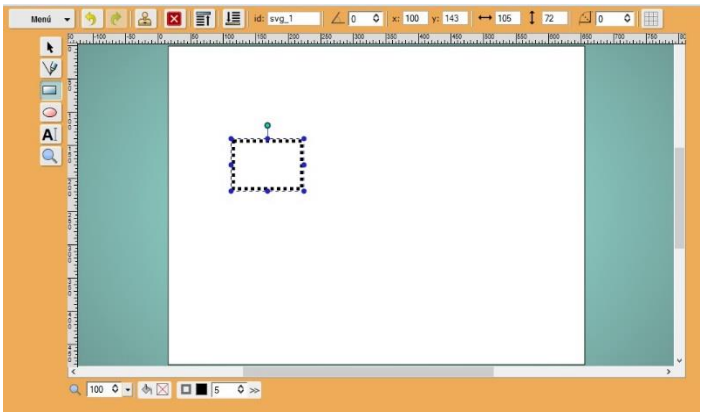
		Color de relleno. Color de línea. Rotación.
	Imagen	Ancho. Largo. Posición. Rotación.
	Texto	Posición. Rotación. Tipo de texto. Tamaño de fuente.
Descripción	*1. Seleccionar el elemento en la barra de herramientas* *2. Seleccionar la parte de la plantilla en que se quiere que se inserte.*	
Validaciones		
Post-condiciones	Se inserta el elemento	
Prototipo		

Tabla 14: RF2.1.2 Modificar propiedades de los elementos insertados.

Propósito	Modificar las propiedades de los elementos que fueron previamente insertados en la plantilla que se está diseñando	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. La plantilla debe estar creada. Debe existir algún elemento en esta plantilla.	
Elementos tratados	Elemento	Propiedades

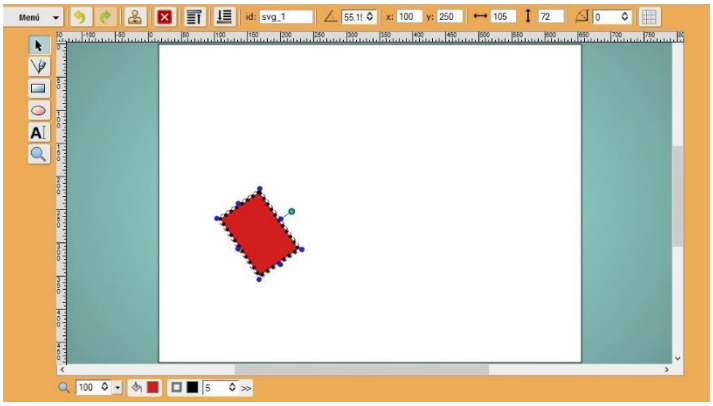
	Figuras Geométricas (rectángulo, elipse, línea)	Ancho. Largo. Posición. Ancho de línea. Tipo de línea. Color de relleno. Color de línea. Rotación.
	Imagen	Ancho. Largo. Posición. Rotación.
	Texto	Posición. Rotación. Tipo de texto. Tamaño de fuente.
Descripción	*1. Se selecciona el elemento al cual se le quieren modificar las propiedades.* *2. Se Modifican las propiedades en el panel de propiedades.*	
Validaciones	1. Las propiedades numéricas no pueden contener caracteres alfabéticos.	
Post-condiciones	Se modifica las propiedades del elemento seleccionado.	
Prototipo		

Tabla 15: RF2.1.3 Insertar atributos.

Propósito	Insertar los atributos con los que cuenta el tipo de documento al cual se le

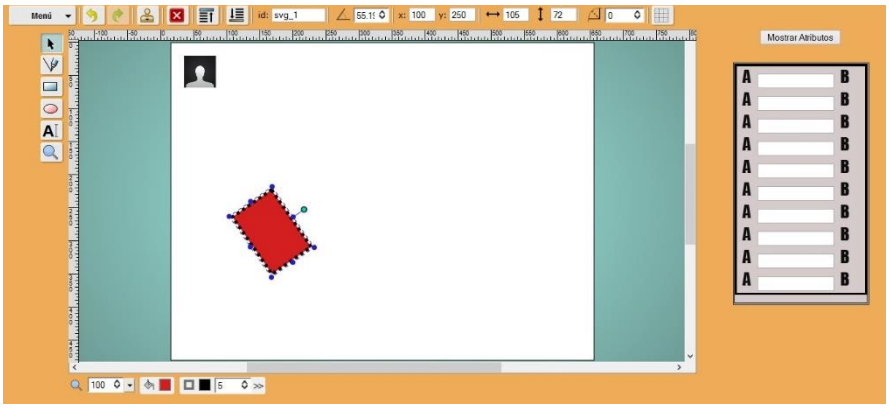
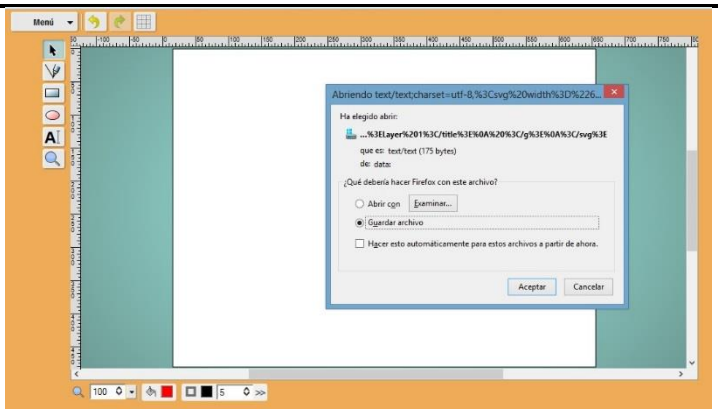
	está creado la plantilla.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. La plantilla debe estar creada.	
Elementos tratados	Elemento	Propiedades
	Imagen	Ancho. Largo. Posición. Rotación.
	Texto	Posición. Rotación. Tipo de texto. Tamaño de fuente.
Descripción	1. *Se selecciona la opción de "Mostrar Atributos".* 2. *Se insertan los atributos que tiene el tipo de documento en cuestión.*	
Validaciones		
Post-condiciones	Se insertan los atributos del tipo de documento a la plantilla	
Prototipo		

Tabla 16: RF2.1.4 Guardar la plantilla creada.

