Universidad de las Ciencias Informáticas (UCI)

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

Título: Análisis y Diseño de una herramienta de diseño de plantillas para la personalización de documentos de identificación.

Autores:
Abel Espinosa Cañive
Rosdanys Ramírez Arcas
Tutores:
Ing. Pedro Enrique Novales Hernández
MSc. Yurdik Cervantes Mendoza

Ciudad de La Habana, Junio 2009

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado:

Análisis y Diseño de una herramienta de diseño de plantillas para la personalización de documentos de identificación.

Y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Abel Espinosa Cañive	Rosdanys Ramírez Arcas
MSc. Yurdik Cervantes Mendoza	Ing. Pedro Enrique Novales Hernández

Agradecimientos

Abel

A mis padres y mis hermanos que siempre me han apoyado en todas mis decisiones y por ser mi inspiración eterna y por saberme enseñar el camino a seguir en la vida.

A mi hermano Osbel que ha sabido estar ahí cuando me hace falta, siempre vas a ser un ejemplo para mí.

A Yurdik y Pedro, nuestros tutores, por guiarnos en todo momento, por sus consejos y ayuda siempre útiles.

A Rosdanys, por ser tan buen colega y amigo, y aguantar todas las pesadeces que le digo.

A Liudnet, por ser mis ojos cuando no vi como iniciar este trabajo.

A Dolennis y Lourdes por ser tan buenas conmigo y ofrecerme su ayuda tan desinteresadamente y saber que siempre he podido contar con ellas.

A mis tíos Tata, Eloy, Cary y Pucha, por preocuparse siempre por mí todos estos años.

A todos mis compañeros de grupo, los que están ahora y los que una vez estuvieron. A los que se fueron antes de llegar a 5to, y que hicieron que conociera otras personas, costumbres y cosas.

A todos los que de una forma u otra me apoyaron para realizar este trabajo.

¡Muchas Gracias!

Rosdanys

A toda mi familia por el apoyo que me han brindaron todos estos años. En especial a mi mamá por ser como es y hacerme sentir lo que soy.

A todas las personas que he conocido en estos 5 años de universidad, en especial a mis compañeros del grupo.

A todos los amigos que conviven conmigo en la residencia.

A Abel por ser un compañero y amigo en todo momento en los bueno y malos.

A Yadira por ser mi amiga favorita.

A Juan Carlos Céspedes Capote por ser mi único amigo, guiarme y enseñarme el camino durante estos 5 años.

Dedicatoria

Abel

A mi mamá y mi papá, que tanto me han dado en esta vida y se han sacrificado tanto para que llegase de la mejor manera a este día.

A mis hermanos que son los mejores hermanos que pueda haber deseado tener.

Rosdanys

A toda mi familia y amigos y a todos los que han convivido conmigo estos últimos 5 años.

A mi amigo Juan Carlos Céspedes Capote por haberme ayudado en todo lo que necesitaba.

Resumen

Un desafío en la informatización de la sociedad es la implantación de documentos que identifiquen a sus portadores. En el mundo el uso de documentos de identificación es regido por estándares para proteger los datos y evitar que se cometan falsificaciones. Sin embargo, aún se utilizan documentos que no se rigen por estos estándares o no se ajustan a sus últimas especificaciones.

En este trabajo se realiza el análisis y diseño de una herramienta para el diseño de plantillas de documentos de identificación para la personalización, que utilice los estándares internacionales para la creación de este tipo de documentos y haga uso de las tecnologías libres de desarrollo. Para ello se presenta el estado del arte sobre las actuales tecnologías y herramientas utilizadas en el proceso de creación de credenciales o documentos de identificación en el mundo. Se describe la propuesta de análisis y diseño de la herramienta de diseño de plantillas de documentos de identificación y se analizan críticamente las herramientas para el desarrollo de la misma. Finalmente se describen los pasos necesarios para la elaboración de la propuesta y se elabora un prototipo.

Índice

Introducción 7
Capítulo 1 - Fundamentación Teórica 14
1.1. Conceptos relacionados 14
1.1.1. Documento de identificación 14
1.2. Estándar Internacional ISO/IEC 7810 14
1.2.1. ID-1 15
1.2.2. ID-2 15
1.2.3. ID-3 16
1.2.4. ID-000 16
1.3. Estándares Internacionales de la Organización de la Aviación Civil Internacional (OACI) 16
1.4. Estado del Arte 17
1.5. Metodologías de Desarrollo de Software 18
1.5.1. XP (eXtreme Programming) 18
1.5.2. SCRUM 19
1.5.3. Crystal Methodologies 20
1.5.4. RUP 20
1.6. Lenguaje de Modelado UML 22
1.7. Herramientas para el modelado 24
1.7.1. Rational Rose 24
1.7.2. Visual Paradigm 25
1.8. Tecnologías de desarrollo 26
1.8.1. XML 26
1.8.2. SVG 28
1.8.3. Análisis de Plataformas de Desarrollo 30
1.9. Herramientas de Desarrollo 34
1.9.1. NetBeans 34
1.9.2. Eclipse 35
1.10. Elección de las principales herramientas a utilizar. 36

1.11. Conclusiones Parciales del Capítulo 37
Capítulo 2 - Características del Sistema 38
2.1. Descripción de los procesos involucrados en la personalización 38
2.2. Análisis de otras soluciones 39
2.2.1. DataCard® 39
2.2.2. Grupo Mühlbauer 40
2.2.3. XYMA Emipas 41
2.2.4. CredentialBuilder 41
2.2.5. IDSVG 41
2.2.6. CardFive 42
2.3. Solución Propuesta 42
2.4. Modelo de Dominio 44
2.4.1. Conceptos asociados al dominio del problema 44
2.4.2. Diagrama de Clases del Modelo de Dominio 46
2.5. Especificación de los requisitos de software 46
2.5.1. Requerimientos Funcionales 46
2.5.2. Requerimientos No Funcionales 48
2.5.3. Definición de los casos de uso 49
2.5.4. Breve Descripción de los Casos de Uso 50
2.5.5. Diagrama de Casos de Uso del Sistema 56
2.6. Descripción de los Casos de Uso del Sistema 57
2.7. Conclusiones Parciales del Capítulo 73
Capítulo 3 - Análisis y Diseño del Sistema 74
3.1. Análisis y Diseño 74
3.1.1. Definición del Modelo de Análisis 75
3.1.2. Diagrama de Clases de Análisis 75
3.1.3. Arquitectura 76
3.1.4. Patrones de Diseño 77
3.1.5. Definición del Modelo de Diseño 78
3.1.6. Diagramas de Interacción 78

3.1.7. Diagrama de Clases del Diseño 80
3.1.8. Descripción de las Principales Clases Utilizadas 87
3.1.9. Interfaz 99
3.1.10. Tratamiento de errores 101
3.1.11. Concepción de la ayuda 102
3.2. Conclusiones Parciales del Capítulo 103
Conclusiones Generales 104
Recomendaciones 105
Bibliografía 106
Anexos 109
Anexo 1 109
Representación Gráfica de los Diagramas de Clases del Análisis 109
Anexo 2 112
Diagramas de Secuencia 112
Anexo 3 122
Representación Gráfica de las Clases del Diseño 122

Índice de Figuras

Figura 1: Ejemplo de campos personalizables en el DNI electrónico español. (Fuente:(Policía)) 39
Figura 2: Servicios disponibles que ofrece el grupo DataCard®. (Fuente: (DataCard)) 40
Figura 3: La figura muestra el Diagrama de Clases del Modelo de Dominio. 46
Figura 4: La figura muestra el Diagrama de Casos de Uso del Sistema. 56
Figura 5: Representación del Caso de Uso Crear Plantilla. 57
Figura 6: Representación del Caso de Uso Abrir Plantilla. 58
Figura 7: Representación del Caso de Uso Guardar Plantilla. 59
Figura 8: Representación del Caso de Uso Crear Objeto. 60
Figura 9: Representación del Caso de Uso Copiar Objeto. 61
Figura 10: Representación del Caso de Uso Cortar Objeto. 62
Figura 11: Representación del Caso de Uso Pegar Objeto. 63
Figura 12: Representación del Caso de Uso Mover Objeto. 64
Figura 13: Representación del Caso de Uso Eliminar Objeto. 65
Figura 14: Representación del Caso de Uso Editar Codificación. 66
Figura 15: Representación del Caso de Uso Mostrar Vista Previa. 67
Figura 16: Representación del Caso de Uso Especificar Elemento de Seguridad. 68
Figura 17: Representación del Caso de Uso Gestionar Estándar. 69
Figura 18: Representación del Caso de Uso Asociar Propiedades. 72
Figura 19: Diagrama de Clases del Modelo de Análisis. Casos de Uso: Crear Plantilla, Abrir Plantilla y Guardar Plantilla. 75
Figura 20: Representación gráfica de la arquitectura del sistema. 77
Figura 21: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Crear Plantilla. 80
Figura 22: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Abrir Plantilla. 80
Figura 23: Representación gráfica del Diagrama de Clases del Diseño de la aplicación. 81
Figura 24: Representación gráfica del Diagrama de Clases del Diseño: Crear, Abrir y Guardar Plantilla. 81
Figura 25: Representación gráfica del Diagrama de Clases del Diseño: Crear Objeto. 82
Figura 26: Representación gráfica del Diagrama de Clases del Diseño: Copiar, Cortar, Pegar y Eliminar Objeto. 83
Figura 27: Representación gráfica del Diagrama de Clases del Diseño: Editar Codificación. 83
Figura 28: Representación gráfica del Diagrama de Clases del Diseño: Gestionar Estándar. 84
Figura 29: Representación gráfica del Diagrama de Clases del Diseño: Especificar Elemento de Seguridad. 85
Figura 30: Representación gráfica del Diagrama de Clases del Diseño Mostrar Vista Previa. 85
Figura 31: Representación gráfica del Diagrama de Clases del Diseño: Asociar Propiedades. 86
Figura 32: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Crear Objeto. 109
Figura 33: Diagrama de Clases del Modelo de Análisis. Casos de Uso: Copiar Objeto, Cortar Objeto, Pegar Objeto y Eliminar Objeto. 110
Figura 34: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Editar Codificación. 110

Figura 35: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Mostrar Vista Previa. 110
Figura 36: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Especificar Elemento de Seguridad. 111
Figura 37: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Gestionar Estándar. 111
Figura 38: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Asociar Propiedades. 111
Figura 39: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Guardar Plantilla. ..112
Figura 40: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Guardar Plantilla. (Curso Alterno 1) 112
Figura 41: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Guardar Plantilla. (Curso Alterno 2) 113
Figura 42: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Crear Objeto. 113
Figura 43: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Copiar Objeto. 113
Figura 44: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Cortar Objeto. 114
Figura 45: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Pegar Objeto. 114
Figura 46: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Mover Objeto. 115
Figura 47: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Eliminar Objeto. 115
Figura 48: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Editar Configuración. 116
Figura 49: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Mostrar Vista Previa. 116
Figura 50: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Especificar Elemento de Seguridad. 117
Figura 51: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Especificar Elemento de Seguridad. (Curso Alterno) 117
Figura 52: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Agregar). 118
Figura 53: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Modificar). 119
Figura 54: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Eliminar). 120
Figura 55: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Eliminar). (Curso Alterno) 120
Figura 56: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Asociar Propiedades. 121
Figura 57: Representación gráfica de la clase CardDesigner. 122
Figura 58: Representación gráfica de la clase DesignerView. 122
Figura 59: Representación gráfica de la clase MainForm. 123
Figura 60: Representación gráfica de la clase IDesignableComponent. 124
Figura 61: Representación gráfica de la clase PropertiesView. 124
Figura 62: Representación gráfica de la clase ToolBox. 124
Figura 63: Representación gráfica de la clase ToolsView. 124

Figura 64: Representación gráfica de la clase PropertyEditor. 125
Figura 65: Representación gráfica de la clase Ruler. 125
Figura 66: Representación gráfica de la clase Card. 127
Figura 67: Representación gráfica de la clase CardObject. 129
Figura 68: Representación gráfica de la clase ImageDraw. 130
Figura 69: Representación gráfica de la clase TextDraw. 130
Figura 70: Representación gráfica de la clase LineDraw. 131
Figura 71: Representación gráfica de la clase PencilDraw. 131
Figura 72: Representación gráfica de la clase PolygonDraw. 131
Figura 73: Representación gráfica de la clase RectangleDraw. 132
Figura 74: Representación gráfica de la clase TextFieldDraw. 132
Figura 75: Representación gráfica de la clase EllipseDraw. 132
Figura 76: Representación gráfica de la clase CardBeanInfo. 132
Figura 77: Representación gráfica de la clase CardObjectBeanInfo. 133
Figura 78: Representación gráfica de la clase ImageDrawBeanInfo. 133
Figura 79: Representación gráfica de la clase TextDrawBeanInfo. 133
Figura 80: Representación gráfica de la clase LineDrawBeanInfo. 133
Figura 81: Representación gráfica de la clase PencilDrawBeanInfo. 133
Figura 82: Representación gráfica de la clase PolygonDrawBeanInfo. 133
Figura 83: Representación gráfica de la clase RectangleDrawBeanInfo. 134
Figura 84: Representación gráfica de la clase TextFieldDrawBeanInfo. 134
Figura 85: Representación gráfica de la clase EllipseDrawBeanInfo. 134

Índice de Tablas

Tabla 1: Operacionalización de las Variables. 11
Tabla 2: Codificación de colores de la metodología Crystal Methodologies 20
Tabla 3: Requerimientos Funcionales del Sistema. 48
Tabla 4: Requerimientos No Funcionales del Sistema. 49
Tabla 5: Definición y descripción de los actores del sistema. 50
Tabla 6: Definición y descripción del Casos de Uso del Sistema Crear Plantilla. 50
Tabla 7: Definición y descripción del Casos de Uso del Sistema Abrir Plantilla. 50
Tabla 8: Definición y descripción del Casos de Uso del Sistema Guardar Plantilla. 51
Tabla 9: Definición y descripción del Casos de Uso del Sistema Crear Objeto. 51
Tabla 10: Definición y descripción del Casos de Uso del Sistema Copiar Objeto. 51
Tabla 11: Definición y descripción del Casos de Uso del Sistema Cortar Objeto. 52
Tabla 12: Definición y descripción del Casos de Uso del Sistema Pegar Objeto. 52
Tabla 13: Definición y descripción del Casos de Uso del Sistema Mover Objeto. 52
Tabla 14: Definición y descripción del Casos de Uso del Sistema Eliminar Objeto. 53
Tabla 15: Definición y descripción del Casos de Uso del Sistema Editar Configuración. 53
Tabla 16: Definición y descripción del Casos de Uso del Sistema Mostrar Vista Previa. 53
Tabla 17: Definición y descripción del Casos de Uso del Sistema Especificar Elemento de Seguridad. 54
Tabla 18: Definición y descripción del Casos de Uso del Sistema Gestionar Estándar. 54
Tabla 19: Definición y descripción del Casos de Uso del Sistema Asociar Propiedades. 55
Tabla 20: Descripción del Casos de Uso Crear Plantilla. 58
Tabla 21: Descripción del Casos de Uso Abrir Plantilla. 59
Tabla 22: Descripción del Casos de Uso Guardar Plantilla. 60
Tabla 23: Descripción del Casos de Uso Crear Objeto. 61
Tabla 24: Descripción del Casos de Uso Copiar Objeto. 61
Tabla 25: Descripción del Casos de Uso Cortar Objeto. 62
Tabla 26: Descripción del Casos de Uso Pegar Objeto. 63
Tabla 27: Descripción del Casos de Uso Mover Objeto. 65
Tabla 28: Descripción del Casos de Uso Eliminar Objeto. 65
Tabla 29: Descripción del Casos de Uso Editar Codificación. 67
Tabla 30: Descripción del Casos de Uso Mostrar Vista Previa. 68
Tabla 31: Descripción del Casos de Uso Especificar Elemento de Seguridad. 69
Tabla 32: Descripción del Casos de Uso Gestionar Estándar. 72
Tabla 33: Descripción del Casos de Uso Asociar Propiedades. 73
Tabla 34: Descripción de la Clase del Diseño: CardDesigner. 87
Tabla 35: Descripción de la Clase del Diseño: ToolsView. 87
Tabla 36: Descripción de la Clase del Diseño: DesignerView. 87
Tabla 37: Descripción de la Clase del Diseño: PropertiesView. 88
Tabla 38: Descripción de la Clase del Diseño: MainForm. 88
Tabla 39: Descripción de la Clase del Diseño: EstandarView. 88

Tabla 40: Descripción de la Clase del Diseño: Card....89
Tabla 41: Descripción de la Clase del Diseño: CardObject.....89
Tabla 42: Descripción de la Clase del Diseño: ImageDraw.....90
Tabla 43: Descripción de la Clase del Diseño: TextDraw.....90
Tabla 44: Descripción de la Clase del Diseño: LineDraw.....90
Tabla 45: Descripción de la Clase del Diseño: EllipseDraw.....91
Tabla 46: Descripción de la Clase del Diseño: PolygonDraw....91
Tabla 47: Descripción de la Clase del Diseño: RectangleDraw.....91
Tabla 48: Descripción de la Clase del Diseño: PencilDraw.....92
Tabla 49: Descripción de la Clase del Diseño: ItemSecurityDraw....92
Tabla 50: Descripción de la Clase del Diseño: TextFieldDraw....92
Tabla 51: Descripción de la Clase del Diseño: BarCodeDraw.....93
Tabla 52: Descripción de la Clase del Diseño: ImageDrawBeanInfo....93
Tabla 53: Descripción de la Clase del Diseño: TextDrawBeanInfo.....93
Tabla 54: Descripción de la Clase del Diseño: LineDrawBeanInfo.....94
Tabla 55: Descripción de la Clase del Diseño: EllipseDrawBeanInfo.....94
Tabla 56: Descripción de la Clase del Diseño: PolygonDrawBeanInfo....94
Tabla 57: Descripción de la Clase del Diseño: RectangleDrawBeanInfo.....95
Tabla 58: Descripción de la Clase del Diseño: CardObjectBeanInfo.....95
Tabla 59: Descripción de la Clase del Diseño: PencilDrawBeanInfo.....95
Tabla 60: Descripción de la Clase del Diseño: ItemSecurityDrawBeanInfo....96
Tabla 61: Descripción de la Clase del Diseño: TextFieldDrawBeanInfo....96
Tabla 62: BarCodeDrawBeanInfo.....96
Tabla 63: Descripción de la Clase del Diseño: CardBeanInfo....97
Tabla 64: Descripción de la Clase del Diseño: IDesignableComponent....97
Tabla 65: Descripción de la Clase del Diseño: GestionarConfiguración.....97
Tabla 66: Descripción de la Clase del Diseño: Estándar.....98
Tabla 67: Descripción de la Clase del Diseño: Ruler.....98
Tabla 68: Descripción de la Clase del Diseño: PropertyEditor.....98
Tabla 69: Descripción de la Clase del Diseño: ToolBox.....99
Tabla 70: Descripción de la Clase del Diseño: UnitConverter.....99

Introducción

The nice thing about standards is, there are so many to choose from.

C. Northcote Parkinson

"Recordarás algo de lo que leas, bastante de lo que oigas, mucho de lo que veas, y todo lo que hagas".

*(**Harrison**)*

La identificación de las personas, ha sido un proceso que se ha ido desarrollando día a día. Ha sido utilizada en China desde al menos el siglo XIV y llega a las culturas occidentales hacia finales del siglo XIX, como una forma de control de acceso a las ciudades y países.

Uno de los mecanismos para el control de las personas es la utilización de documentos de identificación. Estos incluyen datos personales de cada individuo (nombre, sexo, foto, entre otros), y se encuentran regidos por estándares internacionales que garantizan un formato mucho más generalizado, y que se utilicen tecnologías asociadas a esos estándares para la lectura de los datos solicitados a una persona.

Dada la gran importancia que tienen los documentos de identificación para la identidad personal, porque identifican a quien lo porta y describe muchas de sus características, cada día surgen nuevas formas de crear estos documentos de una manera más innovadora, utilizando nuevos materiales y aportes tecnológicos que brindan vías más seguras ante la falsificación.

Hoy en día se ha ido incrementando el nivel de seguridad y confiabilidad en los documentos de identificación, donde el desarrollo de las TIC [1] juega un papel muy importante ante la posibilidad de la informatización de los procesos que rigen la identificación de las personas de un estado o de una institución de manera local y ante el mundo, con un nivel de seguridad que los proteja ante las falsificaciones.

