

Universidad de las Ciencias Informáticas

Facultad 1



Título: Componentes y utilitarios para la impresión de
documentos de identificación en software libre
“IDSVG”.

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

Autores: Abdel Sadín Odio

Yainier Labrada Nueva

Tutor: Ing. Yurdik Cervantes Mendoza

Ciudad de La Habana, Cuba Julio 2008

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizamos a la Facultad 1 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Abdel Sadín Odio

Yainier Labrada Nueva

Ing. Yurdik Cervantes Mendoza

Datos de Contacto

Agradecimientos

Yainier Labrada Nueva

A los profesores que durante estos cinco años nos formaron, al Tutor que siempre estuvo a nuestro lado, a mi compañero de tesis y de largas y tensas horas de trabajo y con quien compartimos la amistad y la convivencia en la UCI, a nuestros padres que durante estos duros y largos meses también presentaron y defendieron una nueva tesis de paciencia, tesón y aliento. Nuestro agradecimiento también para todos aquellos que nos apoyaron: a Yurdik, Jeandy, Adonis, Irving, Yailin, a ambos Dayron, Ismar, Luis, Michel, Joel Saez, a Landrián por sus peleaderas que nos sirvieron de mucho en fin, al Equipo de Desarrollo de Identidad que en todo momento nos brindó su sostén. Al Decano y vicedecanos de la facultad Nro. 1.

Abdel Sadín Odio

Además de los mencionados más arriba agradecer a Damian por su paciencia, a Reynier, en especial a Yisel y a Yeisa por todo su apoyo en los no pocos momentos difíciles que tuve que enfrentar a lo largo de esta investigación. Gracias por estar a mi lado cuando lo necesité mis queridas y entrañables amigas. A Adriana, a Diana y a Alina quienes desde la distancia han sabido ayudarme, valorarme y tener esa indestructible confianza en mí.

En fin, a todos los que de alguna manera me ayudaron o tuvieron alguna vez la intención de hacerlo.

Dedicatoria

Abdel Sadín Odio

A nuestro querido Comandante en Jefe Fidel Castro, artífice de nuestra gloriosa Revolución Cubana y de la brillante idea de erigir esta fabulosa Universidad de las Ciencias Informáticas, dedico este trabajo de Tesis de Grado, fruto de largas horas de trabajo y de insomnio.

A mis padres y a los de Labrada, que nos alentaron siempre y que con el ejemplo nos educaron y nos guiaron y, no por último la menos importante, dedico todo este esfuerzo a una persona muy especial para mí, alguien a quien desgraciadamente la vida no le dio la oportunidad de poder disfrutar momentos tan irrepetibles como lo fueron mi entrada a esta singular universidad y ahora mi graduación en la misma, a mi abuela querida Eva.

Yainier Labrada Nueva

A mis padres por ser la alegría del presente y del futuro, a los padres de Abdel por estar siempre junto a nosotros ayudándonos en todo, a Fidel y a toda mi familia; abuelos, tíos, primos y, especialmente, a mi hermanita. No por ser la última la menos importante, dedico este trabajo de Tesis de Grado a la indestructible obra de la Revolución Cubana.

RESUMEN

Un reto en la informatización de la sociedad, es el establecimiento de credenciales o documentos que identifiquen a los usuarios ante los distintos procesos de las organizaciones sociales a las que estos pertenecen. Tales documentos, generalmente llevan impresos los principales datos representativos de la persona con el mayor nivel de actualización posible respecto a una base de datos y presentan un nivel determinado de seguridad ante falsificaciones. La tendencia en este tema, es la de utilizar grandes centros de personalización con tecnologías y materias primas costosas, por lo que es necesaria, una alternativa rápida, de bajo costo y libre para estar a tono con la estrategia productiva de la Universidad de las Ciencias Informáticas y para brindar esta funcionalidad a aplicaciones futuras que se desarrollen bajo este modelo.

En la presente investigación, se proponen entonces un conjunto de componentes que permitan la impresión de tarjetas de identificación, cédulas, pasaportes y otros. La solución se basa en un diseño flexible que facilite su integración a cualquier sistema de identificación, además se especifica la presentación (contenido a dibujar y forma de hacerlo) y el enlace de datos (cómo obtener los datos desde los objetos del sistema) mediante plantillas XML para hacer independiente la aplicación de nuevos cambios de diseño o dispositivos de impresión, igualmente se incluyen herramientas de validación y diseño visual para la especificación de estos elementos.

PALABRAS CLAVES

Componentes para impresión, documentos de Identificación, software libre.

ÍNDICE

Introducción	1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA	9
1.1 Introducción	9
1.2 Conceptos asociados al dominio del problema.....	9
1.2.1 ¿Qué es un componente?	9
1.2.2 ¿Qué se entiende por impresión?.....	9
1.2.3 ¿Qué son los documentos de identificación?.....	9
1.3 Tendencias y tecnologías actuales.....	10
1.4 Conclusiones	39
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA	40
2.1 Introducción	40
2.3 Descripción del Sistema Propuesto.....	43
2.4 Modelo de Casos de Uso del Sistema.....	47
2.5 Conclusiones	59
CAPÍTULO 3: CONSTRUCCIÓN Y LEABORACIÓN DEL SISTEMA	60
3.1 Introducción	60
3.2 Modelo de Análisis	60
3.3 Patrones de Diseño.....	62
3.4 Modelo de Diseño	62
3.5 Diagrama de Clases de Diseño	63
3.6 Arquitectura.....	98
3.8 Diagrama de despliegue	99
3.9 Diagrama de componentes	99
3.10 Conclusiones.....	101
CONCLUSIONES	102
RECOMENDACIONES	103
BIBLIOGRAFÍA	104
ANEXOS.....	106
GLOSARIO DE TÉRMINOS.....	109

ÍNDICE DE TABLAS

Tabla 1: Operacionalización de las Variables.....	4
Tabla 2: Recursos humanos	5
Tabla 3: Recursos Materiales.....	5
Tabla 4: Cronograma de las Actividades.....	6
Tabla 2.1 Actores del sistema	48
Tabla 2.2 Resumen del Caso de Uso 1	48
Tabla 2.3 Resumen del Caso de Uso 2.....	48
Tabla 2.4 Resumen del Caso de Uso 3.....	48
Tabla 2.5 Resumen del Caso de Uso 4.....	48
Tabla 2.6 Resumen del Caso de Uso 5.....	49
Tabla 2.7 Resumen del Caso de Uso 6.....	49
Tabla 2.8 Resumen del Caso de Uso 7	49
Tabla 2.9 Resumen del Caso de Uso 8.....	49
Tabla 2.10 Descripción del caso de uso Crear Modelo de Datos	50
Tabla 2.11 Caso de uso: Crear enlace de datos	50
Tabla 2.12 Caso de uso: Cargar diseño SVG	51
Tabla 2.13 Caso de uso: Cargar modelo de datos	52
Tabla 2.14 Caso de uso: Cargar enlace de datos	52
Tabla 2.15 Caso de uso: Crear plantilla de documentos de identificación.....	53
Tabla 2.16 Caso de uso: Cargar plantilla de documentos de identificación.....	53
Tabla 2.17 Caso de uso: Crear plantilla compilada	54
Tabla 2.18 Resumen Caso de Uso 1	54
Tabla 2.19 Resumen del Caso de Uso 2.....	55
Tabla 2.20 Resumen del Caso de Uso 3.....	55
Tabla 2.21 Resumen del Caso de Uso 4.....	55
Tabla 2.22 Caso de uso: Cargar datos personalización	56
Tabla 3.1 Descripción de la clase Enlace.....	67
Tabla 3.2 Descripción de la clase Campo	67
Tabla 3.3 Descripción de la clase Diseño.....	68
Tabla 3.4 Descripción de la clase EnlaceDatos	68
Tabla 3.5. Descripción de la clase ModeloDatos.....	68
Tabla 3.6 Descripción de la clase PlantillaCompilada	69

Tabla 3.7 Descripción de la clase CargadorPlantillaCompilada.....	70
Tabla 3.8 Descripción de la clase GeneradorPlantillaXML	71
Tabla 3.9 Descripción de la clase DatosPersonalizacion	72
Tabla 3.10 Descripción de la clase Personalizacion.....	72
Tabla 3.11 Descripción de la clase Compilador	72
Tabla 3.12 Descripción de la clase ValidadorXMLContraXSD.....	73
Tabla 3.13 Descripción de la clase PrintinManager.....	73
Tabla 3.14 Descripción de la clase DocumentBuilder	74
Tabla 3.15. Descripción de la clase Documento.....	74
Tabla 3.16 Descripción de la clase PageConfiguration	74
Tabla 3.17 Descripción de la clase XMLPageConfiguration	75
Tabla 3.18 Descripción de la clase SVGDisposer	75
Tabla 3.19 Descripción de la clase SVGLoader	76
Tabla 3.20 Descripción de la clase SVGPainter	76
Tabla 3.21 Descripción de la clase CargadorDatosPersonalizacion.....	76
Tabla 3.22 Descripción de la clase GeneradorPlantillaCXML	76
Tabla 3.23 Descripción de la clase GeneradorDatosPersonalizacion.....	77
Tabla 3.24 Descripción de la clase TempFileHandler	78
Tabla 3.25 Descripción de la clase frmPrincipalPersonalizacionImpresion.....	78
Tabla 3.26 Descripción de la clase frmPrincipalDiseñador	79

ÍNDICE DE FIGURAS

Figura 1. Ciclo de vida de RUP	19
Figura 2. Ejemplo gráfico modelo del pintor	29
Figura 4 Propuesta de solución.....	41
Figura 5 Modelo de Dominio	43
Figura 6 Diagrama de caso de uso del editor de plantillas.	58
Figura 8 Diagrama de clases del análisis del caso de uso: Crear plantilla.....	60
Figura 9 Diagramas de clases del análisis del caso de uso: Crear Modelo de datos.....	61
Figura 10 Diagrama de clases del análisis del caso de uso: Crear enlace de datos.....	61
Figura 11 Diagrama de clases del análisis del caso de uso: Cargar diseño SVG.....	61
Figura 12 Diagrama de clases del análisis del caso de uso: Cargar modelo de datos.	61
Figura 13 Diagrama de clases del análisis del caso de uso: Cargar enlace de datos.....	61
Figura 14 Diagrama de clases del análisis del caso de uso: Cargar enlace de datos.....	62
Figura 15. Diagrama de contrato de paquetes del caso de uso: Crear plantillas.	83
Figura 16. Diagrama de contrato de paquetes del caso de uso: Personalizar documento SVG.	84
Figura 17. Formulario principal del Diseñador de plantillas.	85
Figura 18. Formulario principal con una plantilla cargada.	86
Figura 19. Formulario principal de la personalización e impresión.	96
Figura 20. Figura que muestra la validación de los datos de la personalización.....	97
Figura 21. Personalización e impresión de los documentos.	98
Figura 22. Diagrama de despliegue	99
Figura 23. Diagrama de componentes	100

INTRODUCCIÓN

Con el desarrollo de las Tecnologías de la Informática y las Comunicaciones se lleva a cabo paralelamente la informatización de la sociedad. En estas condiciones juega un papel muy importante el uso de documentos identificativos lo cual posibilitaría en un futuro que se informaticen los principales procesos de identificación en los registros de dirección de los ciudadanos y en las instituciones a las que estos pertenecen; con un nivel determinado de seguridad ante falsificaciones.

Estos documentos poseen impresos datos significativos de los individuos (nombres, apellidos sexo, foto, dirección, entre otros), por lo que se hace necesario la existencia de componentes que brinden la funcionalidad de personalizar e imprimir de manera eficiente los mismos.

Para el desarrollo de los componentes en una forma viable y más económica, se recomienda realizarlos en software libre para seguir las normas que exige nuestra institución y lograr una futura integración a cualquier sistema de identificación.

De lo anteriormente expuesto se plantea la siguiente **Situación Problemática**: Las actuales aplicaciones de identificación de personas desarrolladas en la UCI carecen de un conjunto de componentes que provean la funcionalidad de personalización de documentos (tarjetas de identificación, cédulas, pasaportes, entre otros).

En la actualidad no se asegura la óptima precisión y rendimiento del proceso de impresión. No se cuenta con un conjunto de utilitarios que permitan la especificación de la presentación de los documentos y el enlace de los datos mediante ficheros XML, no existiendo independencia de la aplicación a nuevos cambios de diseño o de dispositivos de impresión. Tampoco se posee un editor de plantillas que facilite el diseño de los documentos de identificación.

Constituye una necesidad que el desarrollo sea basado totalmente en tecnologías de software libre, dadas las proyecciones y estrategias trazadas por la Oficina de Informatización del Ministerio de la Informática y las Comunicaciones (MIC) conjuntamente con la dirección de informatización de la Universidad de las Ciencias Informáticas (UCI), al encontrarse, la misma, en un proceso de transición a software libre de todos los sistemas elaborados, definiendo como **Problema**: las actuales aplicaciones de identificación de personas de la UCI, carecen de un conjunto unificado de componentes que provean la funcionalidad de impresión y personalización de documentos.

Se define como **Hipótesis** que: al implementarse un conjunto de componentes y utilitarios para la impresión de documentos de identificación de personas en la UCI, se permitirá dar una **Mejor**

Respuesta a la funcionalidad de personalización e impresión de documentos y al incremento de solicitudes en las distintas instituciones donde se realicen estos procesos.

El **Objetivo General** es desarrollar componentes y utilitarios para la impresión de documentos de identificación de personas en software libre, así como un editor de plantillas para el enlace de los datos mediante ficheros XML, de forma tal que esta funcionalidad se pueda proveer a futuras aplicaciones de identificación que se desarrollen bajo este modelo.

Para lograrlo se proponen como **Objetivos Específicos**:

- Diseñar un formato XML estándar para la especificación de la presentación de los documentos de identificación (plantillas) de manera que se hagan independiente los componentes de software a nuevos cambios de diseño o dispositivos de impresión, utilizando como base el formato SVG (Scalable Vector Graphics).
- Crear una herramienta de diseño de documentos de interfaz simple y amena con posibilidades de edición, que permita asociar las fuentes de datos a la presentación de estos en el proceso de impresión. El formato de persistencia a utilizar debe ser el estándar de plantillas XML definido.
- Desarrollar componentes de software para la impresión de documentos de identificación en software libre, que permita realizar los procesos necesarios para la personalización como la recuperación y formato de datos, lectura y validación de las plantillas XML, así como el dibujo en el dispositivo o contexto gráfico especificado.
- Crear un sistema de configuración, excepciones y eventos de impresión, que permita una fácil adaptabilidad a cualquier contexto de desarrollo.

El **Objeto de estudio** consiste en componentes de software para la impresión de documentos.

El **Campo de Acción**: componentes de impresión de documentos en Software Libre.

De esta forma se obtienen las siguientes **variables de la investigación**:

Variable independiente: son los componentes y utilitarios para la impresión de documentos de identificación.

Variable dependiente: mejor respuesta al incremento de solicitudes de documentos de identificación, en las distintas instituciones donde se realicen estos procesos. **Mejor respuesta**, será el tiempo en que se demora la realización de un proceso.

Métodos Investigativos Métodos Empleados: Métodos Teóricos: Hipotético-Deductivo, Análisis Histórico-lógico.

Histórico-Lógico: Este método posibilita el análisis histórico del proceso de gestión de información. Se emplea durante el proceso de diseño y mejoramiento de las funcionalidades que satisfacen o podrían satisfacer las demandas de los componentes de impresión.

Hipotético-Deductivo: Se emplea partiendo de la hipótesis, siguiendo la lógica de deducción obtenida para arribar a nuevos conocimientos y predicciones. Los resultados se someten a verificaciones. Este método se emplea en cada fase en que se obtenga un resultado o se analice sobre la base de algún conocimiento.

Métodos empíricos: Describen y explican las características fenomenológicas del objeto. Representan un nivel de la investigación cuyo contenido procede de la experiencia y se somete a cierta elaboración racional.

Este tipo de método es empleado durante todo el proceso de implementación de la herramienta.

Observación: Se emplea en todo momento. Con este método se analiza cada fase del proceso y se toma experiencia de cada tarea, para aplicarse en otras.

Las **Tareas** a desarrollar son:

- Consultar la bibliografía existente acerca de sistemas de impresión de documentos de identificación.
- Revisar herramientas de impresión y de edición de documentos de identificación existentes en la actualidad a nivel mundial.
- Analizar el estado actual de la identificación de personas en la UCI.

A continuación se muestra el proceso de operacionalización de las variables

Tabla 1: Operacionalización de las Variables

VARIABLE CONCEPTUAL	DIMENSIÓN	INDICADORES	UNIDAD DE MEDIDA
Componentes y utilitarios para la impresión de documentos de identificación.	Factibilidad	Tiempo de desarrollo	Extenso
			Moderado
			Breve
		Costo	Costoso
			Moderado
			Barato
		Esfuerzo	Alto
			Moderado
			Despreciable
Mejor respuesta al incremento de solicitudes de documentos de identificación, en las distintas instituciones donde se realicen estos procesos.	Mejor rendimiento	Complejidad	Alta
			Media
			Baja
		Control	Eficiente
			Deficiente
		Organización del trabajo.	Eficiente
			Deficiente
		Importancia	Alta
			Media
			Baja
		Tiempo de ejecución.	Rápido
			Moderado
			Lento

Tabla 2: Recursos humanos

Nombre del recurso	Rol	Fecha entrada al proyecto	fecha salida del proyecto	Horas mensuales en el proyecto	Salario básico mensual	Costo total
Yainier Labrada Nueva	Líder de proyecto	nov-07	jun-08	100	100MN	800MN
Abdel Sadín Odio	Líder de proyecto	nov-07	jun-08	100	100MN	800MN
Yurdik Cervantes Mendoza	Líder de proyecto	nov-07	jun-08	24	400MN	3200MN

Tabla 3: Recursos Materiales

Recursos	Cantidad	Costo en CUC	Costo Total	Fuente de Financiamiento
Material gastable				
Materiales de oficina				
• Paquete de Hojas	3	5.65	16.95	Institución(UCI)
• Tóner	1	25.00	25.00	
Equipamiento necesario				
Equipos de computación				
• Computadora	2	900.00	1800.00	Institución(UCI)
• Impresora	1	300.00	300.00	
Total			2141.95	

Tabla 4: Cronograma de las Actividades

No.	Acciones a realizar	Responsable	Participantes	Fecha DE INICIO	FECHA FIN
1.	<p>Búsqueda bibliográfica sobre el tema.</p> <ul style="list-style-type: none"> • Materiales relacionados con el proceso de diseño de documentos de identificación e impresión de los mismos. • Materiales relacionados con diferentes tecnologías a utilizar. 	<p>Abdel Sadín Odio</p> <p>Yainier Labrada Nueva</p>		11/07	11/07
2.	Realizar una entrevista para determinar la situación problemática.	Yainier y Abdel	Yurdik Cervantes	11/07	11/07
3.	<p>Procesamiento y análisis de los resultados obtenidos.</p> <ul style="list-style-type: none"> • Diseño Teórico. • Diseño Metodológico. • Fundamentación teórica. 	Abdel y Yainier		11/07	12/07
4.	<p>Especificación de los Requisitos del software.</p> <ul style="list-style-type: none"> • Definición de Requisitos Funcionales • Definición de Requisitos no Funcionales. 	<p>Abdel Sadín Odio y Yainier Labrada Nueva</p>	<p>Yurdik Cervantes Mendoza</p>	01/08	01/08
5.	<p>Modelamiento del Sistema</p> <ul style="list-style-type: none"> • Definición de Actores. 	<p>Yainier Labrada Nueva y Abdel</p>		01/08	02/08

	<ul style="list-style-type: none"> Definición de Casos de Uso. Diagramas de Casos de Uso. Descripción expandida de los Casos de Uso. 	Sadín Odio			
6.	<ul style="list-style-type: none"> Análisis Modelo de clases del análisis. 	<ul style="list-style-type: none"> Abdel Sadín Odio y Yainier Labrada Nueva 	•	• 02/08	• 02/08
7.	<ul style="list-style-type: none"> Diseño Diseño Paquetes y sus relaciones. Diagramas de clases del diseño. Diagramas de interacción. Diagrama Entidad Definiciones de diseño que se apliquen. Interfaz. Tratamiento de errores. 	<ul style="list-style-type: none"> Abdel Sadín Odio y Yainier Labrada Nueva 	•	• 02/08	• 02/08

Este trabajo está estructurado de la siguiente forma:

Capítulo 1 Fundamentación Teórica: contiene la fundamentación teórica de la propuesta presentada, un análisis crítico y tendencia de las tecnologías, técnicas y metodologías a emplear para dar solución

al problema, al igual que se realiza una investigación de Scalable Vector Graphics (SVG), se hace una descripción, se exponen las ventajas y las desventajas, limitantes entre otras cosas.

Capítulo 2 se describe el negocio a través de un modelo de Dominio y, a partir de este, se comienza a hacer el análisis del sistema a desarrollar. Se enumeran los requisitos funcionales y no funcionales que debe tener la aplicación, y se muestran en forma de diagrama los Casos de Usos que de ellos se derivan.

Capítulo 3 se tratan aspectos relacionados con la construcción de la solución que se propone, se modelan los diagramas de clases de análisis, los diagramas de clases del diseño y los diagramas de interacción. Por último, se definen los principios de diseño de la interfaz y el tratamiento de errores. Se aborda la implementación. Contiene los diagramas de despliegue y de componentes, así como una breve descripción de estos procesos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se brinda una visión general de los aspectos relacionados con los procesos de impresión de documentos de identificación, además se describen los principales conceptos asociados al dominio del problema. Se expone un análisis de las tendencias tecnológicas y metodologías actuales, contiene la fundamentación teórica de la propuesta a presentar al igual que se realiza una investigación acerca Scalable Vector Graphics (SVG) describiéndose las ventajas, desventajas y limitantes entre otras características.

1.2 Conceptos asociados al dominio del problema

1.2.1 ¿Qué es un componente?

En esta investigación se utiliza la definición de Sterling Software, que plantea que un componente es: Una parte reutilizable que encapsula elementos del modelo (por ejemplo una DLL, documentos, tablas o bibliotecas).

Un paquete de software el cual ofrece servicios a través de sus interfaces.

Un paquete de software que puede ser usado para construir aplicaciones o componentes más grandes. [1]

1.2.2 ¿Qué se entiende por impresión?

La **impresión** es un proceso para la producción de textos e imágenes, típicamente con tinta sobre papel usando una impresora. A menudo se realiza como un proceso industrial a gran escala, y es una parte esencial de la edición de libros y documentos.

Una **impresora** es un periférico de computadora que permite producir una copia permanente de textos o gráficos de documentos almacenados en formato electrónico, imprimiéndolos en medios físicos, normalmente en papel o transparencias, utilizando cartuchos de tinta o tecnología láser.

1.2.3 ¿Qué son los documentos de identificación?

Se entiende por **Documento de Identificación** o **Documento Nacional de Identidad** al documento emitido por una autoridad administrativa competente para permitir la identificación personal de los ciudadanos. No todos los países emiten documentos de identidad, aunque la extensión de la práctica

acompañó el establecimiento de sistemas nacionales de registro de la población y la elaboración de los medios de control administrativo del Estado.

1.3 Tendencias y tecnologías actuales

Migración hacia Software Libre

El Software Libre es el software que garantiza, sin costo adicional, la libertad de usar el programa con cualquier propósito; de estudiar cómo funciona y adaptarlo según las necesidades. La libertad de distribuir copias, de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad que lo usa se beneficie. La migración hacia el software libre evita la dependencia de los proveedores y se reciben mejores servicios y productos. Cuba y, en particular la Universidad de las Ciencias Informáticas (UCI), han sustentado la posibilidad de migrar hacia Software Libre, por las ventajas que representa con respecto a los de tipo propietario. Desde el ámbito político, primeramente representa la no utilización de productos informáticos que demanden la autorización de sus propietarios (licencias) para su explotación. En el presente Cuba se encuentra a merced de la empresa norteamericana Microsoft, que tiene la capacidad legal de reclamar a Cuba que no siga utilizando un sistema operativo de su propiedad, basada en leyes de propiedad industrial por las cuales también Cuba se rige; esto provocaría una interrupción inmediata del programa de informatización de la sociedad, que como parte de la batalla de ideas está desarrollando el país. Asimismo, pudiera implementarse una campaña de descrédito a la isla, abogando el uso de la piratería informática por parte de las instituciones estatales cubanas.

Además, el software libre representa la alternativa para los países pobres y, es por concepción, propiedad social. Económicamente, su utilización no implica gastos adicionales por concepto de cambio de plataforma de software, por cuanto es operable en el mismo soporte de hardware con que cuenta el país. La adquisición de cualquiera de sus distribuciones, puede hacerse de forma gratuita, descargándolas directamente de Internet o en algunos casos a muy bajos precios.

Por último, desde el punto de vista tecnológico permite su adaptación a los contextos de aplicación, al contar con su código fuente, lo cual garantiza un mayor porcentaje de efectividad, además la corrección de sus errores de programación y obtención de las actualizaciones y nuevas versiones.

Software libre

Software libre es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio

del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente. Análogamente, el software gratuito (denominado usualmente Freeware) incluye en algunas ocasiones el código fuente; sin embargo, este tipo de software no es libre en el mismo sentido que el software libre, al menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.[2]

En 1984, Richard Stallman comenzó a trabajar en el proyecto GNU (es un acrónimo recursivo para "Gnu No es Unix"), fundando la Fundación de Software Libre (*FSF, Free Software Foundation*) un año más tarde. Stallman introdujo una definición para software libre (en inglés, *free software*), el cual desarrolló para dar a los usuarios libertad y para restringir las posibilidades de apropiación del software.

De acuerdo con tal definición, el software es "libre" si garantiza:

- la libertad para ejecutar el programa con cualquier propósito (llamada "libertad 0").
- la libertad para estudiar y modificar el programa ("libertad 1").
- la libertad de copiar el programa de manera que puedas ayudar a tu vecino ("libertad 2").
- la libertad de mejorar el programa y hacer públicas tus mejoras de forma que se beneficie toda la comunidad ("libertad 3").

Es importante señalar que las libertades 1 y 3 obligan a que se tenga acceso al código fuente.

Software libre y software de código fuente abierto

El término *free software* resulta ambiguo en el idioma inglés, ya que *free* significa además de libre, gratis. Para evitar esta ambigüedad es acuñado entonces por Christine Peterson, el término "open source" para definir todo aquel software que fuera de código fuente abierta. "El significado obvio para software de código fuente abierto es: usted puede mirar el código fuente. Este es un criterio más pobre que software libre. Software de código fuente abierto incluye software libre, pero también incluye programas semi libres". [3]

El movimiento código abierto (*Open Source*) apareció en 1998 con un grupo de personas, entre los que cabe destacar a Eric S. Raymond y Bruce Perens, que formaron la Open Source Initiative (OSI). Buscaban darle mayor relevancia a los beneficios prácticos de compartir el código fuente e interesar a las principales casas de software y otras empresas de la industria de la alta tecnología en el concepto. [2]

El movimiento del software libre hace especial énfasis en los aspectos morales o éticos del software, viendo la excelencia técnica como un producto secundario deseable de su estándar ético. El movimiento *Open Source* ve la excelencia técnica como el objetivo prioritario, siendo la compartición del código fuente un medio para dicho fin. Por dicho motivo, la FSF se distancia tanto del movimiento *Open Source* como del término "*Open Source*".

Plataformas de Java

Una plataforma no es tan solo un procesador y el software correspondiente; incluye hardware, software y otros servicios. La Plataforma Java incluye Plataforma Java Edición Estándar (J2SE), Plataforma Java Edición Empresa (J2EE) y Plataforma Java Edición Micro (J2ME).