Propósito	Guardar la plantilla creada.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento de la plantilla a guardar debe existir en la base de datos.	
Entidades tratadas	Entidad	Atributos

	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Seleccionar la opción "Menú".* 2. *Seleccionar la opción "Salvar Documento".* 3. *Seleccionar la opción "Aceptar".* 4. *Escoger ruta donde se desea guardar la plantilla.* 5. *Seleccionar la opción "Guardar".*	
Validaciones		
Post-condiciones	Se guarda la plantilla.	
Prototipo		

RF2.2 Modificar plantilla.

Tabla 17: RF2.2.1. Insertar elementos.

Propósito	Insertar o eliminar elementos en la plantilla previamente diseñada	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. La plantilla debe estar creada.	
Elementos tratados	Elemento	Propiedades
	Figuras Geométricas (rectángulo, elipse, línea)	Ancho. Largo. Posición.

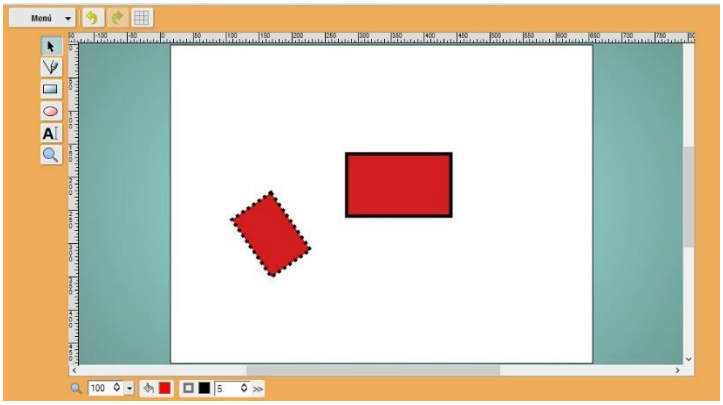
		Ancho de línea. Tipo de línea. Color de relleno. Color de línea. Rotación.
	Imagen	Ancho. Largo. Posición. Rotación.
	Texto	Posición. Rotación. Tipo de texto. Tamaño de fuente.
Descripción	1. *Se carga la plantilla previamente diseñada.* 2. *Seleccionar el elemento en la barra de herramientas* 3. *Seleccionar la parte de la plantilla en que se quiere que se inserte.*	
Validaciones		
Post-condiciones	Se inserta el elemento	
Prototipo		

Tabla 18: RF2.2.2 Modificar propiedades de los elementos insertados

Propósito	Modificar las propiedades de los elementos que fueron previamente insertados en la plantilla diseñada	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. La plantilla debe estar creada. Debe existir algún elemento en esta plantilla.	
Elementos tratados	Elemento	Propiedades

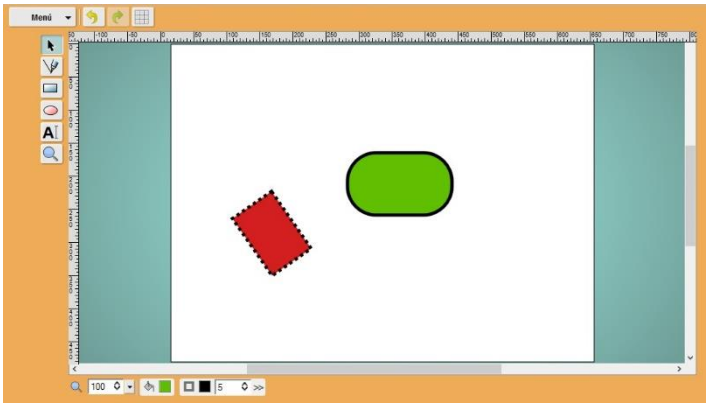
	Figuras Geométricas (rectángulo, elipse, línea)	Ancho. Largo. Posición. Ancho de línea. Tipo de línea. Color de relleno. Color de línea. Rotación.
	Imagen	Ancho. Largo. Posición. Rotación.
	Texto	Posición. Rotación. Tipo de texto. Tamaño de fuente.
Descripción	*1. Se carga la plantilla previamente diseñada* *2. Se selecciona el elemento al cual se le quieren modificar las propiedades.* *3. Se Modifican las propiedades en el panel de propiedades.*	
Validaciones	1. Las propiedades numéricas no pueden contener caracteres alfabéticos.	
Post-condiciones	Se modifica las propiedades del elemento seleccionado.	
Prototipo		

Tabla 19: RF2.2.3 Insertar atributos

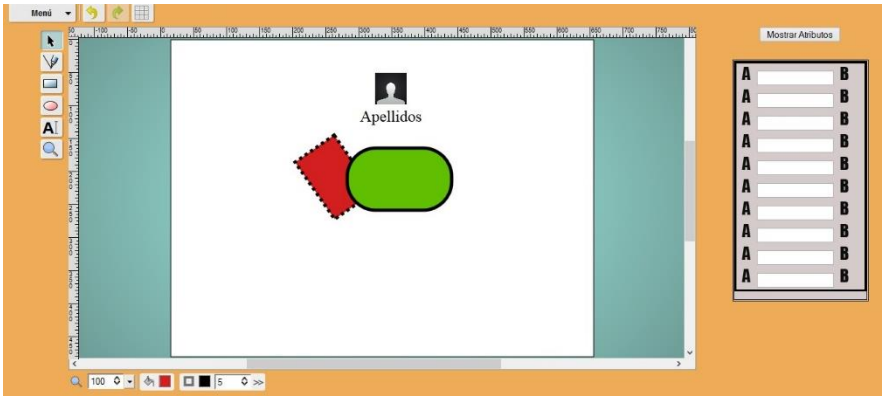
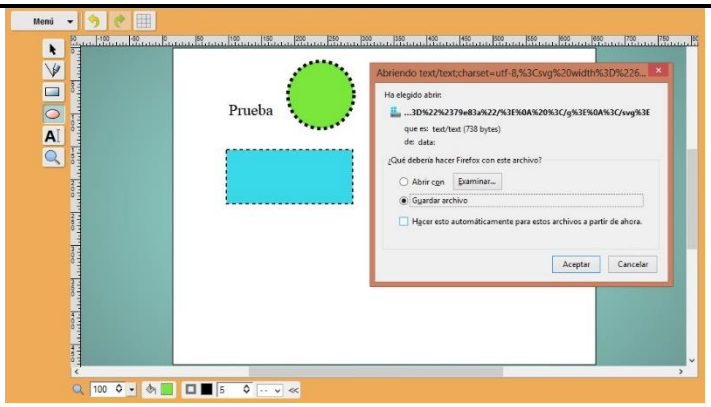
Propósito	Insertar los atributos con los que cuenta el tipo de documento al cual se le creo la plantilla	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. La plantilla debe estar creada.	
Elementos tratados	Elemento	Propiedades
	Imagen	Ancho. Largo. Posición. Rotación.
	Texto	Posición. Rotación. Tipo de texto. Tamaño de fuente.
Descripción	1. *Se carga la plantilla previamente diseñada* 2. *Se selecciona la opción de "Mostrar Atributos".* 3. *Se insertan los atributos que tiene el tipo de documento.*	
Validaciones		
Post-condiciones	Se insertan los atributos del tipo de documento a la plantilla	
Prototipo		

Tabla 20: RF2.2.4 Guardar la plantilla modificada.