Actualmente se diseñan estos documentos utilizando formatos propietarios los cuales crean dependencia tecnológica de las máquinas de personalización de su proveedor. El diseño de los documentos de identificación se solicita y se devuelve una solución, lo que no da posibilidades de cambios sin provocar otro proceso de importación de servicios y productos. Esto nos lleva a un enorme gasto de recursos, y viene a la par de que todos los tipos de documentos no necesariamente los

[1] Siglas de Tecnologías de la Informática y las Comunicaciones

diseña una misma empresa, además puede quelos documentos de viaje no estén regidos por los estándares de la Organización de la Aviación Civil Internacional (OACI[2]) o que el formato de salida de los documentos no sea compatible con las impresoras que tengamos para imprimirlos.

De aquí que se haga notable la necesidad de crear una herramienta que permita crear plantillas que apliquen dichos estándares internacionales a la hora del diseño de los documentos de identificación, que lo haga de una manera cómoda, con cierto nivel de personalización, que cree plantillas de documentos de identificación con un formato que pueda ser utilizado en la gran mayoría de las impresoras o que pueda ser fácilmente transformado a otro formato que sea compatible y que evite en algún grado la dependencia tecnológica.

Dada esta situación se plantea el siguiente **problema de investigación**:
¿Cómo mejorar el proceso de diseño de plantillas para documentos de identificación?

El **objeto de estudio** de este trabajo consiste en las herramientas de diseño de plantillas para documentos de identificación.

El **campo de acción** del presente trabajo es el diseño de plantillas para documentos de identificación desarrolladas utilizando plataforma Java.

Como **objetivo general** se tiene: Realizar el análisis y el diseño de una herramienta de diseño de plantillas para la personalización de documentos de identificación.

Los **objetivos específicos** son:

- Especificar las funcionalidades de una herramienta de confección de plantillas para la personalización de documentos de identificación
- Diseñar la herramienta de diseño de plantillas para la personalización de documentos de identificación
- Realizar prototipo de un diseñador de plantillas para la personalización de documentos de identificación.

Se propone como **hipótesis**: El análisis y diseño de una herramienta para el diseño de plantillas de documentos de identificación, facilitará la personalización de documentos de identificación.

[2] Siglas de Organización de la Aviación Civil Internacional, en inglés International Civil Aviation Organization (ICAO).

A partir de la hipótesis planteada anteriormente se define como **variable independiente** la herramienta para el diseño de plantillas de documentos de identificación y como **variable dependiente** la personalización de documentos de identificación.

A continuación se muestra el proceso de **operacionalización de las variables**:

Variables	**Dimensiones**	**Indicadores**	**Índices de los indicadores**
Herramienta para el diseño de plantillas de documentos de identificación	Factibilidad	Tiempo de Desarrollo	Extenso
			Moderado
			Breve
		Esfuerzo	Alto
			Moderado
			Despreciable
		Costo	Costoso
			Moderado
			Barato
	Rendimiento	Tiempo de respuesta	Alto
			Medio
			Bajo
	Usabilidad	Preparación previa de los usuarios	Sí
			No
Personalización de documentos de identificación	Estructuración de la Plantilla	Complejidad	Alta
			Media
			Baja
		Organización del trabajo	Eficiente
			Deficiente
		Ajuste a los estándares de la OACI	Se ajusta
			No se ajusta

Tabla 1: Operacionalización de las Variables.

Los métodos de investigación teóricos que se han aplicado en la realización de este trabajo son:

Histórico - Lógico: Coneste método podemos hacer un análisis histórico del proceso de diseños e impresión de documentos de identificación. Se utiliza este método durante el proceso

de diseño y mejoras de las funcionalidades que cumplirán con las demandas del componente de diseño de plantillas de documentos de identificación.

Hipotético - Deductivo: Este método es utilizado para comprobar experimentalmente lo que se quiere lograr a partir del problema concreto donde se plantearon objetivos específicos e hipótesis que serán resueltos en el transcurso de la investigación.

El método de investigación empírico que se ha aplicado a la realización de este trabajo es:

Método de la observación: La observación científica es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado. Este método es utilizado en este trabajo definiendo una visión del mismo y de cómo se ha planificado dirigiéndose a una meta determinada.

Teniendo en cuenta el valor metodológico y práctico de este trabajo se obtienen como **aportes prácticos**el ajuste a los estándares internacionales de diseño de documentos de identificación; permitir un ambiente ameno en la edición y configuración de los documentos de identificación y la mejora consecuente del proceso gestión de documentos de identificación.

Tareas a realizar en el transcurso de este trabajo:

- Estudio de los problemas y posibles mejoras en el actual proceso de impresión de documentos de identificación.
- Revisión bibliográfica de los estándares internacionales de impresión de documentos de identificación y selección de los que se aplicarán en el sistema a diseñar.
- Estudio de las diferentes tecnologías para la impresión.
- Interacción con diferentes impresoras de documentos de identificación para conocer las particularidades de las mismas.
- Descripción textual de todos los Casos de Uso del sistema.
- Realización de los diagramas de clases del diseño de todos los casos de uso del sistema.
- Revisión de la existencia de otras aplicaciones o soluciones similares.
- Estudio y selección de la tecnología de software libre que permite la interacción con dispositivos de impresión de alta tecnología y para el diseño de plantillas.
- Identificación de los requisitos funcionales y no funcionales del sistema a diseñar.

- Definición de aportes teóricos y prácticos e impacto social.
- Identificación y selección de herramientas a utilizar en el proceso.
- Diseño de todas las clases de la herramienta.
- Diseño de los prototipos no funcionales de la herramienta.
- Realización de todos los diagramas de secuencia de los principales escenarios de cada CUS.
- Definición de la arquitectura del sistema a diseñar.

El presente documento consta de tres capítulos:

Capítulo 1 - Fundamentación Teórica: Se presenta el estado del arte. Se plantean los principales conceptos y aspectos relacionados con el dominio del problema a resolver. Se exponen las principales tendencias tecnológicas actuales para el diseño de documentos de identificación y se escogen las herramientas que se utilizarán en este trabajo.

Capítulo 2 - Características del Sistema: En este capítulo como parte de la propuesta de solución, se hace un estudio del negocio. Se describen las características del sistema, se presenta el modelo de dominio, elaborado con los principales conceptos del dominio del problema y las relaciones que entre ellos se establecen. Se realiza la especificación de los requisitos funcionales y no funcionales del sistema, y se determinan los casos de uso y los actores. Para cada uno de los casos de uso se elabora una descripción en formato expandido.

Capítulo 3 – Análisis y Diseño del Sistema: Este capítulo contiene lo referente al análisis y diseño del sistema. Como parte de la solución, se desarrollan los diagramas de clases del análisis y del diseño para cada uno de los casos de uso. Así como los diagramas de interacción correspondientes al análisis y al diseño.

Capítulo 1 - Fundamentación Teórica

En el presente capítulo se expone la fundamentación teórica que sustenta la solución propuesta. Se abordan los principales conceptos relacionados con el dominio del problema, se hace un estudio del arte para poder comprender mejor el mismo. Se realiza el análisis de las principales tendencias tecnológica actuales y se eligen las propicias para el desarrollo del trabajo. De forma general, se ofrece una visión de los aspectos relacionados con los procesos de impresión de documentos de identificación, las herramientas y metodologías utilizadas para la realización de este software.

1.1.Conceptos relacionados

A continuación se presentan los principales conceptos relacionados con el dominio del problema y que facilitarán una mejor comprensión de todos los aspectos que se tratarán posteriormente.

1.1.1. Documento de identificación

Etimológicamente, identificación deriva del verbo latino *identificare*, vocablo integrado por los también latinos *identitas* y *facere* (*identitatem facere*), que significa comprobar, hacer patente la identidad de alguien o algo.

Según el diccionario de la Real Academia de la Lengua Española, identificación es la acción y efecto de identificar o identificarse, entendiendo por identificar: reconocer que una persona o cosa es la misma que se supone o se busca.

Un carné de identidad, documento nacional de identidad (DNI), o cédula de identidad es un documento emitido por una autoridad administrativa competente para permitir la identificación personal de los ciudadanos.

1.2.Estándar Internacional ISO/IEC 7810

La creación de los documentos de identificación y tarjetas está regida por los estándares internacionales definidos por la Organización Internacional para la Normalización (ISO).

A partir de estas disposiciones, la ISO define en el estándar ISO/IEC 7810:2003[1] cuatro formatos para tarjetas de identidad o identificación, el ID-1, ID-2, ID-3 e ID-000. Estos formatos establecen las normas para la creación de todo tipo de documento de identificación a nivel internacional, de aquí la importancia de la definición de estos estándares.

1.2.1. ID-1

El formato ID-1 es comúnmente usado para tarjetas de banco (tarjetas ATM, tarjetas de crédito, tarjetas de débito, etc.) y tiene una especificación de tamaño de 85.60 × 53.98 mm (3.370 × 2.125 in). También hoy se utiliza en muchos países (incluyendo los Estados Unidos, Canadá, Australia, Nueva Zelanda, Noruega y países de la Unión Europea). Este formato también es usado como tarjeta de identificación personal en algunos otros países como Chile, Pakistán, y es un formato bastante común en las tarjetas de negocio.

Este formato a su vez define otros cuatro estándares que especifican las características de otros sub-formatos que se muestran a continuación:

* ISO 7813 define características adicionales de las tarjetas de plástico de bancos, por ejemplo el espesor y las esquinas redondeadas de determinadas dimensiones.
* ISO 7811 define las técnicas tradicionales para registrar los datos en las tarjetas de identificación ID-1, es decir con caracteres en relieve y otros formatos de grabación magnética.
* ISO 7816 define las tarjetas ID-1 de identificación con un chip incrustado (tarjeta inteligente o smartcard) y las superficies de contacto para la energía, el reloj, y las señales de datos en serie y de reseteo.
* ISO 14443 define las tarjetas de identificación con un chip incrustado (tarjeta de proximidad) y una antena de bucle magnético que opera a 13,56 MHz (RFID[3]). Las especificaciones más recientes definidas por la OACI en la norma ISO 14443 de chips RFID para los documentos de viaje de lectura mecánica (DVLM), especifica que el formato de los archivos han de tener una firma criptográfica y protocolo de autenticación para el almacenamiento de datos biométricos (fotos de la cara, huellas dactilares y / o del iris).

1.2.2. ID-2

El formato ID-2 especifica un tamaño de 105 × 74 mm (4,134 a 2,913 in). Este tamaño es el formato A7. El ID-2 es el formato utilizado, por ejemplo, en el documento de identidad alemán (Personalausweis). Este formato, ligeramente más grande, proporciona suficiente espacio para una claramente reconocible foto facial, pero es aún lo suficientemente pequeño como para ser transportado en una billetera.

[3] RFID: (del inglés Radio Frequency Identification), en español Identificación por Radiofrecuencia.

1.2.3. ID-3

El formato ID-3 especifica un tamaño de 125 × 88 mm (4,921 a 3,465 in). Este tamaño es el formato B7. Este formato se utiliza en todo el mundo para los pasaportes y visados.

1.2.4. ID-000

La ID-000 especifica un tamaño de 25 mm x 15 mm. Este formato es el utilizado por las tarjetas SIM[4].

1.3. Estándares Internacionales de la Organización de la Aviación Civil Internacional (OACI)

La Organización de la Aviación Civil Internacional (OACI) es la encargada de definir los estándares de los documentos de viaje, basándose en los tamaños estándares de los documentos de identificación definidos por la ISO, anteriormente mencionados.

La tarea de creación de estos estándares tuvo su origen en el Convenio de Chicago[5], donde se presenta una gama completa de requisitos para que las operaciones de aviación civil se ejerzan en forma eficiente y ordenada, lo cual comprende disposiciones para el despacho de personas por los puntos de control fronterizos, o sea:

a) el requisito de que las personas que viajen por vía aérea y las tripulaciones obedezcan los reglamentos de inmigración, aduanas y pasaportes;
b) el requisito de que los Estados simplifiquen las formalidades para el cruce de fronteras y eviten todo retardo innecesario; y
c) el requisito de que los Estados elaboren y adopten normas y procedimientos internacionales respecto a las formalidades de aduana e inmigración.

La OACI, en documento 9303 parte 1, define los formatos de documentos de viaje de lectura mecánica. La 6 ª edición de la Parte 1 del documento 9303, publicado por la OACI en 2006, se divide en 2 volúmenes: En el Doc. 9303, parte 1, Tomo 1 se describen "los pasaportes de lectura mecánica con los datos almacenados en el formato de reconocimiento óptico de caracteres". En el Doc. 9303, Parte 1, Volumen 2 contiene las especificaciones para los pasaportes electrónicos habilitado con la capacidad de identificación biométrica ", utilizando chips incrustados con RFID sin contacto.

[4] SIM: Es el acrónimo de Subscriber Identify Module y significa Módulo de Identificación del Suscriptor.
[5] Convenio de Chicago: La Convención de Chicago del año 1944, llamada oficialmente Conferencia Internacional de Aviación Civil tuvo por objeto actualizar la Convención de Paris de 1919 sobre normas de aviación civil.

1.4. Estado del Arte

Una herramienta de diseño de plantillas para documentos de identificación es una aplicación que permita crear, editar, salvar, y la personalización e impresión de plantillas (tarjetas de identificación, cédulas, pasaportes, entre otros).

No siempre se ha contado con herramientas que permitan tales beneficios a la hora de crear un documento de identificación, tiempos atrás las cosas eran muy diferentes.

En España, en la época de la conquista de América existió un proto-D.N.I (Documento Nacional de Identificación). Llamado cédula de composición, que daba fe de la identidad del que se embarcaba hacia el Nuevo Mundo, pues los *manchados de sangre* (judíos conversos y sus descendientes) tenían prohibido viajar a América. El pre-D.N.I. oficial de España parece ser la cédula de identidad del siglo XIX a la que no se le hacía demasiado caso y que se falsificaba con facilidad (entre otras cosas no tenía foto).

En la actualidad no pocas empresas se han dedicado a hacer aplicaciones que permiten el diseño de documentos de identificación y tarjetas, entre ellas se encuentran el Grupo Mühlbauer y DataCard®. Esta última es una de las impulsoras de la gran mayoría de los programas más importantes de emisión de tarjetas de todo el mundo.

En Cuba, existe una aplicación creada por la empresa de tecnologías y sistemas DATYS, perteneciente al MININT (Ministerio del Interior), llamada EMIPAS, la cual es el sistema cubano de emisión de pasaportes. Este sistema abarca tres módulos, el de personalización, el de administración y el de reportes.

En la Universidad de las Ciencias Informáticas (UCI) se estuvo trabajando, en el proyecto Control de Acceso, en la elaboración de un prototipo de programa para ser utilizado en la creación de documentos de identificación para su uso local, CredentialBuilder. La aplicación estuvo en pruebas pero luego no se llegó a concretar su uso y nunca se finalizó.

Como parte del proyecto Identidad de la UCI con la República Bolivariana de Venezuela, se desarrolló un sistema de emisión de pasaportes que se encuentra en explotación en ese país, con el nombre de Sistema de Identidad Pasaporte Electrónico, el cual ha tenido muy buen desempeño.

Otra de las aplicaciones desarrolladas en este centro es el Sistema de Emisión de Solapines el cual es utilizado internamente en la institución para la impresión de estos documentos locales. Este sistema y el desarrollado en el proyecto Identidad no poseen un editor de plantillas y estas son creadas mediante la modificación directa del código de un documento XML.

También se diseñó otra solución llamada IDSVG, para su uso local, al igual que la anterior, y que fue desarrollada utilizando Software Libre y lenguaje de programación Java. Esta solución estuvo llamada a cubrir las mismas necesidades que se trata de cubrir en este trabajo. Poseía dos módulos, uno de creación de plantillas de los documentos de identificación y otro de impresión, pero sólo se llegó a finalizar este último.

Estas soluciones no brindan una interfaz para el diseño de las plantillas del documento de identificación por lo que no cubren las necesidades actuales en este campo de la personalización de los documentos de identificación.

1.5.Metodologías de Desarrollo de Software

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Una metodología es un proceso. No existe una metodología de software universal. Se desarrollan con el objetivo de dar solución a los problemas existentes en la producción de software, que cada vez son más complejos. Estas engloban procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software.
Clasificación de las metodologías de desarrollo:

Tradicionales:

- METRICA 3
- RUP (Rational Unified Process).
- OPEN

Ágiles

- XP (eXtreme Programming).
- SCRUM
- FDD (Feature Driven Development).
- CRYSTAL

A continuación se hace un análisis de las principales características de algunas de ellas:

1.5.1. XP (eXtreme Programming)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Se define adecuada para proyectos con requisitos imprecisos

cambiantes y con alto riesgo técnico. Esta metodología promueve el trabajo en equipo, además de preocuparse por el aprendizaje de los desarrollares. XP se basa principalmente en la realimentación continua entre el cliente y el equipo de desarrollo, existiendo una comunicación fluida entre ellos.
Las características fundamentales del método son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas.
- Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenimiento pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se haya introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto.
- Simplicidad en el código: La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo. (Beck, 2000)

1.5.2. SCRUM

Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante la última década. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutara durante un proyecto. Los roles principales en Scrum son el **ScrumMaster**, que mantiene los procesos y trabaja de forma similar al director de proyecto, el **ProductOwner**, que representa a los **Stakeholders**[6] (clientes externos o internos), y el **Team** que incluye a los desarrolladores.

[6] Stakeholders: palabra de origen inglés que significa parte implicada, parte afectada o interesada.

Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

1.5.3. Crystal Methodologies

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Se basa en ciclos iterativos de desarrollo incremental (de 1 a 4 meses máximo), a lo que añade una reunión previa y posterior al ciclo, en la que reflexiona sobre el proyecto y sobre cómo ha ido ese ciclo. Antes de comenzar el siguiente ciclo al menos dos usuarios finales deben revisar, de forma independiente, lo desarrollado y validarlo.

Característica del equipo:

- Crystal aconseja que el tamaño del equipo sea reducido (Pocos componentes).
- La mejora de la comunicación entre los miembros del equipo del proyecto:

  Mismo lugar de trabajo → Disminuye el coste de la comunicación.

Políticas de equipo:

- Se utilizarán políticas diferentes para equipos diferentes:

Codificación por colores de Crystal: dependiendo del tamaño del equipo (Crystal Clear para 3-8 personas, Crystal Yellow para 10-20 personas, Crystal Orange para 25-50,…)

3-8	10-20	25-50	50-100	100-200	200-500	800+

Tabla 2: Codificación de colores de la metodología Crystal Methodologies

1.5.4. RUP

El Proceso Unificado de Desarrollo (RUP) es una metodología para la Ingeniería de Software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que

soportan el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. RUP está 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 organizado 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.
- 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:

- 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 uno o varios releases [7] del producto que han pasado las pruebas. Se ponen estos releases 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 del software al de la explotación de este.

El ciclo de vida de RUP se caracteriza por estar:

- 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.

1.6. Lenguaje de Modelado UML

El Lenguaje Unificado de Modelación (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. El UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. UML no es método, es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Principales diagramas UML:

[7] Release: palabra inglesa para referirse en informática a la entrega o liberación de una versión del producto.

Diagramas de estructura estática: Describen las propiedades estructurales del sistema.

- Diagrama de clases: Conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
- Diagrama de objetos: Conjunto de objetos y sus relaciones.
- Diagrama de casos de uso: Conjunto de casos de uso y actores y sus relaciones.

Diagramas de comportamiento:

- Diagramas de interacción (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
- Diagrama de estados: Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.
- Diagrama de actividad: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

Diagramas de Implementación:

- Diagramas de despliegue.
- Diagrama de componentes.

(UCI)

1.7.Herramientas para el modelado

1.7.1. Rational Rose

Características:

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- Soporte de ingeniería directa y/o reversa para algunos de los conceptos más comunes de Java 1.5
- La generación de código C++, Java y Visual Basic, con capacidad de sincronización modelo-código configurables.
- Soporte Enterprise Java Beans™ 2.0.
- Capacidad de análisis de calidad de código.
- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones Web.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear Definiciones de Tipo de Documento XML (DTD) para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Capacidad para integrarse con cualquier sistema de control de versiones SCC-compliant, incluyendo a Rational ClearCase.
- Publicación Web y generación de informes para optimizar la comunicación dentro del equipo.

Rational Rose es una herramienta de modelado visual, líder en el mundo:

- Rational Rose domina el mercado de herramientas para el análisis, modelado, diseño y construcción orientado a objetos.
- Rational es reconocido como el líder tecnológico por su rol en el desarrollo del UML, logrado en gran parte por los esfuerzos de Grady Booch, Ivar Jacobson, y Jim Rumbaugh, los tres más importantes autores del UML.