Plataforma J2EE

J2EE es una plataforma que habilita soluciones para desarrollo, uso efectivo y manejo de multicapas en aplicaciones centralizadas en el servidor. Define los estándares para desarrollar aplicaciones empresariales en Java. J2EE simplifica el desarrollo de este tipo de aplicaciones, basándolas en componentes modulares y estandarizados, y proporcionando un conjunto de especificaciones que aseguran la portabilidad de las aplicaciones entre un amplio número de productos comerciales y de códigos abiertos existentes, capaces de soportar J2EE. Esta plataforma cuenta con las siguientes características: Portable; puede utilizarse en cualquier plataforma para la que haya disponible una Máquina Virtual Java (JVM). Escalable; soporta el aumento de clientes, sin tener que reescribir todo el código de nuevo, tan solo añadiendo nuevos componentes J2EE a una aplicación Web. Altamente Soportada; prácticamente cualquier gran empresa de software tiene un contenedor de componentes (o servidor de aplicaciones) Web compatibles con J2EE. Segura; el entorno de seguridad de la plataforma J2EE permite que se definan unas restricciones de seguridad en el momento de despliegue de la aplicación, aislando las aplicaciones de la complejidad de las implementaciones de seguridad, la plataforma J2EE hace portables una gran complejidad de implementaciones de seguridad [4].

Java como alternativa para el desarrollo

Java es un lenguaje de programación, desarrollado a principios de los años 90 por Sun Microsystems. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos mucho más simple.

Las características principales que ofrece Java respecto a cualquier otro lenguaje de programación, son:

Simple

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Reduce en un 50% los errores más comunes de programación con lenguajes como C y C++, al eliminar muchas de las características de éstos, entre las que se encuentran la aritmética de punteros y la herencia múltiple.

Orientado a objetos

Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.

Distribuido

Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales. Proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

Robusto

Java realiza verificaciones en busca de problemas, tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

Multihebrado

Java es un entorno de ejecución multihebrado, lo que significa que un mismo programa Java puede subdividirse en varias unidades de proceso concurrentes, es decir, que se ejecutan al mismo tiempo, en el mismo espacio de direcciones y compartiendo las mismas variables.

Lo anterior, permite crear de forma sencilla programas que realicen más de una cosa a la vez, aunque exige un gran cambio de mentalidad a la hora de programar.

Multiplataforma

Los programas en Java pueden ejecutarse en cualquier plataforma, sin necesidad de hacer cambios. La compatibilidad es total:

- A nivel de fuentes: El lenguaje es exactamente el mismo en todas las plataformas.

- A nivel de bibliotecas: En todas las plataformas están presentes las mismas bibliotecas estándares.
- A nivel del código compilado: el código intermedio que genera el compilador es el mismo para todas las plataformas. Lo que cambia es el intérprete del código intermedio.[5]

NetBeans como entorno de desarrollo

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun Microsystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos. [6]

XML

XML, siglas en inglés de *Extensible Markup Language* (traducido: —lenguaje de marcas extensible||), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C1). Es una simplificación y adaptación del SGML (Standard Generalized Markup Language o Lenguaje estándar genérico por etiquetas) y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, incluso con posibilidades mucho mayores. Tiene un papel muy importante en la actualidad, pues permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

XSL

XSL (siglas en inglés de Extensible Stylesheet Language, traducido: —lenguaje extensible de hojas de estilo|)) es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera, debe ser transformada o formateada para su presentación en un medio. [7]

XSLT

XSLT o **Transformaciones XSL** es un estándar de la organización **W3C** que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. Las hojas de estilo **XSLT** (aunque el término de hojas de estilo no se aplica sobre la función directa del XSLT) realizan la transformación del documento utilizando una o varias reglas de plantilla: unidas al documento fuente a transformar, esas reglas de plantilla alimentan a un procesador de XSLT, el cual realiza las transformaciones deseadas colocando el resultado en un archivo de salida o, como en el caso de una página web, directamente en un dispositivo de presentación, como el monitor de un usuario.

Actualmente, **XSLT** es muy usado en la edición web, generando páginas HTML o XHTML. La unión de XML y **XSLT** permite separar contenido y presentación, aumentando así la productividad. [8]

XSLT es usado para transformar un documento XML en otro documento XML u otro tipo de documento (como HTML o XHTML) que sea reconocido por un explorador. XSLT normalmente hace eso transformando cada elemento XML en un elemento XHTML.

Con XSLT se puede adicionar o quitar elementos y atributos desde o hacia el fichero de salida. Se pueden reagrupar y ordenar elementos, personalizar pruebas y tomar decisiones acerca de cada elemento que se mostrará o no. Para esto cuenta con una especie de navegador de página llamado XPath.

Una manera común de describir el proceso de transformación es decir que XSLT transforma un árbol con fuente XML en un árbol con resultado XML.

A través de las transformaciones XSL es que se crean las plantillas del lenguaje para el que se decide generar el acceso a datos, de esta manera al cargar las plantillas el núcleo generador de la herramienta genera en el código la información que lee de las plantillas XML del lenguaje y se obtienen las clases que conforman la solución del acceso a datos.

XPath

- Es una sintaxis para definir partes de un documento XML.

- Utiliza expresiones de camino (path expressions) para navegar en los documentos XML.
- Contiene una librería de funciones estándares.
- Es el elemento más importante del XSLT.
- Es un estándar de la W3C.

XPath es un lenguaje para encontrar información en un documento XML, se utiliza para navegar a través de elementos y atributos en un documento XML. Un buen entendimiento de XPath posibilita un uso avanzado de XML.

XPath emplea expresiones de camino para seleccionar nodos o conjuntos de nodos en un documento XML. Estas expresiones de camino son muy parecidas a las expresiones que se ven al trabajar con un sistema tradicional de ficheros de computadora (ej: C:\Archivos de programa\Altova\XMLSpy2007).

Funciones Estándares de XPath

XPath incluye más de 100 funciones integradas. Contiene funciones para cadenas, valores numéricos, comparaciones de fecha y hora, manipulación de nodos, trabajo con secuencias y valores booleanos entre otras.

XMLSpy2007 de Altova

XMLSpy es un editor de XML. Algunas de sus características especiales son:

- Fácil de utilizar.
- Completamiento automático de etiquetas (tags).
- Chequeo automático de buen formato.
- Colores de las sintaxis y buena impresión.
- Validación acoplada de DTD y/o XML Schema.
- Cambio fácil entre las vistas de texto y cuadrículas.
- Editor gráfico de esquemas XML integrado.
- Potentes utilidades de conversión.
- Importaciones y exportaciones de base de datos.
- Plantillas integradas para varios tipos de documentos XML.
- Analizador XPath 1.0/2.0 integrado.
- Editor, profile y debugger de XSLT 1.0/2.0.
- Generación de código en Java, C++, y C#.

- Soporte para Office 2007 / OOXML.

UModel 2007 de Altova

Es el punto de entrada para el desarrollo de software exitoso. Permite crear e interpretar diseños de software mediante la potencia del estándar UML 2.1. Dibuja el diseño de la aplicación y genera código Java o C# a partir de sus planos. Es una herramienta que permite realizar la ingeniería inversa de programas existentes, a diagramas UML claros y precisos para abarcar rápidamente la arquitectura. Además puede corregir el código generado sincronizando con el modelo y viceversa.

UML

UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. Fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación y plataformas de objetos distribuidos (como por ejemplo J2EE, .NET). El hecho de que UML sea un lenguaje de propósito general proporciona una gran flexibilidad y expresividad a la hora de modelar sistemas.

Es importante resaltar que UML es un "lenguaje" para especificar y no un método o un proceso, se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Es el lenguaje en el que está descrito el modelo. UML puede aplicar en una gran variedad de formas para soportar una metodología de desarrollo de software, (tal como el Proceso Unificado de Rational), pero no especifica en sí mismo qué metodología o proceso usar.

Metodologías

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Indican, paso a paso, todas las actividades a realizar para lograr el producto informático deseado, además de las personas que deben participar en el desarrollo de las actividades y qué papel deben tener. Detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. Las metodologías se encargan de elaborar estrategias de desarrollo de software que promuevan prácticas centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente. Una metodología de desarrollo de software permite producir organizada y económicamente software de alta calidad, siguiendo una serie de pasos donde se utilizan un conjunto de técnicas, notación y normas de documentación preestablecidas.

RUP

El Proceso Unificado de Rational, es un proceso de desarrollo de software, orientado a objetos, preparado para desarrollar grandes y complejos proyectos, unifica los mejores elementos de metodologías anteriores y utiliza el Lenguaje Unificado de Modelado UML, como lenguaje de representación visual. En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Flujos de trabajo:

- Modelamiento del negocio
- Requerimientos
- Análisis y diseño
- Implementación
- Prueba (Testeo)
- Instalación
- Administración del proyecto
- Administración de configuración y cambios
- Ambiente

El proceso de ciclo de vida de RUP se divide en cuatro fases bien conocidas llamadas Concepción, Elaboración, Construcción y Transición.

Esas fases se dividen en iteraciones, cada una de ellas produce una pieza de software demostrable.

Fases:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema, que orientarán la funcionalidad.
- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales), identificados de acuerdo con el alcance definido.
- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios release del producto que han pasado las pruebas.

Se ponen estos release a consideración de un subconjunto de usuarios. Es la fase más prolongada de todas.

- **Transición:** El release ya está listo para su instalación en las condiciones reales. Se corrigen los últimos errores. Se llama transición porque se transfiere a las manos del usuario, pasando del entorno de desarrollo al de producción.

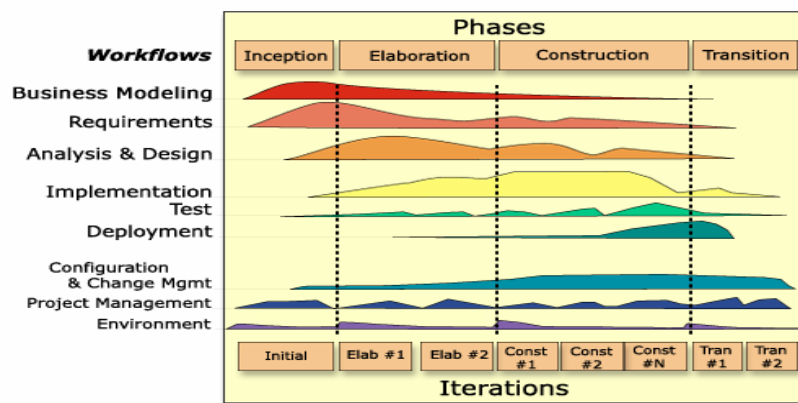


Figura 1. Ciclo de vida de RUP

El ciclo de vida de RUP se caracteriza por:

- Dirigido por Casos de Uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo pues los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Programación Extrema (XP)

Es una de las metodologías de desarrollo de software utilizadas para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

La metodología se basa en:

- **Pruebas Unitarias:** Son las pruebas realizadas a los principales procesos, de tal manera que, se puedan hacer pruebas de las fallas que pudieran ocurrir.
- **Refabricación:** Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares:** Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.[9]

Herramientas para el modelado

La misión de cualquier herramienta CASE, que utiliza UML como notación para elaborar los modelos, es comunicar, de la manera más eficiente posible, a los agentes del proyecto, todas aquellas decisiones que se toman con respecto a la arquitectura del sistema en discusión y que son determinantes para cumplir con los objetivos de las distintas fases de un proyecto.

Rational Rose Enterprise Suite

El Rational Rose, es una herramienta CASE desarrollada por Rational Corporation, basada en UML, que permite crear los diagramas que se van generando durante el proceso de Ingeniería en el desarrollo del software. Las personas que desarrollaron RUP son miembros de Rational Corporation, por lo que el mismo es completamente compatible con esta metodología, brinda muchas facilidades en la generación de la documentación del software que se está desarrollando, además posee un gran número de estereotipos predefinidos que facilitan el proceso de modelación del software.

Dicha herramienta es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño, pero tiene la limitación de que aún hay varios lenguajes de programación que no soporta o que solo lo hace a medias. Por otra parte, una vez que se tiene el diagrama de clases persistentes a

partir del cual se genera la base de datos del sistema, no existe la posibilidad de exportar ese modelo hacia algún sistema gestor de bases de datos. [10]

Visual Paradigm

Visual Paradigm es una herramienta de desarrollo que se integra al IDE de NetBeans. Diseñada para desarrollar software con Programación Orientada a Objetos, busca reducir la duración del ciclo de desarrollo brindando ayuda tanto a arquitectos, analistas, diseñadores y desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores.

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

Herramientas para el control de versiones del software

Subversion

Es un sistema de control de versiones libre y de código fuente abierta. Es decir, maneja ficheros y directorios a través del tiempo. Existe un árbol de ficheros en un *repositorio* central. El repositorio es como un servidor de ficheros ordinario, con la excepción que recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. Brinda la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas estaciones de trabajo. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros. Existen varias interfaces a Subversion, ya sea programas individuales como interfaces que lo integran en entornos de desarrollo, como es el caso de TortoiseSVN, el cual

provee integración con el explorador de Windows, siendo este la interfaz más popular en este sistema operativo.

TortoiseSVN

Es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. Maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un repositorio central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio. Es fácil de usar, permite ver el estado de los archivos desde en el explorador de Windows y permite también crear gráficos de todas las revisiones/asignadas.

Estado actual de las herramientas de diseño e impresión de documentos de identificación en el mundo. Herramientas más utilizadas.

En el mundo existe un gran número de aplicaciones que de una forma u otra, se relacionan con la identificación. Algunos de los más conocidos son:

Number Five CardFive es un software para el diseño de credenciales con diseño personalizado y soporte de las mayores impresoras de tarjetas PVC (un material plástico). El mismo ofrece flexibilidad para la creación y diseño de las propias credenciales PVC. El diseño a utilizar es avanzado y amigable. En sus distintas versiones permite utilizar los datos automáticamente mediante el acceso a una base de datos existente, donde se encuentra la información necesaria para la impresión en la tarjeta plástica.

Evolis eMedia Software es un programa que permite al usuario el diseño personalizado de credenciales para su posterior salida en las impresoras de tarjetas de PVC. En sus distintas versiones permite utilizar los datos automáticamente accediendo a una base de datos existente, la cual posee la información necesaria para la impresión en la tarjeta plástica. Posee un alto nivel de configuración y trabajo con imágenes. Este software es líder en impresoras de PVC de alta calidad a nivel mundial. Tiene más de 10 años de experiencia y ha sido usado en más de 70 países.

DataCard Id Work Id Cards Software es un diseño de aplicación personalizable para el programa de tarjetas de identificación. El software de identificación de ID Works mejora cada aspecto del diseño de tarjeta y reportes, así como la producción. Es extremadamente fácil de usar, pues tiene un diseño flexible y modular, permite seleccionar sólo los componentes específicos que se necesitan. DataCard

es una compañía con más de 30 años de experiencia, con una marcada connotación internacional en lo que a soluciones informáticas respecta.

Expreso Creador de Tarjeta Fácil (Easy Card Creator Express) es un software que permite la creación e impresión de tarjetas profesionales de identificación al instante, esta herramienta provee una amplia variedad de instrumentos, entre ellos textos, gráficos, efectos para el diseño entre otros, lo que le permite al diseñador más facilidad cuando tiene que diseñar algún documento.

Pequeño Editor Comercial del Belltech, proporciona una solución profesional para pequeños negocios, es un software que permite imprimir encabezamientos de calidad, entre ellos folletos, catálogos, boletines de noticias, tarjetas postales, sobres, etiquetas de dirección, credenciales de nombre, tarjetas de invitación.

Estado actual de este tipo de herramientas en la UCI

Actualmente en la UCI existen varios software con funcionalidades similares al que se espera desarrollar, el más significativo es el **CredentialBuilder**. El mismo está en la fase de pruebas, el cual se realizó con una tecnología propietaria y costosa (En la plataforma Visual Studio.NET y C# como lenguaje de programación), esta herramienta goza de una interfaz amena para los usuarios, luciendo la paleta de dibujo, componentes entre otros., permitiendo el diseño de documentos de identificación, y la impresión de los mismos, así como también provee la forma de como obtener desde una fuente de datos (Base de Datos) los datos.

Antes de ir al estudio de cada editor por separado se debe dar una definición de que es SVG.

SVG Significa Scalable Vector Graphics una gramática XML para gráficos vectoriales 2D, utilizable como XML de nombres. Ser **Escalable**: Significa aumentar o disminuir uniformemente. En términos gráficos, escalable significa que no se limita a un solo tamaño fijo de pixel. En la Web, escalable significa que una tecnología en particular puede crecer a un gran número de archivos, un gran número de usuarios, una amplia variedad de aplicaciones. SVG, al ser una tecnología de gráficos para la Web, es escalable en ambos sentidos de la palabra. Los gráficos SVG son escalables a diferentes resoluciones de pantalla, un ejemplo de ellos puede ser que la salida impresa utilice la plena resolución de la impresora y se pueda mostrar en el mismo tamaño de las diferentes resoluciones de las pantallas. El mismo gráfico SVG se puede colocar en diferentes tamaños en la misma página web, y volver a utilizarse en diferentes tamaños en diferentes páginas. Los gráficos SVG pueden ser ampliados para ver detalles finos, o para ayudar a las personas con baja visión. **Vector**: Los gráficos vectoriales contienen objetos geométricos como líneas y curvas, esto da una mayor flexibilidad en

comparación con los formatos de raster-sólo (como JPEG y PNG) que se han de almacenar la información para cada píxel de la imagen, normalmente, los vectores también pueden integrar los formatos de imágenes de trama y pueden ser combinarlos con el vector de la información para producir una completa ilustración; SVG no es una excepción. Ya que todas las pantallas modernas están orientadas a la trama, la diferencia entre la trama y gráficos vectoriales sólo se reduce a donde están rasterized. **Gráficos:** La mayoría de las gramáticas de XML existentes representan cualquier información textual, o representan los datos en bruto. Ellos suelen proporcionar sólo rudimentarias capacidades gráficas, a menudo menos capaces que el HTML 'img' elemento. SVG llena un vacío en el mercado mediante el suministro de una rica y estructurada descripción de los vectores. **Scalable Vector Graphics:** Es un formato vectorial basado en la tecnología XML que permite escalar e imprimir los dibujos realizados en cualquier resolución sin perder calidad ni obtener efectos granulosos sino más bien respetando el nivel de detalle del dibujo sin pérdidas de ningún tipo.

A continuación se exponen características del SVG.

Descripción de SVG

Gráficos de vectores escalables (Scalable Vector Graphics - SVG) es un estándar de gráficos basados en XML del consorcio W3C (World Wide Web Consortium), que abre nuevas y poderosas maneras para que los programadores generen gráficos para una gran variedad de dispositivos. SVG está concebido para la creación de gráficos en 2 dimensiones, siendo de gran utilidad para desplegar una gran variedad de información incluyendo estadísticas cuantitativas, gráficos, mapas y diagramas técnicos. SVG es particularmente útil para la elaboración de gráficos que son creados a partir de datos en formato XML.

SVG está previsto para desplegar tres tipos de entidades visuales:

Formas gráficas: SVG cuenta con formas gráficas predefinidas así como también elementos para trayectorias que pueden describir cualquier forma bidimensional.

Texto: SVG provee de texto y elementos para desplegar dicho texto.

Gráficos Bitmap: SVG permite gráficos convencionales (tipo raster) como PNG, para ser mezclados en un documento o imagen SVG.

A continuación las ventajas

Ventajas de SVG

En esta sección se presentan brevemente algunas ventajas de SVG:

- El Consorcio W3C especialmente preveía a SVG como un subtítulo para los gráficos raster (bitmaps) en la web. Una de las aspiraciones era que el tamaño de los archivos SVG fuera menor que el de los gráficos raster o bitmaps. En la práctica es posible crear logos, banners y otras imágenes significativamente de menor tamaño que el gráfico en raster. Desde luego que el tamaño del SVG está en función del detalle del gráfico vectorial, pero además este gráfico vectorial puede ofrecer la funcionalidad de varios gráficos raster además de proveer paneo y zoom de la imagen, que los gráficos raster no proporcionan.
- En las imágenes SVG se puede hacer zoom sin la necesidad de cargar otra versión de la imagen. Esta capacidad es una gran ventaja en campos como la cartografía y la ingeniería.
- Paneo de imágenes, cuando a una imagen se le hace zoom. El usuario tiene un control sobre el área a manipular, pero el área que se despliega después de hacer el acercamiento a la imagen puede no ser exactamente el área deseada.
- Visualización selectiva de elementos, en algunas situaciones ver imágenes que se enfocan en ciertos tipos de datos puede ser de utilidad. SVG, cuando es combinado con un lenguaje de programación más convencional como Java Script, permite una visualización selectiva de información.
- Es código abierto, el código fuente puede ser desplegado por el visualizador de SVG. En el visualizador de Adobe, un clic derecho sobre la imagen muestra un menú que incluye la opción de ver el código fuente. Elegir esa opción provoca que el navegador abra una ventana con el código fuente. Ese código puede ser inspeccionado para un mayor entendimiento de cómo fue creada la imagen, también se tiene la opción de guardar el código fuente al disco duro.
- Internacionalización del contenido, cada vez más, los sitios son diseñados para alcanzar una audiencia internacional. SVG cuenta con soporte para la internacionalización de texto. En adición proporciona soporte para desplegar texto condicionalmente al lenguaje correspondiente a la ubicación o características del browser que despliega el contenido.

- Accesible para motores de búsqueda, el texto dentro de las imágenes SVG permanece como tal porque es desplegado el carácter que es contenido por uno de los elementos SVG que determina como se despliega el texto en la imagen. Este texto puede ser buscado mediante motores de búsqueda de XML. Esto contrasta favorablemente con gráficos raster usados en páginas web donde el texto es convertido simplemente a un patrón de píxeles sólo descifrable por el ojo humano.
- Manejo de datos gráficos, los gráficos SVG pueden ser estáticos o creados dinámicamente al tiempo que una página o una imagen es servida al cliente. El documento SVG cuando es creado dinámicamente puede hacer uso de los datos contenidos en cualquier lugar, como lo puede ser un documento XML o en una base RDBMS para proveer información a desplegar en el gráfico.
- Independiente de resolución. Las imágenes SVG son dibujadas de cierta manera que son independientes de la resolución del dispositivo en que se dibujan. A diferencia de las imágenes raster que consisten básicamente en un vector bidimensional con información de píxeles individuales, las imágenes SVG son dibujadas a partir de instrucciones contenidas en el código fuente.
- Visualización en dispositivos móviles. Una de las metas de los requerimientos del documento original de SVG era que estuviera disponible en una gran cantidad de dispositivos, no solamente en el escritorio convencional del navegador.
- Rollovers. Un uso común de las imágenes raster en páginas web es proveer funcionalidad de rollovers, por ejemplo un cambio de forma o color de un botón u otro objeto en respuesta a un evento del Mouse. SVG introduce la capacidad de crear rollovers usando elementos prestados de animación SMIL.

Limitantes de SVG

- El tamaño del visualizador. Un punto negativo del visualizador de Adobe es que es mucho más grande que el de la competencia Macromedia Flash.
- Código abierto. Algunos creadores de SVG se preocupan de que el código fuente de sus gráficos esté disponible a los usuarios del sitio, esto puede crear temor.

- Uso intensivo de CPU. Un visualizador de SVG hace uso extensivo del CPU durante los múltiples cálculos requeridos por complejas animaciones SVG. La carga del CPU es determinada por el tamaño de la imagen SVG, una imagen grande ocupa más CPU, así como por el número, tipo y complejidad de las animaciones. Esta limitante se ha ido mejorando con el tiempo. El visualizador de adobe y otros se están volviendo más eficientes.
- Herramientas limitadas. Inevitablemente toma tiempo después de que una tecnología es liberada para que se desarrollen herramientas y se hagan disponibles al público.

Herramientas básicas

Para hacer uso de imágenes SVG se requieren de 2 tipos de módulos: uno para la creación de la imagen SVG y otro para la visualización de la imagen. Un editor de textos puede ser usado para escribir código SVG, si se tiene el conocimiento de la sintaxis, también se pueden utilizar aplicaciones para editar XML, por ejemplo se puede usar XMLSPY versión 4.4 o posterior. La funcionalidad SVG ha sido agregada a las herramientas de dibujo existentes tales como Adobe Ilustrador y CorelDRAW, pero la primera herramienta dedicada al dibujo y animación SVG es Jasc WebDraw. Su interfaz tiene ciertas ventajas para visualizar el dibujo, canvas en una pestaña y en otra el código fuente y si una se altera automáticamente la otra se modifica.

Se tienen herramientas para visualizar imágenes SVG. A continuación una breve descripción de una de ellas:

Batik es una herramienta para aplicaciones o applets basadas en Java que desean utilizar imágenes en formato SVG para diversos fines, como la visualización en pantalla, la generación o la manipulación de estos.

El proyecto tiene como objetivo darle a los desarrolladores un conjunto de módulos básicos que pueden utilizarse, juntos o por separado, para apoyar soluciones específicas SVG. Algunos ejemplos de módulos son los Parseadores SVG (SVG Parsers), el generador SVG y el SVG DOM. Otra ambición del proyecto Batik es que éste sea altamente ampliable, por ejemplo, Batik permite que los desarrolladores puedan manejar elementos SVG personalizados. A pesar de que el objetivo del proyecto es proporcionar un conjunto de módulos básicos, uno de los resultados más palpables fue la salida de la implementación completa de un navegador o browser SVG que valida los distintos módulos y su interoperabilidad.

Con Batik, es posible manipular los documentos SVG en cualquier lugar donde Java esté disponible. También se pueden utilizar los diversos módulos de Batik para generar, manipular y codificar las imágenes SVG en aplicaciones o applets.

Batik hace que, aplicaciones basadas en Java o en applets, puedan interactuar fácilmente con el contenido SVG. Por ejemplo, utilizando el módulo generador de SVG de Batik, una aplicación Java puede exportar sus gráficos al formato SVG. Otra posibilidad es utilizar los módulos de Batik para convertir imágenes vectoriales SVG a diferentes formatos, tales como imágenes de trama (JPEG, PNG o TIFF) o de otros formatos de vectores (EPS o PDF, los dos últimos son proporcionadas por re codificadores de tipo Apache FOP).