Propósito	Guardar los cambios realizados en una plantilla.
Roles	Usuario
Pre-condiciones	El usuario debe estar autenticado.

	La plantilla debe estar creada con anterioridad.	
Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Seleccionar la opción "Menú".* 2. *Seleccionar la opción "Abrir Documento".* 3. *Buscar la ruta en donde se encuentra la plantilla.* 4. *Seleccionar la opción "Abrir".* 5. *Modificar plantilla.* 6. *Seleccionar la opción "Menú".* 7. *Seleccionar la opción "Salvar Documento".* 8. *Seleccionar la opción "Aceptar".* 9. *Escoger ruta donde se desea guardar la plantilla.* 10. *Seleccionar la opción "Guardar".*	
Validaciones		
Post-condiciones	Se guarda la plantilla con los cambios realizados.	
Prototipo		

RF3 Gestionar atributos de documentos.

Tabla 21: RF3.1 Adicionar atributos de documentos.

Propósito	Adicionar atributos de documentos.

Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El atributo no debe existir.	
Entidades tratadas	Entidad	Atributos
	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Seleccionar la opción "Gestionar nomencladores".* 2. *Introducir el nombre del atributo.* 3. *Seleccionar el tipo de dato.* 4. *Seleccionar la opción "Adicionar".* 5. *Seleccionar la opción "Aceptar".*	
Validaciones	1. El campo "Nombre" no debe estar vacío. 2. El campo "Tipo de dato" no debe estar vacío.	
Post-condiciones	Se adicionan los atributos de documentos.	
Prototipo		

Tabla 22: RF3.2 Modificar atributos de documentos.

Propósito	Modificar atributos de documentos atendiendo a uno o varios parámetros.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El atributo debe existir.	
Entidades tratadas	Entidad	Atributos

	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Seleccionar la opción "Gestionar nomencladores".* 2. *Seleccionar el atributo que se quiere modificar.* 3. *Seleccionar la opción "Editar".* 4. *Introducir el nombre del atributo.* 5. *Seleccionar el tipo de dato.* 6. *Seleccionar la opción "Aceptar".*	
Validaciones	El campo "Nombre" no debe estar vacío. El campo "Tipo de dato" no debe estar vacío.	
Post-condiciones	Se modifican los atributos de documentos.	
Prototipo		

Tabla 23: RF3.3 Eliminar atributos de documentos.

Propósito	Eliminar atributos de documentos.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El atributo debe existir.	
Entidades tratadas	Entidad	Atributos

	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Seleccionar la opción "Gestionar nomencladores".* 2. *Selecciona el atributo que se desea eliminar.* 3. *Selecciona la opción "Eliminar".*	
Validaciones		
Post-condiciones	Se eliminan los atributos del documento seleccionado.	
Prototipo		

Tabla 24: **RF3.4 Mostrar atributos de documentos.**

Propósito	Mostrar atributos de documentos.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El atributo debe existir.	
Entidades tratadas	Entidad	Atributos

	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Seleccionar la opción "Gestionar nomencladores".* 2. *Selecciona la opción "Atributos".* 3. *Se muestran los atributos de los documentos.*	
Validaciones		
Post-condiciones	Se muestran los atributos de los documentos.	
Prototipo		

RF4 Gestionar tipo de dato.

Tabla 25: RF4.1 Adicionar tipo de dato.

Propósito	Adicionar un tipo de dato.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato no debe existir.	
Entidades tratadas	Entidad	Atributos

	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	1. *Seleccionar la opción "Gestionar nomencladores".* 2. *Selecciona la opción "Tipo de dato".* 3. *Colocar el nombre del tipo de dato.* 4. *Seleccionar la opción "Aceptar".* 5. *Se añade el tipo de dato.*	
Validaciones		
Post-condiciones	Se adiciona el tipo de dato.	
Prototipo		

Tabla 26: RF4.2 Modificar tipo de dato.

Propósito	Modificar el tipo de dato de uno o varios atributos de un documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato debe existir.	
Entidades tratadas	Entidad	Atributos

	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	6. *Seleccionar la opción "Gestionar nomencladores".* 7. *Selecciona la opción "Tipo de dato".* 8. *Selecciona el tipo de dato.* 9. *Selecciona la opción "Modificar".* 10. *Se modifica el tipo de dato.*	
Validaciones		
Post-condiciones	Se modifica el tipo de dato.	
Prototipo		

Tabla 27: RF4.3 Eliminar tipo de dato.

Propósito	Eliminar un tipo de dato para uno o varios atributos de un documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato debe existir.	
Entidades tratadas	Entidad	Atributos

	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	11. *Seleccionar la opción "Gestionar nomencladores".* 12. *Selecciona la opción "Tipo de dato".* 13. *Selecciona el tipo de dato.* 14. *Selecciona la opción "Eliminar".* 15. *Se elimina el tipo de dato.*	
Validaciones		
Post-condiciones	Se elimina el tipo de dato.	
Prototipo		

Tabla 28: RF4.4 Mostrar tipo de dato.

Propósito	Mostrar un tipo de dato para uno o varios atributos de un documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato debe existir.	
Entidades tratadas	Entidad	Atributos

	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	16. *Seleccionar la opción "Gestionar nomencladores".* 17. *Selecciona la opción "Tipo de dato".* 18. *Se muestran los tipos de datos.*	
Validaciones		
Post-condiciones	Se muestran los tipos de datos.	
Prototipo		

RF5 Buscar.

Tabla 29: RF5.1 - Buscar tipo de documento.

Propósito	Buscar un tipo de documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento debe existir.	
Entidades tratadas	Entidad	Atributos

	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	*Introducir el nombre en el campo "Filtrar".*	
Validaciones		
Post-condiciones	Se encuentra el tipo de documento.	
Prototipo		

Tabla 30: RF5.2 - Buscar atributos de documentos.

Propósito	Buscar los atributos correspondientes a un tipo de documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. Los atributos deben existir.	
Entidades tratadas	Entidad	Atributos
	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO

Descripción	*Introducir el nombre en el campo "Filtrar".*
Validaciones	
Post-condiciones	Se muestran los atributos.
Prototipo	

Tabla 31: RF5.3 - Buscar tipo de dato.