- Rational Rose tiene todas las características que los desarrolladores, analistas, y arquitectos están exigiendo – soporte UML incomparable, ingeniería round-trip[8] multi-lenguaje, completo soporte al equipo, desarrollo basado en componentes con soporte para arquitecturas líderes en la industria y modelos de componentes tales como WinDNA y EJB, facilidad de uso, integración optimizada, y mucho más.(Rational Software Corporation, 2002)

Desventajas:

- Necesita de mucha memoria RAM para poder ser manejado de forma rápida y eficiente.
- Es una herramienta propietaria.

1.7.2. Visual Paradigm

Básicamente se trata de una herramienta para trabajar con UML. Se puede conseguir con integración para Eclipse/IBM WebSphere, Borland JBuilder, NetBeans IDE[9]/Sun ONE, IntelliJ IDEA, Oracle JDeveloper y BEA Weblogic Workshop.

El Visual Paradigm está diseñado 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 2.1.
- 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.
- Apoya la sincronización de código Java

[8] Ingeniería Round-Trip: Solución para el proyecto estructural que permite la combinación perfecta de CAD (Diseño Asistido por Computador) y estructuras. Optimiza los procesos de trabajo cotidianos y logra evitar una introducción de datos repetitiva, laboriosa y susceptible de errores.

[9] IDE: (del inglés Integrated Development Enviroment), que significa Entorno de Desarrollo Integrado.

- Visual Paradigm para UML soporta un conjunto de lenguajes, tanto en la generación de código y de ingeniería inversa en Java, C++, PHP [10], XML Schema y Python. Además, apoya la generación de código C#, VB.NET, Object Definition Language (ODL), Flash ActionScript, Delphi, Perl, Objective-C, y Ruby. Ingeniería inversa también apoya clase Java, .NET dll y exe, JDBC [11], y los archivos de mapeo Hibernate.
- Visual Paradigm para UML soporta la importación y exportación de XMI [12] de versiones 1.0, 1.2 y 2.1. Los archivos de proyecto (.MDL / .CAT) de Rational Rose también pueden ser importados en Visual Paradigm para UML a través de la Rose Importer. Para aprovechar al máximo la interoperabilidad de productos de Visual Paradigm con otras aplicaciones, se han introducido a la importación / exportación de modelado de proyectos desde / a un formato XML abierto. Usuarios y proveedores de tecnología pueden integrar modelos de Visual Paradigm en sus soluciones con un mínimo esfuerzo.(Visual Paradigm, 2009)

1.8.Tecnologías de desarrollo

1.8.1. XML

XML, siglas en inglés de Extensible Markup Language (Lenguaje de Marcas Extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). 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 como son XHTML [13], SVG [14] y MathML [15].

Varias son las características del XML:

[10] PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser embebido dentro de código HTML.
[11] JDBC: (del inglés Java Database Connectivity), su significado es Conectividad de Bases de Datos de Java. Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.
[12] XMI: (del inglés XML Metadata Interchange), significa XML de Intercambio de Metadatos. Es una especificación para el Intercambio de Diagramas.
[13] XHTML: (del inglés eXtensible Hypertext Markup Language), significa Lenguaje Extensible de Marcado de Hipertexto. Es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.
[14] SVG: (del inglés Scalable Vector Graphics), su traducción es Gráficos de Vectores Escalables.
[15] MathML: (del inglés Mathematical Markup Language), significa Lenguaje de Marcado Matemático, utilizado en las etiquetas HTML para realizar operaciones matemáticas.

- Aunque hoy día XML aún no está tan extendido como HTML [16], su uso futuro en la Web mejorará la eficiencia de las búsquedas, al proporcionar cada documento XML metadatos sobre sí mismo.
- Permite proporcionar diferentes vistas sobre los datos (HTML, PDF [17], voz, etc.).
- Facilita la integración desde fuentes de datos heterogéneas, por ejemplo, páginas Web, distintas bases de datos, etc.
- Los documentos tienen una estructura que los hace legibles e inteligibles no sólo para los ordenadores, sino también para los humanos.
- Las aplicaciones de XML son fácilmente extensibles mediante nuevas definiciones de tipo de documentos (DTD).

XML, proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos, sirve para marcar lo siguiente:

- Un documento normal.
- Un registro estructurado, como un registro de citas o un pedido de compra.
- Un objeto con datos y métodos, como el formulario permanente de un objeto Java o de un control ActiveX [18].
- Un registro de datos, como el conjunto de resultados de una consulta.
- Meta-contenido sobre un sitio Web, como el Formato de Definición de Canal (CDF).
- Representaciones gráficas, como la interfaz de usuario de una aplicación.
- Entidades y tipos de esquema estándar.
- Todos los vínculos entre datos y personas que hay en la Web.

XML será el lenguaje que nos garantizará el intercambio de cualquier tipo de información, sin que ocasione problemas de tipo "contenido" o de tipo "presentación". Este garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes, lo que está originando una nueva generación de aplicaciones en la Web.

[16] HTML: (del inglés HyperText Markup Language), significa Lenguaje de Marcas de Hipertexto. Lenguaje de marcado predominante para la construcción de páginas Web.
[17] PDF: (acrónimo del inglés Portable Document Format), Formato de Documento Portátil. Es un formato de almacenamiento de documentos. Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto) y está especialmente ideado para documentos susceptibles de ser impresos.
[18] En 1996, Microsoft renombró la tecnología OLE 2.0 (Object Linking and Embedding) en ActiveX. Esta se usa comúnmente por diseñadores Web para incrustar archivos multimedia.

A estos fines se le une unos estándares como el Unicode e ISO/IEC 10646 para caracteres, el Internet RFC 1766 para identificación de lenguajes, ISO 639 para códigos de nombres de lenguajes y también el ISO 3166 para códigos de nombres de países, para la normal comprensión de esta versión de XML. (Francisco, 1997)

1.8.2. SVG

Scalable Vector Graphics (SVG) es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL [19]), en XML.

El SVG permite tres tipos de objetos gráficos:

- Formas gráficas de vectores (ej.: caminos consistentes en rectas y curvas, y áreas limitadas por ellos).
- Imágenes de mapa de bits / digitales.
- Texto.

Los objetos gráficos pueden ser agrupados, transformados y compuestos en objetos previamente generados, y pueden recibir un estilo común. El texto puede estar en cualquier espacio de nombres [20] XML admitido por la aplicación, lo que mejora la posibilidad de búsqueda y la accesibilidad de los gráficos SVG. El juego de características incluye las transformaciones anidadas, los *clipping paths* [21], las máscaras alfa [22], los filtros de efectos, las plantillas de objetos y la extensibilidad.

Características de Accesibilidad de SVG

Los Gráficos de Vectores Escalables (SVG) ofrecen un número de prestaciones para crear gráficos en la Web más accesibles de lo que actualmente es posible, a un más amplio grupo de usuarios. 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. La accesibilidad requiere que las prestaciones ofrecidas por SVG sean usadas y soportadas correctamente.

---

[19] SMIL es el acrónimo de *Synchronized Multimedia Integration Language* (Lenguaje de Integración Multimedia Sincronizada) y es un estándar del *World Wide Web Consortium* (W3C) para presentaciones multimedia. El lenguaje SMIL permite integrar audio, video, imágenes, texto o cualquier otro contenido multimedia.

[20] Los espacios de nombres XML se utilizan para diferenciar entre elementos XML con el mismo nombre o para agrupar datos XML del mismo tipo o función.

[21] Clipping Paths: término inglés que se utiliza para designar Máscaras de recorte de imágenes.

[22] Máscaras Alfa: Permiten hacer que partes de un elemento o un objeto visual sean transparentes total o parcialmente.

SVG - Texto plano

Un beneficio mayor de accesibilidad derivado de XML es que una imagen SVG está codificada como texto plano. Los autores pueden crearlo y editarlo con un procesador de texto o una herramienta de autor XML (hay otras propiedades de SVG que hacen esto más fácil de lo que podría parecer al principio).

SVG - Hojas de estilo

La separación del estilo del resto del contenido es muy importante para la accesibilidad. Los autores pueden usar hojas de estilo CSS o XSL para controlar la representación de imágenes SVG, una característica común a todos los lenguajes de marcado escritos en XML. Los usuarios que podrían, de otro modo, ser incapaces de acceder al contenido, pueden definir hojas de estilo para controlar la representación de las imágenes SVG, satisfaciendo sus necesidades sin perder el estilo adicional proporcionado por el autor.

Estilos extendidos

SVG ofrece un número de características de estilo que van más allá de las propiedades definidas en CSS para proporcionar efectos gráficos para controlar cómo se representan las imágenes. Características como máscaras, filtros, y la habilidad de definir fuentes altamente sofisticadas se encuentran disponibles en SVG.

SVG - la interfaz DOM

Otro beneficio de usar XML es que la interacción se puede hacer accesible mediante el Modelo de Objeto de Documento (DOM). La interfaz del DOM puede habilitar el uso de muchas ayudas técnicas con las imágenes SVG. SVG permite el acceso tanto a hojas de estilo como a contenido XML, puesto que usa la versión 2 del **DOM** (**DOM2**).

XML - SVG con otros lenguajes XML

Los documentos SVG pueden ser incluidos en documentos escritos en otros lenguajes XML, y también pueden contener fragmentos de código de otros lenguajes XML. La mezcla de lenguajes de marcado puede incrementar la accesibilidad, pues lo autores podrían utilizar el lenguaje de marcas que más se ajuste a cada parte del documento.

(W3C, 2009)(W3C, 2009)

1.8.3. Análisis de Plataformas de Desarrollo

Para el desarrollo de una solución informática que dé respuesta a la problemática antes planteada es necesario escoger una plataforma de desarrollo que reúna varias particulares acordes a los intereses del sistema y de los usuarios que se beneficiarán del mismo.

Una plataforma de desarrollo es el entorno común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo, sin embargo, también es posible encontrarlas ligadas a una familia de lenguajes de programación o a una Interfaz de Programación de Aplicaciones o API (Application Programming Interface). Se le denomina plataformas a los diferentes ambientes creados para el desarrollo de software. (Almenares, y otros, 2007)

En los últimos años las plataformas de desarrollo han ido evolucionando siempre con el objetivo de ofrecer una interfaz más amigable para el programador y abstraerlo de las funciones más elementales, la creación de aplicaciones con sus prestaciones para diferentes campos de acción como los son la Web y las aplicaciones de escritorio.

Entre las varias plataformas de desarrollo que hay en el mundo se encuentran:

Máquina Virtual de Java

La Máquina Virtual de Java (MVJ) es el núcleo del lenguaje de programación Java. De hecho, es imposible ejecutar un programa Java sin ejecutar alguna implantación de la MVJ. En la MVJ se encuentra el motor que en realidad ejecuta el programa Java y es la clave de muchas de las características principales de Java, como la portabilidad, la eficiencia y la seguridad. Siempre que se corre un programa Java, las instrucciones que lo componen no son ejecutadas directamente por el hardware sobre el que subyace, sino que son pasadas a un elemento de software intermedio, que es el encargado de que las instrucciones sean ejecutadas por el hardware. (García Carballeira, 2000)

Las maquinas virtuales tienen algunas desventajas dentro de las que se encuentran, es que agregan gran complejidad al sistema en tiempo de ejecución. Por ejemplo, la MVJ espera que la computadora sobre la que subyace, soporte el estándar de IEEE para los números de punto flotante de 32 y 64 bits, así como enteros largos de 64 bits. La mayoría de las plataformas lo hacen pero hay algunas que no, lo que implica trabajo extra. (García Carballeira, 2000)

La principal desventaja de los lenguajes basados en máquina virtual, es que efectivamente son más lentos que los lenguajes completamente compilados, debido a la sobrecarga que genera tener una capa de software intermedia entre la aplicación y el hardware de la computadora.

La Máquina Virtual de Java tiene tres (3) deficiencias, un conjunto de instrucciones no ortogonal[23], difícil de entender, ya que se utiliza un byte para codificar el código de operación de las instrucciones del procesador virtual Java y no posee un árbol de análisis sintáctico, o sea el código intermedio, usado en la MVJ, es simple y plano, es decir no incluye información acerca de la estructura del método original. (García Carballeira, 2000)

Plataforma .NET (.NET Framework)

El .NET Framework es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de compilación.
- Bibliotecas de clases de .NET.
- CLR[24].

(Microsoft Corporation, 2008)

.NET Framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes. Existen más de 30 lenguajes adaptados a .Net, desde los más conocidos como C# (C Sharp), Visual Basic o C++ hasta otros lenguajes menos conocidos como Perl o Cobol.

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.

[23] Un lenguaje de programación es ortogonal, si tiene el mismo número de instrucciones asociadas a cada uno de los tipos de datos.
[24] Common Language Runtime, es un componente de maquina virtual del .NET Framework de Microsoft.

El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

La principal ventaja de los espacios de nombres de .Net es que de esta forma se tiene toda la biblioteca de clases de .Net centralizada bajo el mismo espacio de nombres (System).

Esta plataforma cuenta con ciertas ventajas a tener en cuenta como son:

- **Código administrado:** El CLR (Command Language Runtime) realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.
- Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL[25]).
- **Compilación Just-in-Time:** El compilador JIT[26] incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.
- **Recolector de basura:** El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (Garbage Collector[27]). El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma el programador no tiene por que liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente.
- **Seguridad de acceso al código:** Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.
- **Despliegue:** Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

[25] Es un código de bites que la tecnología .NET de Microsoft utiliza para lograr independencia de la plataforma y seguridad en ejecución.
[26] Técnica para mejorar el rendimiento de sistemas de programación que compilan a bytecode, consiste en traducir el bytecode a código máquina nativo en tiempo de ejecución.
[27] Mecanismo implícito de gestión de memoria implementado en algunos lenguajes de programación.

Algunas de las desventajas que presenta son:

- Una de las desventajas que presenta esta plataforma es el mantenimiento en múltiples lenguajes. Mantener un proyecto en múltiples lenguajes es costoso. Si una aplicación está realizada en varios lenguajes se necesitan expertos en varios lenguajes para entenderla y mantenerla, aumentando los costos.
- No es multiplataforma, o sea ella solo está disponible para la familia de Windows.
- El tema de las licencias es también otras de las desventajas por lo que no hay licencias libres.
- La infraestructura para desarrollar en .NET representa un alto costo para las empresas.

(Microsoft Corporation, 2008)

Plataforma Mono

Es un proyecto de código abierto impulsado por Novell y creado por Miguel de Icaza como alternativa al Framework .NET para los programadores de software libre para brindar un grupo de herramientas, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA[28].

Esta plataforma presenta importantes ventajas, es ideal para desarrollo de plataformas cruzadas y se ha posicionado como un entorno que permite ejecutar en Mac OS X y Linux aplicaciones diseñadas para Microsoft .Net en entorno Windows, facilitando la migración de aplicaciones a estas plataformas y aumentando su base de desarrolladores y usuarios. (Cano Ossa, 2006)

Algunas de las ventajas que ofrece son:

- Posee independencia de lenguajes, se puede usar clases escritas en cualquier lenguajes que este soporte (Python, C#, Mono Basic, Java, Nemerle, Boo).
- Posee también independencia de plataforma, lo cual las aplicaciones son muy portables y la mayoría compatibles en binario entre plataformas; tienen un gran soporte para bases de datos MS SQL, MySQL, Postgres, OLE DB.
- Otra característica presente es la GUI[29] multiplataforma, lo cual se pueden escribir aplicaciones con interfaz gráfica que se ejecutan invariantemente en multitud de plataformas.
- Las aplicaciones se traducen a CLI.La ventaja fundamental de CLI frente los formatos usados anteriormente es claramente la independencia del lenguaje, por ejemplo, podrás escribir tus

[28] European Computer Manufacturers Association, ECMA International es una organización basada en membrecías de estándares para la comunicación y la información. Creada en 1961.

[29] Graphical User Interface, tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

librerías en C# y usarlas desde Python sin ninguna dificultad, de hecho, no hay diferencia entre una librería escrita en Python, C# o Fortran si éstos generan CLI. (Seoane, 2004)

Como desventajas:

- La principal desventaja que presenta esta plataforma es que es muy inmadura, todavía no ha madurado lo suficientemente como para implementar grandes proyectos de gestión, y debido a esto pues presentan errores como son en el mecanismo de actualización, que maneja el redibujado de la aplicación, no funciona correctamente cuando ésta es ocultada por otra aplicación.
- Adicionalmente, el mecanismo de eventos presenta algunos problemas con algunos controles de interfaz.
- No se desarrolla a la par del .NET Framework de Microsoft, que actualmente va por su versión 3.5 y la Plataforma Mono usa aún la versión 2.0.
  (Mono, 2009)

Teniendo en cuenta que Cuba busca soluciones libres de ataduras, licencias o patentes que puedan perjudicar de alguna forma su posterior desarrollo tecnológico, asociado además, que la plataforma .Net es básicamente una tecnología no es alternativa debido a las restricciones de los EE.UU sobre Cuba, entre las cuales se incluyen el reciente bloqueo del servicio de mensajería MSN para Cuba; se ha decidido realizar el desarrollo de la aplicación usando Java como plataforma y lenguaje de programación. A partir de esta decisión, se procederá a analizar las herramientas de desarrollo que toman como base esta plataforma.

1.9.Herramientas de Desarrollo

1.9.1. 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.

Características del NetBeans:

- Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas.
- Está escrito en Java pero puede servir para cualquier otro lenguaje de programación.
- Posee un número importante de módulos para extender el IDE NetBeans.

- Es un producto libre y gratuito sin restricciones de uso.
- Soporta el desarrollo de todos los tipos de aplicación Java (J2SE [30], web, EJB [31] y aplicaciones móviles).
- Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones.
- Posee soporte multiplataforma por lo que puede ser ejecutado en sistemas operativos como Microsoft Windows, GNU/Linux, Macintosh OS X y Solaris.

(NetBeans, 2009)

1.9.2. Eclipse

La definición que da el proyecto Eclipse acerca de su software es: "*una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular*".

El IDE Eclipse emplea módulos (en inglés *plug-in*) para proporcionar funcionalidades, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, posibilita al IDE trabajar con lenguajes para procesado de texto como LaTex, aplicaciones en red como Telnet y Sistema de gestión de base de datos. Provee en el SDK [32] soporte para Java y CVS [33].

En cuanto a las aplicaciones clientes, Eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. Por ejemplo, GEF (Graphic Editing Framework - Framework para la Edición Gráfica) es un plugin de Eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto WYSIWYG [34] hasta editores de diagramas UML, Interfaces Gráficas para el Usuario (GUI), etc. Dado

[30] J2SE: acrónimo que significa Java Platform, Standard Edition o Java SE, en español Plataforma Java, Edición Estándar. Es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.

[31] EJB: acrónimo informático que significa Enterprise JavaBeans. Son una de las API que forman parte del estándar de construcción de aplicaciones empresariales.

[32] SDK: (del inglés Software Development Kit), significa Kit de Desarrollo de Software.

[33] Concurrent Versions System (CVS), también conocido como Concurrent Versioning System, es una aplicación informática que implementa un sistema de control de versiones.

[34] WYSIWYG: (del inglés What You See Is What You Get), significa Lo Que Ves Es Lo Que Hay.

que los editores realizados con GEF "viven" dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional.

Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ [35]) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como por ejemplo, BitTorrent Azureus.(Eclipse, 2009)

1.10. Elección de las principales herramientas a utilizar.

Luego de realizar un estudio de las principales tendencias tecnológicas actuales relacionadas con el software que se pretende hacer, se llega a la conclusión que dadas las características de la situación que se enfrenta, las tecnologías a utilizar son:

- Metodología de desarrollo: RUP.
- Herramienta de modelado: UML y Visual Paradigm.
- Herramienta de desarrollo: NetBeans.
- Y lenguaje de programación: Java.

Se tuvo en cuenta para su elección las ventajas que estas presentan. Se utiliza la metodología de desarrollo RUP pues tiene como resultado un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura y es iterativo e incremental y con esto se lograría un rápido y eficiente desarrollo en el más corto tiempo.

Para la modelación de UML se utiliza el Visual Paradigm diseñado para el desarrollo de software con POO, por su fácil uso es capaz de reducir la duración del ciclo de desarrollo ayudando tanto a arquitectos, analistas, diseñadores y desarrolladores. Su integración con varios IDE y multiplataforma. Cumple con la metodología del RUP.