Modelo del Pintor

En la creación de imágenes SVG el relleno de las formas SVG es descrita como *fills* y su periferia es reseñada como *stroke*. El modelo del pintor SVG trabaja de manera similar a pinturas de aceite sobre el óleo. El código que viene primero a una imagen SVG provoca que un objeto sea dibujado. La siguiente sección de código que se refiere a la misma área en la pantalla es dibujada, ocultando la primera imagen (código) donde se traslapan las imágenes. A continuación se muestra el código de un documento SVG y en la siguiente figura se puede observar su imagen gráfica que muestra lo expuesto en este punto.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="250" height="250">
    <!--Poligono azul -->
    <polyline points="19,53 29,199 184,115 131,51"
        transform="translate(69 15) translate(-46 12)"
        style="fill:rgb(0,0,255);stroke:rgb(0,0,0);stroke-width:1"/>
    <!--Poligono rojo -->
    <polygon points="164.5,44.9715 224,148.029 105,148.029"
        transform="matrix(1 0 0 1.4458 0 -43.0198) translate(-46
8.29989)"
        style="fill:rgb(255,0,0);stroke:rgb(0,0,0);stroke-width:1"/>
</svg>
```

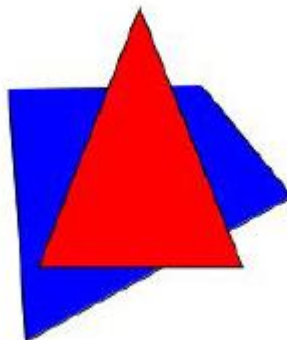


Figura 2. Ejemplo gráfico modelo del pintor

SVG DOM

SVG tiene su propio DOM (Document Object Model), que está basado en el DOM de XML y permite la manipulación específica del modelo de objetos de un documento SVG. El visualizador de Batik permite ver la representación del SVG DOM. Jasc WebDraw también provee una representación del SVG DOM. El propósito del SVG DOM no es primariamente permitir visualizarlo, sino que se puede tener el control para manipular la estructura y contenido de un documento SVG mediante la programación.

Estructura del documento SVG

La estructura de un documento SVG, es como la de un archivo XML, dado que el SVG está basado en XML. A continuación se describe la estructura del lenguaje SVG.

SVG es un documento escrito en sintaxis de XML 1.0. Tiene similitudes con HTML pero si se quiere que las imágenes se desplieguen correctamente se necesita seguir algunas reglas de XML y SVG que los navegadores y HTML no requieren.

Un documento SVG puede empezar con una declaración XML opcional:

```
<?xml version="1.0"?>
```

Si la declaración XML está presente debe estar en la primera línea del documento y ningún carácter debe precederla, ni tan siquiera un espacio. El atributo de la versión es obligatorio y su único valor permitido es "1.0". La declaración también puede tener un atributo de `standalone` con los valores permitidos "yes" o "no". Adicionalmente puede contener el atributo `encoding` utilizado para especificar la codificación de los caracteres en el documento. Todos los parsers de XML deben soportar, al lo menos, UTF-8 y UTF-16.

Un documento XML puede tener una declaración de tipo de documento (DTD), normalmente referido como una declaración DOCTYPE, para indicar información acerca de la estructura del documento. Para un documento SVG 1.0 la declaración DOCTYPE típicamente toma la siguiente forma:

```
<! DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN">  
http://www.w3.org/TD/2001/REC-SVG-20010904/DTD/svg10.dtd
```

SVG en la primera línea indica el nombre del tipo de elemento del documento, es decir, el elemento en el que todos los demás elementos del documento estarán anidados. Después la cadena que le sigue la palabra `PUBLIC` es el identificador público para el tipo de documento. La última cadena encerrada

entre comillas es el url donde se encuentra la definición del tipo de documento (DTD) para un SVG 1.0. El DTD define la estructura permitida para un documento SVG.

Los nombres del tipo de elementos en todos los lenguajes basados en XML son sensibles a mayúsculas y minúsculas. En SVG la mayoría de los nombres de elementos están en minúsculas, pero no todos. En cualquier caso es recomendable que se usen los nombres de los elementos tal cual están definidos en la especificación SVG.

Todos los elementos que tienen contenido deben tener una etiqueta de inicio así como también deben tener una etiqueta de fin. Omitir la etiqueta de fin causará un error. Los elementos vacíos pueden ser escritos en una sola etiqueta con una diagonal "/" antes del símbolo de cierre ">".

Cuando se agregan atributos a un elemento de SVG, se deben utilizar usar comillas para delimitar el valor del atributo. Se pueden usar comillas simples o dobles, pero siempre indicar el valor del atributo con las mismas comillas para cada atributo. Por ejemplo, se puede escribir el siguiente texto para definir la esquina superior derecha de una recta:

```
<rect x="0" y='0.../'>
```

Se pueden agregar comentarios a documentos SVG. La sintaxis es la misma que para HTML/XHTML:

```
<!-- Esto es un comentario.-->
```

Adicionalmente SVG tiene el elemento "des" que puede ser usado para proveer una descripción del documento, una descripción de un elemento individual y su propósito. En SVG se pueden usar instrucciones de procesamiento para asociar cascadas de estilos externas (CSS) con un documento SVG. Una instrucción de proceso inicia con los caracteres <? Seguido por el nombre de la instrucción de proceso seguida de texto indicando los atributos.

Un número de herramientas de dibujo de SVG, incluyendo Adobe Ilustrador, generan entidades en el código SVG que exportan. Una entidad es un termino XML 1.0 que se refiere a la manera de guardar la información que será usada más de una vez, por eso SVG tiene algunas entidades:

- <: el signo menor que.
- >: el signo mayor que.
- ": comillas dobles.
- ': comillas simples o apostrofe.

- `&`: ampersan “&”

El uso de entidades puede significar una reducción en el tamaño del archivo, ya que se suprime cierto texto en el documento.

Para distinguir nombres de elementos de otros elementos de XML se pueden utilizar “declaraciones namespaces”, que es una manera de asociar un prefijo con un URL (Uniform Resource Identifier).

Cuando el código de un script es agregado a un documento SVG simplemente se puede colocar entre las etiquetas de script y el visualizador tratará de parsearlo como un documento XML. El código script típico es significativamente diferente de la sintaxis de un documento XML e inevitablemente ocurre un error. Para prevenir dichos errores se debe colocar el código del script entre secciones CDATA. Esta debe ser colocada entre la etiqueta de inicio y de fin del elemento `script` como sigue:

```
<script type="text/javascript">
<![CDATA[]]>
</script>
```

El elemento SVG

Un fragmento de documento de SVG consiste de un elemento `svg` y su contenido. Cuando en el fragmento del documento SVG existe como “standalone” se le llama documento SVG. Un documento SVG podría estar vacío. Sin embargo puede contener complejos y anidados elementos describiendo un gráfico estático, animación o algún script. El área donde se dibuja es llamada “viewport”. Los atributos `width` y `height` en el elemento SVG del documento son usados para definir las dimensiones del “viewport”. Todos los documentos SVG tienen un elemento SVG como elemento principal del documento, en cuyo interior va todo el documento.

Si el elemento SVG no tiene los atributos de `width` y `height` pueden ser especificados en cualquiera de las siguientes unidades: `em` (igual al tamaño de la letra), `ex` (el tamaño en x del tipo de letra actual), `px` (píxeles), `pt` (puntos, 72 puntos = 1 pulgada), `cm` (centímetros), `mm` (milímetros), `pc` (picas, 6 picas = 1 pulgada), `%` (porcentaje del tamaño relevante a la ventana del navegador), `in` (pulgadas).

El elemento SVG puede tener un atributo `versión` que no necesariamente debe ser incluido y si está presente debe tener el valor de “1.0”. En futuras versiones de SVG a partir de la 1.1 el atributo será requerido.

En los elementos SVG se puede tener anidado otro de igual identidad, lo que implica cierta complejidad, pero tiene varias ventajas. Por ejemplo cada elemento SVG anidado puede ser tratado como componente. Si se ajustan los valores de los atributos de X e Y del elemento SVG anidado todo el componente es movido en la pantalla. Además el contenido del elemento SVG puede ser importado usando el elemento image.

El atributo `viewbox` ayuda a definir donde se encontraran los objetos en la pantalla. Esto se lleva a cabo definiendo un nuevo sistema de coordenadas. Este atributo provee una manera de escalar (zoom) el contenido de un elemento SVG. El atributo `viewbox` contiene 4 valores numéricos separados por espacios. La diferencia entre el primero y el tercero es el ancho y la diferencia entre el segundo y el cuarto es la altura. A continuación se muestra la parte del script que utiliza el atributo `viewbox`. En la figura 2 se puede ver una imagen antes de utilizar el atributo `viewbox` y otra después de haberle aplicado dicho atributo.