Propósito	Buscar un tipo de dato.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	*Introducir el nombre en el campo "Filtrar".*	
Validaciones		
Post-condiciones	Se muestra el tipo de dato.	
Prototipo		

Servicio web *SPDIService*

RF6 Gestionar Tipo de documento.

Tabla 32: RF6.1 Adicionar tipo de documento.

Propósito	Insertar un tipo de documento para la creación de una plantilla	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El documento no debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Se adiciona el tipo de documento*	
Validaciones	1. Validar que el campo “Tipo de documento” no este vacío.	
Post-condiciones	Se adiciona el tipo de documento.	
Prototipo	No procede	

Tabla 33: RF6.2 Modificar tipo de documento.

Propósito	Modificar un tipo de documento atendiendo a uno o varios parámetros.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento debe existir.	
Entidades tratadas	Entidad	Atributos

	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Se modifica el tipo de documento.*	
Validaciones	1. Validar que el campo “Tipo de documento” no este vacío.	
Post-condiciones	Se modifica el tipo de documento.	
Prototipo	No procede.	

Tabla 34: RF6.3 Eliminar tipo de documento.

Propósito	Eliminar un tipo de documento	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Se elimina el tipo de documento.*	
Validaciones		
Post-condiciones	Se elimina el tipo de documento.	
Prototipo	No procede.	

RF7 Gestionar tipo de dato.

Tabla 35: RF7.1 Adicionar tipo de dato.

Propósito	Adicionar un tipo de dato.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato no debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	1. *Se adiciona el tipo de dato.*	
Validaciones	1. Validar que el campo “Nombre” no este vacío.	
Post-condiciones	Se adiciona el tipo de dato.	
Prototipo	No procede.	

Tabla 36: RF7.2 Modificar tipo de dato.

Propósito	Modificar el tipo de dato de uno o varios atributos de un documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	1. *Se modifica el tipo de dato.*	

Validaciones	1. Validar que el campo “Nombre” no este vacío.
Post-condiciones	Se modifica el tipo de dato.
Prototipo	No procede.

Tabla 37: RF7.3 Eliminar tipo de dato.

Propósito	Eliminar un tipo de dato para uno o varios atributos de un documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	1. *Se elimina el tipo de dato.*	
Validaciones		
Post-condiciones	Se elimina el tipo de dato.	
Prototipo	No procede.	

RF8 Gestionar atributos de documentos.

Tabla 38: RF8.1 Adicionar atributos de documentos.

Propósito	Adicionar atributos de documentos.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El atributo no debe existir.	
Entidades tratadas	Entidad	Atributos

	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Se adiciona el atributo.*	
Validaciones	1. Validar que el campo “Nombre” no este vacío.	
Post-condiciones	Se adicionan los atributos de documentos.	
Prototipo		

Tabla 39: RF8.2 Modificar atributos de documentos.

Propósito	Modificar atributos de documentos atendiendo a uno o varios parámetros.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El atributo debe existir.	
Entidades tratadas	Entidad	Atributos
	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Se modifica el atributo.*	
Validaciones	1. Validar que el campo “Nombre” no este vacío.	
Post-condiciones	Se modifican los atributos de documentos.	
Prototipo		

Tabla 40: RF8.3 Eliminar atributos de documentos.

Propósito	Eliminar atributos de documentos.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El atributo debe existir.	
Entidades tratadas	Entidad	Atributos
	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Se elimina el atributo.*	
Validaciones		
Post-condiciones	Se eliminan los atributos del documento seleccionado.	
Prototipo		

RF9 - Buscar.

Tabla 41: RF9.1 - Buscar tipo de documento.

Propósito	Buscar un tipo de documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de documento debe existir.	
Entidades tratadas	Entidad	Atributos
	NTIPODOCUMENTO	IDTIPODOCUMENTO DESCRIPCION FECHAREGISTRO
Descripción	1. *Buscar tipo de documento.* 2. *Realizar la búsqueda.* 3. *Mostrar el tipo de documento.*	

Validaciones	1. Validar que el campo “Filtrar” no se encuentre vacío.
Post-condiciones	Se encuentra el tipo de documento.
Prototipo	No procede

Tabla 42: RF9.2 - Buscar atributos de documentos.

Propósito	Buscar los atributos correspondientes a un tipo de documento.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. Los atributos deben existir.	
Entidades tratadas	Entidad	Atributos
	NATRIBUTODOCUMENTO	IDATRIBUTODOCUMENTO DESCRIPCION FECHAREGISTRO NOMBRE IDTIPODATO
Descripción	1. *Buscar los atributos del documento.* 2. *Realizar la búsqueda.* 3. *Mostrar los atributos del documento.*	
Validaciones	1. Validar que el campo “Filtrar” no se encuentre vacío.	
Post-condiciones	Se muestran los atributos.	
Prototipo		

Tabla 43: RF9.3 - Buscar tipo de dato.

Propósito	Buscar un tipo de dato.	
Roles	Usuario	
Pre-condiciones	El usuario debe estar autenticado. El tipo de dato debe existir.	
Entidades tratadas	Entidad	Atributos

	NTIPODATO	IDTIPODATO DESCRIPCION EXPRESIONREGULAR FECHAREGISTRO
Descripción	1. *Buscar el tipo de dato.* 2. *Realizar la búsqueda.* 3. *Mostrar el tipo de dato.*	
Validaciones	1. Validar que el campo “Filtrar” no se encuentre vacío.	
Post-condiciones	Se muestra el tipo de dato.	
Prototipo	No procede.	

Anexo 3. Descripción de Entidades del Editor de Plantillas para el SPDI v2.0.

Tabla 44: Descripción de la Entidad DTIPODOCATRIBUTO.

Nombre de la Entidad: DTIPODOCATRIBUTO.			
Descripción de la Entidad: Nomenclador de atributos de tipo de documentos.			
Servicio: *SPDIService*			
Atributo	**Tipo de Dato**	**Nulo**	**Descripción**
IDTIPODOCUMENTO	*Number(4,0)*	No	Identificador del tipo de documento.
IDATRIBUTODOCUMENTO	*Number(4,0)*	No	Identificador de atributos de documentos.
NULO	*Varchar(5000)*	No	Muestra si el atributo es nulo para ese tipo de documento.

SEMODIFICA	*Varchar(500)*	No	Muestra si el atributo se modifica para ese tipo de documento.

Tabla 45 : Descripción de la Entidad NTIPODATO.