Como herramienta de desarrollo se hace uso del IDE NetBeans por ser un producto libre y gratuito sin restricciones de uso, multiplataforma, soporta el desarrollo de todos los tipos de aplicación Java, es un producto desarrollado en java con soporte para varios lenguajes de POO.

Se emplea el lenguaje de programación Java, que cumple con el paradigma de POO y hace más fácil y eficiente el desarrollo de software. El uso de este lenguaje en plataformas de desarrollo libre hace más extensible y confiable su uso.

[35] EJC: (del inglés Eclipse Compiler for Java), es un compilador de Java para Eclipse.

1.11. Conclusiones Parciales del Capítulo

En estos días estar identificado resulta de vital importancia. Las tecnologías han posibilitado que largos y extenuantes procesos de identificación se hagan en minutos. A nivel internacional existen exitosos programas que hacen del proceso creación de los documentos de identificación algo más cómodo y sencillo, con la contraparte de ser muy costosos y dependientes de tecnologías de terceros. En Cuba también ha habido adelantos con respecto al tema, pero aún queda un largo camino que recorrer.

Teniendo en cuenta el estudio realizado, existe la necesidad de desarrollar programas que contribuyan a la creación de los documentos de identificación rigiéndose por los estándares internacionales de la OACI y a precios asequibles.

Tras haber realizado un análisis profundo de las metodologías y las herramientas que le dan soportes a estas, se decidió escoger RUP con UML junto a la herramienta CASE Visual Paradigm. Se pensó que es una buena elección por las grandes facilidades que esta presenta y que fueron mencionadas anteriormente en el análisis de la herramienta.

Se ha escogido Java como lenguaje de programación para desarrollar el sistema y NetBeans como IDE de desarrollo por las grandes facilidades de diseño y programación que muestra en sus más recientes versiones.

Capítulo 2 - Características del Sistema

Elaborar un software que contribuya al diseño de plantillas de documentos de identificación es una tarea en la que hay que tener en cuenta una serie de aspectos para lograr una solución con calidad.

En el siguiente capítulo se exploran posibles soluciones, además se hace una propuesta de solución y se abordan los principales aspectos a tener en cuenta para su elaboración, entre los que se encuentran las funcionalidades con que contará el sistema, los principales casos de uso a desarrollar y el modelado de dominio descrito.

2.1.Descripción de los procesos involucrados en la personalización

Cuando se va a crear un documento de identificación, uno de los pasos a los cuales se procede luego de saber qué información presentar, es a la personalización del documento (Ver Fig.1). Los documentos de identificación suelen construirse de tamaños estándares, para que puedan ser procesados por todos los lectores y máquinas de personalización del mercado. Por eso es tan importante asimilar los estándares internacionales en cuanto a la fabricación de artículos que se usan tan regularmente como pueden ser las tarjetas de crédito o los pasaportes, y por supuesto, los equipos que se usan para leer estos. En el paso de creación del documento de identificación, llamado “Personalización”, es la parte del proceso de creación del documento de identificación donde se escriben los datos personales del portador, así como los elementos de seguridad que tengan algunos de esos datos, de manera tal que todas estas se rijan por los estándares internacionales y que así, a su vez, provean de seguridad al documento por la estrategia de diseño seguida. Estos diseños, la mayoría de las veces se realizan escribiendo los códigos y probando a ver como se muestran los documentos con las medidas elegidas y el enfoque dado a su organización. Esto provoca que se demore más la creación del producto final, pues se crea una plantilla para el documento a prueba y ensayo, desperdiciando más documentos de prueba; documentos que vienen a veces con medidas de seguridad pre-escritas, y por tanto cuestan bastante.

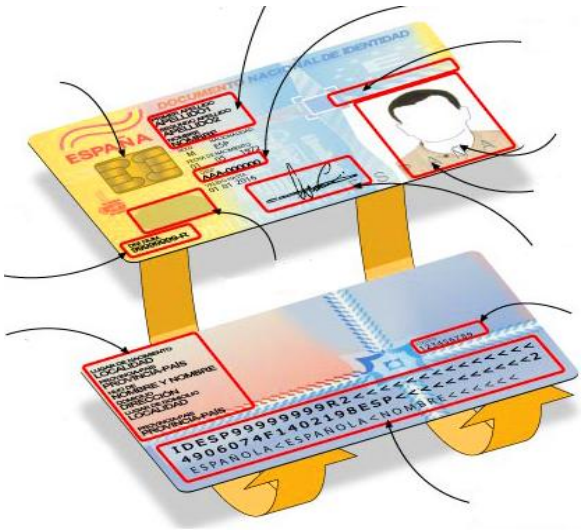


Figura 1: Ejemplo de campos personalizables en el DNI electrónico español. (Fuente:(Policía))

2.2.Análisis de otras soluciones

A nivel mundial prestigiosas empresas incursionan en el mundo del diseño de documentos de identificación, entre ellas se encuentran el Grupo Mühlbauer y DataCard®.

2.2.1. DataCard®

El grupo DataCard®, líder de ventas mundial de soluciones para la identificación segura y la personalización de tarjetas, ofrece a sus clientes una amplia gama de servicios que incluye sistemas de emisión de tarjetas en grandes cantidades, inserciones, ensobrados y acabados, emisión de identificaciones seguras y producción de pasaportes; así como una gran variedad de opciones de asistencia y suministros (Ver Fig.2). Representa además, una entidad privada con ingresos anuales de aproximadamente 400 millones de dólares, con más de 1 400 empleados y que realiza operaciones en todo el mundo incluyendo centros de desarrollo en el Reino Unido, Alemania, Francia, India, Japón y Estados Unidos.

DataCard utiliza aplicaciones propias y no las distribuye al cliente, sino que se encarga de crear el mismo los diseños al cliente. Utiliza aplicaciones personalizadas como FlashCredit para el diseño de tarjetas de sobremesa, la cual permite emitir de forma instantánea tarjetas de crédito de forma segura para los clientes que se encuentren en el punto de venta. De las aplicaciones más sofisticadas de este grupo no se tiene registro de nombres.(DataCard, 2009)

Una solución con las características que ofrece DataCard no es válida si se decide en algún momento cambiar de proveedor o si Cuba decide en un futuro hacer un sistema de personalización propio.

Figura 2: Servicios disponibles que ofrece el grupo DataCard®. (Fuente: (DataCard))

2.2.2. Grupo Mühlbauer

El Grupo Mühlbauer es uno de los líderes del mercado mundial en cuanto a sistemas innovadores y soluciones de software para la producción y personalización de tarjetas, pasaportes y aplicaciones de RFID. Esta empresa tiene aproximadamente 1800 empleados en 29 localidades en los cinco continentes y cuenta con soluciones que permiten la fabricación de cualquier tipo de chip de la tarjeta, incluyendo las tarjetas de identificación, tarjetas de contacto, así como tarjetas inteligentes para el control de acceso. (Mühlbauer Group)

Comprar el producto que se necesita para el diseño de documentos de identificación a empresas con estas características, resolvería muchos problemas actuales en el campo de la identificación por tarjetas y resultaría medianamente asequible. Aún así, el hecho de comprar el servicio a estas compañías extranjeras provocaría una dependencia tecnológica y por consiguiente gastos provocados por el mantenimiento del servicio y del equipamiento; dependencia de la cual nuestro país, al ser una nación en ascenso en el campo de la informática, no debería incluirse, siempre que tenga como evitarla, para así ganar terreno en garantizar su soberanía tecnológica. Una solución de producción nacional evitaría todos estos gastos y la dependencia de los productos extranjeros.

2.2.3. XYMA Emipas

Previendo este tipo de situaciones, *DATYS*[36] - *Tecnología y Sistemas*, empresa perteneciente al MININT (Ministerio del Interior), elaboró un software para la personalización y emisión de pasaportes de lectura mecánica que actualmente está en siendo utilizado por el MINREX (Ministerio de Relaciones Exteriores) en la creación de los pasaportes para ciudadanos cubanos. Es un software que en términos de tecnologías de diseño de pasaportes no cuenta con los últimos avances, pues no es capaz de emitir pasaportes electrónicos. La aplicación se nombra *XYMA*[37] *Emipas* y ha sido desarrollada usando la tecnología propietaria .NET, perteneciente a la corporación norteamericana Microsoft Corporation®, y de la cual no se le otorgan licencias de comercialización a Cuba producto del bloqueo económico que ha impuesto EE.UU.

Esta aplicación no constituye un editor de plantillas sino que va directamente a la personalización y a otros aspectos del servicio de emisión de documentos. Por lo que no provee una solución a la creación de plantillas de documentos de identificación entre sus funcionalidades.

2.2.4. CredentialBuilder

En la UCI ha habido algunos avances en cuanto a la creación de programas que faciliten el diseño de documentos de identificación. Un ejemplo de ello es el prototipo de programa para la creación de documentos de identificación de su uso local (CredentialBuilder), creado por estudiantes del proyecto Control de Acceso. Esta aplicación fue elaborada con una tecnología .NET, en el IDE Visual Studio 2005 y utilizando C#.NET como lenguaje de programación[38]. Su interfaz es amigable y basa su funcionamiento solamente en la inclusión de elementos de líneas, imágenes y textos, que pudieran ser impresos en documentos de identificación.

2.2.5. IDSVG

También se diseñó para uso local otra solución llamada IDSVG, desarrollada utilizando lenguaje de programación Java y utilizando Software Libre. La aplicación está compuesta por la tecnología necesaria solo para la parte de personalización, transformando un diseño parametrizable (plantilla) en

[36] DATYS: El nombre no significa nada, ha sido un identificador creado para identificar a la empresa, como lo es Sony o Intel.
[37] XYMA: El nombre no significa nada, ha sido un identificador creado para identificar la serie de productos que posee como parte del nombre esta denominación.
[38] Anteriormente se hizo alusión a los problemas de licencias de esta tecnología en el país

un documento SVG personalizado y además ofrece la posibilidad de imprimirlo. La dificultad de IDSVG es que adolece de un editor de plantillas, hay que hacer los documentos XML sin ayuda visual.

Estas últimas dos soluciones no se encuentran en disposición de cubrir las necesidades actuales en el campo de la impresión de los documentos de identificación, porque la primera tiene serias limitaciones de dibujo, y la segunda adolece de la parte del diseño gráfico de plantillas, sólo se centra en el formato y las transformaciones de estas plantillas (a nivel conceptual).

2.2.6. CardFive

Una aplicación que más cumple con los requerimientos buscados es la incluida en las impresoras de tarjetas marca *Zebra*, llamada *CardFive*. Esta aplicación personaliza tarjetas y desde ella misma se imprimen, una a una, las tarjetas personalizadas cuando se conecta el ordenador con la aplicación a la impresora. Como desventajas de su uso, esta aplicación realiza la personalización de las tarjetas con un formato propio y no estándar, pues viene específicamente para la impresora que acompaña. Posee limitadas capacidades de dibujo, el formato que crea es específico de esa aplicación y se codifica binario, por lo que no se puede leer, ni editar con otro programa. Tampoco se puede ver el código fuente del documento en ninguna etapa de su creación y es una aplicación totalmente propietaria. Esta aplicación posee todas las funcionalidades unidas y requiere de supervisión humana.

2.3. Solución Propuesta

La propuesta de solución comprende el análisis y diseño de una aplicación dedicada al diseño de plantillas de documentos de identificación, para que sistemas más abarcadores las utilicen para su personalización o sirva como aplicación complementaria a otras como la mencionada IDSVG, y así evitar, o reducir grandemente, los cambios directamente en el código en la creación de las plantillas de los documentos de identificación.

Las plantillas de la aplicación serán creadas yguardadasen formato de salida SVG, el cual está basado en el formato estándar XML. Esto contribuirá en gran medida, al éxito del producto final que garantizará una mejor portabilidad de los estándares internacionales a la impresión de los documentos de identificación, a través de una interfaz funcional, rápida y agradable al usuario.

La aplicación está basada en los estándares internacionales de diseño de la OACI para los documentos de identificación, documentos de viaje y documentos de lectura mecánica, lo cual se ajusta a los requerimientos del sistema.

Para evitar las trabas comerciales de leyes y licencias, esta solución informática será desarrollada utilizando Software Libre y con lenguaje de programación Java, lo que hará de la aplicación una solución multiplataforma.

Los documentos que se diseñarán gráficamente en la aplicación propuesta serán los documentos de identificación como tarjetas de crédito y débito, cédulas de identificación personal, documentos de identificación electrónicos; pasaportes, visas y tarjetas de lectura mecánica, entre otros documentos semejantes, reduciendo, considerablemente, el tiempo empleado en la creación de la plantilla de documento de identificación.

Se realiza la propuesta de esta solución tratando de hacer una personalización de mayor calidad, algo más fácil y rápida. El formato de salida SVG que será usado por la aplicación, al estar basado en XML, es muy ligero, puede verse el código fuente en cualquier editor de texto plano, es un formato libre y crea imágenes vectoriales lo que hace que no pierdan calidad al cambiar su tamaño. Este formato es extensible, por lo que si se desea, se puede traducir la estructura del SVG a otro formato que pueda ser creado con una transformación XML, como por ejemplo, el PDF de Adobe[39]. Esta solución, al implementarse completamente, contribuirá con su uso en Cuba y también podría convertirse en un producto exportable como aplicación ligera o integrada a algún sistema de emisión de documentos de identificación.

[39]Adobe Systems Incorporated: Empresa de software con sede en San José (California, USA) fundada en diciembre de 1982 por John Warnock y Charles Geschke. Destaca en el mundo del software por sus programas de edición de páginas web, vídeo e imagen digital.

2.4.Modelo de Dominio

Un *modelo* es una abstracción del sistema, especificándolo desde un determinado punto de vista y un determinado nivel de abstracción.

Un *modelo de dominio* captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. El modelo del dominio se considera en RUP un subconjunto del llamado modelo de objetos del negocio. Se realiza cuando se determina que no es necesario un modelo completo del negocio y servirá para poder ayudar a comprender mejor los conceptos que aparecen en el sistema, en este caso se realiza su descripción a través de un diagrama de clases UML definiendo las principales clases conceptuales que intervienen en el sistema.

El Modelo de Dominio es una representación visual estática del entorno real objeto del proyecto. Es decir, un diagrama con los objetos que existen (reales) relacionados con el proyecto que vamos a acometer y las relaciones que hay entre ellos. Pero no son clases de software (aunque algunos objetos del Modelo de Dominio pueden terminar siéndolo). (Larman, 2008)

2.4.1. Conceptos asociados al dominio del problema

Objeto: Son todos los elementos que pueden ser ubicados en el área de trabajo. Ej.: Línea, cuadrado, elipse, imagen, entre otros.

Plantilla SVG: Es la representación del diseño del documento de personalización y la descripción de los datos que sobre el serán personalizados. Es el resultado final al guardar el diseño del documento de identificación creado o modificado en la aplicación, siempre y cuando se le hayan especificado los campos personalizables que este contendrá y sus identificadores correspondientes.

Usuario: Es la persona que utiliza la aplicación con el fin de diseñar plantillas de documentos de identificación.

Estándar: Puede ser conceptualizado como la definición clara de un modelo, criterio, regla de medida o de los requisitos mínimos aceptables para la operación de procesos específicos, con el fin asegurar la calidad. Es decir, las reglas por las cuales regirse para que se desarrollen los documentos de identificación en correspondencia con las normas establecidas.

Aplicación Principal: Es la solución general del sistema que integra módulos y funcionalidades que están basadas en los requerimientos especificados.

Diseñador: Es la parte de la aplicación que se dedica específicamente a la creación del diseño del documento de identificación

Objeto de Diseño: Es cualquier elemento de diseño que se le pueda añadir a la tarjeta como líneas, códigos de barras, rectángulos imágenes, entre otros.

Elemento de Seguridad: Es un tipo especial de objeto de diseño. Son las impresiones especiales que se le hacen a los documentos de identificación para que estos no sean falsificados. Algunas pueden realizarse a través de una impresión sobre el documento, pero otras requieren de impresiones especiales sobre el documento. Ej.: Impresión con láser, etc.

Documento: Es el documento de identificación que está siendo diseñado y que se encuentra abierto en el diseñador. Sobre este documento es donde se dibujan y ubican los objetos de diseño.

Propiedades: Son las características propias de cada objeto del área de diseño, y que modifican al objeto de acuerdo a la configuración especificada.

2.4.2. Diagrama de Clases del Modelo de Dominio

El Modelo de Dominio queda recogido en un diagrama (parecido al Diagrama de Clases pero más simplificado). (Ver Fig. 3)

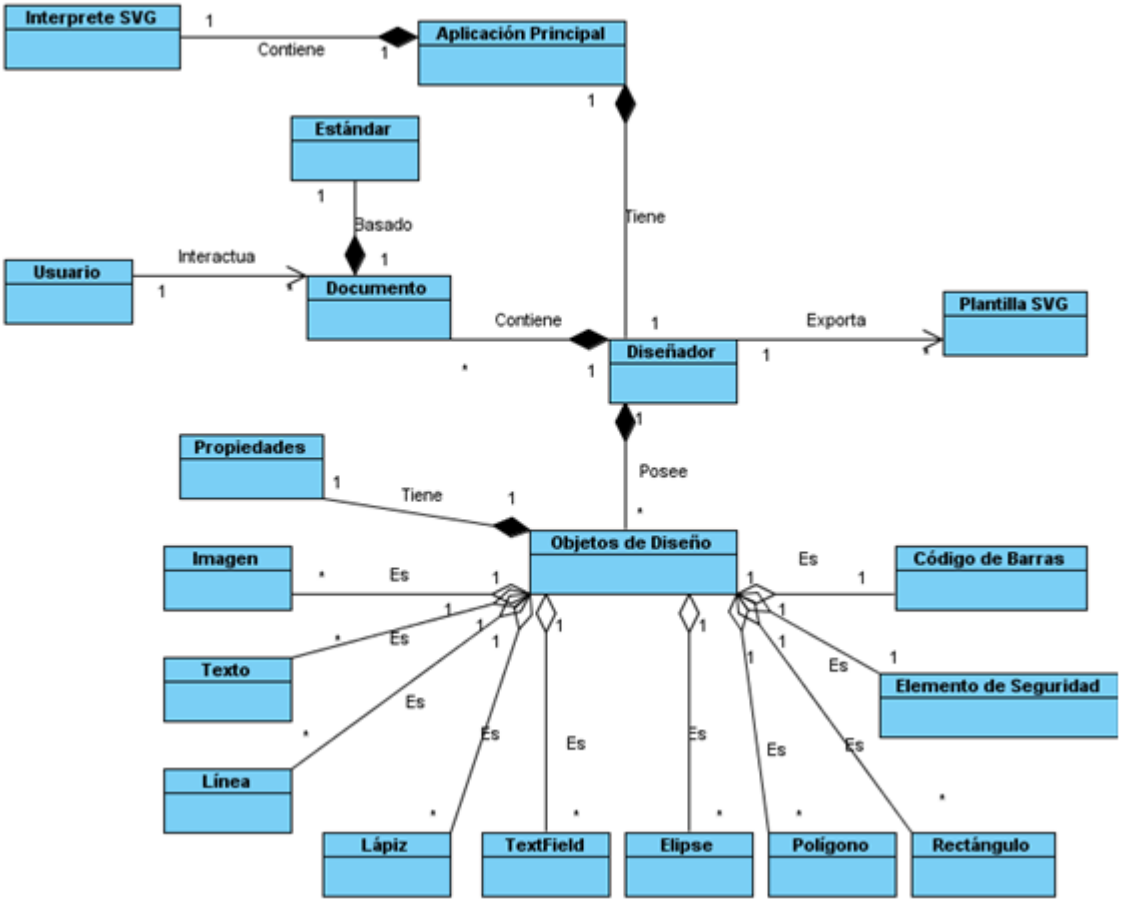


Figura 3: La figura muestra el Diagrama de Clases del Modelo de Dominio.