```
svgMainViewport = svgdoc.getElementById ("allMap");  
svgMainViewport.setAttribute ("viewBox", new Viewport);
```



Figura 3 Atributo Viewbox

En SVG se pueden agrupar elementos mediante el elemento de agrupación `g`, que a su vez pueden contener elementos `desc` y `title` que proveen la información acerca del contenido del elemento `g`, lo que resulta de gran utilidad al permitir aplicar un estilo a todos los elementos, uno a uno, o agrupándolos en un elemento `g` y que al aplicarle ese estilo todos sus subelementos también lo tengan. También es aplicable a las traslaciones.

El código de script es un documento SVG está relacionado siempre con un elemento `script`. Típicamente el elemento `script` no tiene prefijos de namespaces. A pesar de que los `script` de

HTML/XHTML tienen una función similar, su sintaxis no es la misma. El código script como JavaScript no satisface las sintaxis de los requerimientos de XML 1.0. Para evitar errores cuando el visualizador intenta interpretar el código JavaScript u otro código script como XML, se debe encerrar entre secciones CDATA.

Alternativamente el código script puede estar contenido en un archivo externo que es referenciado usando el atributo `xlink:href`. En el elemento `script`.

SVG soporta hacer acercamientos (zoom) y paneos (panning). Los acercamientos permiten desplegar la imagen con diferentes magnitudes, mientras que el paneo posibilita que la imagen SVG sea movida a lo largo de la pantalla. SVG ofrece la oportunidad de deshabilitar el acercamiento y el paneo. Esto se realiza mediante el atributo `zoomAndPan` que es colocado en el elemento SVG. En cuanto a la magnificación, esta se puede realizar hasta 16 veces ya sea haciendo “zoom out.” o “zoom in”. El paneo (panning) permite al viewport, en teoría, cambios infinitos de canvas. Diferentes visualizadores de SVG varían en las técnicas de usar el paneo.

El hacer enlaces (linking) en SVG depende del lenguaje de enlaces de XML. Todos los enlaces internos y externos en imágenes SVG, incluyendo aquellas que usan identificadores de fragmentos hacen uso de los atributos de XLink. XLink es una especificación de w3c para la descripción de enlaces entre recursos. Toda la semántica de XLink está expresada en atributos globales. XLink provee enlaces extendidos y enlaces simples. SVG solo implementa los enlaces simples.

Un elemento SVG asocia exactamente 2 recursos: un recurso de inicio local y el otro, final remoto, mientras las partes más complejas de los enlaces permanecen ocultas. El comportamiento por defecto (default) de un enlace simple en SVG es reemplazar el documento en donde el recurso inicial esta por el recurso final. XLink provee una forma de abrir el recurso final en una ventana nueva.

SVG proporciona una implementación parcial del lenguaje de apuntador XML (XML pointer “XPointer”), que es el lenguaje identificador del fragmento para documentos genéricos XML. Un conjunto limitado de instrucciones XPointer es usado en SVG para habilitar la reutilización de elementos contenidos en etiquetas `defs` en los documentos SVG. El elemento `view` permite cambiar la apariencia del contenido del elemento SVG que lo contiene.

Interfaces SVG DOM

El principal propósito de SVG DOM es permitir al programador tener una interfaz de programación de aplicación (API) para acceder y modificar el documento SVG al cual se le relaciona el DOM.

El documento SVG en si mismo está representado por un documento nodo. En términos de XML DOM el documento nodo es la raíz jerárquica de DOM. Todos los demás nodos en el DOM son vistos en

relación a su nodo. El documento nodo tiene nodos hijos, y unos de nodos hijos pueden a su vez tener más nodos hijos y así sucesivamente.

El documento nodo de SVG tiene todos sus métodos y propiedades descritas por un documento nodo en la especificación DOM nivel 2. En la especificación de eventos de DOM nivel 2, además tiene las siguientes propiedades específicas de SVG:

- `title`, de tipo `DOMString` – sirve para poner un título al documento.
- `Referrer`, de tipo `DOMString` – permite decir qué documentos hacen referencia al documento actual.
- `Domain`, de tipo `DOMString` – posibilita asociar un url a este documento.
- `URL`, de tipo `DOMString` – sirve para asociar un url a este documento.
- `rootElement`, un elemento de tipo `SVGSVGElement` – es el elemento raíz del documento.

Solo se tienen los métodos `get` de estas propiedades, y son de la forma `get<propiedad empezando con mayúscula>`.

El objeto `SVGElement` tiene todas las propiedades de un elemento XML DOM, además de tener las siguientes propiedades:

- `Id` – de tipo `DOMString`.
- `xmlbase` – de tipo `DOMString`.
- `ownerSVGElement` – de tipo `SVGElement`.
- `viewportElement` – de tipo `SVGElement`. La propiedad `id` corresponde a un atributo `id` del documento SVG, si es que este posee uno. La propiedad `xmlbase` corresponde a la base uri, como está definido en xml. La propiedad `ownerSVGElement` corresponde al elemento SVG del documento. La propiedad `viewport` corresponde al elemento SVG que define el viewport por el elemento.

Objeto `SVGSVGElement`

El objeto `SVGSVGElement` es específico solo para un elemento SVG. Este objeto tiene un número sustancial de propiedades y métodos específicos. Las propiedades de un objeto `SVGSVGElement` son las siguientes:

- `x` – de tipo `SVGAnimatedLength`, muestra la longitud máxima del documento en el eje X.
- `y` – de tipo `SVGAnimatedLength`, muestra la longitud máxima del documento en el eje Y.
- `width` – de tipo `SVGAnimatedLength`, muestra el ancho del documento.
- `height` – de tipo `SVGAnimatedLength`, muestra la altura del documento.

- `contentScriptType` – de tipo `DOMString`, muestra el tipo de Script que contiene el documento.
- `contentStyleType` – de tipo `DOMString`, muestra el estilo que posee el documento.
- `viewport` – de tipo `SVGRect`, dice lo que actualmente se está visualizando del documento.

Objeto `SVGLength`

El elemento `SVGLength` da la longitud del documento SVG y tiene las siguientes constantes:

- `SVGLength.SVG_LENGTHTYPE_UNKNOWN` – es de tipo short y tiene un valor de 0.
- `SVGLength.SVG_LENGTHTYPE_NUMBER` – es de tipo short y tiene un valor de 1. Esto aplica cuando se asigna un valor numérico a un atributo como `x`.
- `SVGLength.SVG_LENGTHTYPE_PERCENTAGE` – es de tipo short y tiene un valor de 2. Esto se aplica cuando se asigna un valor de por ciento a un atributo como `x`.
- `SVGLength.SVG_LENGTHTYPE_EMS` – es de tipo short y tiene un valor de 3. Esto se aplica cuando se asigna un valor numérico con unidades `em` a un atributo como `y`.
- `SVGLength.SVG_LENGTHTYPE_EXS` – es de tipo short y tiene un valor de 4. Esto se aplica cuando se asigna un valor numérico con unidades `ex` a un atributo como `x`.
- `SVGLength.SVG_LENGTHTYPE_PX` – es de tipo short y tiene un valor de 5. Esto se aplica cuando se asigna un valor numérico con unidades `px` (píxeles) a un atributo como `y`.
- `SVGLength.SVG_LENGTHTYPE_CM` – es de tipo short y tiene un valor de 6. Esto se aplica cuando se asigna un valor numérico con unidades `cm` (centímetros) a un atributo como `y`.
- `SVGLength.SVG_LENGTHTYPE_MM` – es de tipo short y tiene un valor de 7. Esto se aplica cuando se asigna un valor numérico con unidades `mm` (milímetros) a un atributo como `y`.
- `SVGLength.SVG_LENGTHTYPE_IN` – es de tipo short y tiene un valor de 8. Esto se aplica cuando se asigna un valor numérico con unidades `in` (pulgadas) a un atributo como `y`.
- `SVGLength.SVG_LENGTHTYPE_PT` – es de tipo short y tiene un valor de 9. Esto se aplica cuando se asigna un valor numérico con unidades `pt` (puntos) a un atributo como `font-size`.
- `SVGLength.SVG_LENGTHTYPE_PC` – es de tipo short y tiene un valor de 10. Esto se aplica cuando se asigna un valor numérico con unidades `pc` (picas) a un atributo como `y`.

El objeto `SVGLength` también tiene las siguientes propiedades:

- `unitType` – de tipo `unsigned short`, muestran el tipo de unidades que se usa.
- `Value` – de tipo flotante, muestran el valor de la longitud del documento.
- `valueInSpecifiedUnits` – de tipo flotante, proporciona el valor del documento en las unidades específicas.
- `valueAsString` – de tipo `DOMString`, proporciona la longitud del documento como una cadena

El objeto `SVGLength` también tiene dos métodos que son específicos y extras además de los `set` y los `get`:

- `newValuesSpecifiedUnits (unitType, valueInSpecifiedUnits)` – este método regresa un valor de tipo `void`. El argumento `unitType` es de tipo `unsigned short`. El argumento `valueInSpecifiedUnits` es de tipo flotante.
- `convertToSpecifiedUnits (unitType)` – el método regresa un valor de tipo `void`. El argumento `unitType` es de tipo `unsigned short`.

Otros Objetos SVG DOM

A continuación se describen otros objetos `SVGDOM` que se pueden usar para el manejo de documentos SVG.

Objeto `GetSVGDocument`

Es usado para regresar un documento SVG usando código script. Este objeto no tiene constantes ni propiedades específicas y tiene el siguiente método:

- `getSVGDocument ()` – el documento regresa un valor de tipo `SVGDocument`. El método no toma argumentos.

Objeto `SVGElement`

Este objeto corresponde a un elemento en el código SVG. Este objeto tiene las propiedades y métodos de los objetos `SVGElement`, `SVGURIReference`, `SVGTest`, `SVGLangSpace`, `SVGExternalResourceRequired`, `SVGStylable`, `SVGTransformable` y el elemento `EventTarget` de los eventos de DOM nivel 2. Este objeto no tiene métodos, pero la siguiente propiedad:

- `target` – de tipo `SVGAnimatedString`.

Estados de los editores SVG más utilizados a nivel mundial.

Después de haber dado un concepto extendido acerca de SVG, se está en condiciones de hacer un estudio de cada uno de los distintos tipos de editores existentes en la actualidad.

Entre los editores SVG más usados se encuentran el Amaya, que es un software libre, disponible para sistemas de tipo Unix, Mac os X, Windows y otras plataformas. Es una herramienta combinada del W3C compuesta por un navegador Web, cualquier página que se abra puede ser editada inmediatamente. Se pueden ver y generar páginas HTML y XHTML con hojas de estilo CSS, expresiones MathML y dibujos SVG. Una de sus características consiste en que se pueden ver los enlaces que se crean con el editor. Visualiza imágenes, como las que tienen formato PNG y un subconjunto de Gráficos Vectoriales Escalables (SVG), como las figuras básicas, texto, imágenes. Los gráficos están escritos en XML y pueden ser mezclados libremente con HTML y MathML.

El **Sketsa SVG Editor**, está escrito en Java y es multiplataforma, es decir que se puede usar tanto en Linux como en Mac OS X, Windows y demás sistemas Unix. Sketsa viene acompañado con herramientas de autoría vectorial, entre otras las paletas de dibujo, el editor DOM (Document Object Model) o herramientas específicas SVG, opciones de transformación y de ilustración, así como algún que otro utensilio de optimización. Modificar las propiedades de los objetos es muy fácil mediante su editor centralizado, del mismo modo que las fuentes y recursos, donde se puede localizar toda la información acerca de los documentos SVG y editar de forma visual el código fuente SVG mediante un documento XML. Por último, Sketsa soporta plug-ins, disponibles desde la página Web del autor, por lo que podrás implementar nuevas funciones y herramientas a Sketsa según éstas vayan apareciendo.

El **Jasc Webdraw** es una utilidad para diseñar imágenes y animaciones en formato SVG (Scalable Vector Graphics) útiles para páginas Web, pues que cualquier navegador que pueda leer el formato XML será capaz de mostrarlas. Dispone de herramientas para crear diseños desde cero, importar imágenes, aplicar ligeros efectos, añadir textos, aplicar degradados, generar las animaciones, puedes acceder al código SVG, logrando hacer cambios directamente sobre éste. También posibilita importar y editar imágenes SVG creadas con otros programas.

Entre los editores más usados se tiene el **Sodipodi**, que es un editor de gráficos vectoriales libre distribuido bajo la GNU GPL. Está disponible para GNU/Linux y Microsoft Windows, emplea un estándar SVG como formato nativo de almacenamiento, Es capaz de importar y exportar archivos SVG estándar, así como gráficos de mapa de bits en formato PNG. El mismo está concebido para proveer

un editor útil para gráficos vectoriales y una herramienta de dibujo para artistas, si bien no una implementación completa del estándar SVG. Después del **Sodipodi** surge el **Inkscape** como herramienta de dibujo libre y multiplataforma para gráficos vectoriales SVG. Este software surgió como resultado de una bifurcación del proyecto **Sodipodi**. Las características de SVG soportadas incluyen formas básicas, caminos, texto, canal alfa, transformaciones, gradientes, edición de nodos, exportación de SVG a PNG, agrupación de elementos. La principal motivación de Inkscape es proporcionar a la comunidad una herramienta de código abierto de dibujo SVG que cumpla completamente con los estándares XML, SVG y CSS2. Inkscape ha sido desarrollado principalmente para el sistema operativo GNU/Linux, pero es una herramienta multiplataforma que funciona en Windows, Mac OS X, y con otros sistemas derivados de Unix. Soporta meta-datos, edición de nodos, capas, operaciones complejas con trazos, vectorización de archivos gráficos, texto en trazos, alineación de textos, edición de XML directo.

1.4 Conclusiones

Se concluye que a partir de las características de las metodologías establecidas anteriormente, se decidió utilizar como metodología RUP, para el control y planificación de este trabajo; atendiendo a sus características y a las facilidades que aporta.

Como herramienta de modelado se seleccionó Visual Paradigm pues es una herramienta multiplataforma, además la institución posee la licencia, al igual que el mismo brinda muchas facilidades a la hora de modelar y se puede integrar fácilmente al NetBeans.

Como herramienta para la creación de los gráficos de los documentos de identificación se seleccionó al Inkscape, ya que es una herramienta libre multiplataforma. Después de una breve descripción de todas las herramientas y metodologías usadas se puede afirmar que el SVG es una tecnología muy eficiente para poder hacer la visualización de los datos geográficos gracias a que permiten manejar gráficos de manera dinámica a través de los scripts y de DOM. Esto le permite dar al usuario una visualización en un contexto abierto. Además de ofrecer ventaja de que está basado en XML, proporciona una gran flexibilidad en el manejo de gráficos vectoriales.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se describe la propuesta de solución de este trabajo, para ello se identifican los procesos del negocio que tienen que ver con el objeto de estudio. De acuerdo con esto, se arriba a la conclusión que debido a la poca estructuración de esos procesos, se requieren definir conceptos que se puedan agrupar en un Modelo de Dominio, que permita especificar correctamente los requisitos y de esa manera poder construir un sistema correcto.

También se determinan los requisitos funcionales y no funcionales que debe tener el sistema, se identifican mediante Diagramas de Casos de Uso, las relaciones de los actores que interactúan con el sistema y las secuencias de acciones con las que interactúan.

Descripción de los procesos del negocio propuestos

Uno de los principales objetivos de esta investigación es realizar el desarrollo de una herramienta multiplataforma libre que permita crear, salvar y editar plantillas de documentos de identificación y la personalización e impresión de los mismos (tarjetas de identificación, cédulas, pasaportes, carné de identidad entre otros).

La solución se fundamenta en un diseño manuable que facilite su integración a cualquier sistema de identificación, además especifica el contenido a dibujar y forma de hacer el enlace de datos mediante plantillas XML para hacer independiente la aplicación de nuevos cambios de diseño o dispositivos de impresión.

El sistema estará compuesto por dos aplicaciones: El diseñador de plantillas (plantillas compiladas y plantillas no compiladas), que mostrará al usuario la forma en que van a estar organizados los distintos elementos que conformarán un documento de identificación, por ejemplo la plantilla no compilada tendrá un formato en el que especifica el modelo de datos, el enlace de datos y el diseño gráfico que tendrá el documento de identificación, lo cual constituye una información de la que dispondrá el usuario de los elementos que conformaran la plantilla del documento.

Además, existe otro módulo donde se encuentran los componentes de impresión, que permitirán personalizar el documento validando los datos contra un esquema de datos y al hacer las transformaciones correspondientes se personalizará el documento, dejándolo listo para la impresión del mismo.

En la actualidad este proceso no se lleva a cabo. Para alcanzar lo antes expuesto, se propone desarrollar un software, donde un usuario llamado (Diseñador de Plantillas) sea el encargado de crearlas, editarlas y salvarlas y otro (Administrador de impresión de documentos), sería el responsable y capacitado para manejar todos los procesos y eventos que se operarán en la personalización y la impresión de los distintos tipos de documentos de identificación. Tanto el diseñador de plantillas como el administrador de impresión son los máximos responsables de todos los eventos que se desarrollen a la hora de conformar un documento de identificación.

A continuación se muestra una imagen para un mayor entendimiento de los procesos existentes

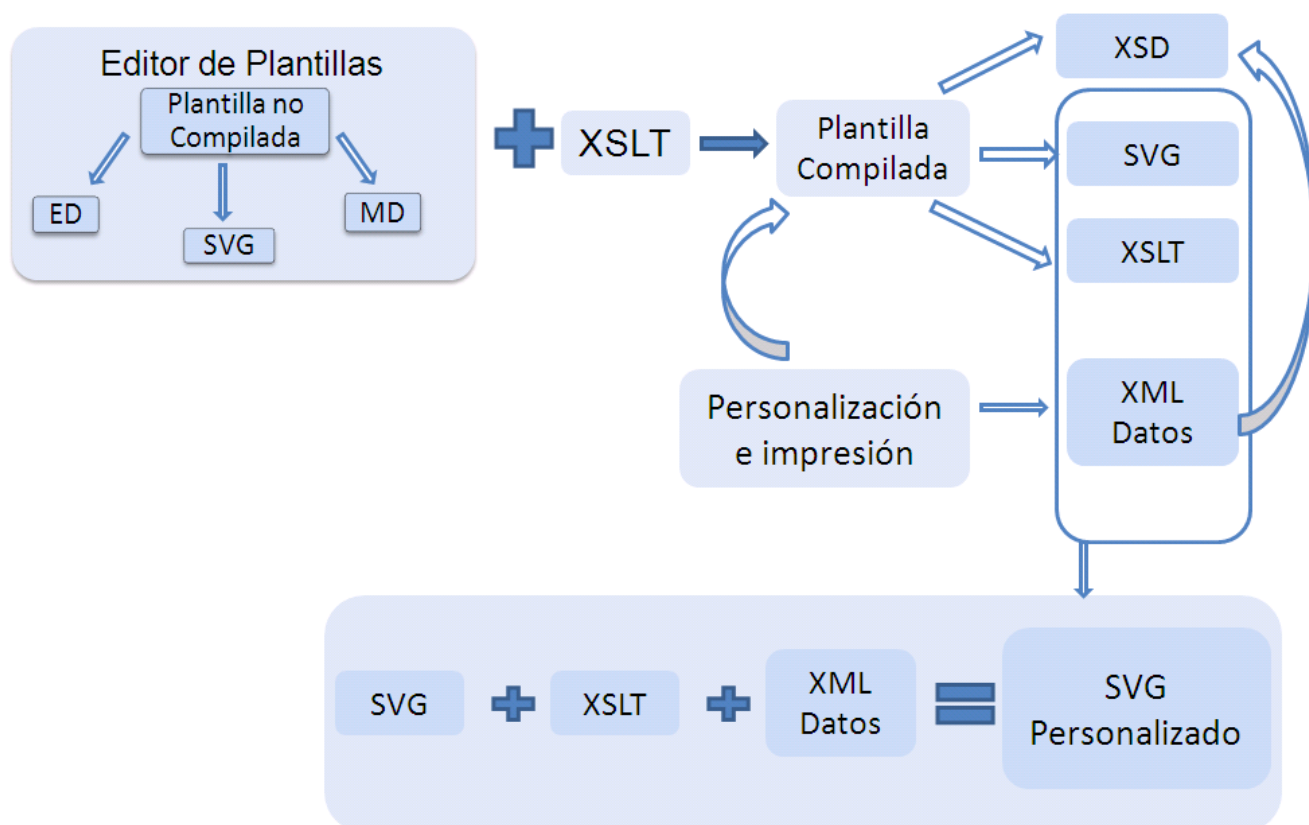


Figura 4 Propuesta de solución

2.2 Modelo del Dominio

A partir de una breve descripción de los procesos del negocio expuestos, se puede observar que el mismo tiene un bajo nivel de estructuración. Por lo que esta investigación se basará en un modelo de

Capítulo 2: Características del sistema

dominio, que permitirá mostrar al usuario, los principales conceptos que se operan en el dominio de la aplicación en desarrollo. De esta forma los usuarios utilizarán un vocabulario común, para lograr entender el contenido en que se enmarca la aplicación. Este modelo va a contribuir más adelante a describir las clases más importantes dentro del contexto la aplicación.

A continuación se identifican los conceptos más significativos que se utilizarán en el modelo de dominio:

Administrador de impresión de documentos de identificación individuo que se encarga de hacer un conjunto de operaciones referentes a la personalización e impresión de los documentos de identificación.

Diseñador de documentos de identificación, es el encargado de diseñar una **Plantilla XML** (**PlantillaNoCompilada**) y de generar una **PlantillaCompilada**, la cual tendrá el diseño gráfico del documento de identificación, el modelo de datos referente a la personalización y el enlace de datos entre los elementos visuales del SVG y los elemento de los campos del modelo de datos.

XMLDatosPersona, es el documento XML que contendrá los datos para la personalización de los documentos de identificación.

PlantillaCompilada, contiene el esquema para validar el **XMLDatosPersona** provisto por el usuario, las transformaciones llevadas a cabo para personalizar el diseño SVG y el diseño del documento de identificación.

DiseñoSVG al fichero-imagen en formato XML diseñado por el Inkscape.

DiseñoSVGPersonalizado al fichero-imagen en formato XML diseñado por el Inkscape, con los datos de la persona.

Esquemadatos, fichero **XSD** empotrado en la plantilla compilada que permite validar el **XMLDatosPersona** de los datos de personalización.

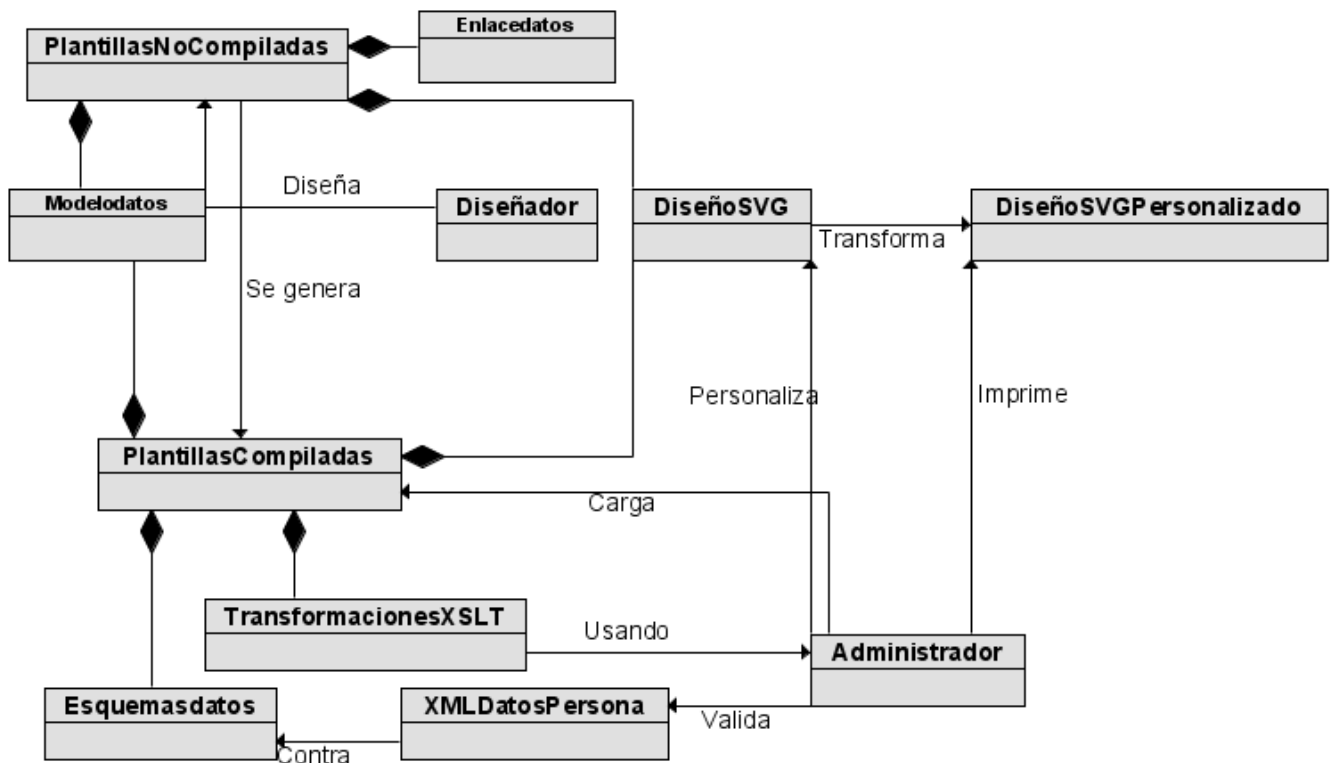


Figura 5 Modelo de Dominio

2.3 Descripción del Sistema Propuesto

Para dar cumplimiento a los objetivos trazados y asumir todos los requerimientos planteados al inicio de esta investigación, la realización del análisis, diseño e implementación de componentes y utilitarios para la personalización y la impresión de documentos de identificación, deben permitir la integración a cualquier sistema de esta índole, validar los datos de la persona contra esquemas de validación, personalizar e imprimir documentos de identificación. El logro de ese producto garantizará un mejor rendimiento en los procesos de personalización e impresión de documentos de identificación.

Requerimientos funcionales

Diseñador de plantillas de documentos de identificación

- 1 Especificar o cargar los elementos constitutivos de las plantillas de documentos de identificación.
 - 1.1 Cargar el diseño gráfico del documento de identificación.

- 1.1.1 Cargar el fichero SVG generado por diseñador externo. (Ej.: Inkscape Vector Illustrator)
- 1.1.2 Validar el contenido SVG contra el esquema W3C correspondiente, para verificar su correcta conformación.
- 1.2 Especificar modelo de datos de personalización.
 - 1.2.1 Especificar el esquema XSD con la descripción de los datos a utilizar en la personalización.
 - 1.2.1.1 Especificar visualmente los campos y subcampos así como los tipos de datos y las restricciones correspondientes.
 - 1.2.1.2 Especificar los valores demostrativos para cada campo.
 - 1.2.2 Permitir cargar alternativamente el esquema XSD desde un fichero.
- 1.3 Especificar los valores demostrativos.
 - 1.3.1 Conformar el XML de valores demostrativos.
 - 1.3.1.1 Si el esquema XSD ha sido editado (variante 2.1) conformar el XML a partir de los valores demostrativos especificados por el usuario para cada campo.
 - 1.3.1.2 Si el esquema XSD ha sido cargado desde un fichero XSD (variante 2.2), cargar el XML desde un fichero.
 - 1.3.1.3 Verificar el XML de datos demostrativos contra el XSD del modelo de datos.
- 1.4 Establecer asociaciones de enlace de datos.
 - 1.4.1 Establecer los tipos de elementos SVG que son enlazables y especificar cuales de sus atributos tienen esta cualidad.
 - 1.4.2 Mostrar visualmente un listado de los identificadores de los elementos SVG cuyos tipos sean definidos como enlazables.
 - 1.4.3 Mostrar visualmente un listado de los campos del modelo de datos XSD.
 - 1.4.4 Permitir seleccionar un valor de cada uno de los dos listados referidos en 1.4.2 y 1.4.3 y crear una asociación entre estos.

- 1.4.4.1 Permitir especificar parámetros en el momento de la asociación, como el atributo de ese elemento SVG que va a ser asociado al elemento XSD del modelo de datos.
 - 1.4.4.2 Mostrar visualmente un listado de todas las asociaciones realizadas.
- 1.5 Generar transformaciones correspondientes a la plantilla del documento de identificación.
 - 1.5.1 Generar una transformación XLST preparada para convertir un XML con los datos de personalización en un SVG que represente el diseño gráfico del documento de identificación completamente personalizado.
 - 1.5.1.1 Lograr que la transformación XLST parta del diseño SVG y sustituya los elementos y/o atributos enlazables por los valores especificados en el XML con los datos de personalización.
 - 1.5.1.2 Utilizar las asociaciones de enlaces de datos para determinar cuales de los valores del XML de los datos de personalización debe ser utilizado.
- 2 Crear las plantillas de documentos de identificación.
 - 2.1 Crear el contenido de las plantillas como un documento XML que tenga los siguientes componentes:
 - 2.1.1 Modelo de datos de personalización (Esquemas XSD).
 - 2.1.2 Diseño gráfico del documento de identificación. (SVG)
 - 2.1.3 Asociaciones de enlace de datos. (XML)
 - 2.1.4 Valores demostrativos. (XML)
 - 2.1.5 Transformaciones de dibujo.(XSLT)
- 3 Generar vista previa de la plantillas del documento de identificación.
 - 3.1 Invocar a la capa de impresión especificándole la plantilla, el modo de impresión de vista previa y el contexto gráfico donde se desea dibujar la pre visualización
- 4 Salvar las plantillas de documentos de personalización en ficheros XML.
- 5 Cargar plantillas de documentos de personalización.

Capa de impresión de documentos

- 6 Cargar la plantilla de impresión del documento de identificación.
 - 6.1 Cargar el documento XML que representa la plantilla.
 - 6.2 Comprobar la validez de la plantilla del documento de identificación.
- 7 Cargar los datos de personalización a imprimir.
 - 7.1 Obtener documento XML con los datos de personalización.
 - 7.2 Validar los datos de personalización contra el modelo de datos de personalización.
- 8 Verificar estado listo de la impresora.
- 9 Personalizar documento SVG.
 - 9.1 Obtener la transformación XLST de la plantilla del documento de identificación.
 - 9.2 Aplicar la transformación XLST a los datos de personalización, obteniendo así un SVG que es la representación gráfica del documento de identificación.
 - 9.3 Invocar al motor de dibujo especificándole el SVG y el contexto gráfico de dibujo.
 - 9.3.1 Si el modo de impresión es de pre visualización se debe especificar en que contexto gráfico se realizará el dibujo.
 - 9.3.2 Si el modo de impresión NO es de pre visualización se debe utilizar el contexto gráfico de la impresora.

Luego de una breve descripción de los requisitos funcionales se pasa a enumerar los requisitos no funcionales.

Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Apariencia o interfaz externa

- La interfaz de esta aplicación debe ser sencilla y de fácil uso, con rápida respuesta la aplicación.
- Debe tener diferentes opciones para mostrar.
- Debe tener colores refrescantes a la vista.

Capítulo 2: Características del sistema

Usabilidad

- La aplicación debe ser de fácil manejo para los usuarios que tengan niveles básicos sobre computación o hayan realizado algún trabajo sobre diseño de documentos.

Rendimiento

- La aplicación está concebida para un ambiente donde la misma se integre a un sistema de identificación, por lo que los tiempos de respuestas deben ser rápidos, así como la velocidad de procesamiento de información.
- La aplicación requiere de un buen rendimiento en máquinas de pocos recursos de Hardware.

Portabilidad

- Necesidad de que la aplicación sea multiplataforma.

Confiabilidad

- La información manejada por la aplicación está protegida de acceso no autorizado y de divulgación.

2.4 Modelo de Casos de Uso del Sistema

Una vez capturados los requisitos funcionales del sistema, estos se representarán mediante dos Diagrama de Casos de Uso. Para ello primero se debe definir claramente cuáles son los actores que van a interactuar con el sistema, y los Casos de Uso que representarán todas las funcionalidades del sistema.

Un Caso de Uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica.

Un actor representa un conjunto coherente de roles que juegan los usuarios de los Casos de Uso cuando interactúan con éstos. Los actores pueden ser personas (roles que desempeñan las personas), aparatos eléctricos o mecánicos, y otros sistemas de cómputo. En este caso con el sistema interactúan dos actores (Diseñador de plantillas de documentos de identificación y Administrador de impresión de documentos).

Capítulo 2: Características del sistema

Tabla 2.1 Actores del sistema

Actor del Sistema	Justificación
Administrador de impresión de documentos de identificación	Representa una persona que es la encargada de hacer todos los procesos que se llevan a cabo para realizar la creación de un documento de identificación, la personalización y por último la impresión.
Diseñador de plantillas de documentos de identificación	Representa una persona que es la encargada diseñar las plantillas de los documentos de identificación.

Tabla 2.2 Resumen del Caso de Uso 1

CU-1	Crear plantillas de documentos de identificación.
Actor	Diseñador de plantillas de documentos de identificación.
Descripción	El diseñador de plantillas de documentos de identificación solita crear una nueva plantilla de documento de identificación.
Referencia	R2.

Tabla 2.3 Resumen del Caso de Uso 2

CU-2	Cargar plantillas de documentos de identificación.
Actor	Diseñador de plantillas de documentos de identificación.
Descripción	El diseñador de plantillas de documentos de identificación solita cargar una plantilla de documento de identificación.
Referencia	R1.

Tabla 2.4 Resumen del Caso de Uso 3

CU-4	Crear modelo de datos.
Actor	Diseñador de plantillas de documentos de identificación.
Descripción	El diseñador de plantillas de plantillas de documentos de identificación solicita crear el modelo de datos que va a tener la plantilla de los documentos de identificación.
Referencia	R1.2.

Tabla 2.5 Resumen del Caso de Uso 4

CU-5	Crear enlace de datos.
Actor	Diseñador de plantillas de documentos de identificación.

Capítulo 2: Características del sistema

Descripción	El diseñador de plantillas de plantillas de documentos de identificación solicita crear el enlace de datos donde se enlazaran los elementos visuales del documento SVG y los elementos de los campos del modelo de datos.
Referencia	R1.4.

Tabla 2.6 Resumen del Caso de Uso 5

CU-6	Cargar diseño SVG.
Actor	Diseñador de plantillas de documentos de identificación.
Descripción	El diseñador de plantillas de plantillas de documentos de identificación solicita cargar el diseño SVG que va a tener la plantilla de los documentos de identificación.
Referencia	R1.1.

Tabla 2.7 Resumen del Caso de Uso 6

CU-7	Cargar modelo de datos.
Actor	Diseñador de plantillas de documentos de identificación.
Descripción	El diseñador de plantillas de plantillas de documentos de identificación solicita cargar el modelo de datos que va a tener la plantilla de los documentos de identificación.
Referencia	R1.2.

Tabla 2.8 Resumen del Caso de Uso 7

CU-8	Cargar enlace de datos.
Actor	Diseñador de plantillas de documentos de identificación.
Descripción	El diseñador de plantillas de plantillas de documentos de identificación solicita cargar el enlace de datos que va a tener la plantilla de los documentos de identificación.
Referencia	R1.4.

Tabla 2.9 Resumen del Caso de Uso 8

CU-8	Crear plantilla compilada.
Actor	Diseñador de plantillas de documentos de identificación.
Descripción	El diseñador de plantillas de plantillas de documentos de identificación solicita cargar el enlace de datos que va a tener la plantilla de los documentos de identificación.
Referencia	R3.

Expansión de los casos de uso.

Tabla 2.10 Descripción del caso de uso Crear Modelo de Datos

Nombre del caso de uso: Crear modelo de datos.	
Actores:	Diseñador de plantillas de documentos de identificación.
Propósito:	Crear el modelo de datos de la plantilla de los documentos de identificación.
Resumen:	El caso de uso inicia cuando diseñador de plantillas de documentos de identificación solicita crear el modelo de datos de la plantilla.
Referencias:	R1.2.
Precondiciones:	Se ha iniciado el proceso de crear plantillas.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El diseñador solicita la creación del modelo de datos de la plantilla.	2. El sistema le muestra una interfaz donde aparecen los campos que se van a especificar (nombre y tipo de elemento), al igual que hay un botón donde se aceptan los datos anteriormente especificados y otro botón para eliminar dichos datos.
3. El diseñador introduce los elementos del modelo de datos	4. El sistema le muestra un listado de los datos y sus tipos.
Flujo alterno	
	5. En caso de que se especifiquen mal los datos el sistema muestra un mensaje.
Poscondiciones:	Se creo el modelo de datos.

Tabla 2.11 Caso de uso: Crear enlace de datos

Nombre del caso de uso: Crear enlace de datos.	
Actores:	Diseñador de plantillas de documentos de identificación.
Propósito:	Crear el enlace de datos de la plantilla de los documentos de identificación.
Resumen:	El caso de uso inicia cuando diseñador de plantillas de documentos de identificación solicita crear el enlace de datos de la plantilla.
Referencias:	R1.4.
Precondiciones:	Se ha iniciado el proceso de crear plantillas.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El diseñador solicita la creación del enlace	2. El sistema le muestra una interfaz donde aparecen los

Capítulo 2: Características del sistema

de datos de la plantilla.	campos que se van a especificar (campos fuente, cambio contenido), al igual que hay un botón donde se aceptan los datos anteriormente especificados y otro botón para eliminar dichos datos.
3. El diseñador introduce los elementos del enlace de datos	4. El sistema le muestra un listado todos los elementos y los enlaces.
Flujo alterno	
	5. En caso de que se especifiquen mal los datos el sistema muestra un mensaje.
Poscondiciones:	Se creo el enlace de datos.

Tabla 2.12 Caso de uso: Cargar diseño SVG

Nombre del caso de uso: Cargar diseño SVG.	
Actores:	Diseñador de plantillas de documentos de identificación.
Propósito:	Cargar el diseño que tendrá la plantilla de los documentos de identificación.
Resumen:	El caso de uso inicia cuando diseñador de plantillas de documentos de identificación solicita cargar el diseño SVG que tendrá dicha plantilla (lo mismo puede cargar el diseño desde un fichero que desde una plantilla).
Referencias:	R1.1.
Precondiciones:	Se ha iniciado el proceso de crear plantillas.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El diseñador solicita cargar el diseño SVG de los documentos de identificación.	2. El sistema le muestra una interfaz donde hay un menú de opciones, encontrándose en el mismo la opción de cargar diseño.
	3. El sistema le muestra una interfaz donde aparece el diseño mostrado en pantalla, lo mismo cargado desde una plantilla como de un fichero SVG.
Flujo alterno	
	4. En caso de que el fichero este mal conformado el sistema mostrara un mensaje de error.
Poscondiciones:	Se cargo el diseño SVG de la plantilla.

Capítulo 2: Características del sistema

Tabla 2.13 Caso de uso: Cargar modelo de datos

Nombre del caso de uso: Cargar modelo de datos.	
Actores:	Diseñador de plantillas de documentos de identificación.
Propósito:	Cargar el modelo de datos que tendrá la plantilla de los documentos de identificación.
Resumen:	El caso de uso inicia cuando diseñador de plantillas de documentos de identificación solicita cargar el modelo de datos que contiene dicha plantilla.
Referencias:	R1.2.
Precondiciones:	El sistema debe estar en condiciones óptimas para comenzar todo el proceso de carga de la plantilla.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El diseñador solicita cargar modelo de datos.	2. El sistema le muestra una interfaz donde hay un menú de opciones, encontrándose en el mismo la opción de cargar modelo de datos.
3. El diseñador carga la plantilla.	4. El sistema le muestra la interfaz donde se muestran los datos correspondientes al modelo de datos que esta en la plantilla, al igual el sistema permite adicionar más campos o eliminar más campos, es decir editar ese modelo de datos.
Flujo alterno	
	5. En caso de que el fichero plantilla este mal conformado el sistema mostrara un mensaje de error.
Poscondiciones:	Se cargo desde la plantilla el modelo de datos y se edito nuevamente.

Tabla 2.14 Caso de uso: Cargar enlace de datos

Nombre del caso de uso: Cargar enlace de datos.	
Actores:	Diseñador de plantillas de documentos de identificación.
Propósito:	Cargar el enlace de datos de la plantilla.
Resumen:	El caso de uso inicia cuando diseñador de plantillas de documentos de identificación solicita cargar el enlace de datos que contiene dicha plantilla.
Referencias:	R1.4.
Precondiciones:	El sistema debe estar en condiciones óptimas para comenzar todo el proceso de carga de la plantilla.
Flujo normal de eventos	

Capítulo 2: Características del sistema

Acción del actor		Respuesta del sistema
1. El diseñador solicita cargar enlace de datos.		2. El sistema le muestra una interfaz donde hay un menú de opciones, encontrándose en el mismo la opción de cargar enlace de datos.
3. El diseñador carga la plantilla.		4. El sistema le muestra la interfaz correspondiente donde se muestran los datos del modelo de datos que esta en la plantilla, al igual el sistema permite editar ese enlace de datos.
Flujo alternativo		
		5. En caso de que el fichero plantilla este mal conformado el sistema mostrara un mensaje de error.
Poscondiciones:	Se cargo desde la plantilla el enlace de datos y se edito nuevamente.	

Tabla 2.15 Caso de uso: Crear plantilla de documentos de identificación

Nombre del caso de uso: Crear plantilla de documentos de identificación.	
Actores:	Diseñador de plantillas de documentos de identificación.
Propósito:	Crear plantilla de documentos de identificación.
Resumen:	El caso de uso inicia cuando el diseñador de plantillas de documentos de identificación solicita crear una nueva plantilla.
Referencias:	R2.
Precondiciones:	Que estén creados el modelo de datos, el enlace de datos y cargado el diseño SVG.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El diseñador de plantillas solicita la creación de una nueva plantilla.	2. El sistema muestra la interfaz principal con un menú de opciones donde aparece la opción nueva plantilla.
	3. El sistema permite crear modelo d datos, crear enlace de datos y cargar diseño SVG.

Tabla 2.16 Caso de uso: Cargar plantilla de documentos de identificación

Nombre del caso de uso: Cargar plantilla de documentos de identificación.		
Actores:	Diseñador de plantillas de documentos de identificación.	
Propósito:	Cargar plantilla de documentos de identificación.	

Capítulo 2: Características del sistema

Resumen:	El caso de uso inicia cuando el diseñador de plantillas de documentos de identificación solicita crear una nueva plantilla.
Referencias:	R1.
Precondiciones:	Se deben ver creado el modelo de datos el enlace de datos y ver cargado el diseño SVG.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El diseñador de plantillas solicita cargar plantilla.	2. El sistema muestra la interfaz principal con un menú de opciones donde aparece la opción cargar plantilla.
	3. El sistema permite cargar modelo de datos, cargar enlace de datos y cargar diseño SVG.

Tabla 2.17 Caso de uso: Crear plantilla compilada

Nombre del caso de uso: Crear plantilla compilada.	
Actores:	Diseñador de plantilla de documentos de identificación.
Propósito:	Crear la plantilla compilada de los documentos de identificación.
Resumen:	El caso de uso inicia cuando el diseñador de plantillas de documentos de identificación solicita crear una plantilla compilada.
Referencias:	R3.
Precondiciones:	Se tiene que ver comenzado el proceso de creación de una plantilla.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El diseñador de plantillas solicita crear una plantilla compilada.	2. El sistema muestra la opción crear plantilla compilada, el sistema salva la plantilla compilada en disco.

Casos de uso de la parte de personalización.

Tabla 2.18 Resumen Caso de Uso 1

CU-1	Personalizar documento SVG.
Actor	Administrador de impresión de documentos de identificación.
Descripción	El administrador de impresión solicita la personalización del documento de identificación.
Referencia	R9.

Capítulo 2: Características del sistema

Tabla 2.19 Resumen del Caso de Uso 2

CU-2	Imprimir documento de identificación.
Actor	Administrador de impresión de documentos de identificación.
Descripción	El administrador de impresión solicita la impresión del documento de identificación.
Referencia	R8.

Tabla 2.20 Resumen del Caso de Uso 3

CU-3	Cargar plantilla compilada.
Actor	Administrador de impresión de documentos de identificación.
Descripción	El administrador de impresión solicita cargar la plantilla compilada.
Referencia	R6.

Tabla 2.21 Resumen del Caso de Uso 4

CU-4	Cargar datos personalización.
Actor	Administrador de impresión de documentos de identificación.
Descripción	El administrador de impresión solicita cargar los datos de la personalización de los documentos de identificación.
Referencia	R7.

Tabla 2.22 Caso de uso: Cargar plantilla compilada de documentos de identificación

Nombre del caso de uso: Cargar plantilla compilada de documentos de identificación.	
Actores:	Administrador de impresión de documentos de identificación.
Propósito:	Cargar plantilla compilada de documentos de identificación.
Resumen:	El caso de uso inicia cuando el administrador de impresión de documentos de identificación solicita cargar la plantilla compilada de los documentos de identificación.
Referencias:	R6.
Precondiciones:	El sistema debe estar listo para cargar la plantilla compilada.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El administrador solicita cargar plantilla compilada.	2. El sistema muestra la interfaz principal con un menú de opciones donde aparece la opción cargar plantilla compilada.

Capítulo 2: Características del sistema

3. El administrador carga dicha plantilla.	4. El sistema dibuja el contenido grafico del documento de identificación en la parte izquierda de la pantalla.
Flujo alternativo	
	5. En caso de que la plantilla esté mal configurada el sistema muestra un mensaje de error.

Tabla 2.22 Caso de uso: Cargar datos personalización

Nombre del caso de uso: Cargar datos personalización.	
Actores:	Administrador de impresión de documentos de identificación.
Propósito:	Cargar desde fichero XML los datos de la personalización.
Resumen:	El caso de uso inicia cuando el administrador de impresión solicita cargarlos datos necesarios para la personalización.
Referencias:	R7.
Precondiciones:	El sistema debe estar en condiciones para empezar todo el proceso de personalización.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El administrador solicita cargar los datos de la personalización.	2. El sistema le muestra una interfaz donde hay un menú de opciones, encontrándose en el mismo la opción de Cargar datos.
3. El administrador carga los datos.	4. El sistema le muestra un mensaje que los datos fueron validados y cargados correctamente, mostrándose en la parte inferior del formulario.
Flujo alternativo	
	5. En caso de que el fichero XML este mal conformado el sistema mostrara un mensaje de error.
Poscondiciones:	Se cargaron los datos de la personalización desde XML

Tabla 2.27 Caso de uso: Personalizar documento SVG

Nombre del caso de uso: Personalizar documento SVG.	
Actores:	Administrador de impresión de documentos de identificación.
Propósito:	Personalizar documento SVG.
Resumen:	El caso de uso inicia cuando el administrador de impresión de documentos de identificación solicita personalizar documento de identificación.

Capítulo 2: Características del sistema

Referencias:	R9.
Precondiciones:	El sistema debe tener la plantilla compilada cargada.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El administrador solicita personalizar el documento SVG.	2. El sistema muestra la interfaz principal con un menú de opciones donde aparece la opción personalizar.
3. El administrador personaliza el documento SVG.	4. El sistema dibuja el contenido grafico del documento de identificación en la parte derecha de la pantalla.
Flujo alterno	
	5. En caso de que exista algún problema con la personalización, el sistema muestra un mensaje de error.

Tabla 2.28 Caso de uso: Imprimir documento

Nombre del caso de uso: Imprimir documento.	
Actores:	Administrador de impresión de documentos de identificación.
Propósito:	Personalizar el documento SVG.
Resumen:	El caso de uso inicia cuando el administrador de impresión de documentos de identificación solicita personalizar el documento SVG.
Referencias:	R8.
Precondiciones:	El sistema debe tener la plantilla compilada cargada.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. El administrador solicita imprimir el documento de identificación.	2. El sistema le muestra una ventana para realizar la impresión del documento de identificación.
3. El administrador imprime el documento.	4. El sistema muestra una vista previa del documento impreso, saliendo dicho documento por la bandeja de salida de la impresora.
Flujo alterno	
	5. En caso de que exista algún problema con el proceso de impresión el sistema muestra un mensaje de error.

Diagrama de casos de uso de la parte del diseñador de plantillas:

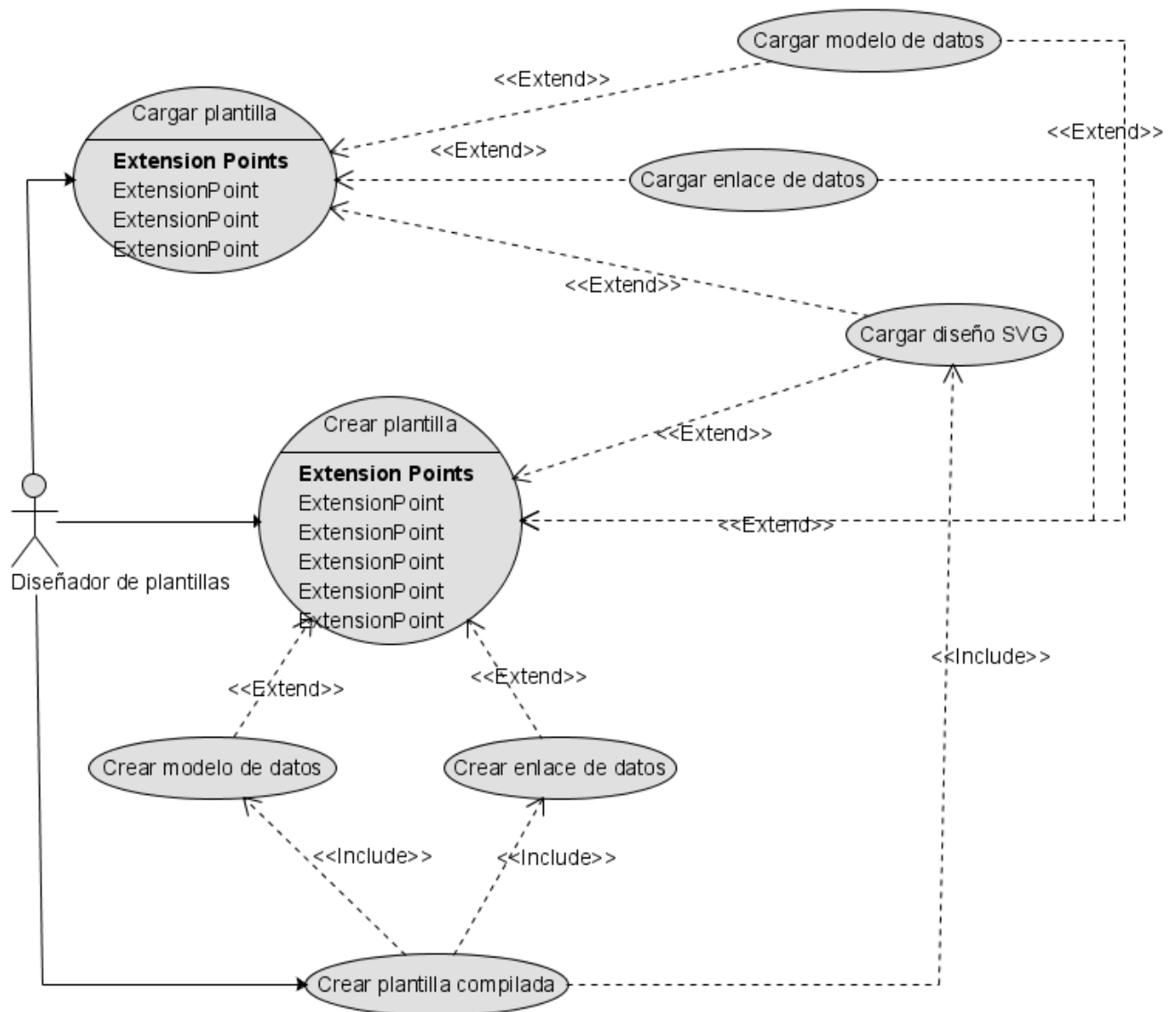


Figura 6 Diagrama de caso de uso del editor de plantillas.

Consecuentemente se representa el diagrama de caso de uso de la personalización.

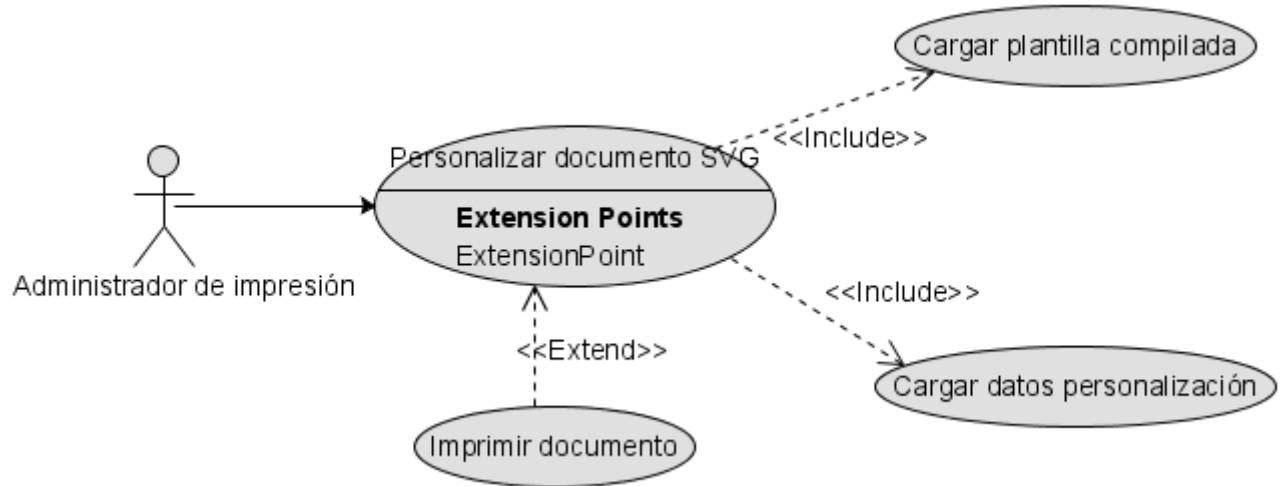


Figura 7 Diagrama de caso de uso de la personalización.

2.5 Conclusiones

En este capítulo se comenzó a realizar la propuesta de solución de la futura aplicación, se modeló el negocio mediante un modelo de dominio, se expusieron los requisitos funcionales y no funcionales y se escribieron los casos de usos.

CAPÍTULO 3: CONSTRUCCIÓN Y LEABORACIÓN DEL SISTEMA

3.1 Introducción

En este capítulo se realiza el diseño de la propuesta de solución, se definen los principios de diseño gráfico, los estándares de interfaz de la aplicación, así como una concepción general de la ayuda y el tratamiento de excepciones. Se expondrán además el diagrama de despliegue y el de implementación. Ellos son sumamente importantes dentro del flujo de trabajo de implementación, pues ambos conforman lo que se conoce como modelo de implementación. Además se describen y explican los conceptos relacionados con dichos diagramas.

3.2 Modelo de Análisis

En la construcción del modelo de análisis se identifican las clases que describen la realización de los Casos de Uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto comúnmente en el diseño e implementación de la solución. El Modelo de Análisis es el resultado de la actividad de analizar los Casos de Uso y su realización suaviza la transición al Diseño y se utiliza para tener una visión general de la propuesta de sistema [11].

Diagrama de Clases de Análisis

Las clases de análisis se centran en los requisitos funcionales, y son evidentes en el dominio del problema, porque representan conceptos y relaciones del dominio. Tienen atributos y se establecen relaciones entre ellas. Encajan en tres estereotipos básicos:

Clase Interfaz: Modelan la interacción entre el sistema y sus actores.

Clase Entidad: Modelan información que posee larga vida y que es a menudo persistente.

Clase Control: Coordinan la realización de uno o unos pocos Casos de Uso coordinando las actividades de los objetos que implementan su funcionalidad.

A continuación se muestran los diagramas de clases del análisis para el diseñador de plantillas. Los diagramas de la personalización e impresión se encuentran en los Anexos I, II, III y IV:



Figura 8 Diagrama de clases del análisis del caso de uso: Crear plantilla



Figura 9 Diagramas de clases del análisis del caso de uso: Crear Modelo de datos.

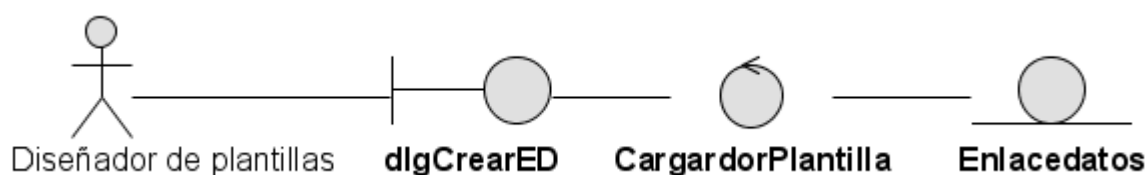


Figura 10 Diagrama de clases del análisis del caso de uso: Crear enlace de datos.

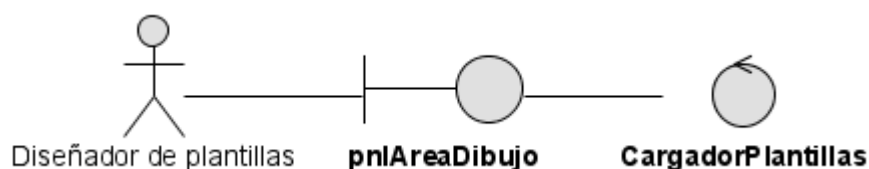


Figura 11 Diagrama de clases del análisis del caso de uso: Cargar diseño SVG.

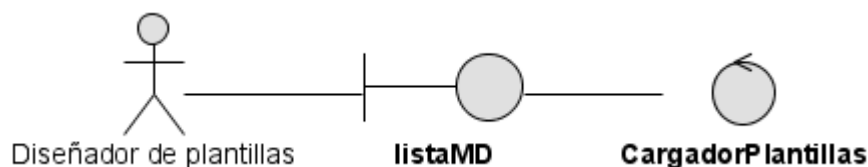


Figura 12 Diagrama de clases del análisis del caso de uso: Cargar modelo de datos.

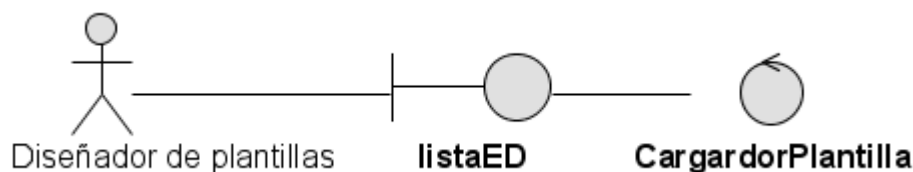


Figura 13 Diagrama de clases del análisis del caso de uso: Cargar enlace de datos.



Figura 14 Diagrama de clases del análisis del caso de uso: Cargar plantilla.

3.3 Patrones de Diseño

El patrón es una descripción de un problema y su solución, con un nombre, que codifica buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Los patrones de diseño contribuyen a reutilizar diseño, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. La importancia de la reutilización del diseño reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad, eficiencia y consistencia de los diseños, y proporciona un considerable ahorro en la inversión.

En el diseño de la aplicación se tuvieron en cuenta los patrones Experto, Creador, Bajo Acoplamiento y **Alta Cohesión**.

El patrón **Experto** plantea que se debe asignar la responsabilidad al experto en información, que es la clase que cuenta con la información necesaria para cumplir la responsabilidad.

El patrón Creador expresa que la responsabilidad de crear una instancia de una determinada clase, debe asignarse a otra clase, siempre que esta agregue, contenga, registre o utilice específicamente los objetos de aquella.

El patrón **Bajo Acoplamiento** impulsa la asignación de responsabilidades, de manera que su localización no incremente el acoplamiento, hasta un nivel que lleve a los resultados negativos que puede producir un acoplamiento alto.

El patrón **Alta Cohesión** asigna una responsabilidad, de modo que la cohesión permanezca alta. Este patrón incrementa la claridad y facilita la comprensión del diseño, simplifica el mantenimiento y las mejoras en funcionalidad, e incrementa las capacidades de reutilización.

3.4 Modelo de Diseño

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales. Este modelo está muy cercano al modelo de implementación, lo que es natural para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software.

Entre los propósitos del diseño se encuentran adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y de las restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario. Además de descomponer los trabajos de implementación en partes más manejables, que puedan ser ejecutadas por diferentes equipos de desarrollo.

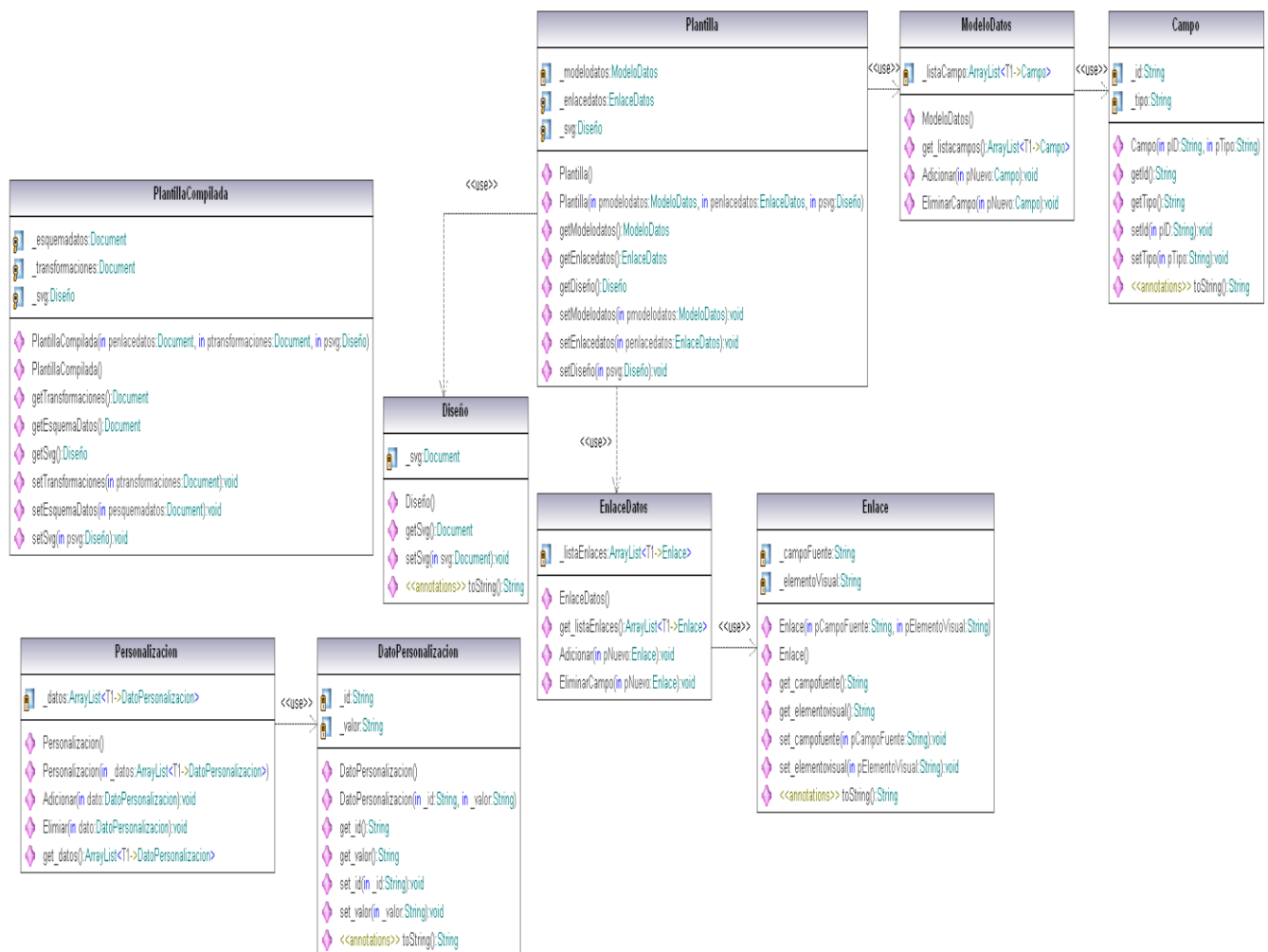
Capítulo 3: Construcción y Elaboración del Sistema

3.5 Diagrama de Clases de Diseño

Las clases de diseño se especifican utilizando la sintaxis del lenguaje de programación elegido y tienen correspondencia directa con los métodos en la implementación. Un diagrama de clases de diseño es una representación más concreta que el diagrama de clases del análisis, representa la parte estática del sistema, las clases y sus relaciones.

A continuación se muestra el diagrama de clases del diseño por paquetes y la descripción de cada una de las clases que conforman los paquetes:

- Paquete número 1: **IDSVG.Comun**
- Paquete número 2: **IDSVG.Dibujo**
- Paquete número3: **IDSVG.Persistencia**
- Paquete número4: **IDSVG.Impresion**



SVGDisposer

◆ `DisposeSVG(in spPanelRedimensionable:JScrollPane, in panel:JPanel, in canvas:JSVGCanvas):void`





SVGPainter

◆ `Paint(in forma:JFrame, in panel:JPanel, in canvas:JSVGCanvas, in diseño:Diseño):void`












SVGLoader










◆ `Load(in url:String, in forma:JFrame, in etiqueta:JLabel, in panel:JPanel, in canvas:JSVGCanvas):void`

Capítulo 3: Construcción y Elaboración del Sistema











CargadorDatosPersonalizacion	
	_personalizacion: Personalizacion
	xmlDoc: Document
	CargadorDatosPersonalizacion(in _personalizacion: Personalizacion)
	CargarXMLDatos(in ruta: String): void













TempFileHandler	
	SaveTempFileXMLType(in documento: Document, in nombreFichero: String): void
	LoadTempFileXMLType(in nombreFichero: String): Document

CargadorPlantilla	
	_plantilla: Plantilla
	_document: Document
	CargadorPlantilla()
	CargadorPlantilla(in _plantilla: Plantilla)
	getPlantilla(): Plantilla
	Cargar(in camino: String): void
	CrearDocumento(in camino: String): Document
	DocumentoXMLaTexto(): String
	CargarModeloDatos(in camino: String, in cargarXSeparado: boolean): ModeloDatos
	CargarEnlaceDatos(in camino: String, in cargarXSeparado: boolean): EnlaceDatos
	CargarDiseño(in camino: String, in cargarXSeparado: boolean): Diseño

GenerarDatosPersonalizacion	
	_personalizacion: Personalizacion
	xmlDoc: Document=null
	datoXML: Element=null
	TAG_PRINCIPAL: String="Datos"
	TAG_DATOSP: String="Personalizacion"
	GenerarDatosPersonalizacion(in _personalizacion: Personalizacion)
	GenerarDocumento(): void
	ObtenerDocumento(): void
	grabaDocumentoXML(in docEncadenaTexto: String, in ruta: String): void

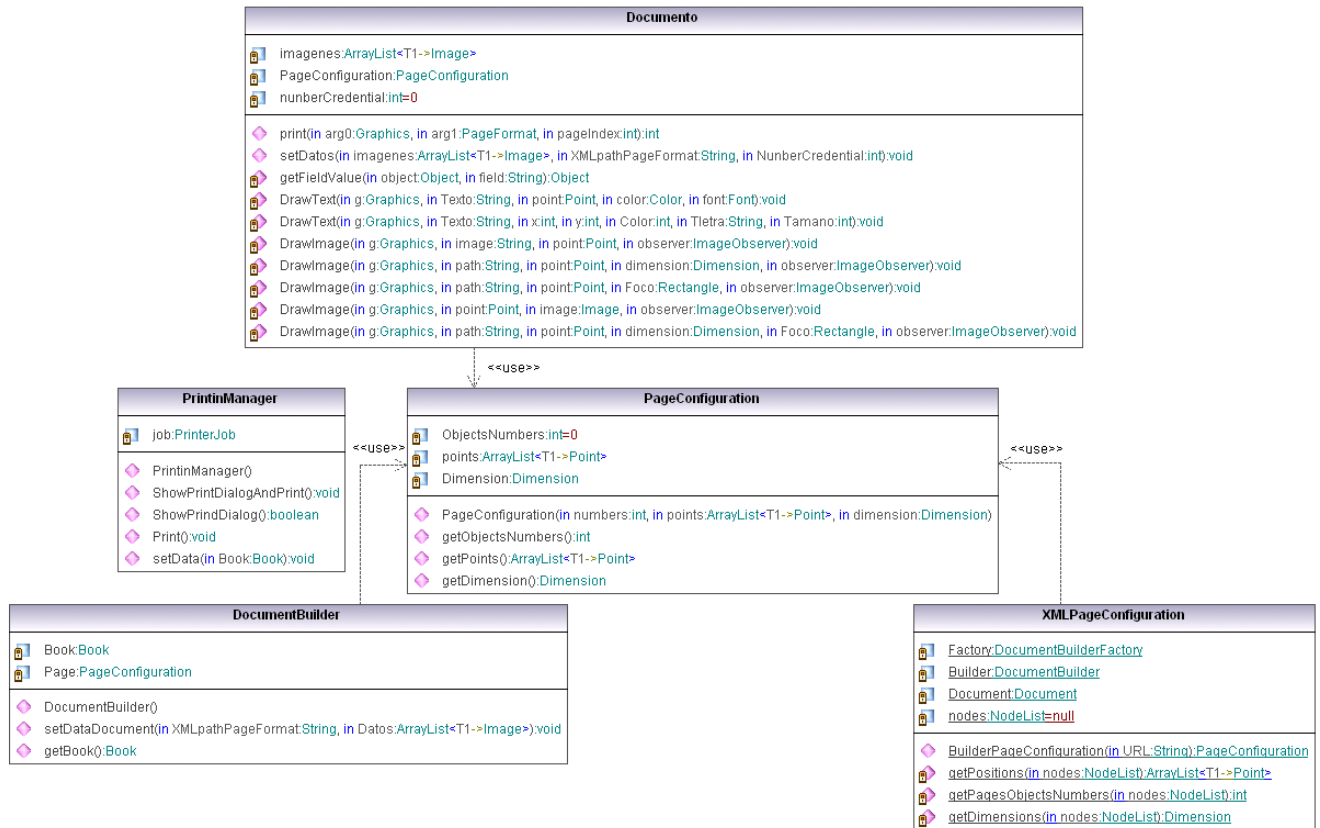
GeneradorPlantillaXML	
	_objPlantilla: Plantilla
	_plantilla: Document=null
	elementoPlantilla: Element=null
	_TAGDISEÑO: String="diseño"
	_TAGPLANTILLA: String="plantilla compilada"
	_ATRVERSION: String="versionidsvg"
	_TAGMODELODATOS: String="modelodatos"
	_TAGCAMPO: String="campo"
	_ATRID: String="id"
	_ATRTIPO: String="tipo"
	_TAGENLACEDATOS: String="enlacedatos"
	_TAGENLACE: String="enlace"
	_ATRCAMPOFUENTE: String="camposfuente"
	_ATRELEMENTOVISUAL: String="elemento visual"
	_XML_VERSION: String="1.0"
	_XML_ENCODING: String="ISO-8859-1"
	_JAVA_ENCODING: String="8859_1"
	_NOMBRE_ARCHIVO_XML: String="FicheroXML"
	_TAGCAMBIOCONTENIDO: String="cambio contenido"
	GeneradorPlantillaXML(in Plantilla: Plantilla)
	Generar(): void
	GenerarModeloDatos(in modelodatos: ModeloDatos): void
	GenerarDiseño(in diseño: Diseño): void
	GenerarEnlaceDatos(in enlacedatos: EnlaceDatos): void
	GrabarPlantillaNoCompilada(in XMLEnTexto: String): void
	SalvarEnDisco(in ruta: String): void
	DocumentoXMLaTexto(): String

GeneradorDatosPersonalizacion	
	_personalizacion: Personalizacion
	xmlDoc: Document=null
	datoXML: Element=null
	<<final>> TAG_PRINCIPAL: String="datos"
	<<final>> TAG_DATOSP: String="personalizacion"
	GeneradorDatosPersonalizacion(in _personalizacion: Personalizacion)
	GenerarDocumento(): void
	ObtenerDocumento(): String
	grabaDocumentoXML(in docEncadenaTexto: String, in ruta: String): void
	main(in args: String[]): void

CargadorPlantillaCompilada	
	_plantillaCompilada: PlantillaCompilada
	_document: Document
	CargadorPlantillaCompilada()
	CargadorPlantillaCompilada(in _plantillaCompilada: PlantillaCompilada)
	getPlantillaCompilada(): PlantillaCompilada
	Cargar(in camino: String): void
	DocumentoXMLaTexto(): String
	CargarModeloDatos(in camino: String, in cargarXSeparado: boolean): ModeloDatos
	CargarEnlaceDatos(in camino: String, in cargarXSeparado: boolean): EnlaceDatos
	CargarDiseño(): Diseño
	CargarEsquema(): void
	CargarTransformacion(): void

GeneradorPlantillaCXML	
	_objPlantillaCompilada: PlantillaCompilada
	textoXSD: String=""
	_plantillaCompilada: Document=null
	elementoPlantillaCompilada: Element=null
	_TAGDISEÑO: String="diseño"
	_TAGPLANTILLA: String="plantilla compilada"
	_ATRVERSION: String="versionidsvg"
	_TAGMODELODATOS: String="modelodatos"
	_TAGESQUEMADATOS: String="esquemadatos"
	_TAGCAMPO: String="campo"
	_ATRID: String="id"
	_ATRTIPO: String="tipo"
	_TAGENLACEDATOS: String="enlacedatos"
	_TAGENLACE: String="enlace"
	_ATRCAMPOFUENTE: String="camposfuente"
	_ATRELEMENTOVISUAL: String="elemento visual"
	_XML_VERSION: String="1.0"
	_XML_ENCODING: String="ISO-8859-1"
	_JAVA_ENCODING: String="8859_1"
	_NOMBRE_ARCHIVO_XML: String="FicheroXML"
	GeneradorPlantillaCXML(in PlantillaCompilada: PlantillaCompilada)
	getTextoXSD(): String
	Generar(): void
	GenerarModeloDatos(in modelodatos: ModeloDatos): void
	GenerarEsquemas(in modelodatos: ModeloDatos): void
	GenerarDiseño(in diseño: Diseño): void
	GenerarEnlaceDatos(in enlacedatos: EnlaceDatos): void
	GrabarPlantillaCompilada(in XMLEnTexto: String): void
	SalvarEnDisco(in ruta: String): void
	DocumentoXMLaTexto(): String

Capítulo 3: Construcción y Elaboración del Sistema



Capítulo 3: Construcción y Elaboración del Sistema

Tabla 3.1 Descripción de la clase Enlace

Nombre: Enlace	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_campofuente	String
_elementovisual	String
Responsabilidades	
Nombre	Descripción
Enlace()	Constructor sin parámetros.
Enlace(String pcf, String pev)	Constructor con parámetros.
getCampofuente() : String	
getElementovisual() : String	
setCampofuente(String pcampofuente) : void	Cambia _campofuente por pcampofuente.
setElementovisual(String pelementvis):void	Cambia _elementovisual por pelementvis.
Override toString() : String	Convierte a String.

Tabla 3.2 Descripción de la clase Campo

Nombre: Campo	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_id	String
_tipo	String
Responsabilidades	
Nombre	Descripción
getId() : String	Retorna el _id.
getTipo() : String	Retorna el _tipo.
setId(String pid) : void	Cambia el _id por el pid del parámetro.
setTipo(String ptipo)	Cambia el _tipo por el ptipo del parámetro.
Campo(String pid, String ptipo)	Constructor con parámetros.
Campo()	Constructor sin parámetros.

Capítulo 3: Construcción y Elaboración del Sistema

Tabla 3.3 Descripción de la clase Diseño

Nombre: Diseño	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_svg = null	Document
Responsabilidades	
Nombre	Descripción
Diseño()	Constructor sin parámetros.
Diseño(Document svg)	Constructor con parámetros.
setSvg(Document psvg) : void	Cambia el _svg por el psvg del parámetro.
Override toString() : String	Convierte a String.

Tabla 3.4 Descripción de la clase EnlaceDatos

Nombre: EnlaceDatos	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_listaenlaces	ArrayList<Enlace>
Responsabilidades	
Nombre	Descripción
EnlaceDatos()	Constructor sin parámetros.
get_listaEnlaces():ArrayList<Enlace>	Retorna _listaenlaces.
Adicionar(Enlace pnuevo) : void	Adiciona un elemento a la lista.
Eliminar(Enlace pnuevo) : void	Elimina un elemento de la lista.

Tabla 3.5. Descripción de la clase ModeloDatos

Nombre: ModeloDatos	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_listacampo	ArrayList<Campo>
Responsabilidades	
Nombre	Descripción
ModeloDatos()	Constructor sin parámetros.
get_listaEnlaces():ArrayList<Enlace>	Retorna _listacampo.
Adicionar(Campo pnuevo) : void	Adiciona un elemento a la lista.
Eliminar(Campo pnuevo) : void	Elimina un elemento de la lista.

Capítulo 3: Construcción y Elaboración del Sistema

Nombre: Plantilla	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_modeloDatos	ModeloDatos
_enlaceDatos	EnlaceDatos
_svg	Diseño
Responsabilidades	
Nombre	Descripción
Plantilla()	Constructor sin parámetros.
Plantilla(ModeloDatos, EnlaceDatos, Diseño)	Constructor con parámetros.
getModelodatos():ModeloDatos	Retorna _modeloDatos.
getEnlacedatos():EnlaceDatos	Retorna _enlaceDatos.
getSvg():Diseño	Retorna _svg.
setModeloDatos(ModeloDatos pmodeloD)	Cambia _modeloDatos por pmodeloD.
setEnlaceDatos(EnlaceDatos penlaceD)	Cambia _enlaceDatos por penlaceD.
setSvg(Diseño psvg)	Cambia _svg por psvg.

Tabla 3.6 Descripción de la clase PlantillaCompilada

Nombre: PlantillaCompilada	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_esquemaDatos	Document
_transformaciones	Document
_svg	Diseño
Responsabilidades	
Nombre	Descripción
Plantilla()	Constructor sin parámetros.
Plantilla(ModeloDatos, EnlaceDatos, Diseño)	Constructor con parámetros.
getEsquemadatos():Document	Retorna _esquemaDatos
getTransformaciones():Document	Retorna _transformaciones
getSvg():Diseño	Retorna _svg
setEsquemaDatos(Document pesquemaD)	Cambia _esquemaDatos por pesquemaD
setTransformaciones(Document ptranf)	Cambia _transformaciones por ptranf
setSvg(Diseño psvg)	Cambia _svg por psvg
Nombre: CargadorPlantilla	

Capítulo 3: Construcción y Elaboración del Sistema

Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_plantilla	Plantilla
_document	Document
Responsabilidades	
Nombre	Descripción
CargadorPlantilla()	Constructor sin parámetros.
CargadorPlantilla(Plantilla _plantilla)	Constructor con parámetros.
getPlantilla():Plantilla	Retorna _plantilla
Cargar(String camino):void	Carga la plantilla desde un directorio.
CargarModeloDatos(String,bool):ModeloDatos	Carga modelo de datos desde la plantilla.
CargarEnlaceDatos(String,bool):EnlaceDatos	Carga enlace de datos desde la plantilla.
CargarDiseño(String,bool):Diseño	Carga diseño desde la plantilla.

Tabla 3.7 Descripción de la clase CargadorPlantillaCompilada

Nombre: CargadorPlantillaCompilada	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_plantillacompilada	PlantillaCompilada
_document	Document
Responsabilidades	
Nombre	Descripción
CargadorPlantilla()	Constructor sin parámetros.
CargadorPlantilla(Plantilla _plantillacompilada)	Constructor con parámetros.
getPlantillacompilada ():PlantillaCompilada	Retorna _ plantillacompilada
Cargar(String camino):void	Carga la plantillacompilada desde un directorio.
CargarEsquema():void	Carga esquema desde plantilla.
CargarTransformacion():void	Carga transformación desde plantilla.
CargarDiseño(String,bool):Diseño	Carga diseño desde la plantilla.

Capítulo 3: Construcción y Elaboración del Sistema

Tabla 3.8 Descripción de la clase GeneradorPlantillaXML

Nombre: GeneradorPlantillaXML	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_objplantilla	Plantilla
_plantilla	Document
_elemento	Element
_TAGDISEÑO	String
_TAGPLANTILLA	String
_ATRVERSION	String
_TAGMODELODATOS	String
_TAGCAMPO	String
_ATRID	String
_ATRTIPO	String
_TAGENLACEDATOS	String
_TAGENLACE	String
_ATRCAMPOFUENTE	String
_ATRELEMENTOVISUAL	String
_ATRATRIBUTO	String
_XML_VERSION	String
_XML_ENCODING	String
_JAVA_ENCODING	String
_NOMBRE_ARCHIVO_XML	String
Responsabilidades	
Nombre	Descripción
GeneradorPlantillaXML(Plantilla Plantilla)	Constructor con parámetros.
Generar():void	Genera la plantilla XML.
GenerarModeloDatos(ModeloDatos MD):void	Genera modelo de datos.
GenerarEnlaceDatos(EnlaceDatos ED):void	Genera enlace de datos.
GenerarDiseño(Diseño diseño)	Generar y cargar diseño svg.
GrabarPlantillaNoCompilada(String):void	Graba la plantilla no compilada.
SalvarEnDisco(String ruta):void	Salvar plantilla en disco.
DocumentoXMLaTexto():String	Llevar de documento XML a texto.

Capítulo 3: Construcción y Elaboración del Sistema

Tabla 3.9 Descripción de la clase DatosPersonalizacion

Nombre: DatosPersonalizacion	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_id	String
_tipo	String
Responsabilidades	
Nombre	Descripción
getId() : String	Retorna el _id.
getTipo() : String	Retorna el _tipo.
setId(String pid) : void	Cambia el _id por el pid del parámetro.
setTipo(String ptipo)	Cambia el _tipo por el ptipo del parámetro.
Campo(String pid, String ptipo)	Constructor con parámetros.
Campo()	Constructor sin parámetros.
toString():String	Convierte a String.

Tabla 3.10 Descripción de la clase Personalizacion

Nombre: Personalizacion	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_datos	ArrayList<DatoPersonalizacion>
Responsabilidades	
Nombre	Descripción
Personalizacion ()	Constructor sin parámetros.
Personalizacion (ArrayList<DatoPersonalizacion> _da)	Constructor con parámetros.
get_datos() :ArrayList<Enlace>	Retorna _dato.
Adicionar(DatoPersonalizacion dato) : void	Adiciona un elemento a la lista.
Adicionar(DatoPersonalizacion dato) : void	Elimina un elemento de la lista.

Tabla 3.11 Descripción de la clase Compilador

Nombre: Compilador	
Tipo de clase: IDSVG.Compilacion	
Atributo	Tipo
Responsabilidades	

Capítulo 3: Construcción y Elaboración del Sistema

Nombre	Descripción
Depura(String cadena) : void	Imprime una cadena de texto.
transformar(String xmlOrigen, String ficheroGendor)	Ejecuta transformación XSLT.

Tabla 3.12 Descripción de la clase ValidadorXMLContraXSD

Nombre: ValidadorXMLContraXSD	
Tipo de clase: IDSVG. Compilacion	
Atributo	Tipo
_blsXmlOk	boolean
_NAME_SPACE_AWARE	boolean
_VALIDATING	boolean
_SCHEMA_LANGUAGE	String
_SCHEMA_LANGUAGE_VAL	String
_SCHEMA_SOURCE	String
_sCtrlErr	String
Responsabilidades	
Nombre	Descripción
ValidarXMLVsXSD(String Xml, String Xsd) : void	Validar fichero XML contra fichero XSD.

Tabla 3.13 Descripción de la clase PrintinManager

Nombre: PrintinManager	
Tipo de clase: IDSVG.Impresion	
Atributo	Tipo
_job	PrinterJob
Responsabilidades	
Nombre	Descripción
PrintinManager()	Constructor sin parámetros.
ShowPrintDialogAndPrint() : void	Muestra el dialogo de la impresora e imprime.
Print() : void	Imprime según la configuración de la impresora.
setData(Book Book) : void	Le pasa a la impresora la información a imprimir.

Capítulo 3: Construcción y Elaboración del Sistema

Tabla 3.14 Descripción de la clase DocumentBuilder

Nombre: DocumentBuilder	
Tipo de clase: IDSVG.Impresion	
Atributo	Tipo
_Book	Book
_PageConfiguration	Page
Responsabilidades	
Nombre	Descripción
DocumentBuilder ()	Constructor sin parámetros.
setDataDocument (String , ArrayList<Image>) : void	Recibe datos necesarios para crear documento.
getBook() : Book	Facilita el documento listo para imprimir.

Tabla 3.15. Descripción de la clase Documento

Nombre: Documento	
Tipo de clase: IDSVG.Impresion	
Atributo	Tipo
_image	ArrayList<Image>
_PageConfiguration	PageConfiguration
_nunberCredential	int
Responsabilidades	
Nombre	Descripción
Document ()	Constructor sin parámetros.
Print(Graphics, PageFormat, pageIndex) : int	Dibujar la imagen según la posición.
setDatos(ArrayList<Image>, String, int):void	Recibe datos necesarios para crear una página.
DrawImage(Graphics, Point, Image, ImageObs):void	Pinta la imagen del documento a imprimir.

Tabla 3.16 Descripción de la clase PageConfiguration

Nombre:PageConfiguration	
Tipo de clase: IDSVG.Impresion	
Atributo	Tipo
_ ObjectsNumbers	int
_ points	ArrayList<Point>
_ Dimension	Dimension
Responsabilidades	

Capítulo 3: Construcción y Elaboración del Sistema

Nombre	Descripción
PageConfiguration (int, ArrayList<Point>, Dimension)	Constructor con parámetros.
getObjectsNumbers(): int	Retorna _ ObjectsNumbers.
getPoints():ArrayList<Point>	Retorna _ points.
getDimension():Dimension	Retorna _ Dimension.

Tabla 3.17 Descripción de la clase XMLPageConfiguration

Nombre: XMLPageConfiguration	
Tipo de clase: IDSVG.Impresion	
Atributo	Tipo
_ Factory	DocumentBuilderFactory
_ Builder	DocumentBuilder
_ document	Document
_nodes	NodeList
Responsabilidades	
Nombre	Descripción
BuilderPageConfiguration(String):PageConfiguration	Inicializar objetos de la clase PageConfiguration.
getPosition():ArrayList<Point>	Retorna una lista de posiciones.
getPagesObjectsNumbers(NodeList):int	Retorna valor de un tag que esta en el XML.
getDimensions(NodeList): Dimension	Retorna dimensiones de la página.

Tabla 3.18 Descripción de la clase SVGDisposer

Nombre: SVGDisposer	
Tipo de clase: IDSVG.Dibujo	
Atributo	Tipo
Responsabilidades	
Nombre	Descripción
DisposeSVG(JScrollPane, JPanel, JSVGCanvas):void	Borra la imagen SVG del panel.

Capítulo 3: Construcción y Elaboración del Sistema

Tabla 3.19 Descripción de la clase SVGLoader

Nombre: SVGLoader	
Tipo de clase: IDSVG.Dibujo	
Atributo	Tipo
Responsabilidades	
Nombre	Descripción
Load (String,JFrame,JLabel,JPanel,JSVGCanvas):void	Carga documento imagen SVG.

Tabla 3.20 Descripción de la clase SVGPainter

Nombre: SVGPainter	
Tipo de clase: IDSVG.Dibujo	
Atributo	Tipo
Responsabilidades	
Nombre	Descripción
Paint(JFrame, JPanel, JSVGCanvas,Diseño):void	Pinta el SVG en un contexto gráfico.

Tabla 3.21 Descripción de la clase CargadorDatosPersonalizacion

Nombre: CargadorDatosPersonalizacion	
Tipo de clase: IDSVG.Impresion	
Atributo	Tipo
_personalizacion	Personalizacion
_ xmlDoc	Document
Responsabilidades	
Nombre	Descripción
CargadorDatosPersonalizacion(Personalizacion _pers)	Constructor con parámetros.
CargarXMLDatos (): void	Carga los datos desde un fichero XML.

Tabla 3.22 Descripción de la clase GeneradorPlantillaCXML

Nombre: GeneradorPlantillaCXML	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
_objplantillacompilada	PlantillaCompilada
_plantillaCompilada	Document
_elementplantillacompilada	Element

Capítulo 3: Construcción y Elaboración del Sistema

_textoXSD	String
_TAGDISEÑO	String
_TAGPLANTILLA	String
_ATRVERSION	String
_TAGMODELODATOS	String
_TAGESQUEMADATOS	String
_TAGCAMPO	String
_ATRID	String
_ATRTIPO	String
_TAGENLACEDATOS	String
_TAGENLACE	String
_ATRCAMPOFUENTE	String
_ATRELEMENTOVISUAL	String
_ATRATRIBUTO	String
_XML_VERSION	String
_XML_ENCODING	String
_JAVA_ENCODING	String
_NOMBRE_ARCHIVO_XML	String
Responsabilidades	
Nombre	Descripción
GeneradorPlantillaCXML(PlantillaCompilada Plantilla)	Constructor con parámetros.
getTextoXSD():String	Retorna Texto del fichero XSD.
Generar():void	Genera la plantilla XML.
GenerarModeloDatos(ModeloDatos MD):void	Genera modelo de datos.
GenerarEnlaceDatos(EnlaceDatos ED):void	Genera enlace de datos.
GenerarEsquemaDatos(ModeloDatos MD):void	Genera el esquema XSD.
GenerarDiseño(Diseño diseño):void	Generar y cargar diseño svg.
GrabarPlantillaCompilada(String):void	Graba la plantilla compilada.
SalvarEnDisco(String ruta):void	Salvar plantilla en disco.
DocumentoXMLaTexto():String	Llevar de documento XML a texto.

Tabla 3.23 Descripción de la clase GeneradorDatosPersonalizacion

Nombre: GeneradorDatosPersonalizacion	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo

Capítulo 3: Construcción y Elaboración del Sistema

_personalizacion	Personalizacion
_xmlDoc	Document
_datoXML	Element
_TAG_PRINCIPAL	String
_TAG_DATOSP	String
Responsabilidades	
Nombre	Descripción
GeneradorDatosPersonalizacion (Personalizacion per)	Constructor con parámetros.
GenerarDocumento():void	Generar dinámicamente el documento XML.
ObtenerDocumento():void	Obtener el documento generado.
grabaDocumentoXML(String docEncte,String R):void	Graba en el disco el documento XML.

Tabla 3.24 Descripción de la clase TempFileHandler

Nombre: TempFileHandler	
Tipo de clase: IDSVG.Comun	
Atributo	Tipo
Responsabilidades	
Nombre	Descripción
SaveTempFileXMLType(Document do, String no):void	Salva temporalmente un fichero en disco.
LoadTempFileXMLType():Document	Carga temporalmente un fichero desde disco.

Tabla 3.25 Descripción de la clase frmPrincipalPersonalizacionImpresion

Nombre: frmPrincipalPersonalizacionImpresion	
Tipo de clase: IDSVG.Interfaz	
Atributo	Tipo
CargarDatosMenuItem	JMenuItem
CargarPlantillaCMenuItem	JMenuItem
MenuAplicación	JMenu
MenuDemo	JMenu
MenuFicheros	JMenu
Menú	JMenuBar
PersonalizarMenuItem	JMenuItem
SalirMenuItem	JMenuItem
pnlBordeInferior	JPanel
pnlDerecha	JPanel

Capítulo 3: Construcción y Elaboración del Sistema

pnIzquierdo	JPanel
pnISVGNoPeronalizado	JPanel
pnISVGPeresonalizado	JPanel
spSVGNoPersonalizado	JScrollPane
spSVGPersonalizado	JScrollPane
spTabla	JScrollPane
tablaDatos	JTable
_plantillaCompilada	PlantillaCompilada
_personalizacion	Personalizacion
_SVGNopersonalizado	JSVGCanvas
_SVGPersonalizado	JSVGCanvas
Responsabilidades	
Nombre	Descripción
frmPrincipaPersonalizacionImpresionI()	Constructor del formulario.
initComponents():void	Inicializar componentes.
CargarPlantilla():int	Carga la plantilla compilada.
HabilitadorAreaSVGNoPersonalizado():void	Habilita el área del SVG no personalizado.
HabilitadorAreaSVGPersonalizado():void	Habilitar el área del SVG personalizado.
LimpiarAreaTrabajo():void	Limpiar el área de trabajo.
DeshabilitarAreaTrabajo():void	Deshabilitar el área de trabajo.
HabilitadorMenuDemo():void	Habilitar el menú demo.
PintarAreaSvgNoPersonalizado():void	Pinta el SVG no personalizado.
PintarSVGPersonalizado(String SVG, String cam):int	Pinta SVG personalizado.
MenúMouseEntered(ActionEvent evt):void	Verifica el estado de la plantilla.
SalirMenuItemActionPerformed(ActionEvent evt):void	Sale del programa.
CargarPlantillaCMenuItemActionPerformed(evt):void	Cargar la plantilla compilada desde la interfaz.
CargarDatosMenuItemActionPerformed(evt):void	Carga y valida los datos de la personalización.
PersonalizarMenuItemActionPerformed(evt)	Personalizar e imprimir documento.

Tabla 3.26 Descripción de la clase frmPrincipalDiseñador

Nombre: frmPrincipalDiseñador	
Tipo de clase: IDSVG.Interfaz	
Atributo	Tipo
CargarEDMenuItem	JMenuItem
CargarMDMenuItem	JMenuItem

Capítulo 3: Construcción y Elaboración del Sistema

CargarPlantillaMenuItem	JMenuItem
CargarSVMenuItem	JMenuItem
CerrarTodoMenuItem	JMenuItem
CrearEDMenuItem	JMenuItem
CrearMDMenuItem	JMenuItem
MenuAplicacion	JMenu
MenuFicheros	JMenu
MenuPlantilla	JMenu
MenuPrincipal	JMenuBar
NuevaPlantillaMenuItem	JMenuItem
SalirMenuItem	JMenuItem
SalvarPlantillaCompiladaMenuItem	JMenuItem
SalvarPlantillaMenuItem	JMenuItem
btnAdd	JButton
btnAñadir	JButton
btnAñadirDlgCrearED	JButton
btnCancel	JButton
btnEliminar	JButton
btnEliminarDlgCrearED	JButton
btnOk	JButton
btnRemove	JButton
btnSalirDlgCrearED	JButton
btnSalirDlgCrearMD	JButton
cbTipo	JComboBox
dlgCrearED	JDialog
dlgCrearEDXPATH	JDialog
dlgCrearMD	JDialog
jList2	JList
jList3	JList
jList4	JList
jScrollPane3	JScrollPane
jScrollPane4	JScrollPane
jScrollPane5	JScrollPane
jSeparator1	JSeparator
jSeparator2	JSeparator

Capítulo 3: Construcción y Elaboración del Sistema

lblAtributoDlgCrearED	JLabel
lblCampoFuente	JLabel
lblCampoFuenteDlgCrearED	JLabel
lblElemento	JLabel
lblElementoVisualDlgCrearED	JLabel
lblEnlaces	JLabel
lblID	JLabel
lblStatus	JLabel
lblTipo	JLabel
listaED	JList
listaEDdlgCrearED	JList
listaMD	JList
listaMDdlgCrearMD	JList
pnlAreaDibujo	JPanel
pnlAreaTrabajoED	JPanel
pnlAñadirQuitarCamposDlgCrearED	JPanel
pnlCamposDlgCrearMD	JPanel
pnlEnlaceDatos	JPanel
pnlIzquierdo	JPanel
pnlModeloDatos	JPanel
pnlSeparador	JSplitPane
pnlStatus	JPanel
spAreaDibujo	JScrollPane
spListaED	JScrollPane
spListaEDdlgCrearED	JScrollPane
spListaMD	JScrollPane
spListaMDdlgCrearMD	JScrollPane
tbAtributoDlgCrearED	JTextField
tbCampoFuenteDlgCrearED	JTextField
tbElementoVisualDlgCrearED	JTextField
tbID	JTextField
_plantilla	Plantilla
diseñoSVG	JSVGCanvas
Responsabilidades	
Nombre	Descripción

Capítulo 3: Construcción y Elaboración del Sistema

frmPrincipalDiseñador ()	Constructor del formulario.
HabilitadorMenuPlantilla(boolean habilitar):void	Habilita el menú plantilla.
SalvarPlantilla():int	Salva la plantilla.
CargarPlantilla():int	Carga la plantilla compilada.
ListarModeloDatos ():void	Listar modelo de datos.
ListarEnlaceDatos():void	Listar enlace de datos.
LimpiarControlesJList(JList control)	Limpia control.
HabilitadorAreaSVGPersonalizado():void	Habilitar el área del SVG personalizado.
LimpiarAreaTrabajo():void	Limpiar el área de trabajo.
DeshabilitarAreaTrabajo():void	Deshabilitar el área de trabajo.
HabilitadorMenuDemo():void	Habilitar el menú demo.
PintarAreaDibujo():void	Pintar el SVG en el área de trabajo.
initComponents():void	Inicializar componentes.
MenuPrincipalMouseEntered(MouseEvent evt):void	Verifica el estado de la plantilla.
NuevaPlantillaMenuItemActionPerformed(evt):void	Crea una nueva plantilla.
SalirMenuItemActionPerformed(ActionEvent evt):void	Salir del programa.
SalvarPlantillaMenuItemActionPerformed(evt):void	Salva la plantilla.
CargarPlantillaMenuItemActionPerformed(evt):void	Carga la plantilla.
CerrarTodoMenuItemActionPerformed(evt):void	Cierra el contexto de la plantilla.
CrearMDMenuItemActionPerformed(evt):void	Crea modelo de datos desde la interfaz.
listaMDdlgCrearMDValueChanged(evt):void	Lista modelo de datos en la interfaz.
btnAñadirActionPerformed(ActionEvent evt):void	Añade un elemento.
btnEliminarActionPerformed(ActionEvent evt):void	Elimina un elemento.
btnSalirDlgCrearMDActionPerformed(evt):void	Salir del formulario modelo de datos.
CrearEDMenuItemActionPerformed(evt):void	Llama al formulario de crear el enlace de datos.
CargarMDMenuItemActionPerformed(evt):void	Cargar modelo de datos desde la interfaz.
CargarEDMenuItemActionPerformed(evt):void	Cargar enlace de datos desde la interfaz.
CargarSVMenuItemActionPerformed(evt):void	Cargar imagen SVG.
dlgCrearMDWindowActivated(WindowEvent evt):void	Edita modelo de datos.
btnSalirDlgCrearEDActionPerformed(evt):void	Salir de la interfaz de enlace de datos.
btnAñadirDlgCrearEDActionPerformed(evt):void	Adiciona elementos al enlace de datos.
listaEDdlgCrearEDValueChanged(evt):void	Lista valores en el enlace de datos.
dlgCrearEDWindowActivated(WindowEvent evt):void	Verifica estado del enlace de datos en la plantilla.
btnEliminarDlgCrearEDActionPerformed(evt):void	Elimina elemento de visuales de enlace de datos.
SalvarPlantillaCompiladaActionPerformed(evt):void	Salva plantilla compilada en disco.

Capítulo 3: Construcción y Elaboración del Sistema

main(String args[]):void	Levanta el formulario.
--------------------------	------------------------

Diagramas de contrato entre paquetes.

Un diagrama de contrato entre paquetes es un diagrama de secuencia que muestra el paso de mensajes entre las diferentes carpetas, sin mostrar todo el flujo de eventos que ocurre dentro de las mismas cuando llega una determinada petición. Es un diagrama de secuencia de alto nivel, su objetivo es brindar a los programadores, sin incurrir en demasiado nivel de detalle, lo necesario para la realización de los casos de uso. Este artefacto ahorra tiempo de diseño, ya que su realización no requiere demasiadas especificaciones, y simplifica de manera considerable los diagramas necesarios para las realizaciones.

A continuación se muestran los diagramas de interacción más significativos:

Diagrama de contrato de paquetes para el Caso de Uso: Crear plantillas.

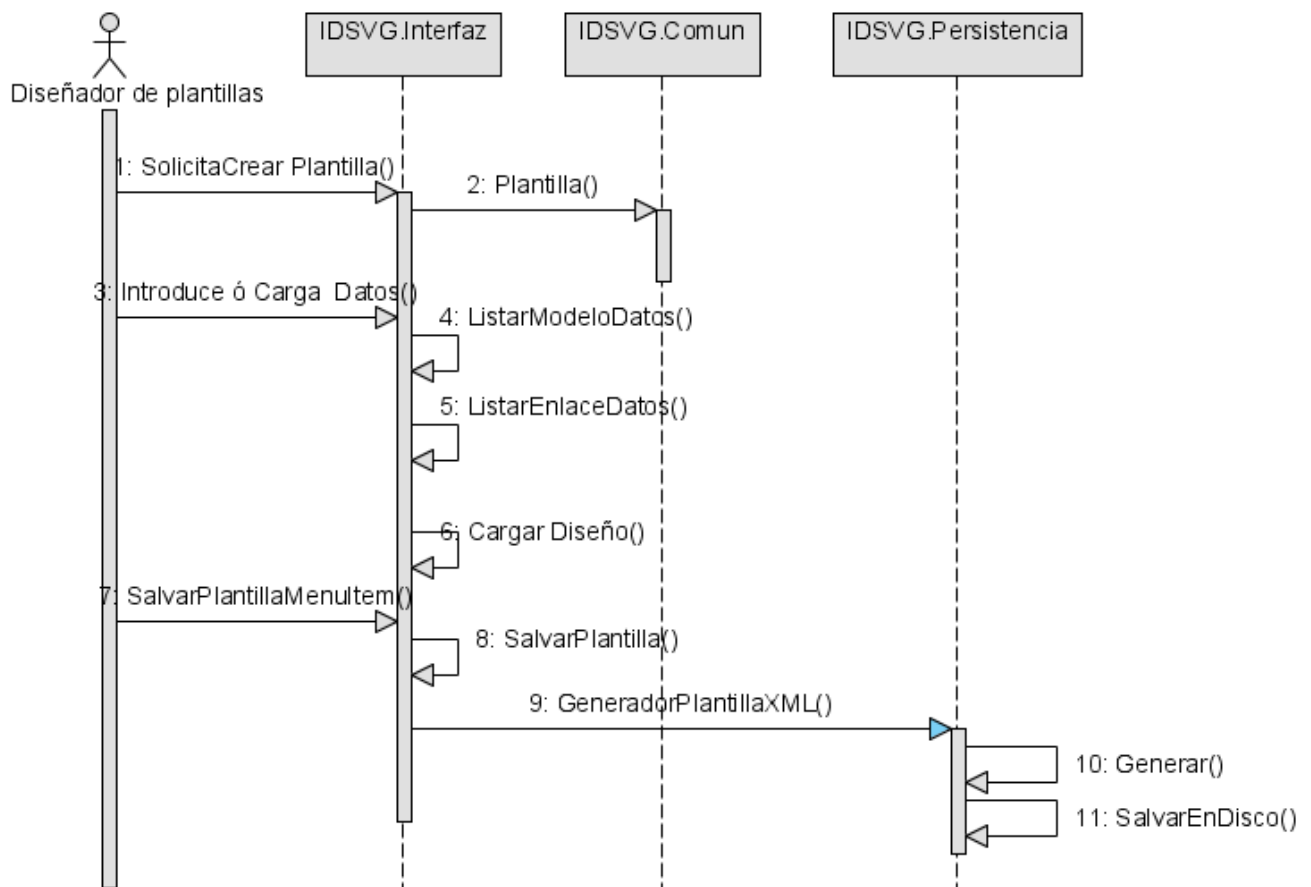


Figura 15. Diagrama de contrato de paquetes del caso de uso: Crear plantillas.

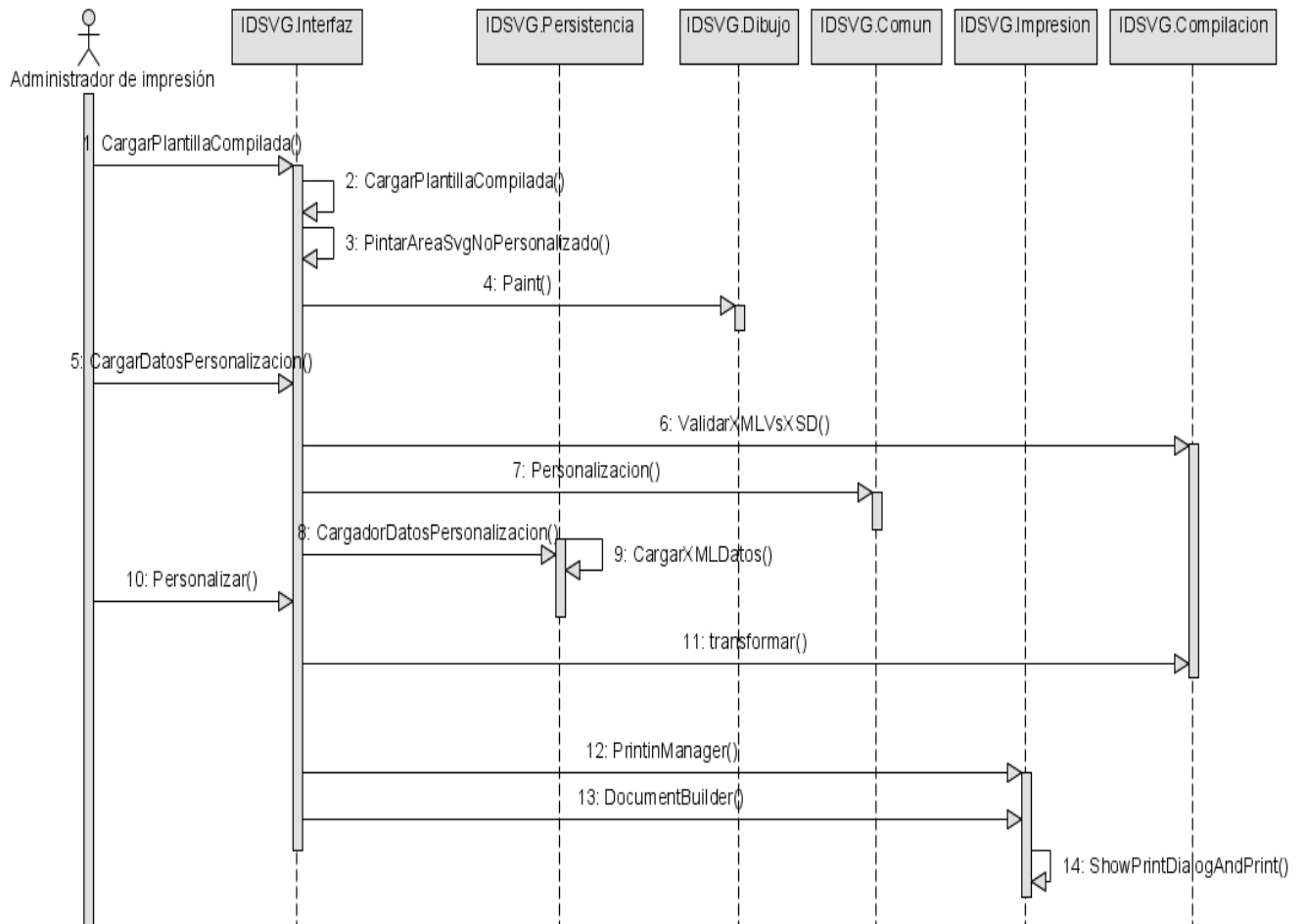


Figura 16. Diagrama de contrato de paquetes del caso de uso: Personalizar documento SVG.

Principios de Diseño de Interfaz

La interfaz gráfica del usuario es el medio por el cual este interactúa con el sistema, por lo que su diseño debe ser amigable, consistente, necesita contener retroalimentación, minimizar las posibilidades de error y la memorización, debe proveer la recuperación de errores y que no requiera usuarios con un alto nivel informático.

Una aplicación con una interfaz bien diseñada, además de un buen diseño gráfico, debe tener una buena navegabilidad, usabilidad y distribución de los contenidos. Este trabajo utilizará en el diseño los principios antes mencionados.

Capítulo 3: Construcción y Elaboración del Sistema

Estándares de la interfaz de la aplicación

Las interfaces permiten la comunicación y el intercambio de información entre el usuario y el sistema. El sistema que se presenta, va a disponer de dos aplicaciones con interfaces distintas, el diseñador de plantillas y la aplicación que personaliza e imprime el documento de identificación.

El diseñador de plantillas posee un menú principal donde están un conjunto de operaciones o acciones que se realizarán en el mismo, al igual que la aplicación de la parte de personalización e impresión.

Después de una breve reseña de las aplicaciones se muestran las interfaces para que se tenga un mayor entendimiento de lo que hacen las distintas aplicaciones.

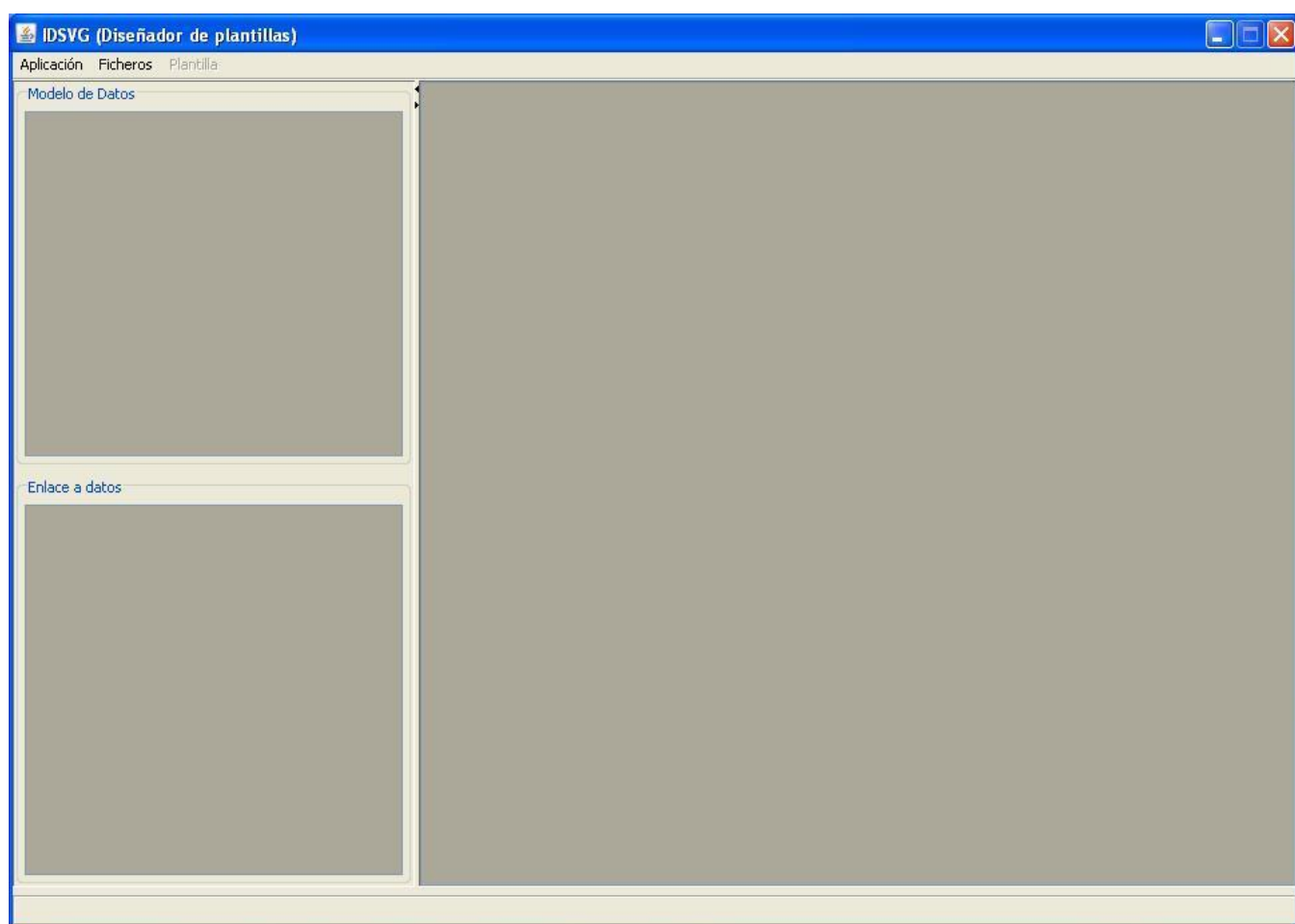


Figura 17. Formulario principal del Diseñador de plantillas.

En este caso el diseñador de plantillas lo mismo puede cargar una plantilla que ya esté hecha editarla o también puede crear una plantilla en blanco al igual que la puede salvar, e sistema le da la ventaja al

Capítulo 3: Construcción y Elaboración del Sistema

usuario que pueda hacer lo que estime conveniente, para un mayor entendimiento del proceso de creación de la plantillas mostramos la siguiente figura:

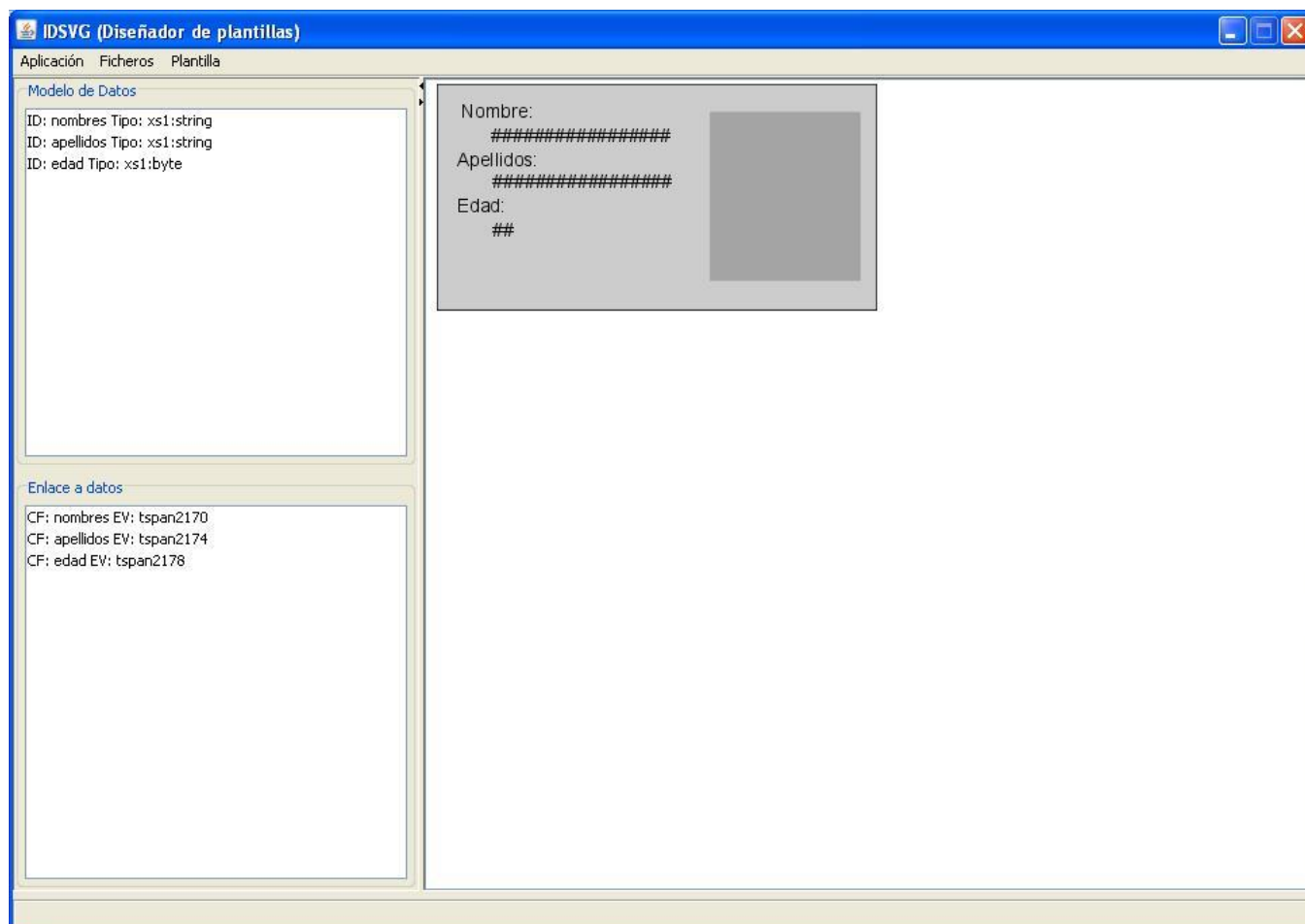


Figura 18. Formulario principal con una plantilla cargada.

En este caso ya el diseñador hizo una nueva plantilla donde cargo el diseño SVG hecho por el Inkscape, fue a las opciones del menú plantilla creo el modelo de datos el enlace de datos o también puede ver cargado ambas cosas en vez de crearlos, esta plantilla que está cargada, el sistema le da la oportunidad al usuario en este caso el diseñador de plantillas a salvarla cuando esa plantilla es salvada en disco el resultado de la misma es un fichero XML que tiene empotrado secciones de XML, los cuales son el modelo de datos para la personalización, el enlace de datos y el diseño SVG anteriormente mencionado, cuando esa plantilla es salvada en disco tiene la siguiente estructura:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plantilla versionidsvg="1.0">
  <modelodatos>
```