Nombre de la Entidad: NTIPODATO.			
Descripción de la Entidad: Nomenclador de tipos de datos de los atributos de un documento.			
Servicio: *SPDIService*			
Atributo	**Tipo de Dato**	**Nulo**	**Descripción**
IDTIPODATO	*Number(4,0)*	No	Identificador del tipo de dato.
DESCRIPCION	*Varchar(5000)*	No	Descripción del nomenclador.
EXPRESIONREGULAR	*Varchar(5000)*	No	Formato que debe tener un determinado tipo de dato.
FECHAREGISTRO	*Date*	No	Fecha de registro del tipo de dato.

Anexo 4: Diseño de Casos de Pruebas.

Tabla 46. Diseño de Caso de prueba de la funcionalidad *GetAllLotsByDocType.*

Escenario	**Descripción**	**Parámetros**			**Respuesta del Sistema**	**Flujo Prueba**
GetAllLotsByDocType	Devuelve todos los lotes por tipo de documento.	*Tipo de nomenclador*	*Inicio*	*Tamaño*	Lista de los primeros 2 nomenclad ores para lote de tipo de documento	Crear el objeto de entrada y compararl o con la constante del
		docTypeId	0	2		

						identificador del tipo de documento.

Tabla 47: Diseño de Caso de prueba de la funcionalidad *GetSAttributeByDocTypeId*.

Escenario	Descripción	Parámetros			Respuesta del Sistema	Flujo Prueba
GetSAttributeByDocTypeId	Devuelve los atributos por el identificador de un tipo de documento	*Tipo de Nomenclador*	*Inicio*	*Tamaño*	Lista de los 2 primeros atributos de un tipo de documento.	Crear el objeto de entrada para compararlo con el identificador del tipo de documento.
		docTypeId	1	2		

Anexo 5: Prueba Unitaria y resultado.

```
[TestMethod]
public void GetAllLotsByDocTypeTest()
{
    var target = new SPDIServiceWcf();
    const int docTypeId = 0;
    var actual = target.GetAllLotsByDocType(docTypeId);
    Assert.AreNotEqual(null, actual);
}
```

Figura 15: Prueba Unitaria a GetAllLostByDocType.

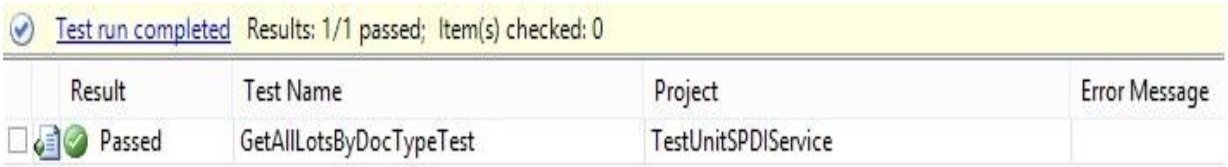


Figura 16: Resultado de la prueba unitaria a GetAllLostByDocType.

```
[TestMethod]
public void GetAttributeByDocTypeIdTest()
{
    var target = new SPDIServiceWcf();
    const int docTypeId = 1;
    var actual = true;
    try
    {
        target.GetAttributeByDocTypeId(docTypeId);
    }
    catch
    {
        actual = false;
    }
    Assert.AreEqual(true, actual);
}
```

Figura 17: Prueba Unitaria a GetAttributeByDocTypeId.

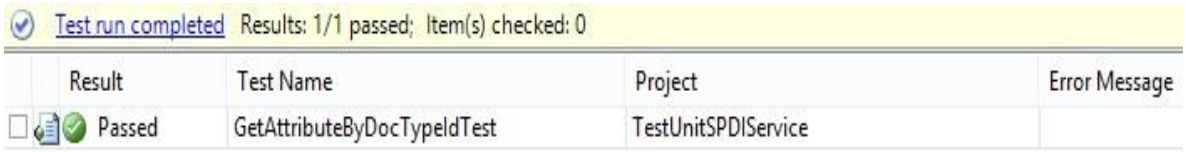


Figura 18: Resultado de la prueba unitaria a GetAttributeByDocTypeId.

Tabla 48: Caso de prueba: RF 1.1 - Adicionar tipo de documento.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Adicionar tipo de documento.	E.C 1.1 Listar tipos de documento.	Lista los documentos existentes en la base de datos.	1. Seleccionar la opción “Gestionar documento”.
	E.C 1.2 Adicionar tipo de documento.	Adiciona un documento a la base de datos.	1. Pulsar el botón “Adicionar”. 2. Introducir el nombre del tipo de documento. 3. Seleccionar el conector. 4. Seleccionar el botón “Adicionar”

Tabla 49: Caso de prueba: RF 1.2 - Modificar tipo de documento.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Modificar tipo de documento.	E.C 1.1 Listar tipos de documento.	Lista los documentos existentes en la base de datos.	1. Seleccionar la opción “Gestionar documento”.

	E.C 1.2 Modificar tipo de documento.	Modifica los datos del documento seleccionado.	1. Selecciona el documento que desee modificar. 2. Introducir el nuevo nombre. 3. Seleccionar el conector. 4. Seleccionar la opción “Guardar”.

Tabla 50: Caso de prueba: RF 1.3 - Eliminar tipo de documento.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Eliminar tipo de documento.	E.C 1.1 Eliminar tipo de documento.	Lista los documentos existentes en la base de datos.	1. Seleccionar la opción “Gestionar documento”. 2. Seleccionar el documento que se quiere eliminar. 3. Pulsar el botón “Eliminar”.

Tabla 51: Caso de prueba: RF 2.1 - Crear plantilla.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Crear Plantilla	E.C 1.1 Crear plantilla.	Crea una plantilla en blanco	1. Ir al menú principal. 2. Seleccionar la opción “Nueva Plantilla”.

Tabla 52: Caso de prueba: RF 2.1.1 - Insertar elementos.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Insertar Elementos.	E.C 1.1 Insertar figuras geométricas.	Debe mostrarse en el área de diseño dependiendo de la opción seleccionada un rectángulo, una elipse o una línea.	1. Seleccionar la opción de “Rectángulo”, “Elipse” o “Línea” en la barra de herramientas.

			2. Seleccionar el lugar donde se quiere insertar en el área de diseño.
	E.C 1.2 Insertar imágenes.	Debe mostrarse en el área de diseño una imagen importada desde el ordenador.	1. Ir al menú principal. 2. Seleccionar la opción "Importar Imagen".
	E.C 1.3 Insertar texto.	Debe mostrarse en el área de diseño un texto.	1. Seleccionar la opción de "Texto" en la barra de herramientas. 2. Seleccionar el lugar donde se quiere insertar en el área de diseño.