2.5.Especificación de los requisitos de software

2.5.1. Requerimientos Funcionales

RF-1	El sistema debe permitir la creación una nueva plantilla.
RF-2	El sistema debe permitir la especificación el tipo de documento de identificación que se va a diseñar, escogido de una lista preestablecida, para tener en cuenta las restricciones de diseño especificas del estándar internacional correspondiente.
RF-3	El sistema debe permitir que se coloquen elementos bases en la plantilla según las restricciones que imponga el estándar para el tipo de documento seleccionado. (Ejemplos de algunos elementos bases son: cuadro de foto, campos de nombre, fecha de nacimiento, fecha de vencimiento del documento, firma del titular, sexo, lugar de

	procedencia, entre otros que pueden ser escogidos para la plantilla u obviados).
RF-4	El sistema debe permitir que a cada elemento personalizable de la plantilla se le asignen las propiedades visuales y aquellas necesarias para la personalización. Ej. Visuales: Tipo de fuente a utilizar, el tamaño de la fuente, el estilo de la fuente, tamaño de la imagen, proporciones de la imagen, etc. Ej. Personalización: Identificador, fuente de datos, tipo de datos a representar, formato.
RF-5	El sistema debe permitir que se guarde la plantilla en un formato basado en XML, con gráficos vectoriales SVG y que incluya los elementos visuales y la información de la personalización en ese formato.
RF-6	El sistema debe posibilitar que se carguen las plantillas creadas.
RF-7	El sistema debe permitir el cierre de la aplicación.
RF-8	El sistema debe permitir que se copien, corten, peguen y eliminen los elementos del área de trabajo.
RF-9	El sistema debe permitir mostrar una vista previa del documento impreso con el formato creado de la plantilla.
RF-10	El sistema debe permitir que se agreguen nuevos estándares de plantilla y se ajusten los ya existentes en caso de evolución del estándar.
RF-11	El sistema debe permitir que se edite el código XML de la plantilla.
RF-12	El sistema debe permitir que se especifique la información necesaria para establecer elementos de seguridad gráfica de la plantilla mediante extensiones del formato SVG y representar visualmente de alguna manera estos elementos en Modo de Diseño y en Vista Previa. Elementos de seguridad: ✓ Impresión de fondo de seguridad. ✓ Tinta ópticamente variable (OVI (Optically Variable Ink)). ✓ Micro-letrado. ✓ Impresión Ultravioleta (UV printing). ✓ Impresión Infrarroja (IR printing). ✓ Hilo de seguridad con fluorescencia arcoíris (Security thread with rainbow fluorescence). ✓ Hilo de seguridad con cambio de color (Security thread with colorshift).

	✓ Dispositivo de Varios colores (Multitude Color Device). ✓ Impresión Base segura (Secure Core Printing). ✓ Relieve negativo (Negative embossing). ✓ Relieve positivo (Possitive embossing). ✓ Dispositivo Ópticamente Variable (OVD (Optically Variable Device)). ✓ Firma STEP (STEP sing). ✓ Protección de Imagen por Laser (LPI (Laser Protected Image)). ✓ Características de inscripción táctil por laser (Tactile Laser-engraved features). ✓ Imagen múltiple por laser (MIL (Multiple Laser Image)).

Tabla 3: Requerimientos Funcionales del Sistema.

2.5.2. Requerimientos No Funcionales

Requerimientos no funcionales de apariencia o interfaz externa	
RNF-13	El sistema debe poderse ejecutar en máquinas con una resolución de 800x600 y 32 bits de color y superiores. La resolución mínima recomendada es de 1024x768 y 32 bits de color.
RNF-14	El sistema debe poseer un diseño lo más sencillo e intuitivo posible.
RNF-15	El sistema mostrará el nombre del producto.
RNF-16	El sistema mostrará el logo del producto.
RNF-17	El texto será de color negro.
RNF-18	El idioma que se utilizará será el español.
Requerimientos no funcionales de usabilidad	
RNF-19	Los usuarios deberán poseer un conocimiento previo del manejo de una computadora personal.
RNF-20	La aplicación tendrá la posibilidad de una ayuda básica disponible para el usuario.
RNF-21	La aplicación debe poseer un instructivo básico de las especificaciones de la OACI para que una persona poco entrenada pueda guiarse y no cometa errores.
RNF-22	El sistema debe poseer información de contacto con los desarrolladores para en caso de que el usuario desee contactar para pedir ayuda o algo similar.
Requerimientos no funcionales de portabilidad	
RNF-23	La aplicación debe correr sin dificultad en el sistema operativo Windows, Linux y Mac OS X.

RNF-24	La aplicación debe poseer o asimilar las fuentes que necesita para trabajar, las cuales sean recomendaciones de la OACI de existir la restricción del estándar.
RNF-25	La aplicación no requiere ser instalada.
Requerimientos no funcionales de software	
RNF-26	Los requerimientos mínimos de software necesarios son una computadora personal con plataforma del sistema operativo Microsoft Windows XP o superior, Linux o Mac OS X.
RNF-27	En el S.O. Windows se debe tener instalado previamente el JVM (Java Virtual Machine).
RNF-28	En el S.O. Mac OS X, requiere la versión del sistema 10.5.4 o superior para que funcione correctamente.
Requerimientos no funcionales de hardware	
RNF-29	Para Windows: Procesador Intel Pentium 4 de 1.2 GHz (o equivalente) y versiones posteriores y 512 MB de RAM.
RNF-30	Para Linux: Procesador Intel Pentium 4 de 1.2 GHz (o equivalente) y versiones posteriores y 256 MB de RAM.
RNF-31	Para Mac OS X: Procesador Intel Pentium 4 a 3.0 GHz con soporte SSE2 y SSE3, o versiones posteriores de procesadores Intel y 512 MB de RAM, debe poseer una máquina.
RNF-32	Requiere el uso del mouse o algún dispositivo equivalente en su uso.
Requerimientos no funcionales de restricciones en el diseño y la implementación	
RNF-34	La herramienta para el desarrollo de la aplicación será: NetBeans 6.5 y el Altova XMLSpy 2007.
Requerimientos no funcionales de seguridad	
RNF-35	La aplicación será entregada a la dirección del proyecto, son ellos los responsables de velar porque no sufra cambios.

Tabla 4: Requerimientos No Funcionales del Sistema.

2.5.3. Definición de los casos de uso

Definición de los Actores

Actores	Justificación
Usuario	Es quien diseña la personalización de los documentos de identificación. Es el que interactúa con el sistema y es además para

	quien está destinado el mismo.

Tabla 5: Definición y descripción de los actores del sistema.

2.5.4. Breve Descripción de los Casos de Uso

CUS – 1	Crear Plantilla
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide crear una nueva plantilla de documento de identificación. Se muestra los tipos de documentos disponibles para esta acción y luego inmediatamente el sistema crea la nueva plantilla, finalizando el caso de uso.
Referencia	RF-1, RF-2

Tabla 6: Definición y descripción del Casos de Uso del Sistema Crear Plantilla.

CUS – 2	Abrir Plantilla
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide abrir una plantilla existente con la aplicación de diseño de plantillas de documentos de identificación. El usuario navega en busca de la plantilla que desea abrir y finalmente el sistema la abre en su área de diseño finalizando así el caso de uso.
Referencia	RF-6

Tabla 7: Definición y descripción del Casos de Uso del Sistema Abrir Plantilla.

CUS – 3	Guardar Plantilla
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide guardar una plantilla que ha creado o modificado en la aplicación. El caso de uso

	finaliza cuando el sistema guarda la plantilla deseada.
Referencia	RF-5

Tabla 8: Definición y descripción del Casos de Uso del Sistema Guardar Plantilla.

CUS – 4	Crear Objeto
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide crear un objeto en el área de diseño de la aplicación. Este selecciona el objeto deseado, define sus parámetros y guías de dibujo y luego finaliza el caso de uso cuando el sistema dibuja finalmente el objeto en el área de diseño.
Referencia	RF-3, RF-4

Tabla 9: Definición y descripción del Casos de Uso del Sistema Crear Objeto.

CUS – 5	Copiar Objeto
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario selecciona un objeto del área de diseño con el fin de hacer en algún momento un duplicado. El caso de uso finaliza cuando el sistema crea una copia del objeto en el *Clipboard* (*Portapapeles*) de la aplicación.
Referencia	RF-8

Tabla 10: Definición y descripción del Casos de Uso del Sistema Copiar Objeto.

CUS – 6	Cortar Objeto
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario desea cortar un objeto para luego restaurarlo pegándolo o sólo como una vía a la eliminación. El caso de uso finaliza cuando el

	sistema realiza una copia en el *Clipboard* de la aplicación y deja de mostrar el objeto en el área de diseño.
Referencia	RF-8

Tabla 11: Definición y descripción del Casos de Uso del Sistema Cortar Objeto.

CUS – 7	Pegar Objeto
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario desea pegar un objeto que tiene en el *Clipboard* de la aplicación. El sistema crea una copia del objeto del Clipboard en el área de diseño y finaliza así el caso de uso.
Referencia	RF-8

Tabla 12: Definición y descripción del Casos de Uso del Sistema Pegar Objeto.

CUS – 8	Mover Objeto
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario desea mover un objeto que tiene en el área de diseño de la aplicación. El sistema moverá el objeto hacia la posición indicada por el usuario y finalizará así el caso de uso.
Referencia	RF-3

Tabla 13: Definición y descripción del Casos de Uso del Sistema Mover Objeto.

CUS – 9	Eliminar Objeto
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario selecciona un objeto que tiene en el área de diseño y finaliza cuando el sistema elimina del área de diseño el objeto designado por el usuario.
Referencia	RF-8

Tabla 14: Definición y descripción del Casos de Uso del Sistema Eliminar Objeto.

CUS – 10	Editar Configuración
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide interactuar con el código de la plantilla que está diseñando. El sistema le muestra el área de codificación al usuario. El usuario hace los cambios que desee y al volver a la vista de diseño el sistema mostrará los cambios realizados finalizando el caso de uso.
Referencia	RF-11

Tabla 15: Definición y descripción del Casos de Uso del Sistema Editar Configuración.

CUS – 11	Mostrar Vista Previa
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide ver cómo queda el diseño actual en un modelo final de documento de identificación. El usuario tiene que especificar sus datos de personalización (Nombre, Apellidos, etc.) para ver la Vista Previa, dado que esta debería ser del documento personalizado. El sistema muestra la vista previa del documento y finaliza así el caso de uso.
Referencia	RF-9

Tabla 16: Definición y descripción del Casos de Uso del Sistema Mostrar Vista Previa.

CUS – 12	Especificar Elemento de Seguridad
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide aplicarle algún elemento de seguridad al diseño de la plantilla del documento de identificación. El sistema le da a escoger al

	usuario que elementos de seguridad aplicar y este al escoger lo configura. El sistema finaliza el caso de uso al insertar el elemento de seguridad indicado en la plantilla y lo representa visualmente en el documento, de alguna manera; o si el usuario cancela la operación.
Referencia	RF-12

Tabla 17: Definición y descripción del Casos de Uso del Sistema Especificar Elemento de Seguridad.

CUS – 13	Gestionar Estándar
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario decide gestionar los estándares de documentos de identificación. El usuario podrá crear un nuevo diseño base de estándar, modificar alguno que puede haber estado mal creado o duplicado, o en otro caso, eliminar algún estándar de documento de identificación por error de configuración al crearlo. En el primer escenario el sistema pide intervención del usuario para realizar la configuración del nuevo estándar y tras guardar cambios finaliza el caso de uso; en el segundo pide intervención del usuario para que este reconsidere la configuración de un estándar presente en la aplicación y tras guardar cualquier cambio realizado por el usuario, finaliza el caso de uso; en el último caso le solicita confirmación de la acción antes de proceder a eliminar el estándar y finalizar así el caso de uso.
Referencia	RF-10

Tabla 18: Definición y descripción del Casos de Uso del Sistema Gestionar Estándar.

CUS – 14	Asociar Propiedades
Actor	Usuario
Descripción	El caso de uso comienza cuando el usuario desea asociar los datos de personalización de los objetos del área de diseño.
Referencia	RF-4

Tabla 19: Definición y descripción del Casos de Uso del Sistema Asociar Propiedades.

2.5.5. Diagrama de Casos de Uso del Sistema

Un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software que proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. Estos describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, estos permiten establecer un acuerdo entre clientes y desarrolladores sobre las condiciones y requisitos que debe cumplir el sistema.

Un diagrama de casos de uso (Use Case Diagram) es una representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones (Ver Fig. 4). Todo sistema tiene como mínimo un diagrama Main Use Case, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso). (Vilas, 2001)

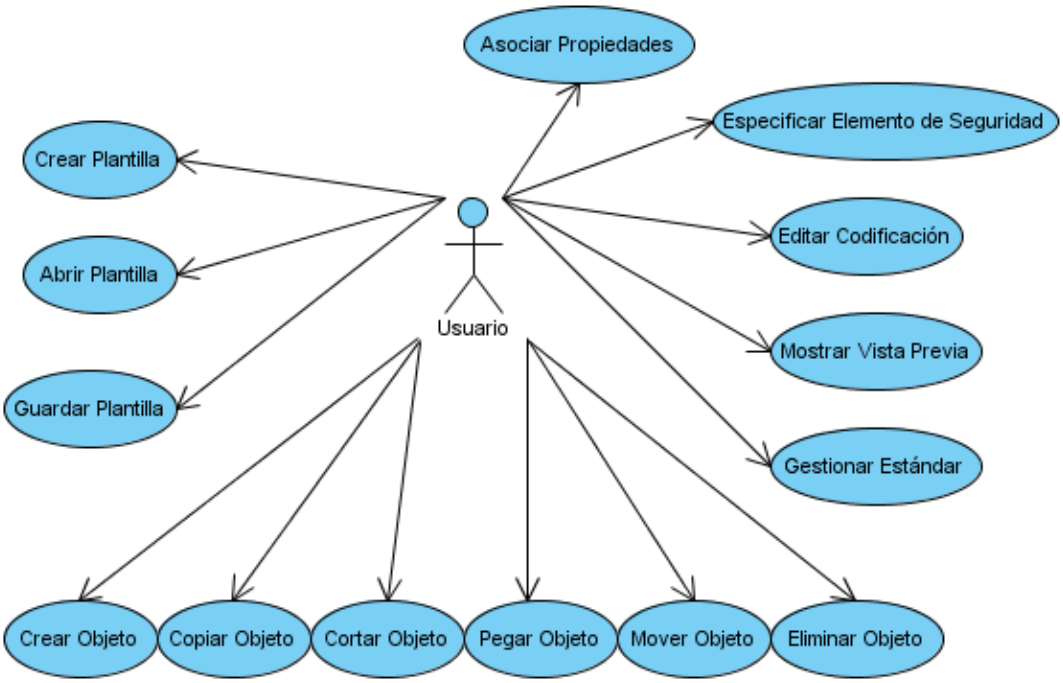


Figura 4: La figura muestra el Diagrama de Casos de Uso del Sistema.

2.6. Descripción de los Casos de Uso del Sistema[40]

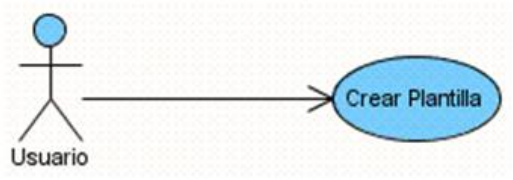


Figura 5: Representación del Caso de Uso Crear Plantilla.

Caso de Uso	**Crear Plantilla**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide crear una nueva plantilla de documento de identificación. Se muestra los tipos de documentos disponibles para esta acción y luego inmediatamente el sistema crea la nueva plantilla, finalizando el caso de uso.
Propósito	Crear una nueva plantilla para un documento de identificación.
Precondiciones	

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción Crear plantilla.	2. El sistema le muestra al usuario los tipos de documentos que puede diseñar para crear una nueva plantilla.
3. El usuario selecciona el tipo de documento deseado para crear la nueva plantilla.	4. El sistema muestra al usuario una ventana para que escoja que tipo de estándar será aplicado al documento seleccionado.
5. El usuario selecciona el tipo de estándar para el diseño del documento.	6. El sistema crea la nueva plantilla
	7. Termina el caso de uso.
Curso Alternos	

[40] CUS: Siglas en español para significar Caso de Uso del Sistema

Prioridad	Crítico

Tabla 20: Descripción del Casos de Uso Crear Plantilla.

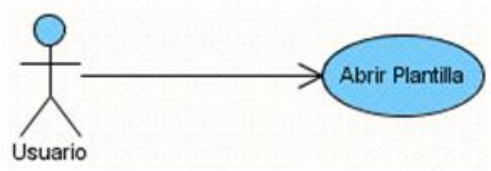


Figura 6: Representación del Caso de Uso Abrir Plantilla.

Caso de Uso	**Abrir Plantilla**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide abrir una plantilla existente con la aplicación de diseño de plantillas de documentos de identificación. El usuario navega en busca de la plantilla que desea abrir y finalmente el sistema la abre en su área de diseño finalizando así el caso de uso.
Propósito	Abrir un documento de plantilla existente.
Precondiciones	El usuario decide interactuar con la aplicación de diseño de plantillas.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario escoge la opción “Abrir una plantilla existente”.	2. El sistema visualiza la pantalla para que el usuario explore el ordenador en búsqueda de un fichero de plantilla.
3. El usuario escoge el fichero de plantilla deseada de su ordenador.	4. El sistema oculta la pantalla de búsqueda y visualiza la plantilla escogida por el usuario, dibujando en pantalla los elementos gráficos de la plantilla. 5. Finaliza el caso de uso.
Curso Alternos	

Prioridad	Crítico

Tabla 21: Descripción del Casos de Uso Abrir Plantilla.

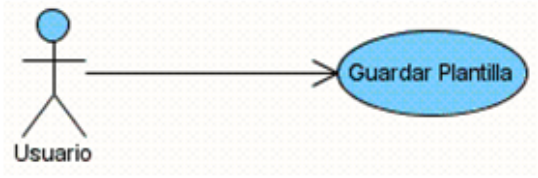


Figura 7: Representación del Caso de Uso Guardar Plantilla.

Caso de Uso	**Guardar Plantilla**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide guardar una plantilla que ha creado o modificado en la aplicación. El caso de uso finaliza cuando el sistema guarda la plantilla deseada.
Propósito	Guardar el diseño de plantilla realizado en la aplicación.
Precondiciones	El usuario ha creado una plantilla o la ha abierto y realizado cambios.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1.El usuario decide guardar la nueva plantilla, para ello debe dirigirse a una de las siguientes opciones: - Guardar en el menú editar. - Guardar en la barra de herramienta. - Guardar por acceso de teclado Ctrl + S. 3. El usuario indica la dirección donde se guardará la plantilla y el nombre correspondiente a esta y presiona la opción Guardar.	2. El sistema muestra la pantalla de exploración para que el usuario indique el destino de la plantilla. 4. El sistema guarda la plantilla. 5. Termina el caso de uso.
Curso Alternos	

1. El usuario modifica la plantilla abierta y decide guardarla	2. El sistema sobrescribe el archivo modificado. 3. Termina el caso de uso.
1. El usuario modifica la plantilla abierta y decide guardar la plantilla con otro nombre.	
Prioridad	Crítico

Tabla 22: Descripción del Casos de Uso Guardar Plantilla.

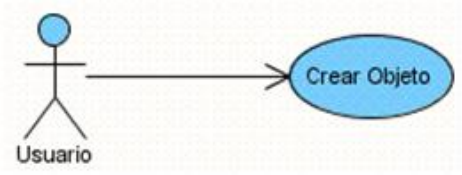


Figura 8: Representación del Caso de Uso Crear Objeto.

Caso de Uso	**Crear Objeto**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide crear un objeto en el área de diseño de la aplicación. Este selecciona el objeto deseado, define sus parámetros y guías de dibujo y luego finaliza el caso de uso cuando el sistema dibuja finalmente el objeto en el área de diseño.
Propósito	Añadir una modificación en el diseño de la plantilla actualmente abierta o creada.
Precondiciones	El usuario crea o abre un plantilla existente.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un objeto del área de herramientas de diseño. 3. El usuario aplica el objeto seleccionado al área de diseño de la plantilla. ¡	2. El sistema marca el objeto seleccionado por el usuario. 4. El sistema muestra la plantilla con el elemento creado por el usuario. 5. Termina el caso de uso.
Curso Alternos	

Prioridad	Crítico

Tabla 23: Descripción del Casos de Uso Crear Objeto.

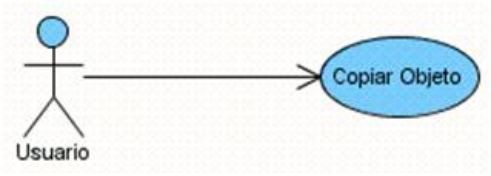


Figura 9: Representación del Caso de Uso Copiar Objeto.