Capítulo 3: Construcción y Elaboración del Sistema

```
<campo id="nombres" tipo="xs:string"/>
<campo id="apellidos" tipo="xs:string"/>
<campo id="edad" tipo="xs:byte"/>
</modelodatos>
<enlacedatos>
  <enlace>
    <cambiocontenido campofuente="nombres" elementovisual="tspan2170"/>
  </enlace>
  <enlace>
    <cambiocontenido campofuente="apellidos" elementovisual="tspan2174"/>
  </enlace>
  <enlace>
    <cambiocontenido campofuente="edad" elementovisual="tspan2178"/>
  </enlace>
</enlacedatos>
<diseño>
  <svg height="177.16534" id="svg2"
    inkscape:output_extension="org.inkscape.output.svg.inkscape"
    inkscape:version="0.45.1"
    sodipodi:docbase="C:\Documents and Settings\Labrada\Escritorio"
    sodipodi:docname="dibujo.svg" sodipodi:version="0.32"
    version="1.0" width="354.33069"
    xmlns="http://www.w3.org/2000/svg"
    xmlns:cc="http://web.resource.org/cc/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
  xmlns:svg="http://www.w3.org/2000/svg">
    <defs id="defs4"/>
    <sodipodi:namedview bordercolor="#666666"
      borderlayer="false" borderopacity="1.0" height="5cm"
      id="base" inkscape:current-layer="layer1"
```