Tabla 53: Caso de prueba: RF 2.1.2 - Modificar propiedades de los elementos insertados.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Modificar propiedades de los elementos insertados.	E.C 1.1 Modificar propiedades de figuras geométricas.	Debe mostrarse en la barra de propiedades: 1. Ancho. 2. Largo. 3. Posición. 4. Ancho de línea. 5. Tipo de línea. 6. Color de relleno. 7. Color de línea. 8. Rotación.	1. Seleccionar el elemento que se le quieren modificar las propiedades. 2. Introducir los valores que el usuario desee.
	E.C 1.2 Modificar propiedades de imágenes.	Debe mostrarse en la barra de propiedades: Ancho. Largo. Posición. Rotación.	1. Seleccionar el elemento que se le quieren modificar las propiedades. 2. Introducir los valores que el usuario desee.

	E.C 1.3 Modificar propiedades de texto.	Debe mostrarse en la barra de propiedades: 1. Posición. 2. Rotación. 3. Tipo de texto. 4. Tamaño de fuente.	1. Seleccionar el elemento que se le quieren modificar las propiedades. 2. Introducir los valores que el usuario desee.

Tabla 54: Caso de prueba: RF 2.1.3 - Insertar atributos.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Insertar atributos.	E.C 1.1 Insertar atributos.	Debe mostrarse los atributos asociados al tipo de documento al cual se le está diseñando la plantilla.	1. Seleccionar la opción “Mostrar atributos”. 2. Seleccionar el atributo que se quiere insertar.

Tabla 55: Caso de prueba: RF 2.2.1 - Insertar elementos.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Insertar Elementos.	E.C 1.1 Insertar figuras geométricas.	Debe mostrarse en el área de diseño dependiendo de la opción seleccionada un rectángulo, una elipse o una línea.	1. Ir al menú principal. 2. Seleccionar la opción “Abrir archivo”. 3. Seleccionar la plantilla que se quiere modificar. 4. Seleccionar la opción de “Rectángulo”, “Elipse” o “Línea” en la barra de herramientas. 5. Seleccionar el lugar donde se quiere insertar en el área de diseño.
	E.C 1.2 Insertar imágenes.	Debe mostrarse en el área de diseño una imagen importada desde el ordenador.	1. Ir al menú principal. 2. Seleccionar la opción “Abrir archivo”.

			3. Seleccionar la plantilla que se quiere modificar. 4. Ir al menú principal. 5. Seleccionar la opción "Importar imagen".
	E.C 1.3 Insertar texto.	Debe mostrarse en el área de diseño un texto.	1. Ir al menú principal. 2. Seleccionar la opción "Abrir archivo". 3. Seleccionar la plantilla que se quiere modificar. 4. Seleccionar la opción de "Texto" en la barra de herramientas. 5. Seleccionar el lugar donde se quiere insertar en el área de diseño.

Tabla 56: Caso de prueba: RF 2.2.2 - Modificar propiedades de los elementos insertados.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Modificar propiedades de los elementos insertados	E.C 1.1 Modificar propiedades de figuras geométricas.	Debe mostrarse en la barra de propiedades: 1. Ancho. 2. Largo. 3. Posición. 4. Ancho de línea. 5. Tipo de línea. 6. Color de relleno. 7. Color de línea. 8. Rotación.	1. Ir al menú principal. 2. Seleccionar la opción "Abrir archivo". 3. Seleccionar la plantilla que se quiere modificar. 4. Seleccionar el elemento que se le quieren modificar las propiedades. 5. Introducir los valores que el usuario desee.
	E.C 1.2 Modificar propiedades de imágenes.	Debe mostrarse en la barra de propiedades: 1. Ancho.	1. Ir al menú principal.

		2. Largo. 3. Posición. 4. Rotación.	2. Seleccionar la opción “Abrir archivo”. 3. Seleccionar la plantilla que se quiere modificar. 4. Seleccionar el elemento que se le quieren modificar las propiedades. 5. Introducir los valores que el usuario desee.
	E.C 1.3 Modificar propiedades de texto.	Debe mostrarse en la barra de propiedades: 1. Posición. 2. Rotación. 3. Tipo de texto. 4. Tamaño de fuente.	1. Ir al menú principal. 2. Seleccionar la opción “Abrir archivo”. 3. Seleccionar la plantilla que se quiere modificar. 4. Seleccionar el elemento que se le quieren modificar las propiedades. 5. Introducir los valores que el usuario desee.

Tabla 57: Caso de prueba: RF 2.2.3 - Insertar atributos.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Insertar atributos.	E.C 1.1 Insertar atributos.	Debe mostrarse los atributos asociados al tipo de documento al cual se le está diseñando la plantilla.	1. Ir al menú principal. 2. Seleccionar la opción “Abrir archivo”. 3. Seleccionar la plantilla que se quiere modificar. 4. Seleccionar la opción “Mostrar atributo”. 5. Seleccionar el atributo que se quiere insertar.

Tabla 58: Caso de prueba: RF 2.1.4/ RF 2.2.4 - Guardar plantilla.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central

Guardar plantilla.	E.C 1.1 Guardar plantilla.	Guarda la plantilla en el ordenador	1. Ir al menú principal. 2. Seleccionar la opción “Guardar archivo”.
	E.C 1.2 Guardar plantilla modificada.	Guarda la plantilla en el ordenador	1. Ir al menú principal. 2. Seleccionar la opción “Abrir archivo”. 3. Hacer alguna modificación en la plantilla. 4. Ir al menú principal. 5. Seleccionar la opción “Guardar archivo.”

Tabla 59: Caso de prueba: RF 3.1 - Adicionar atributo.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Adicionar atributo.	E.C 1.1 Listar atributos.	Lista los atributos existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”.
	E.C 1.2 Adicionar el atributo.	Inserta el atributo en la base de datos	1. Seleccionar la opción “Adicionar”. 2. Introducir el nombre del atributo. 3. Seleccionar el tipo de dato. 4. Seleccionar la opción “Aceptar”.

Tabla 60: Caso de prueba: RF 3.2 - Modificar atributo.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Modificar atributo.	E.C 1.1 Listar atributos.	Lista los atributos existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”.
	E.C 1.2 Modificar atributo.	Modifica el atributo.	1. Seleccionar el atributo que se quiere modificar. 2. Introducir el nombre. 3. Seleccionar el tipo de dato.