Caso de Uso	**Copiar Objeto**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario selecciona un objeto del área de diseño con el fin de hacer en algún momento un duplicado. El caso de uso finaliza cuando el sistema crea una copia del objeto en el *Clipboard* de la aplicación.
Propósito	Crear una copia exacta de un objeto del área de diseño de plantilla.
Precondiciones	El usuario tiene al menos un objeto en el área de diseño.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un objeto del área de diseño. 3.El usuario decide copiar el objeto seleccionado, para ello debe dirigirse a una de las siguientes opciones: - Copiar en el menú editar. - Copiar en la barra de herramienta. - Copiar por acceso de teclado Ctrl + C.	2. El sistema marca el objeto seleccionado por el usuario. 4. El sistema realiza la copia del elemento seleccionado pero no muestra cambio evidente. 5. Termina el caso de uso.
Curso Alternos	
Prioridad	Crítico

Tabla 24: Descripción del Casos de Uso Copiar Objeto.

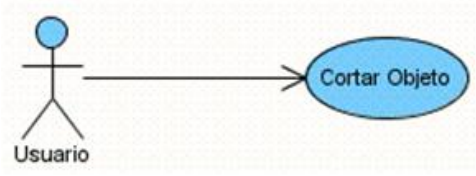


Figura 10: Representación del Caso de Uso Cortar Objeto.

Caso de Uso	**Cortar Objeto**	
Actores	Usuario	
Resumen	El caso de uso comienza cuando el usuario desea cortar un objeto para luego restaurarlo pegándolo o sólo como una vía a la eliminación. El caso de uso finaliza cuando el sistema realiza una copia en el *Clipboard* de la aplicación y deja de mostrar el objeto en el área de diseño.	
Propósito	Pegar el objeto en otro lugar o eliminarlo del área de diseño usando la acción cortar.	
Precondiciones	El usuario tiene al menos un objeto en el área de diseño.	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. El usuario selecciona un objeto del área de diseño. 3.El usuario decide cortar el objeto seleccionado, para ello debe dirigirse a una de las siguientes opciones: - Cortar en el menú editar. - Cortar en la barra de herramienta. - Cortar por acceso de teclado Ctrl + X.		2. El sistema marca el objeto seleccionado por el usuario. 4. El sistema corta el elemento seleccionado del área de diseño y deja de mostrarlo al usuario. 5. Termina el caso de uso.
Curso Alternos		
Prioridad	Crítico	

Tabla 25: Descripción del Casos de Uso Cortar Objeto.

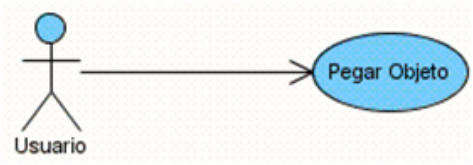


Figura 11: Representación del Caso de Uso Pegar Objeto.

Caso de Uso	**Pegar Objeto**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario desea pegar un objeto que tiene en el *Clipboard* de la aplicación. El sistema crea una copia del objeto del Clipboard en el área de diseño y finaliza así el caso de uso.
Propósito	Pegar el objeto en el área de diseño de la plantilla de documento de identificación.
Precondiciones	El usuario tiene un objeto copiado o cortado presente en el Clipboard de la aplicación.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el área de diseño. 2. El usuario decide pegar el objeto copiado o cortado con anterioridad, para ello debe dirigirse a una de las siguientes opciones: - Pegar en el menú editar. - Pegar en la barra de herramienta. - Pegar por acceso de teclado Ctrl + V.	3. El sistema pega el elemento correspondiente en el área de diseño y lo muestra seleccionado al usuario. 4. Termina el caso de uso.
Curso Alternos	
Prioridad	Crítico

Tabla 26: Descripción del Casos de Uso Pegar Objeto.

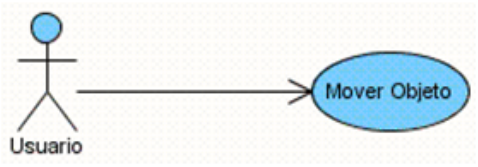


Figura 12: Representación del Caso de Uso Mover Objeto.

Caso de Uso	**Mover Objeto**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario desea mover un objeto que tiene en el área de diseño de la aplicación. El sistema moverá el objeto hacia la posición indicada por el usuario y finalizará así el caso de uso.
Propósito	Cambiar de posición un objeto en el área de diseño de la plantilla de documento de identificación.
Precondiciones	El usuario tiene al menos un objeto en el área de diseño.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un objeto del área de diseño. 3. El usuario mueve el puntero del mouse al área del centro del objeto. 5. El usuario presiona el botón de acción del mouse y mueve el objeto hasta una posición deseada y libera el botón del mouse para posicionar el objeto.	2. El sistema marca el objeto seleccionado por el usuario. 4. El sistema le muestra el puntero de mover objetos para indicarle al usuario que está en la zona apropiada. 6. El sistema reubica el objeto seleccionado y lo visualiza en la posición indicada por el usuario. 7. Termina el caso de uso.
Curso Alternos	

Prioridad	Crítico

Tabla 27: Descripción del Casos de Uso Mover Objeto.

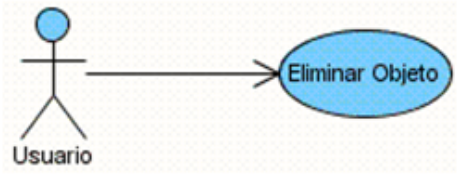


Figura 13: Representación del Caso de Uso Eliminar Objeto.

Caso de Uso	**Eliminar Objeto**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario selecciona un objeto que tiene en el área de diseño y finaliza cuando el sistema elimina del área de diseño el objeto designado por el usuario.
Propósito	Eliminar un objeto existente en el área de diseño de la aplicación.
Precondiciones	El usuario debe tener al menos un objeto en el área de diseño.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el objeto que desee eliminar. 3. El usuario decide eliminar el objeto seleccionado con anterioridad, para ello debe dirigirse a una de las siguientes opciones: - Eliminar en el menú editar. - Eliminar en la barra de herramienta. - Eliminar por acceso de teclado **Delete**.	2. El sistema muestra el objeto seleccionado. 4. El sistema elimina el objeto del área de diseño de la aplicación. 5. Termina el caso de uso.

Curso Alternos	

Prioridad	Crítico

Tabla 28: Descripción del Casos de Uso Eliminar Objeto.

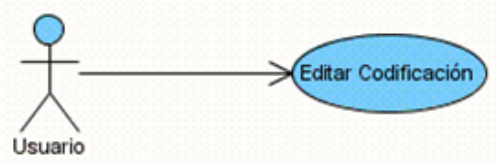


Figura 14: Representación del Caso de Uso Editar Codificación.

Caso de Uso	**Editar Codificación**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide interactuar con el código de la plantilla que está diseñando. El sistema le muestra el área de codificación al usuario. El usuario hace los cambios que desee y al volver a la vista de diseño el sistema mostrará los cambios realizados finalizando el caso de uso.
Propósito	Intervenir directamente en el código y editar algo que no realice el programa de forma visual.
Precondiciones	El usuario debe poseer una plantilla abierta en el área de diseño.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el botón *Ver Código* para ver la sección del código de la plantilla. 3. El usuario realiza la edición deseada en el código. 4. El usuario regresa a la vista de diseño de plantilla.	2. El sistema muestra la sección del código de la plantilla creada hasta el momento. 5. El sistema muestra en pantalla la vista de diseño reflejando los cambios visibles al usuario, si es que existe alguno. 4. Termina el caso de uso.
Curso Alternos	

Prioridad	Crítico

Tabla 29: Descripción del Casos de Uso Editar Codificación.

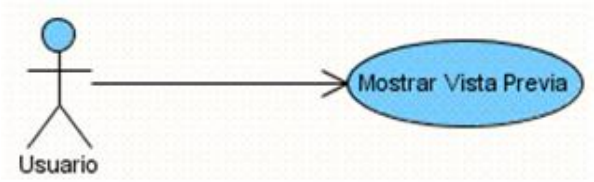


Figura 15: Representación del Caso de Uso Mostrar Vista Previa.

Caso de Uso	**Mostrar Vista Previa**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide ver cómo queda el diseño actual en un modelo final de documento de identificación. El usuario tiene que especificar sus datos de personalización (Nombre, Apellidos, etc.) para ver la Vista Previa, dado que esta debería ser del documento personalizado. El sistema muestra la vista previa del documento y finaliza así el caso de uso.
Propósito	Ver proyección final de la plantilla en un documento de identificación virtual.
Precondiciones	El usuario debe tener abierta o creada una plantilla en el área de diseño de la aplicación.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el botón *Vista Previa* para ver la muestra del documento final. 3. El usuario llena los datos necesarios para crear la Vista Previa del documento que está personalizando y presiona el botón Aceptar.	2. El sistema le muestra una ventana con los campos personalizables que el usuario ha definido en la plantilla para que los llene con datos. 4. El sistema muestra la sección *Vista Previa* con la plantilla generada final y con datos de ejemplo. 3. Termina el caso de uso.

Curso Alternos	
Prioridad	Crítico

Tabla 30: Descripción del Casos de Uso Mostrar Vista Previa.

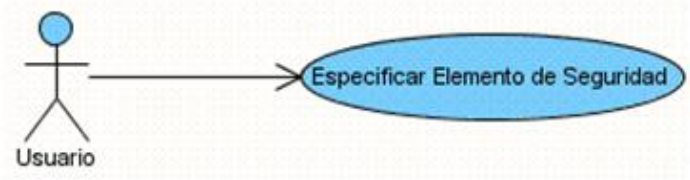


Figura 16: Representación del Caso de Uso Especificar Elemento de Seguridad.

Caso de Uso	**Especificar Elemento de Seguridad**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide aplicarle algún elemento de seguridad al diseño de la plantilla del documento de identificación. El sistema le da a escoger al usuario que elementos de seguridad aplicar y este al escoger lo configura. El sistema finaliza el caso de uso al insertar el elemento de seguridad indicado en la plantilla y lo representa visualmente en el documento, de alguna manera; o si el usuario cancela la operación.
Propósito	Crear una plantilla con elementos de seguridad para su protección.
Precondiciones	El usuario debe tener abierta o creada una plantilla en el área de diseño de la aplicación.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción *Elemento de Seguridad*.	2. El sistema muestra la ventana con los tipos de elementos de seguridad que se le pueden añadir a la plantilla.
3. El usuario selecciona el elemento de seguridad deseado.	4. El sistema le muestra al usuario una ventana para que el usuario configure el elemento de seguridad especificado y muestra los botones *Aceptar* y *Cancelar*.

5. El usuario realiza la personalización del elemento y acepta la acción realizada.	6. El sistema inserta el elemento de seguridad personalizado, lo representa si es posible y oculta la ventana de elementos de seguridad. 7. Termina el caso de uso.
Curso Alternos	
5. El usuario decide cancelar la operación.	6. El sistema oculta la ventana de elementos de seguridad.
Prioridad	Crítico

Tabla 31: Descripción del Casos de Uso Especificar Elemento de Seguridad.

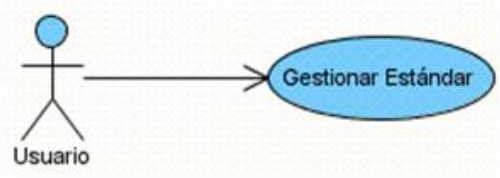


Figura 17: Representación del Caso de Uso Gestionar Estándar.

Caso de Uso	**Gestionar Estándar**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario decide gestionar los estándares de documentos de identificación. El usuario podrá crear un nuevo diseño base de estándar, modificar alguno que puede haber estado mal creado o duplicado, o en otro caso, eliminar algún estándar de documento de identificación por error de configuración al crearlo. En el primer escenario el sistema pide intervención del usuario para realizar la configuración del nuevo estándar y tras guardar cambios finaliza el caso de uso; en el segundo pide intervención del usuario para que este reconsidere la configuración de un estándar presente en la aplicación y tras guardar cualquier cambio realizado por el usuario, finaliza el caso de uso; en el último caso le solicita confirmación de la acción antes de proceder a eliminar el estándar y finalizar así el caso de uso.
Propósito	Habilitar nueva funcionalidad en la creación de plantillas de documentos de identificación. Corregir una deficiencia de los estándares presentes en la creación de plantillas de documentos de identificación. Eliminar un estándar de los

	presentes en la aplicación.
Precondiciones	Escenario 1: El usuario decide añadir un nuevo estándar de plantilla no existente en la aplicación. Escenario 2: El usuario detecta alguna irregularidad en los estándares incluidos o añadidos a la aplicación y decide corregirla. Escenario 3: El usuario decide añadir un nuevo estándar de plantilla no existente en la aplicación.

Flujo normal de eventos	
Escenario 1: "Agregar Estándar"	
Acción del actor	**Respuesta del sistema**
3. El usuario selecciona del menú superior de la sección *Administración*, la opción *Agregar Estándar*. 3. El usuario llena los datos que requiera la configuración del nuevo estándar de plantilla de documento de identificación. 4. El usuario guarda los cambios realizados al estándar creado.	2. El sistema muestra la ventana de configuración de un nuevo estándar. 5. El sistema adiciona el estándar a su configuración y adiciona la entrada al menú inicial. 8. Termina el caso de uso.
Curso Alternos	

Prioridad	Crítico
Escenario 2: "Modificar Estándar"	
Acción del actor	**Respuesta del sistema**

1. El usuario selecciona del menú superior de la sección *Administración*, la opción *Modificar Estándar*. 3. El usuario selecciona el estándar que desea modificar y acepta la selección. 5. El usuario modifica los datos que requiera la nueva configuración del estándar de plantilla de documento de identificación. 6. El usuario guarda los cambios realizados al estándar creado.	2. El sistema muestra una ventana con la lista de estándares con que cuenta actualmente la aplicación para que el usuario seleccione uno. 4. El sistema abre la ventana de configuración del estándar indicado por el usuario para que sea editada. 7. El sistema reemplaza el estándar viejo con el de la nueva configuración. 8. Termina el caso de uso.
Curso Alternos	
Prioridad	Crítico
Escenario 3: "Eliminar Estándar"	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona del menú superior de la Sección *Administración*, la opción *Eliminar Estándar*. 3. El usuario selecciona el estándar que desea eliminar de la aplicación y presiona el botón *Eliminar*. 5. El usuario elige *Sí* a la pregunta realizada por el sistema.	2. El sistema muestra una ventana con la lista de estándares con que cuenta actualmente la aplicación para que el usuario seleccione cual desea eliminar. 4. El sistema pregunta al usuario si está seguro que desea eliminar el estándar seleccionado de la aplicación, y aclarándole que la operación no tiene vuelta atrás. 6. El sistema elimina el estándar y lo suprime del menú inicial.

	7. Termina el caso de uso.
Curso Alternos	
5. El usuario elige *No* para cancelar la operación de eliminado.	6. El sistema cancela la operación.
Prioridad	Crítico

Tabla 32: Descripción del Casos de Uso Gestionar Estándar.

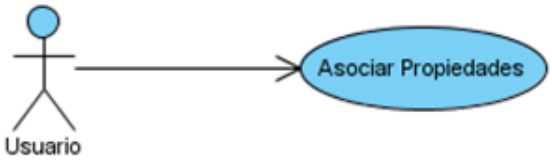


Figura 18: Representación del Caso de Uso Asociar Propiedades.

Caso de Uso	**Asociar Propiedades**
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario desea asociar los datos de personalización de los objetos del área de diseño.
Propósito	Personalizar un objeto presente en el área de diseño.
Precondiciones	Debe existir al menos un objeto en el área de diseño.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un objeto del área de diseño. 3. El usuario selecciona la opción Propiedades del objeto seleccionado. 5. El usuario indica los datos personalizados del elemento. (Ej. Identificador, color, tamaño, tipo de letra)	2. El sistema marca el objeto seleccionado por el usuario. 4. El sistema muestra la interfaz correspondiente. 6. El sistema aplica los datos al objeto seleccionado. 7. Termina el caso de uso.

Curso Alternos	
Prioridad	Crítico

Tabla 33: Descripción del Casos de Uso Asociar Propiedades.

2.7.Conclusiones Parciales del Capítulo

Se puede concluir que no existe una aplicación en el mercado que satisfaga la necesidad básica de creación de plantillas de documentos de identificación en su forma más sencilla. Las ventajas de la solución propuesta abarcan los estándares internacionales de creación de documentos de identificación de la OACI, ejecución en múltiples plataformas (Windows, Linux, Mac OS X), desarrollo con uso de software libre (Java) y formato de salida libre y extensible (SVG). La interrelación de los componentes bases de la aplicación como la imagen, los datos personales y el propio documento de identificación, se refleja a través de la realización de un Modelo de Dominio. Se han planteado las funcionalidades principales de la aplicación que incluyen la creación de plantillas de documentos de identificación con formato de salida SVG, la inclusión de características especiales como elementos de seguridad y la posibilidad de visualizar y editar el código fuente de la plantilla.

Capítulo 3 - Análisis y Diseño del Sistema

En este capítulo se realiza el análisis y 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.

3.1.Análisis y Diseño

El Flujo de Trabajo de Análisis y Diseño se propone comprender perfectamente los requisitos del software y transformarlos a un diseño que indique como debe ser implementado el software. (UCI, 2009)

Los objetivos de este flujo son:

- Transformar los requerimientos en un diseño de cómo va a ser implementado el sistema,
- Evolucionar hacia una arquitectura del software robusta.
- Adaptar el diseño para que coincida con el ambiente de implementación, diseñando el sistema con un enfoque hacia el rendimiento. (UCI, 2009)

El diseño del sistema se ocupa de desarrollar las directrices propuestas durante el proceso de análisis en términos de aquella configuración que tenga más posibilidades de satisfacer los objetivos planteados, tanto del punto de vista funcional como no funcional, guiándose para este por los requerimientos levantados en dicha etapa.

El proceso de diseño de un sistema complejo se puede realizar de forma descendente como se muestra a continuación:

- Diseño de alto nivel (descomposición del sistema a diseñar en subsistemas menos complejos).
- Diseño e implementación de cada uno de los subsistemas.
- Especificación consistente y completa del subsistema de acuerdo con el objetivo establecido en el análisis.
- Desarrollo según las especificaciones.
- Pruebas.
- Integración de todos los subsistemas.
- Validación del diseño.

En el transcurso de este capítulo se verán algunos de los aspectos antes mencionados, donde se explicarán y se presentarán una serie de diagramas pertenecientes al análisis, diseño e implementación del sistema, también se verán algunas de las pruebas que se le hicieron al software, llegando éste a su etapa final, la terminación del producto y la aceptación por parte del cliente. (Rational Software Corporation, 2002)

3.1.1. Definición del 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. (Sadín Odio, y otros, 2008)

3.1.2. 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. (Sadín Odio, y otros, 2008)

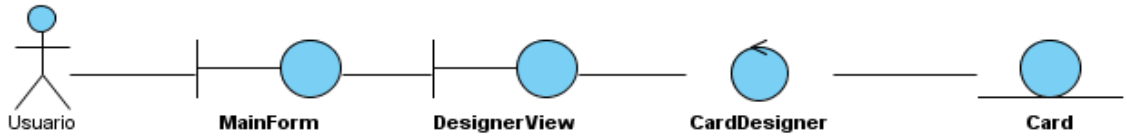


Figura 19: Diagrama de Clases del Modelo de Análisis. Casos de Uso: Crear Plantilla, Abrir Plantilla y Guardar Plantilla.

Representación Gráfica de todos los Diagramas de Clases del Análisis. (Ver Anexo 1)

3.1.3. Arquitectura

Definición

Es una vista estructural de alto nivel, ocurre muy tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales.

En el campo del software, la arquitectura nos identifica los elementos más importantes de un sistema así como sus relaciones. Nos da una visión global del sistema.

Su importancia está dada porque necesitamos arquitectura para entender el sistema, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar.

Un *estilo arquitectónico* define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas. (UCI, 2009)

La arquitectura del sistema se encuentra dividida en 3 capas o módulos sobre los cuales se sustenta el funcionamiento de la aplicación. El módulo de aplicación es donde se encuentra la parte de la aplicación encargada de crear todos los elemento que serán configurables según sus propiedades y tipos definidos por el usuario, además, se encarga de utilizar el módulo núcleo o módulo render para realizar operaciones sobre los elementos creados. El módulo núcleo o módulo render, es una herramienta desarrollada en Java llamada Batik, que es la utilizada por la aplicación para manipular, generar, crear, convertir, renderizar y mostrar el contenido SVG. El módulo de bajo nivel es utilizado por el módulo núcleo para el cumplimiento de sus operaciones. Este módulo normalmente no es utilizado por los desarrolladores directamente.