Capítulo 3: Construcción y Elaboración del Sistema

```
inkscape:cx="213.06174" inkscape:cy="82.158804"
inkscape:document-units="cm" inkscape:pageopacity="0.0"
inkscape:pageshadow="2" inkscape:window-height="682"
inkscape:window-width="1024" inkscape:window-x="-4"
inkscape:window-y="-4" inkscape:zoom="1.979899"
pagecolor="#ffffff" units="cm" width="10cm"/>
<metadata id="metadata7">
  <rdf:RDF>
    <cc:Work rdf:about="">
      <dc:format>image/svg+xml</dc:format>
      <dc:type rdf:resource="http://purl.org/dc/dcmitype/StillImage"/>
    </cc:Work>
  </rdf:RDF>
</metadata>
<g id="layer1" inkscape:groupmode="layer" inkscape:label="Capa 1">
  <rect height="167.62099" id="rect2160"
    style="fill:#cccccc;fill-rule:evenodd;stroke:#000000;stroke-width:0.64802599px;stroke-
linecap:butt;stroke-linejoin:miter;stroke-opacity:1"
    width="324.68439" x="8.6379995" y="3.1937988"/>
  <text id="lblNombres"
    style="font-size:14px;font-style:normal;font-weight:normal;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1;font-
family:Bitstream Vera Sans"
    x="25.927109" xml:space="preserve"
    y="27.566341">
    <tspan id="tspan2173"
      sodipodi:role="line" x="25.927109" y="27.566341">Nombre:</tspan>
    </text>
  <text id="lblApellidos"
    style="font-size:14px;font-style:normal;font-weight:normal;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1;font-
family:Bitstream Vera Sans"
    x="22.521656" xml:space="preserve"
```