			4. Seleccionar la opción “Aceptar”.

Tabla 61: Caso de prueba: RF 3.3 - Eliminar atributo.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Eliminar atributo	E.C 1.1 Eliminar atributo.	Lista los atributos existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”. 2. Seleccionar el atributo que se quiere eliminar. 3. Seleccionar la opción “Eliminar”.

Tabla 62: Caso de prueba: RF 4.1 - Adicionar tipo de dato.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Adicionar tipo de dato.	E.C 1.1 Listar tipos de dato.	Lista los tipos de dato existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”. 2. Seleccionar la pestaña “Tipos de dato”.
	E.C 1.2 Adicionar tipos de dato.	Inserta el tipo de dato en la base de datos	3. Seleccionar la opción “Adicionar”. 4. Introducir el nombre del tipo de dato. 5. Introducir sus restricciones. 6. Seleccionar la opción “Aceptar”.

Tabla 63: Caso de prueba: RF 4.2 - Modificar tipo de dato.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Modificar tipo de dato.	E.C 1.1 Listar tipo de datos.	Lista los tipos de dato existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”. 2. Seleccionar la pestaña “Tipos de dato”.

	E.C 1.2 Modificar tipos de dato.	Modifica el tipo de dato.	1. Seleccionar el tipo de dato que se quiere modificar. 2. Seleccionar la opción “Editar”. 3. Introducir el nombre del tipo de dato. 4. Introducir sus restricciones. 5. Seleccionar la opción “Aceptar”.

Tabla 64: Caso de prueba: RF 4.3 - Eliminar tipo de dato.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Eliminar tipo de dato	E.C 1.1 Eliminar tipo de dato.	Lista los tipos de dato existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”. 2. Seleccionar la pestaña “Tipos de dato”. 3. Seleccionar el tipo de dato que se quiere eliminar. 4. Seleccionar la opción “Eliminar”.

Tabla 65: Caso de prueba: RF 5 - Buscar elemento.

Nombre del flujo	Escenarios del Flujo	Descripción de la funcionalidad	Flujo Central
Buscar elemento.	E.C 1.1 Buscar documento.	Lista los documentos existentes en la base de datos.	1. Seleccionar la opción “Gestionar documentos”. 2. Escribir el nombre en el recuadro filtrar.
	E.C 1.2 Buscar atributo.	Lista los atributos existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”. 2. Escribir el nombre en el recuadro filtrar.

	E.C 1.3 Buscar tipo de dato.	Lista los tipos de dato existentes en la base de datos.	1. Seleccionar la opción “Gestionar nomencladores”. 2. Escribir el nombre en el recuadro filtrar.

Anexo 5: Iteraciones de prueba.

Tabla 66: Iteración 1 de pruebas

Iteración 1					
Elemento	**No**	**No Conformidad**	**Aspecto Correspondiente**	**Etapa de detección del error**	**Importancia**
RF1.1	1	No inserta el documento en la base de datos.	Insertar tipo de documento.	Al seleccionar la opción “Adicionar”.	Significativa.
RF1.3	2	Muestra el mensaje, pero no elimina de la base de datos.	Eliminar tipo de documento.	Al seleccionar la opción “Eliminar”.	Significativa.
RF2.1	3	Aparece “Plantillo” en vez de “Plantilla”.	Crear plantilla.	Al desplegar el menú de la aplicación.	No significativa.
RF2.1.1	4	No inserta el rectángulo.	Insertar Elemento.	Al seleccionar la herramienta rectángulo.	Significativa.
RF2.1.1	5	No inserta el texto.	Insertar Elemento.	Al seleccionar la herramienta texto.	Significativa.
RF2.1.1	6	No inserta a línea.	Insertar Elemento.	Al seleccionar la herramienta línea.	Significativa.
RF2.1.1	7	No permite mover la elipse.	Insertar Elemento.	Al mover el elemento por el área de diseño.	Significativa.
RF2.1.3	8	No inserta los atributos.	Insertar atributo.	Al insertar un atributo.	Significativa.

RF2.2.1	9	No modifica el ancho de los elementos.	Modificar propiedades.	Al cambiar el ancho de un elemento.	Significativa.
RF3.2	10	No modifica el nombre de los atributos.	Modificar atributos.	Al actualizar el nombre de un atributo.	Significativa.
RF3.3	11	Cambiar “Elminar” por “Eliminar”.	Eliminar atributo.	Al listar los atributos.	No Significativa.
RF5.1	12	No busca en la base de datos.	Buscar tipo de documento.	Al filtrar los tipos de documento.	Significativa.
RF2.1.4	13	Cambiar “Guardr” por “Guardar”.	Guardar plantilla.	Al guardar una plantilla.	Significativa.
RF2.1.4	14	Hay que poner manualmente la extensión del archivo.	Guardar plantilla.	Al guardar una plantilla.	No Significativa.
RF5.3	15	Cambiar “filtar” por “filtrar”.	Buscar tipo de dato.	Al filtrar los tipos de dato.	No significativa.

Tabla 67: Iteración 2 de pruebas

Iteración 2					
Elemento	**No**	**No Conformidad**	**Aspecto Correspondiente**	**Etapa de detección del error**	**Importancia**
RF2.1.3	1	No muestra los atributos.	Insertar atributos.	Al seleccionar la opción “Mostrar atributos”.	Significativa.
RF2.2.1	2	No modifica el ancho de línea.	Modificar atributos.	Al cambiar el ancho de la línea de un elemento.	Significativa.
RF2.2.1	3	No modifica el color de relleno de un elemento.	Modificar atributos.	Al cambiar el color de relleno de un elemento.	Significativa.

RF2.1.3	4	Permite insertar más de un atributo del mismo tipo.	Insertar atributo.	Al insertar el atributo “Nombre”.	Significativa.
RF2.1.1	5	No muestra en el área de diseño la cuadricula.	Insertar elemento.	Al seleccionar la opción de “Mostrar Cuadricula”.	No Significativa.
RF2.1	6	Cambiar “Menu” por “Menú”.	Crear plantilla.	Al crear una plantilla.	No significativa.
RF3.1	7	Cambiar “Adiconar” por “Adicionar”.	Adicionar atributos.	Al adicionar un atributo.	No Significativa.
RF4.3	8	No elimina el tipo de dato de la base de datos.	Eliminar tipo de dato.	Al eliminar un tipo de dato.	Significativa.

Anexo 6: Pruebas de integración.

Tabla 68: CPI-SPDIEditor y SPDIServicio.