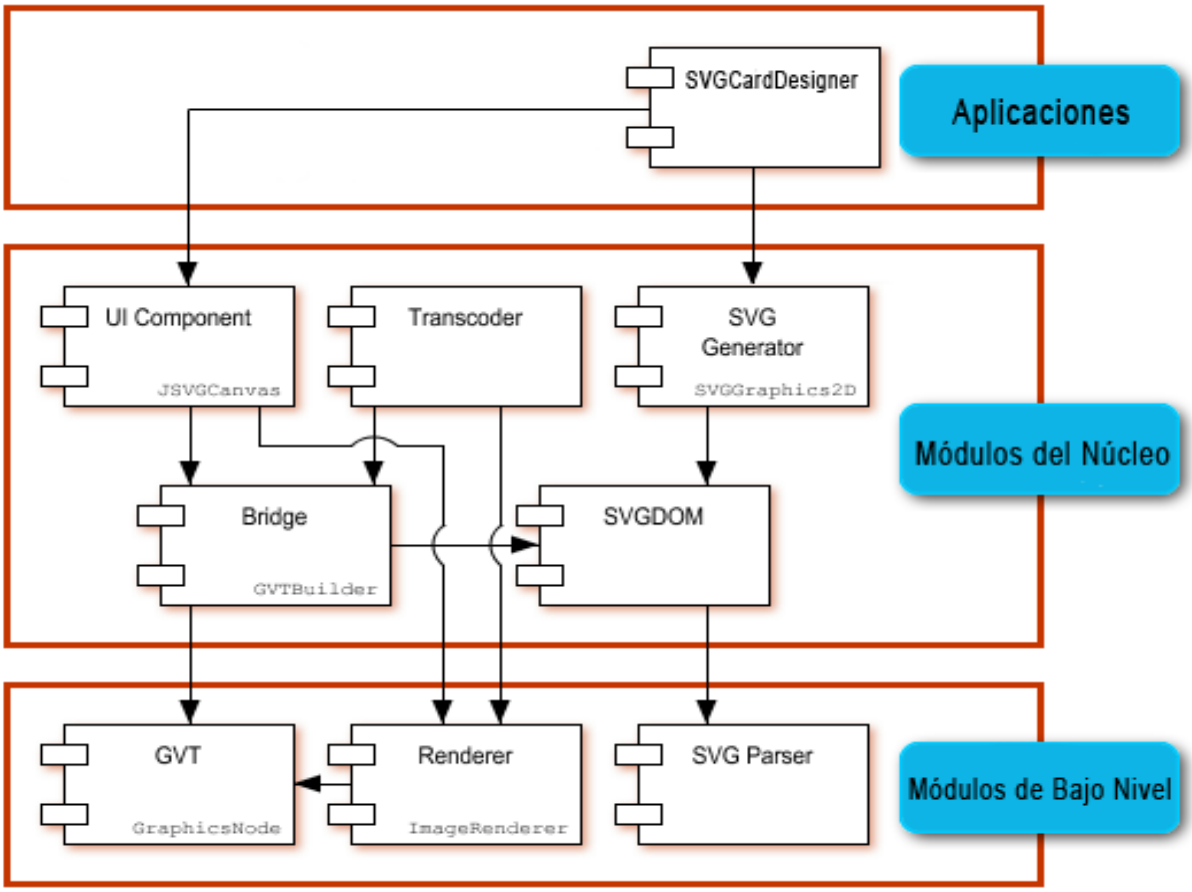


Figura 20: Representación gráfica de la arquitectura del sistema.

3.1.4. 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 el diseño, identificando aspectos claves de la estructura de este que puede ser aplicado en una gran cantidad de situaciones. 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, lo que demuestra su importancia.

En el desarrollo 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. (Sadín Odio, y otros, 2008)

3.1.5. Definición del Modelo de Diseño

Es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a implementación. Representa a los casos de uso en el dominio de la solución.

El Modelo de Diseño puede contener: los diagramas, las clases, paquetes, subsistemas, capsulas, protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo.

Para representar los diagramas del Modelo de Diseño se pueden emplear diferentes diagramas de UML tales como:

1. Diagramas de Clase.
2. Diagramas de Colaboración.
3. Diagramas de Estado.
4. Diagramas de Paquetes.
5. Diagramas de Secuencia.

Este artefacto es considerado necesario para cualquier proyecto. (MeRinde, 2009)

3.1.6. Diagramas de Interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases describen la forma en que grupos de objetos colaboran para proveer un comportamiento. Mientras que un diagrama de casos de uso presenta una visión

externa del sistema, la funcionalidad de dichos casos de uso se recoge como un flujo de eventos utilizando para ello interacciones entre sociedades de objetos.

Existen dos tipos de diagramas de interacción:

- Diagramas de Secuencia (Sequence Diagrams)
- Diagramas de Colaboración (Collaboration Diagrams) (Fernandez Vilas, 2001)

¿Qué es un Diagrama de Secuencia?

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren. (Pozo)

¿Qué es un Diagrama de Colaboración?

Un Diagrama de Colaboración muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). A diferencia de los Diagramas de Secuencia, los Diagramas de Colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia. (Pozo)

En el presente trabajo sólo se le han practicado Diagramas de Secuencia a los Casos de Uso.

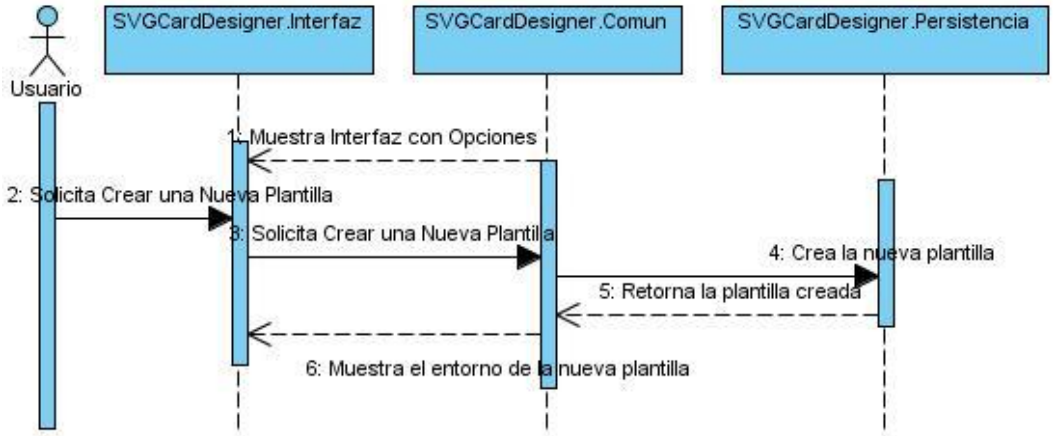


Figura 21: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Crear Plantilla.

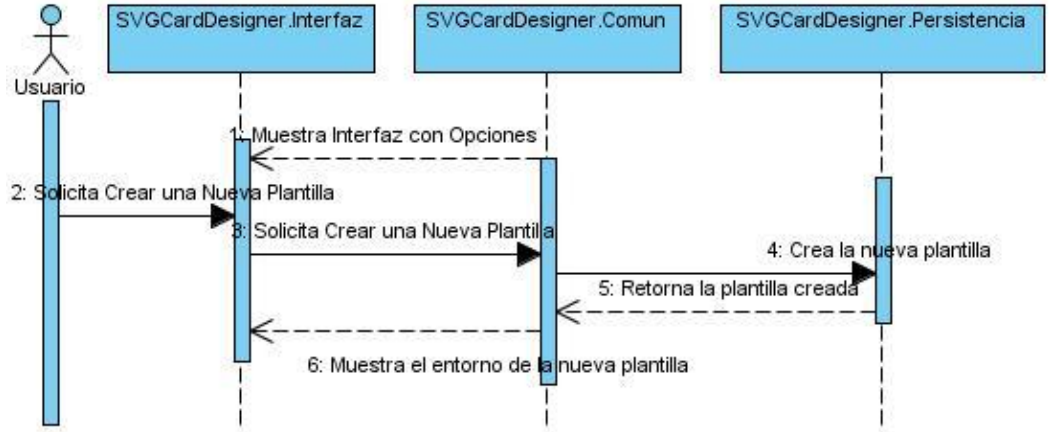


Figura 22: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Abrir Plantilla.

Representación de los restantes Diagramas de Secuencia. (Ver Anexo II)

3.1.7. Diagrama de Clases del Diseño

El Diagrama de Clase es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones.(ibiblio.org)

Las clases de diseño se especifican utilizando la sintaxis del lenguaje de programación elegido y tienen correspondencia directa con los métodos en la implementación. (Sadín Odio, y otros, 2008)

A continuación se muestran los Diagramas de Clases del Diseño:

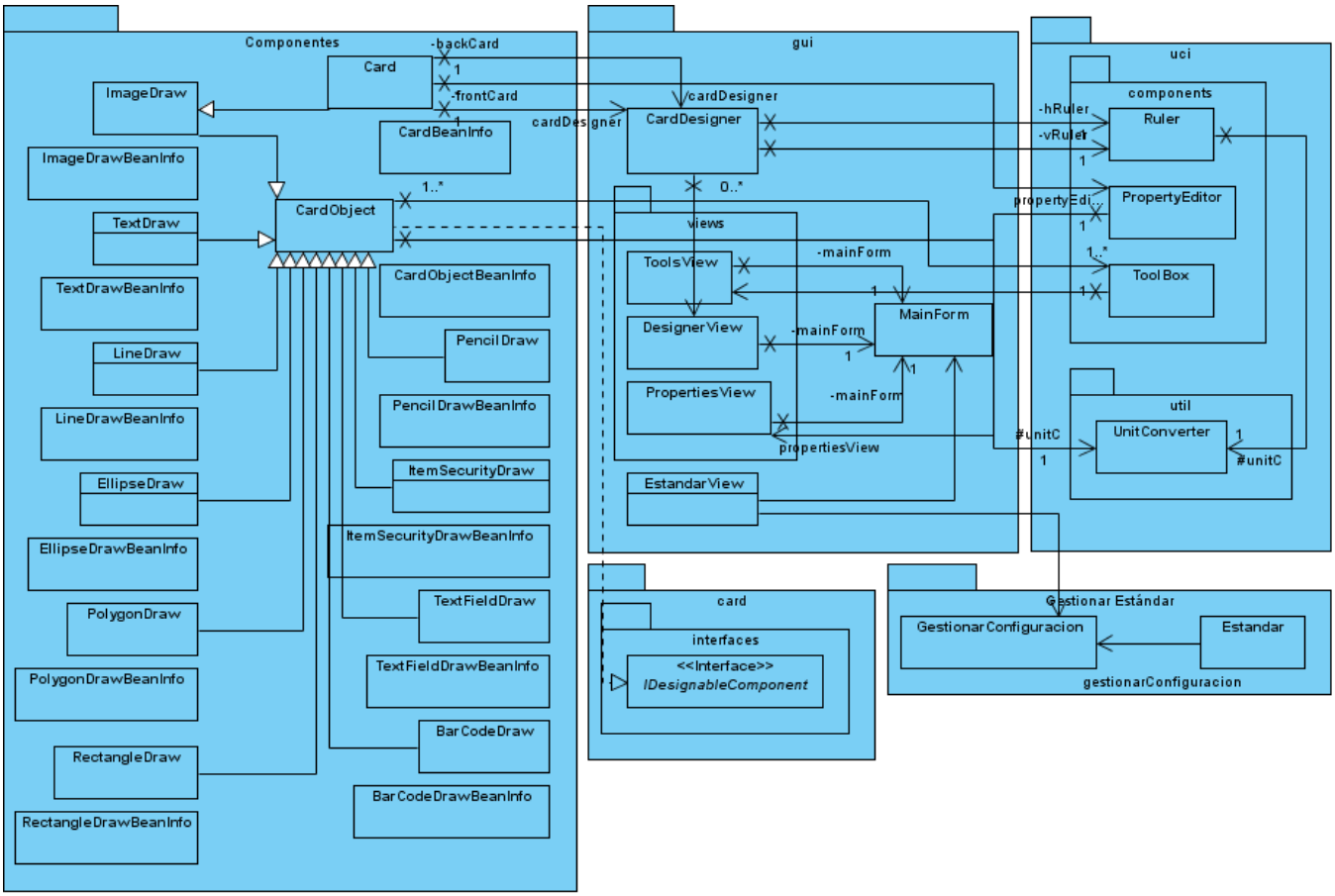


Figura 23: Representación gráfica del Diagrama de Clases del Diseño de la aplicación.

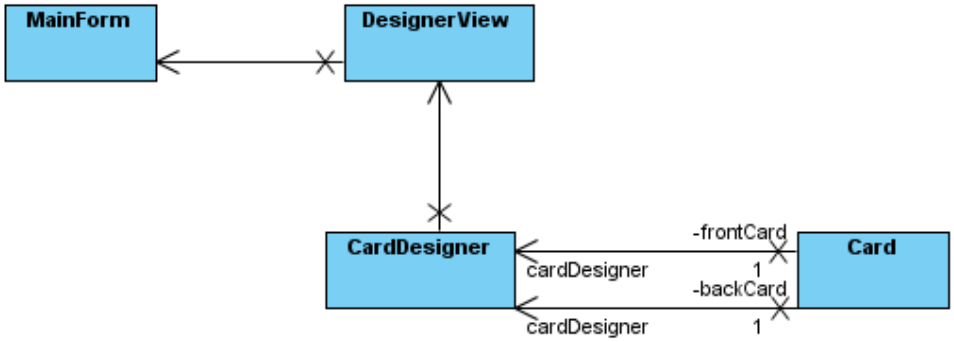


Figura 24: Representación gráfica del Diagrama de Clases del Diseño: Crear, Abrir y Guardar Plantilla.

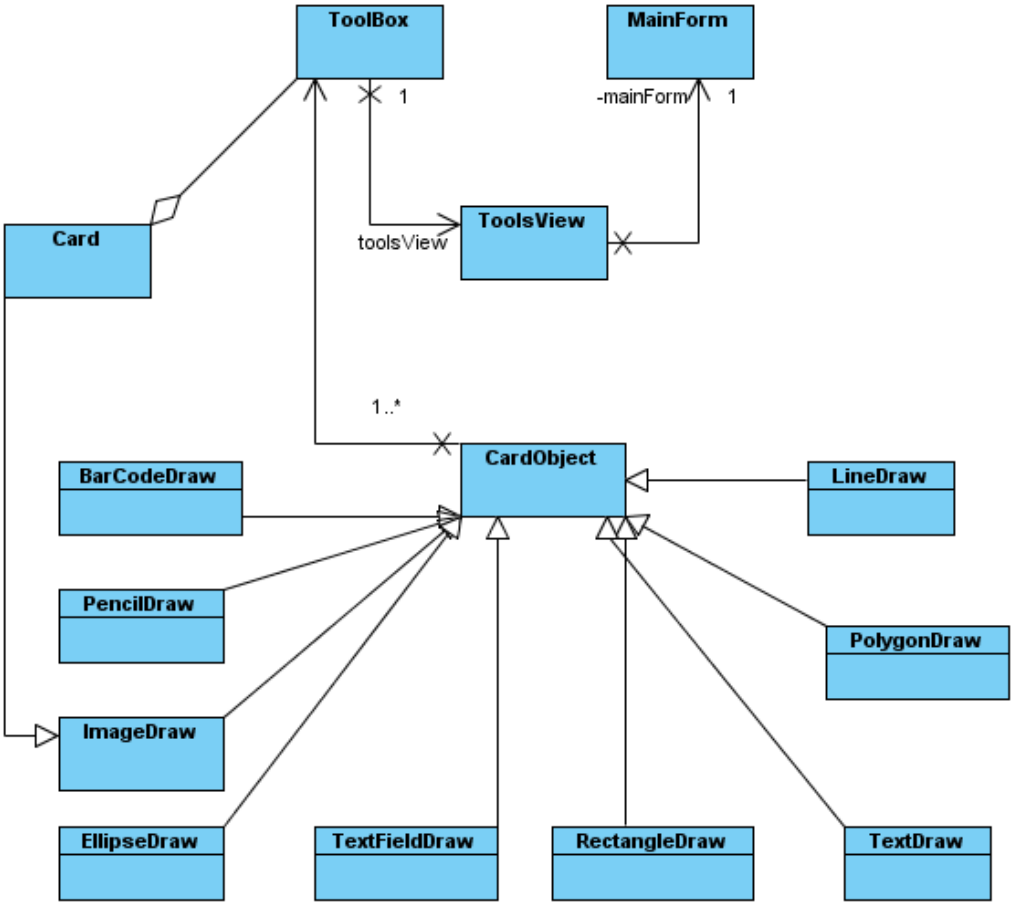


Figura 25: Representación gráfica del Diagrama de Clases del Diseño: Crear Objeto.

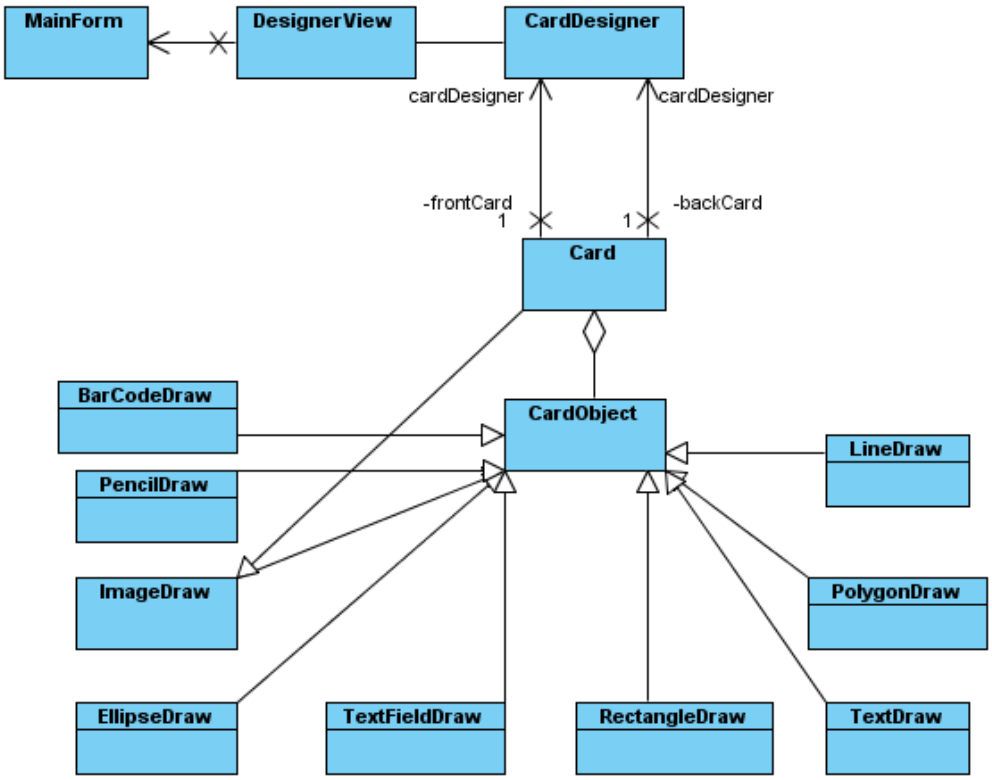


Figura 26: Representación gráfica del Diagrama de Clases del Diseño: Copiar, Cortar, Pegar y Eliminar Objeto.

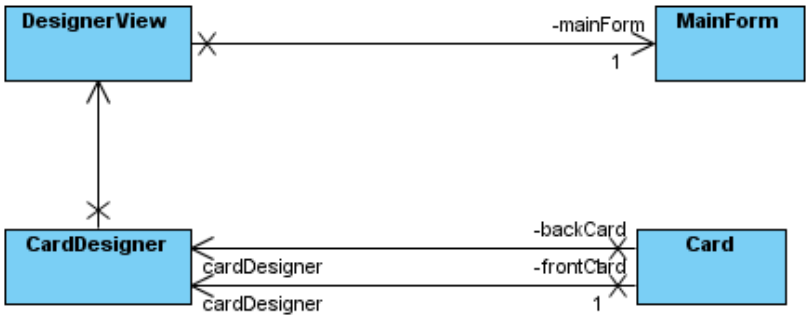


Figura 27: Representación gráfica del Diagrama de Clases del Diseño: Editar Codificación.