Capítulo 3: Construcción y Elaboración del Sistema

```
y="63.906212">
    <tspan id="tspan2162"
sodipodi:role="line" x="22.521656" y="63.906212">Apellidos:</tspan>
</text>
<text id="lblEdad"
style="font-size:14px;font-style:normal;font-weight:normal;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1;font-
family:Bitstream Vera Sans"
x="22.52166" xml:space="preserve"
y="98.251404">
    <tspan id="tspan2166"
sodipodi:role="line" x="22.52166" y="98.251404">Edad:</tspan>
</text>
<text id="campoNombres"
style="font-size:14px;font-style:normal;font-weight:normal;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1;font-
family:Bitstream Vera Sans"
x="48.466324" xml:space="preserve"
y="46.208233">
    <tspan id="tspan2170"
sodipodi:role="line" x="48.466324" y="46.208233">#####</tspan>
</text>
<text id="campoApellidos"
style="font-size:14px;font-style:normal;font-weight:normal;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1;font-
family:Bitstream Vera Sans"
x="49.497471" xml:space="preserve"
y="80.190697">
    <tspan id="tspan2174"
sodipodi:role="line" x="49.497471" y="80.190697">#####</tspan>
</text>
<text id="campoEdad"
```

Capítulo 3: Construcción y Elaboración del Sistema

```
style="font-size:14px;font-style:normal;font-weight:normal;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1;font-
family:Bitstream Vera Sans"
x="49.349102" xml:space="preserve"
y="116.22065">
<tspan id="tspan2178"
sodipodi:role="line" x="49.349102" y="116.22065">##</tspan>
</text>
<rect height="125.25892" id="rect2180"
style="fill-opacity:0.51955307;fill:#808080"
width="111.62186" x="210.11172" y="23.622158"/>
</g>
</svg>
</diseño>
</plantilla>
```

A partir de esta plantilla con una transformación XSLT se va a generar otra plantilla llamada plantilla compilada, la misma poseerá el mismo diseño SVG de la plantilla, un esquema XSD generado dinámicamente a partir del modelo de datos de la personalización de la plantilla para validar un XML de datos de prueba para la personalización, es decir se genera un esquema XSD dinámicamente desde un lugar (modelo de datos de la plantilla) para validar a un XML que viene de otro lugar (XML de personalización que provee el sistema al cual se integrarán los componentes y utilitarios para la impresión de documentos de identificación.) y otra transformación XSLT que es la que pasándole el XML de datos de la personalización y el diseño SVG va a transformar ese diseño SVG, convirtiéndolo en un diseño personalizado, obteniéndose el documento de identificación, para un mayor entendimiento se muestra a continuación la aplicación que se encargará de realizar el proceso completo de personalización e impresión.

Capítulo 3: Construcción y Elaboración del Sistema

La plantilla compilada posee la siguiente estructura:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plantillacompilada xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xs1="http://alias/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsl1="http://www.w3.org/1999/XSL/Transform" versionidsvg="1.0">
  <esquemadatos>
    <xs:schema id="esquemadatos1">
      <xs:element name="datos">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="personalizacion">
              <xs:complexType>
                <xs:all>
                  <xs:element name="nombres"
type="xs1:string"/>
                  <xs:element name="apellidos"
type="xs1:string"/>
                  <xs:element name="edad"
type="xs1:byte"/>
                </xs:all>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </esquemadatos>
  <transformacion>
    <xsl1:stylesheet id="transformacion1" version="2.0">
      <xsl1:output indent="yes" encoding="ISO-8859-1" version="1.0" method="xml"/>
      <xsl1:param select="C:\temp\idsvg\datos.xml" name="fichero-datos"/>
    </xsl1:stylesheet>
  </transformacion>
</plantillacompilada>
```

Capítulo 3: Construcción y Elaboración del Sistema

```
name="datos"/>
<xsl:variable select="document($fichero-datos)/datos/personalizacion"
name="datos"/>
<xsl:template match="/">
    <xsl:call-template name="copianodo"/>
</xsl:template>
<xsl:template name="copianodo">
    <xsl:copy>
        <xsl:for-each select="@*">
            <xsl:call-template name="copiaatributo"/>
        </xsl:for-each>
        <xsl:for-each select="*">
            <xsl:call-template name="copianodo"/>
        </xsl:for-each>
        <xsl:call-template name="copiavalor"/>
    </xsl:copy>
</xsl:template>
<xsl:template name="copiaatributo">
    <xsl:copy/>
</xsl:template>
<xsl:template name="copiavalor">
    <xsl:choose>
        <xsl:when test="">
            <xsl:value-of select=""/>
        </xsl:when>
        <xsl:when test="">
            <xsl:value-of select=""/>
        </xsl:when>
        <xsl:when test="">
            <xsl:value-of select=""/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="text()"/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

Capítulo 3: Construcción y Elaboración del Sistema

```
</xsl:choose>
</xsl:template>
</xsl:stylesheet>
</transformacion>
<diseño>
  <svg xmlns="http://www.w3.org/2000/svg" xmlns:cc="http://web.resource.org/cc/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
xmlns:svg="http://www.w3.org/2000/svg" height="177.16534" id="svg2"
inkscape:output_extension="org.inkscape.output.svg.inkscape" inkscape:version="0.45.1"
sodipodi:docbase="C:\Documents and Settings\Labrada\Escritorio" sodipodi:docname="dibujo.svg"
sodipodi:version="0.32" version="1.0" width="354.33069">
  <defs id="defs4"/>
    <sodipodi:namedview bordercolor="#666666" borderlayer="false"
borderopacity="1.0" height="5cm" id="base" inkscape:current-layer="layer1" inkscape:cx="213.06174"
inkscape:cy="82.158804" inkscape:document-units="cm" inkscape:pageopacity="0.0"
inkscape:pageshadow="2" inkscape:window-height="682" inkscape:window-width="1024"
inkscape:window-x="-4" inkscape:window-y="-4" inkscape:zoom="1.979899" pagecolor="#ffffff"
units="cm" width="10cm"/>
    <metadata id="metadata7">
      <rdf:RDF>
        <cc:Work rdf:about="">
          <dc:format>image/svg+xml</dc:format>
          <dc:type
rdf:resource="http://purl.org/dc/dcmitype/StillImage"/>
        </cc:Work>
      </rdf:RDF>
    </metadata>
    <g id="layer1" inkscape:groupmode="layer" inkscape:label="Capa 1">
```

Capítulo 3: Construcción y Elaboración del Sistema

```
<rect height="167.62099" id="rect2160" style="fill:#cccccc;fill-  
rule:evenodd;stroke:#000000;stroke-width:0.64802599px;stroke-linecap:butt;stroke-  
linejoin:miter;stroke-opacity:1" width="324.68439" x="8.6379995" y="3.1937988"/>  
<text id="lblNombres" style="font-size:14px;font-style:normal;font-  
weight:normal;fill:#000000;fill-opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-  
linejoin:miter;stroke-opacity:1;font-family:Bitstream Vera Sans" x="25.927109" xml:space="preserve"  
y="27.566341">  
    <tspan id="tspan2173" sodipodi:role="line" x="25.927109"  
y="27.566341">Nombre:</tspan>  
    </text>  
    <text id="lblApellidos" style="font-size:14px;font-style:normal;font-  
weight:normal;fill:#000000;fill-opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-  
linejoin:miter;stroke-opacity:1;font-family:Bitstream Vera Sans" x="22.521656" xml:space="preserve"  
y="63.906212">  
    <tspan id="tspan2162" sodipodi:role="line" x="22.521656"  
y="63.906212">Apellidos:</tspan>  
    </text>  
    <text id="lblEdad" style="font-size:14px;font-style:normal;font-  
weight:normal;fill:#000000;fill-opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-  
linejoin:miter;stroke-opacity:1;font-family:Bitstream Vera Sans" x="22.52166" xml:space="preserve"  
y="98.251404">  
    <tspan id="tspan2166" sodipodi:role="line" x="22.52166"  
y="98.251404">Edad:</tspan>  
    </text>  
    <text id="campoNombres" style="font-size:14px;font-style:normal;font-  
weight:normal;fill:#000000;fill-opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-  
linejoin:miter;stroke-opacity:1;font-family:Bitstream Vera Sans" x="48.466324" xml:space="preserve"  
y="46.208233">  
    <tspan id="tspan2170" sodipodi:role="line" x="48.466324"  
y="46.208233">#####</tspan>  
    </text>  
    <text id="campoApellidos" style="font-size:14px;font-style:normal;font-  
weight:normal;fill:#000000;fill-opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-
```


Capítulo 3: Construcción y Elaboración del Sistema

```
linejoin:miter;stroke-opacity:1;font-family:Bitstream Vera Sans" x="49.497471" xml:space="preserve"
y="80.190697">
    <tspan id="tspan2174" sodipodi:role="line" x="49.497471"
y="80.190697">#####</tspan>
    </text>
    <text id="campoEdad" style="font-size:14px;font-style:normal;font-
weight:normal;fill:#000000;fill-opacity:1;stroke:none;stroke-width:1px;stroke-linecap:butt;stroke-
linejoin:miter;stroke-opacity:1;font-family:Bitstream Vera Sans" x="49.349102" xml:space="preserve"
y="116.22065">
    <tspan id="tspan2178" sodipodi:role="line" x="49.349102"
y="116.22065">##</tspan>
    </text>
    <rect height="125.25892" id="rect2180" style="fill-
opacity:0.51955307;fill:#808080" width="111.62186" x="210.11172" y="23.622158"/>
    </g>
</svg>
</diseño>
</plantillacompilada>
```

Capítulo 3: Construcción y Elaboración del Sistema

A continuación se muestra la interfaz de la aplicación que permitirá la personalización y la impresión de los documentos de identificación:

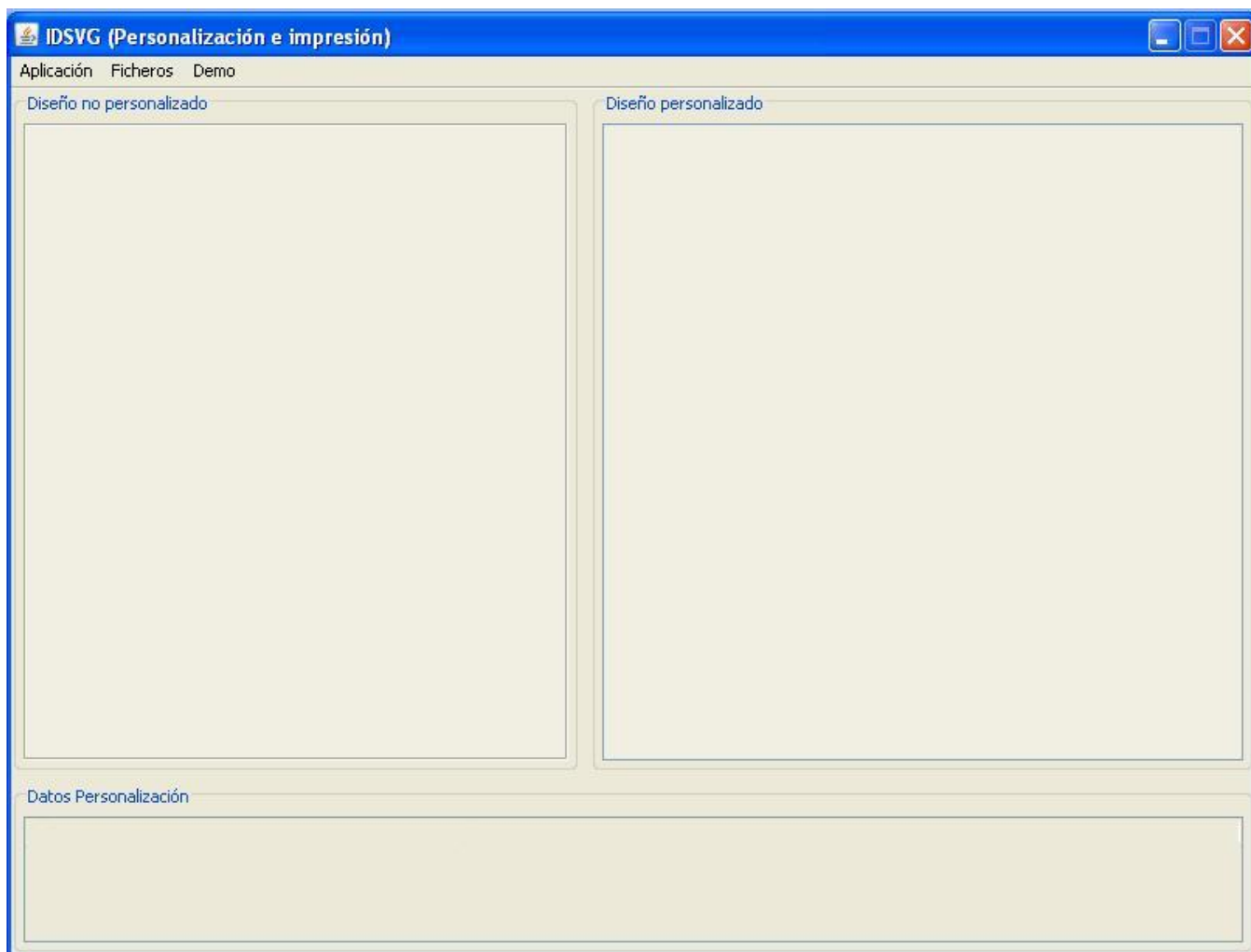


Figura 19. Formulario principal de la personalización e impresión.