Caso de Prueba: INT-ES
Módulo al que se integra: SPDIEditor y SPDIServicio.
Condiciones de ejecución: El módulo SPDIEditor haya solicitado los datos de los documentos a la base de datos mediante el módulo SPDIServicio y exista conexión con la misma.
Descripción de la prueba: Comprobar que el módulo SPDIServicio es capaz de brindar los datos de los tipos de documentos al módulo SPDIEditor.
Entradas/Pasos de ejecución: El módulo SPDIServicio obtiene los datos de los documentos en la base de datos y los proporciona al módulo SPDIEditor.
Resultado esperado: Se obtienen los datos.
Evaluación: Prueba satisfactoria.

Tabla 69: CPI-SPDIEditor y SPDIEntidades.

Caso de Prueba: INT-EEN
Módulo al que se integra: SPDIEditor y SPDIEntidades.

Condiciones de ejecución: El módulo SPDIEditor haya utilizado las entidades existentes en SPDIEntidades.
Descripción de la prueba: Comprobar que el módulo SPDIEditor es capaz de utilizar las entidades contenidas en el módulo SPDIEntidades.
Entradas/Pasos de ejecución: El módulo SPDIEntidades proporciona las entidades existentes para ser usadas por el módulo SPDIEditor.
Resultado esperado: Las entidades son utilizadas.
Evaluación: Prueba satisfactoria.

Anexo 7: Entrevista.

Tabla 70: Entrevista con el cliente.

Entrevistador			Yunior Martínez Suarez		
Entrevistado			Ing. Yanet Silva Fernández.(Cliente)		
Número de Pregunta	**Fecha**	**Hora**	**Lugar**	**Pregunta**	**Respuesta**
1	**13/11/2013**	**09:45 am**	**CISED**	¿Qué es el Sistema de Personalización de Documentos de Identificación?	El Sistema de Personalización de Documentos (SPDI) para la República de Cuba es una solución informática que permite la personalización de documentos relativos a la identificación de los ciudadanos cubanos.
2	**13/11/2013**	**10:15 am**	**CISED**	¿Qué limitaciones tiene el módulo de administración de plantillas de la versión 1.0 del SPDI?	El Editor de plantillas del SPDI en su versión 1.0, brinda la posibilidad de crear pasaportes, y en una actualización reciente, los documentos de acceso a la Universidad de Ciencias Informáticas (UCI), lo que resultó de gran importancia debido a sus funcionalidades, pero no se puede incorporar a la segunda versión puesto que no se ajusta a la arquitectura y tendría que ser modificado el código fuente del mismo. Además solo tiene en cuenta la definición de una plantilla para un tipo de documento en específico, en caso de querer definir una plantilla nueva se debe modificar el código del SPDI; no cuenta con elementos de diseño como figuras geométricas básicas, separadores e imágenes predefinidas

3	**13/11/2013**	**11:00 am**	**CISED**	¿Cuáles son las principales ventajas que brindará el desarrollo del editor de plantillas para el Sistema de Personalización de Documentos de Identificación en su versión 2.0?	Una de las principales ventajas que proporcionará el desarrollo del editor de plantillas para el SPDI v2.0 es que permitirá personalizar cualquier tipo de documento de identificación sin necesidad de transformar el código fuente del sistema cada vez que se desee añadir un nuevo tipo de documento. Permitirá que las plantillas editadas tengan calidad y puedan ser reutilizadas en otras herramientas de diseño, ya que serán exportadas en un formato estándar.
4	**25/11/2013**	**10:45 am**	**CITI**	¿Por qué la necesidad de que las plantillas que se editen tengan un formato que permitan su reutilización y calidad?	Es necesario que las plantillas puedan ser cargadas y editadas en otros editores, lo que garantiza su reutilización, que no solo se limite a que necesariamente tengan que ser editadas en el editor de plantillas del SPDI v2.0. Las plantillas se exportarán en formato SVG, que además de garantizar que sean reutilizables, garantizará la calidad de las mismas, ya que serán construidas de forma vectorial, lo que permitirá que no sufran de distorsión ni pérdida de resolución durante su manipulación.
5	**25/11/2013**	**11:30 am**	**CITI**	¿Se deben construir las plantillas de forma que permitan ubicar los datos del usuario final durante la personalización ?	Sí, es necesario que las plantillas permitan ubicar los datos del usuario final durante la personalización, para que una plantilla pueda ser utilizada para personalizar determinadas cantidades de documentos de identificación. El editor debe contar con una funcionalidad que permita obtener según el tipo de documento, los atributos del mismo en base de datos, y colocarlos en la plantilla.
6	**30/11/2013**	**13:05 pm**	**CITI**	¿Podría citar algunas facilidades	Sí, como parte de añadir algunas facilidades funcionales carentes en el módulo de la versión 1.0, debe contar con

				funcionales que le harían el trabajo de edición más fácil y cómoda?	funcionalidades que permitan tener elementos de diseño como: cuadrado, círculo, separados de bordes, línea, etc. Debe contar con imágenes predefinidas, así como permitir que el usuario coloque las dimensiones que desea para el nuevo tipo de documento a editar, por citar algunas de las funcionalidades que debe tener el editor.
7	**18/01/2014**	**10:15 am**	**CITI**	¿Es posible utilizar el servicio web con que cuenta el sistema en otras aplicaciones que manejen datos?	Sí. El servicio web con que cuenta el sistema puede ser utilizado en cualquier otra herramienta que necesite el acceso a datos.
8	**18/01/2014**	**11:25 am**	**CITI**	¿Se cuenta con dos servicios en el Sistema?	Sí. El editor cuenta con un servicio llamado Servicio.ASMX que es el encargado de llamar al servicio SPDIServicio cuando se soliciten datos de los tipos de documentos.
9	**13/03/2014**	**14:00 pm**	**CITI**	¿Qué diferencias existen entre los paquetes Negocio y Conector contenidos en el paquete SPDIServicio y los paquetes Negocio y Conector contenidos en el paquete SPDIDlls?	En el paquete SPDINegocio se encuentra el paquete Negocio, que es donde se validan las funcionalidades para el acceso a datos de los tipos de documentos, y el paquete Conector es el encargado de realizar estas funcionalidades en la Base de Datos. En el caso del paquete SPDIDlls, el paquete Negocio especifica si existe alguna ruta para guardar las plantillas editadas, esto a modo de organización, y el paquete Conector es el encargado del manejo de estas plantillas según lo definido por el paquete Negocio.
10	**13/03/2014**	**14:30 pm**	**CITI**	¿Por qué se define una	Se define una arquitectura por capas para tener un bajo acoplamiento en el sistema, es decir si se realiza un cambio en alguna

				arquitectura por capas?	de las capas, no se vean afectadas las restantes.