Lo primero que se hace es cargar la plantilla compilada, cuando se hace esa operación se muestra el diseño SVG sin personalizar en la parte izquierda del formulario, luego se pasa a cargar los datos de la personalización que vienen en el XML de personalización, ese XML se valida contra el esquema que posee la plantilla anteriormente cargada, mostrando el sistema un mensaje, en caso que estén bien se muestra un mensaje que los datos han sido validados y se muestran los datos de la parte inferior del formulario, en caso contrario el sistema muestra un mensaje de error, a continuación se muestra la interfaz para un mayor entendimiento:

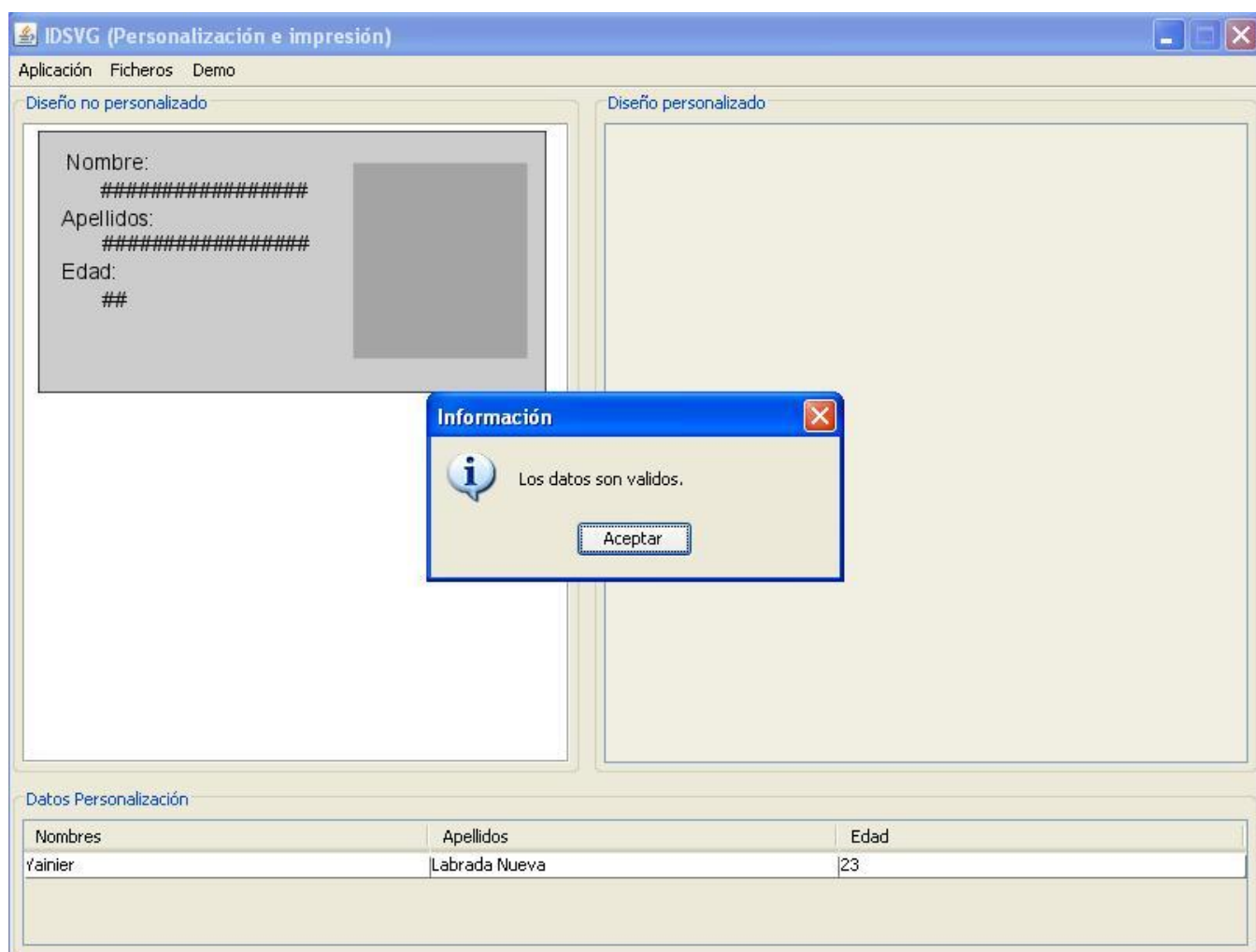


Figura 20. Figura que muestra la validación de los datos de la personalización.

Después de haber validado y mostrado los datos de la personalización, se personaliza el documento SVG y ahí mismo el sistema muestra un mensaje si quiere imprimir o no el documento de identificación, en caso de que no lo quiera imprimir, el usuario, que en este caso es el administrador de impresión de documentos de identificación tiene la opción de cancelar en caso de que lo quiera imprimir lo imprime, a continuación se muestra la imagen:

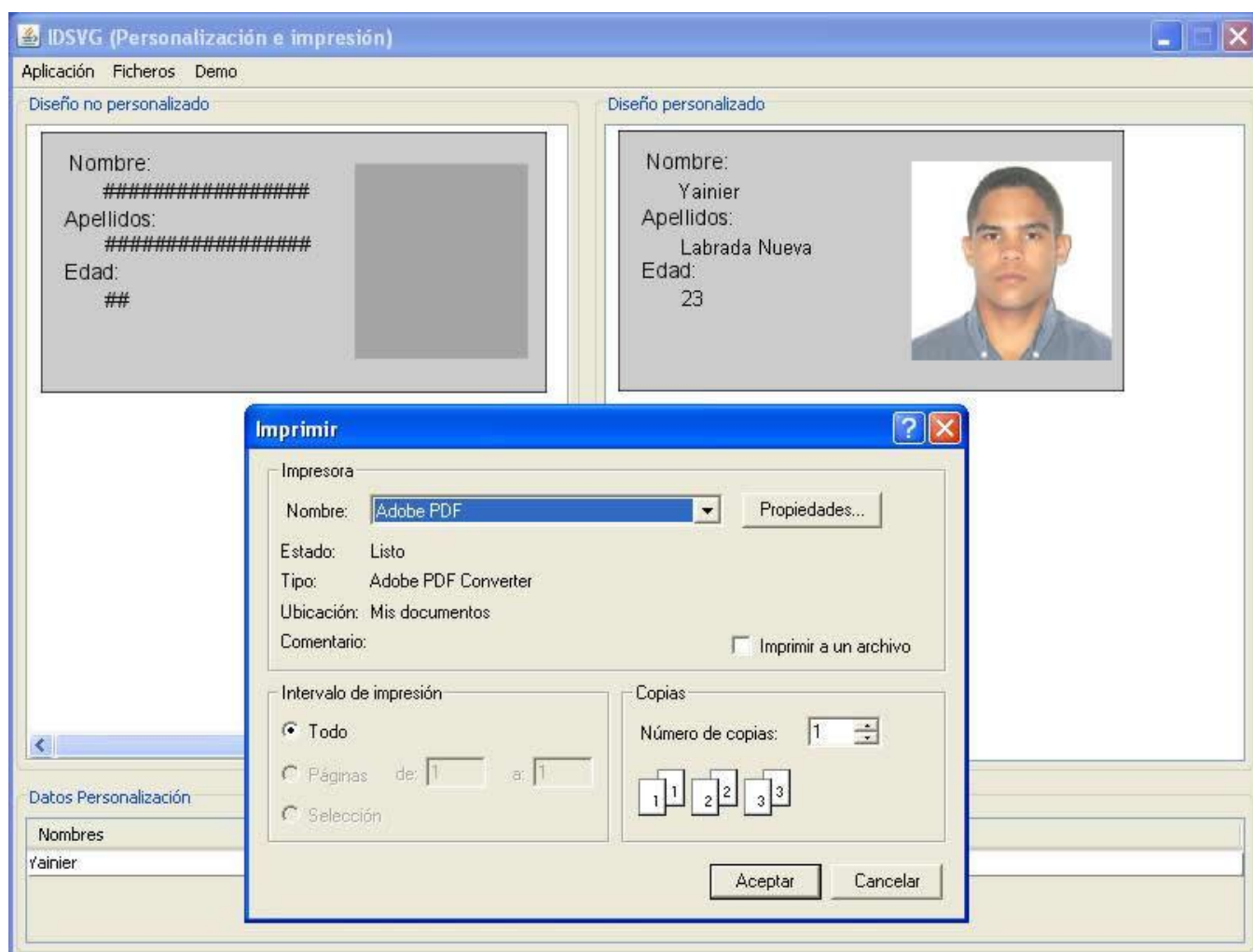


Figura 21. Personalización e impresión de los documentos.

3.6 Arquitectura

Para la definición de la arquitectura de esta propuesta se pensó en el Modelo N Capas el cual utiliza como uno de sus principios, el desarrollo de aplicaciones basadas en componentes. Este tipo de arquitectura presenta varios beneficios:

- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo (es más fácil cambiar un componente que una aplicación monolítica).
- Mayor flexibilidad (se le pueden añadir más componentes que aumenten las funcionalidades del sistema).
- Soluciones altamente flexibles y reutilizables.
- Alta escalabilidad.

Capítulo 3: Construcción y Elaboración del Sistema

La arquitectura usada propone una librería común llamada IDSVG.Comun que contiene las clases que usan las dos aplicaciones IDSVGDesigner e IDSVGPersonalizacion.

3.8 Diagrama de despliegue

El diagrama de despliegue representa cómo va a estar distribuido el sistema de forma física. Se utilizan dos computadoras una donde estará la parte de la aplicación cuya responsabilidad es diseñar plantilla con su diseñador de plantillas y en la otra se encontrará la aplicación de la personalización y la impresión de documentos de identificación con su administrador de impresión de documentos y plantillas, se utiliza una impresora para llevar a cabo la impresión de los documentos de identificación. Los nodos entre los diagramas se conectan con asociaciones de comunicación entre las que pueden estar: enlaces de red, conexiones TCP/IP, microondas. En esta propuesta no se puso un protocolo de impresión en específico, ya que puede ser utilizado cualquiera de ellos, ya sea USB, conexiones por puerto serie o paralelo. La vista representa la distribución de las instancias de componentes en ejecución en dos nodos que se encuentran conectados por enlaces de comunicación. Un nodo es un recurso en ejecución como puede ser un computador, un dispositivo, un procesador o memoria. Esto se identifica utilizando estereotipos.

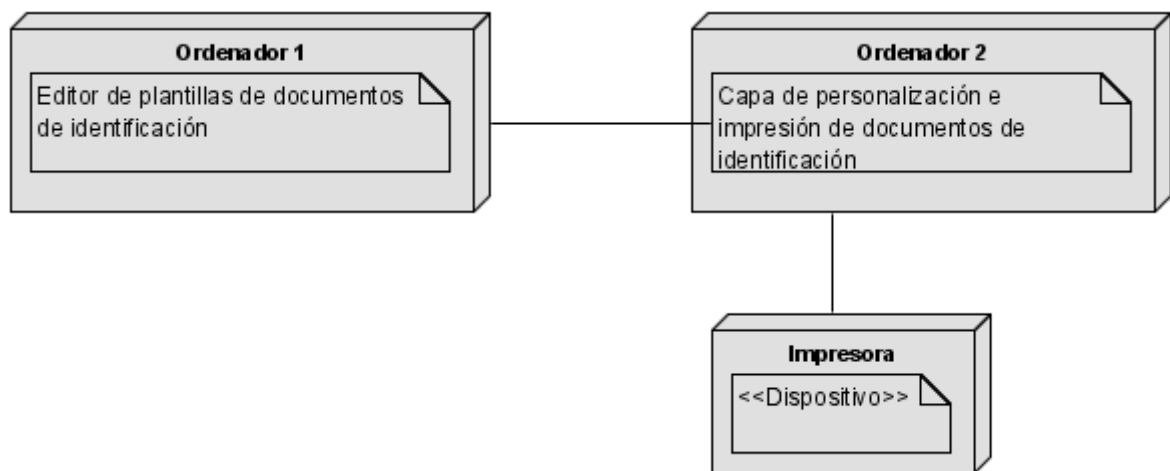


Figura 22. Diagrama de despliegue

3.9 Diagrama de componentes

Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. Un diagrama de

Capítulo 3: Construcción y Elaboración del Sistema

componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, en tiempo de enlace, en tiempo de ejecución y otros en varias de éstas. Los componentes tienen dos características: empaquetan el código que implementa la funcionalidad de un sistema y también empaqueta algunas de sus propias instancias de objetos que constituyen el estado del sistema.

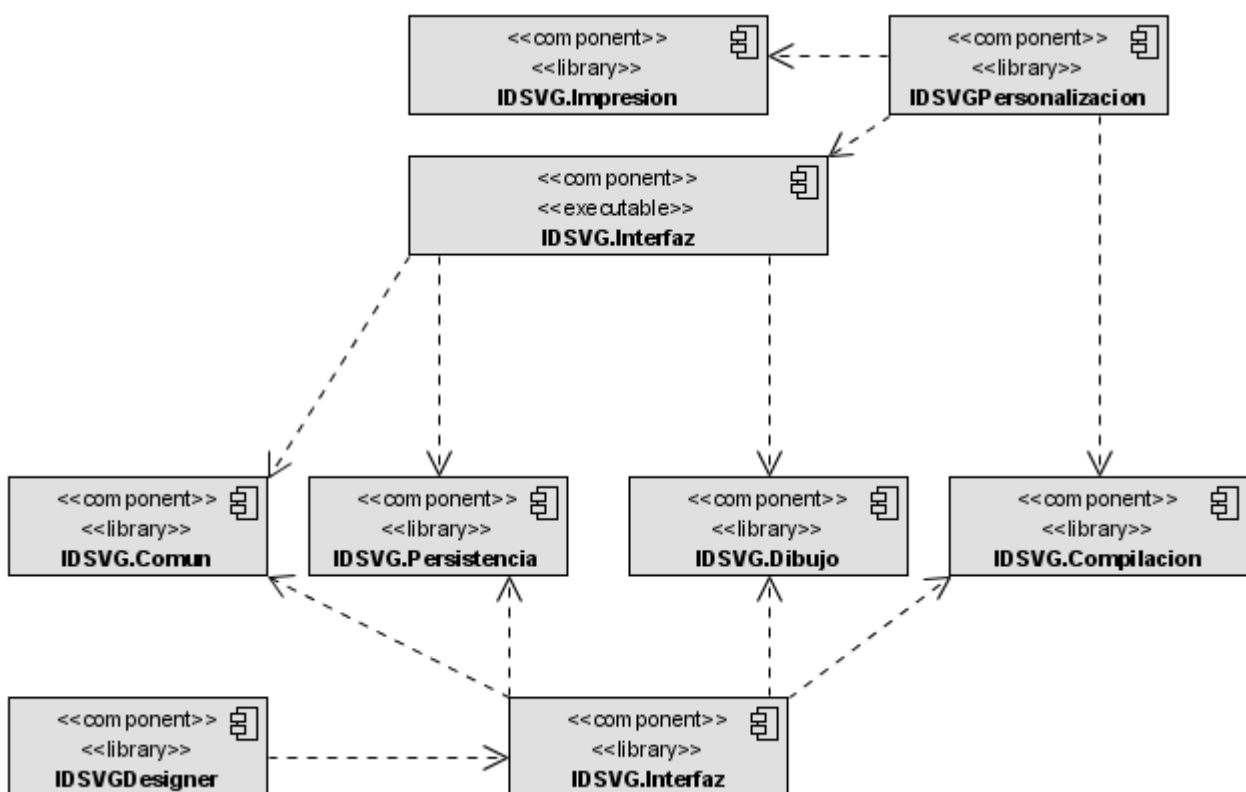


Figura 23. Diagrama de componentes

A continuación se describen cada uno de los paquetes que conforman el diagrama de componentes:

IDSVGDesigner: Paquete referente a la aplicación del diseñador de plantillas.

IDSVGPersonalizacion: Paquete referente a la aplicación de personalización e impresión.

IDSVG.Dibujo: Clases cuyas responsabilidades son el dibujo del grafico SVG.

IDSVG.Impresion: Clases que conforman la capa de impresión.

IDSVG.Persistencia: Clases que permitirán persistir las plantillas, generarlas y salvarlas en disco.

IDSVG.Compilacion: Clases que se encargaran de compilar la plantilla no compilada y de validar contra esquemas de datos los ficheros XML de la personalización.

Capítulo 3: Construcción y Elaboración del Sistema

IDSVG.Comun: Clases que conforman las plantillas.

Existen dos paquetes **IDSVG.Interfaz**, cada paquete pertenece a cada una de las aplicaciones.

3.10 Conclusiones

En el presente capítulo se realizó el análisis y diseño de la solución propuesta. Para un mayor entendimiento se expusieron los diagramas de clases del análisis, los diagramas de clases del diseño; describiéndose cada una de sus clases y los diagramas de secuencia (Diagrama de Secuencia por Contrato de Paquetes), al igual que se da una breve explicación de cómo funciona el sistema mostrándose cada una de sus interfaces.

Se hizo la representación en un diagrama los componentes a ser implementados. Además del Diagrama de Despliegue que brinda información acerca de cómo estará físicamente distribuido el sistema, se hizo una descripción de la arquitectura utilizada en el desarrollo de la aplicación y de las cuáles provienen las ventajas que hicieron que se seleccionaran para esta propuesta.

CONCLUSIONES

- En el presente trabajo, se realizó la implementación de componentes y de utilitarios para la impresión de documentos de identificación en software libre.
- Se estudió y analizó las características de las principales herramientas de diseño e impresión de documentos de identificación existentes en la actualidad a nivel mundial. Se enfatizó en el estudio del formato SVG, buscando las ventajas de su utilización y se profundizó en los principales editores SVG existentes, haciéndose un resumen de cada uno de ellos.
- Luego de un análisis de las tecnologías más empleadas en la actualidad para el desarrollo de sistemas informáticos similares, se propone utilizar Java como lenguaje de programación, en la plataforma J2EE.
- Se llegó a la conclusión de que la metodología idónea para llevar a cabo el proceso de desarrollo es RUP (Proceso Unificado de Rational).
- No se pudieron identificar bien los procesos que ocurren en el negocio, por lo que se modeló el mismo a través de un Modelo de Dominio, en el cual se mostraron los principales conceptos al usuario para lograr un mejor entendimiento del éste.
- Se definieron los requerimientos del sistema, tanto funcionales como no funcionales y se elaboró el Modelo de Casos de Uso del Sistema, describiéndose cada Caso de Uso para propiciar un mejor conocimiento de las funcionalidades que brindan.
- Se diseñó el sistema a través de diagramas de clases de análisis, diagramas de clases de diseño y diagramas de interacción.
- Tras el ejercicio de análisis realizado se puede concluir que se llevó a cabo el análisis y diseño al igual que la implementación de los componentes y utilitarios para la impresión de documentos de identificación en Software Libre, cumpliéndose todos los objetivos generales, específicos conjuntamente con todas las tareas trazadas.

RECOMENDACIONES

- Agregarle al Inkscape todas las funcionalidades expuestas en la propuesta de solución.
- Hacerle pruebas de caja blanca y caja negra a la herramienta propuesta.
- Desplegar la herramienta en oficinas e instituciones donde se desarrollen procesos de personalización e impresión de documentos de identificación.
- Crear una capa intermedia donde se especifiquen como conectar la aplicación a distintos gestores de base de datos de forma tal que permita obtener los datos de la personalización desde en ficheros XML.

BIBLIOGRAFÍA CITADA

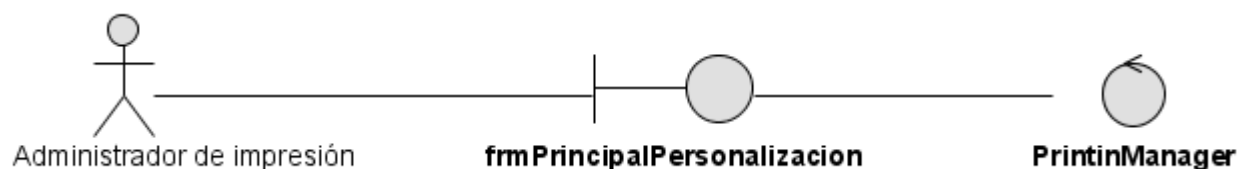
1. Autores, c. d. (2000) About Component-Based Development. Disponible en:
<http://webcabcomponents.com/componentization.shtml>
Software, S. (2000). Component Based Development and Object Modeling.
2. Enciclopedia Wikipedia. "Software libre". [Consultado en enero del 2008]
Disponible en World Wide Web: http://es.wikipedia.org/wiki/Software_libre
3. Stallman, R. Porqué "Software Libre" es mejor que software de "Código Fuente Abierto".
[Consultado en enero del 2008]. Disponible en World Wide Web:
<http://www.gnu.org/philosophy/free-software-for-freedom.es.html>.
4. Ciberaula. (2006). "Java (J2EE)." Retrieved 15/03/2008, 2008, from
http://www.ciberaula.com/curso/java2/que_es/
5. Monografías.com. (2005). "Introducción a Java." Retrieved 31/01/2007, 2007, from
<http://www.monografias.com/trabajos/java/java.shtml>.
6. Foundation, NetBeans. [Online] <http://es.wikipedia.org/wiki/NetBeans>.
7. Foundation, XSL. [Online] <http://es.wikipedia.org/wiki/XSL>.
8. Foundation, XSLT. [Online] <http://es.wikipedia.org/wiki/XSLT>.
9. Sanchez., M. A. M. (2004) "Metodologías de Desarrollo de Software " Grupo Informatizate.
Volume, DOI:
10. Yoandro Hechevarría Toranzo, A. F. D. (2006). Sistema Automatizado para la Planificación
Material y Financiera del MINFAR. Facultad de Ingeniería Industrial Centro de Estudios de
Ingeniería de Sistemas (CEIS). Ciudad de la Habana., Instituto Superior Politécnico "José
Antonio Echeverría". 99.
11. Larman, C. (2004). UML y Patrones. Introducción al análisis y diseño orientado a objetos. La
Habana, Félix Varela.

BIBLIOGRAFÍA CONSULTADA

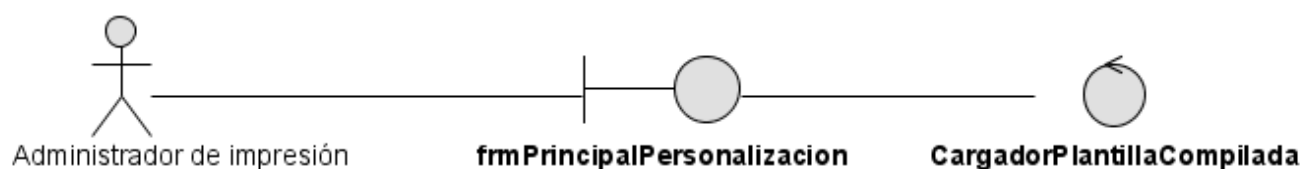
1. Universidad de las Ciencias Informáticas, Conferencias de Ingeniería de Software I, 2008.
2. Universidad de las Ciencias Informáticas, Conferencias de Ingeniería de Software II, 2008.
3. Rumbaugh, J.; Jacobson, I. y Booch, G.; “El Lenguaje Unificado de Modelado”. Manual de referencia. 2000. Addison-Wesley. Páginas 120-121, 157-162, 305-312.
4. Jacobson, I.; Booch, G. y Rumbaugh, J.; “El Proceso Unificado de Desarrollo de software”. 2000. Addison-Wesley., Apéndice A. Visión General de UML, Apéndice B.
5. Jacobson, I.; Booch, G. y Rumbaugh, J.; “El Proceso Unificado de Desarrollo de software”. 2000. Addison-Wesley. Páginas 115-119.
6. Pressman, Roger; Ingeniería de software. Un enfoque práctico. 2002. McGraw.Hill/Interamericana de España.
7. Foundation, XSL. [Online] <http://es.wikipedia.org/wiki/XSL>.
8. Foundation, XSLT. [Online] <http://es.wikipedia.org/wiki/XSLT>.
9. O'Reilly. XSLT. 2001. p. 478. 0-596-00053-7.
10. Foundation, SVG. [Online] <http://es.wikipedia.org/wiki/SVG>.
11. Foundation, Java. [Online] <http://es.wikipedia.org/wiki/Java>.
12. Foundation, NetBeans. [Online] <http://es.wikipedia.org/wiki/NetBeans>.
13. Foundation, Sodipodi. [Online] <http://es.wikipedia.org/wiki/Sodipodi>.
14. Foundation, Inkscape. [Online] <http://es.wikipedia.org/wiki/Inkscape>.

ANEXOS

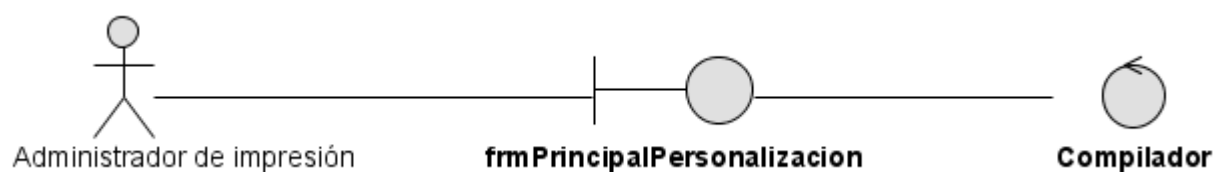
ANEXO I. Diagrama de clases del análisis del caso de uso: Imprimir documento.



ANEXO II. Diagrama de clases del análisis del caso de uso: Cargar plantilla compilada.



ANEXO III. Diagrama de clases del análisis del caso de uso: Personalizar documento SVG.



ANEXO IV. Diagrama de clases del análisis del caso de uso: Cargar datos personalización..



GLOSARIO DE TÉRMINOS

Aqua: es el nombre comercial de la apariencia de la interfaz gráfica de usuario del sistema operativo Mac OS X de Apple Computer.

Bitmaps: es una estructura o fichero de datos que representan generalmente una rejilla rectangular de puntos de color.

CSS: el lenguaje de hojas de estilo en cascada (Cascading Style Sheets) utilizadas para definir la presentación de un documento HTML o XML.

DOM: Modelo de objetos del documento (Document Object Model).

HTML: es el acrónimo inglés de HyperText Markup Language, que se traduce al español como Lenguaje de Marcas Hipertextuales. Es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

Mac OS X: es un sistema operativo basado en UNIX, pero donde el gestor de ventanas X11, característico de estos sistemas, ha sido sustituido por otro denominado Aqua, desarrollado íntegramente por Apple.

MathML o Mathematical Markup Language: es un lenguaje de marcado basado en XML, cuyo objetivo es expresar notación matemática de forma que distintas máquinas puedan entenderla, para su uso en combinación con XHTML en páginas Web, y para intercambio de información entre programas de tipo matemático en general.

Metalenguaje: Lenguaje usado para hacer referencia a otros lenguajes.

Open Source: (Código fuente abierto) un programa que ofrece al usuario la posibilidad de entrar en su interior para poder estudiarlo o modificarlo, libre acceso al código fuente.

PNG (Portable Network Graphics): es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps no sujeto a patentes. Este formato fue desarrollado en buena parte para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de contraste y otros importantes datos.

Personalización: Significa llenar el documento SVG con los datos de las personas.

SGML: Standard Generalized Markup Language o Lenguaje estándar genérico por etiquetas, estándar universal para la escritura de archivos de hipertexto en la Internet.

UNIX: UNIX es un sistema operativo portable, multitarea y multiusuario.

W3C: el World Wide Web Consortium, abreviado W3C, es un consorcio internacional que produce estándares para la World Wide Web.

XHTML: acrónimo inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las

páginas Web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.

XML: eXtensible Markup Language: Es un metalenguaje extensible de etiquetas, se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.