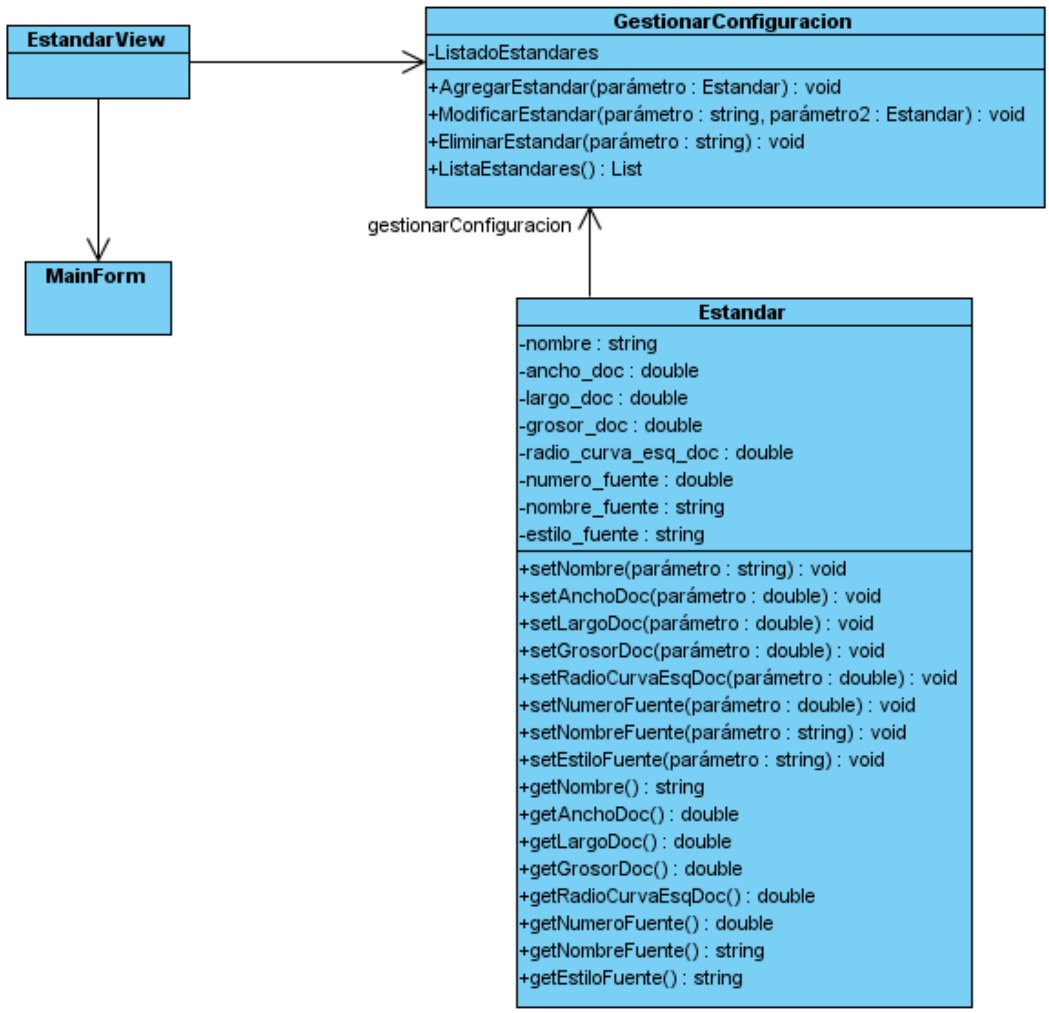


Figura 28: Representación gráfica del Diagrama de Clases del Diseño: Gestionar Estándar.

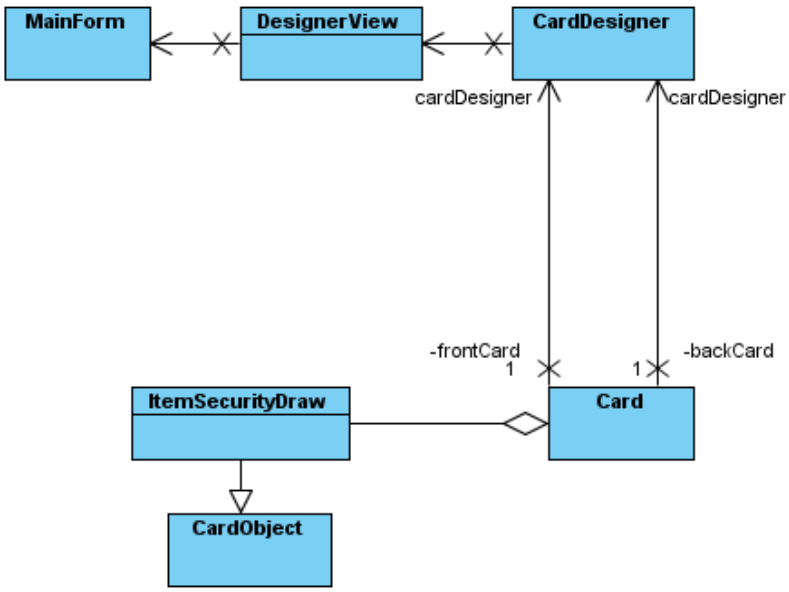


Figura 29: Representación gráfica del Diagrama de Clases del Diseño: Especificar Elemento de Seguridad.

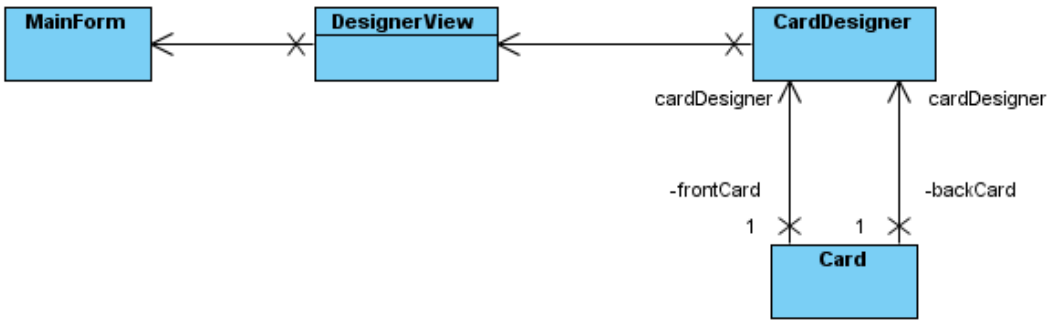


Figura 30: Representación gráfica del Diagrama de Clases del Diseño Mostrar Vista Previa.

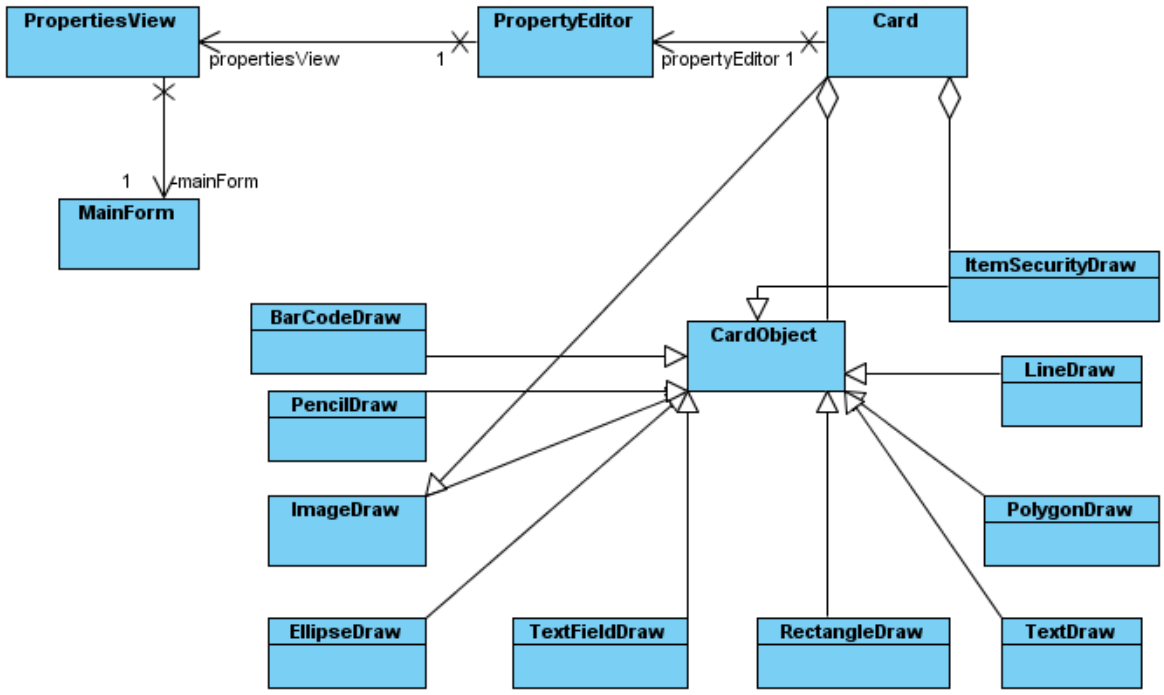


Figura 31: Representación gráfica del Diagrama de Clases del Diseño: Asociar Propiedades.

Para la representación detallada de las clases, refiérase al Anexo 3.

3.1.8. Descripción de las Principales Clases Utilizadas

En esta sección se hará una descripción de aquellas clases usadas en el desarrollo de la aplicación: las clases Entidad, las Controladoras y las clases Interfaz.

Nombre: CardDesigner	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de gestionar las operaciones sobre el documento del área de diseño como la escala del documento, si se muestran las reglas, si se le adicionan objetos al documento, o si se diseña el frente o la parte trasera del documento, entre otras operaciones.

Tabla 34: Descripción de la Clase del Diseño: CardDesigner.

Nombre: ToolsView	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de mostrar la barra de herramientas de diseño.

Tabla 35: Descripción de la Clase del Diseño: ToolsView.

Nombre: DesignerView	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de mostrar el área de diseño.

Tabla 36: Descripción de la Clase del Diseño: DesignerView.

Nombre: PropertiesView	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de mostrar la barra de propiedades.

Tabla 37: Descripción de la Clase del Diseño: PropertiesView.

Nombre: MainForm	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada del control de todas las áreas de diseño creadas, de la creación de nuevos documentos y de la realización de las acciones abrir, salvar, cerrar, salir, cortar, pegar, deshacer, rehacer, mostrar código, mostrar vista frontal, mostrar vista trasera, mostrar propiedades, entre otras.

Tabla 38: Descripción de la Clase del Diseño: MainForm.

Nombre: EstandarView	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de mostrar el panel de configuración de un nuevo estándar.

Tabla 39: Descripción de la Clase del Diseño: EstandarView

Nombre: Card

Tipo de clase: Entidad	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de inicializar los componentes, de especificarle el tamaño al documento al crearse, detectar las acciones sobre el documento como los objetos seleccionados, las selecciones múltiples, el orden de profundidad de los objetos, es decir cual está encima de cual y las alineaciones.

Tabla 40: Descripción de la Clase del Diseño: Card.

Nombre: CardObject	
Tipo de clase: Entidad	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de especificar elementos del documento como la transparencia, la posición en la escena, el estilo del borde, la unidad de medida que se utiliza para mostrar los elementos, y si es un elemento bloqueado o no.

Tabla 41: Descripción de la Clase del Diseño: CardObject.

Nombre: ImageDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto imagen de las características necesarias para que pueda ser ubicado y manipulado en el área

	de diseño.

Tabla 42: Descripción de la Clase del Diseño: ImageDraw.

Nombre: TextDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto texto de las características necesarias para que pueda ser ubicado y manipulado en el área de diseño. Asocia propiedades como el tipo de fuente, el color principal y el de fondo; y la alineación vertical y horizontal.

Tabla 43: Descripción de la Clase del Diseño: TextDraw.

Nombre: LineDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto línea de las características necesarias para que pueda ser ubicado y manipulado en el área de diseño.

Tabla 44: Descripción de la Clase del Diseño: LineDraw

Nombre: EllipseDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto elipse de las características necesarias para que pueda

	ser ubicado y manipulado en el área de diseño.

Tabla 45: Descripción de la Clase del Diseño: EllipseDraw

Nombre: PolygonDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto polígono de las características necesarias para que pueda ser ubicado y manipulado en el área de diseño.

Tabla 46: Descripción de la Clase del Diseño: PolygonDraw

Nombre: RectangleDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto rectángulo de las características necesarias para que pueda ser ubicado y manipulado en el área de diseño.

Tabla 47: Descripción de la Clase del Diseño: RectangleDraw

Nombre: PencilDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto lápiz de las características necesarias para que pueda ser ubicado y manipulado en el área de diseño.

Tabla 48: Descripción de la Clase del Diseño: PencilDraw

Nombre: ItemSecurityDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto elemento de seguridad de las características necesarias para que pueda ser ubicado y manipulado en el área de diseño. Se encarga de gestionar el posicionamiento adecuado de los elementos de seguridad de acuerdo al tipo de documento que se esté manejando.

Tabla 49: Descripción de la Clase del Diseño: ItemSecurityDraw

Nombre: TextFieldDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase semejante a la clase TextDraw pero con la peculiaridad de que se le puede asociar un enlace a un elemento de una base de datos.

Tabla 50: Descripción de la Clase del Diseño: TextFieldDraw

Nombre: BarCodeDraw	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de la asociación al objeto código de barras de las características

	necesarias para que pueda ser ubicado y manipulado en el área de diseño.

Tabla 51: Descripción de la Clase del Diseño: BarCodeDraw

Nombre: ImageDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 52: Descripción de la Clase del Diseño: ImageDrawBeanInfo

Nombre: TextDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 53: Descripción de la Clase del Diseño: TextDrawBeanInfo

Nombre: LineDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle

	formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 54: Descripción de la Clase del Diseño: LineDrawBeanInfo

Nombre: EllipseDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 55: Descripción de la Clase del Diseño: EllipseDrawBeanInfo

Nombre: PolygonDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 56: Descripción de la Clase del Diseño: PolygonDrawBeanInfo

Nombre: RectangleDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle

	formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 57: Descripción de la Clase del Diseño: RectangleDrawBeanInfo

Nombre: CardObjectBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 58: Descripción de la Clase del Diseño: CardObjectBeanInfo

Nombre: PencilDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 59: Descripción de la Clase del Diseño: PencilDrawBeanInfo

Nombre: ItemSecurityDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle

	formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 60: Descripción de la Clase del Diseño: ItemSecurityDrawBeanInfo

Nombre: TextFieldDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 61: Descripción de la Clase del Diseño: TextFieldDrawBeanInfo

Nombre: BarCodeDrawBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 62: BarCodeDrawBeanInfo

Nombre: CardBeanInfo	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de obtener los nombres de todos los métodos de la clase a la que se asocia y darle

	formato para que sean mostrados en la tabla de propiedades del objeto.

Tabla 63: Descripción de la Clase del Diseño: CardBeanInfo

Nombre: IDesignableComponent	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase de tipo interfaz que contiene los principales métodos que va a implementar cada objeto.

Tabla 64: Descripción de la Clase del Diseño: IDesignableComponent

Nombre: GestionarConfiguracion	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Se encarga de la gestión de los estándares, de su creación, de su modificación en caso de error y de su eliminación en caso de que se haya creado duplicado o como método alternativo ante la corrección del estándar.

Tabla 65: Descripción de la Clase del Diseño: GestionarConfiguración

Nombre: Estandar	
Tipo de clase: Entidad	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase básica del estándar donde se especifican todas las características que se tendrán en cuenta pata agregar un nuevo estándar para

	aplicar a los documentos creados en la aplicación.

Tabla 66: Descripción de la Clase del Diseño: Estándar

Nombre: Ruler	
Tipo de clase: Interfaz	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de ajustar el tamaño del documento y de la regla cuando se cambia de una unidad a otra o cuando se realiza el acercamiento o alejamiento del documento.

Tabla 67: Descripción de la Clase del Diseño: Ruler.

Nombre: PropertyEditor	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de controlar la asociación de las propiedades a los elementos seleccionados en el área de diseño y del documento.

Tabla 68: Descripción de la Clase del Diseño: PropertyEditor.

Nombre: ToolBox	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada del control de eventos sobre el área de la barra de herramientas laterales, donde se encuentran los objetos de diseño.

Tabla 69: Descripción de la Clase del Diseño: ToolBox.

Nombre: UnitConverter	
Tipo de clase: Controladora	
Atributos	**Tipo**
-	-
Responsabilidades	
Descripción	Clase encargada de realizar la conversión de unidades de píxeles a pulgadas, milímetros y centímetros.

Tabla 70: Descripción de la Clase del Diseño: UnitConverter.

3.1.9. Interfaz

La interfaz de un sistema es la que permite el intercambio de información entre este y el usuario. El sistema que se presenta consta de una aplicación denotada como **SVGCardDesigner**. Esta posee un menú principal donde se muestran las opciones de trabajo con las plantillas. En uno de sus laterales posee una barra de objetos para el diseño, en el otro lateral posee otra barra dedicada principalmente a la asociación de las propiedades de los objetos y en el centro se encuentra el área de diseño de la aplicación.Luego de esta breve reseña sobre la aplicación, se muestra la interfaz para que se tengauna mayor comprensión del tema.

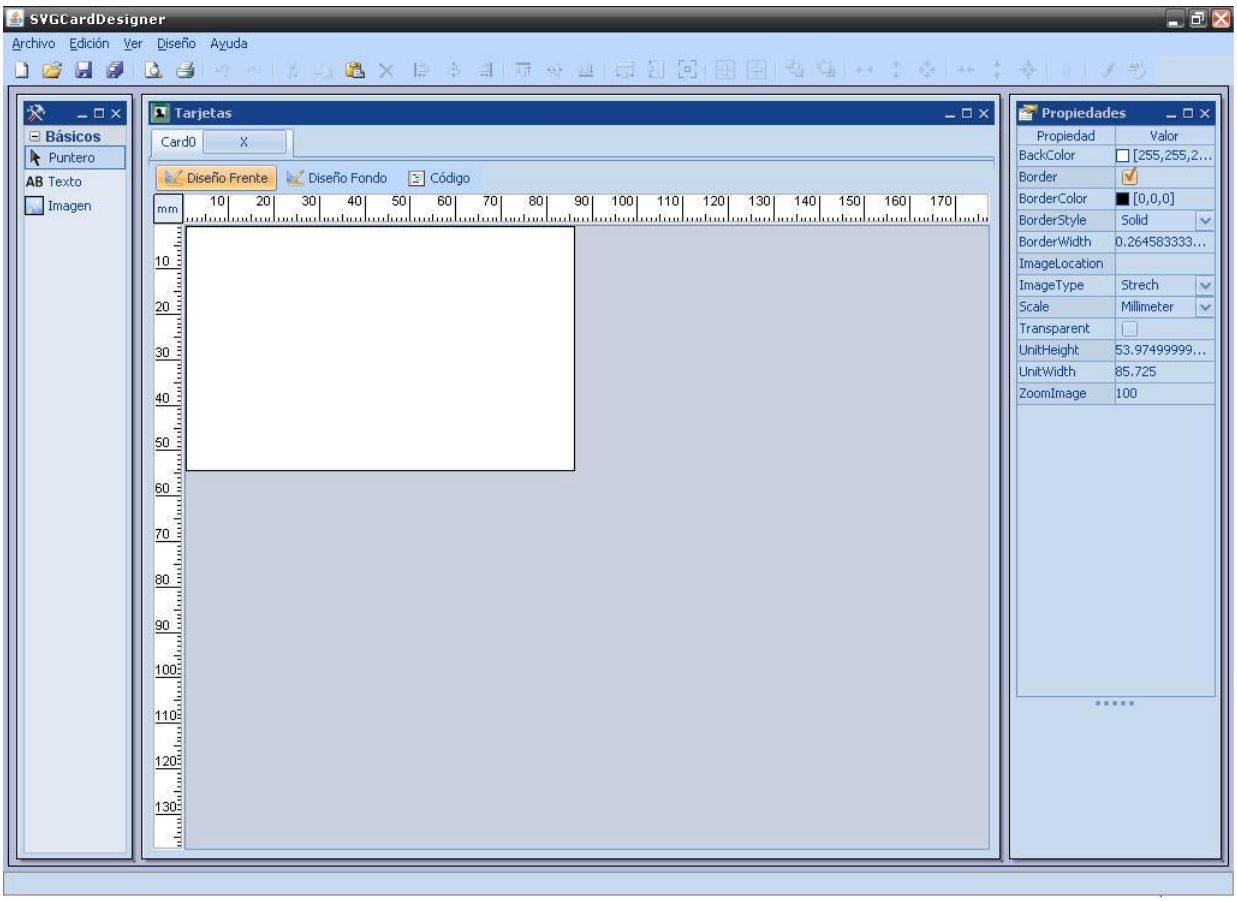


Figura 29: Vista principal del diseñador de plantillas.

En este caso el diseñador de plantillas lo mismo puede cargar una plantilla que ya haya sido creada, editarla o también puede crear una plantilla en blanco al igual que la puede salvar.

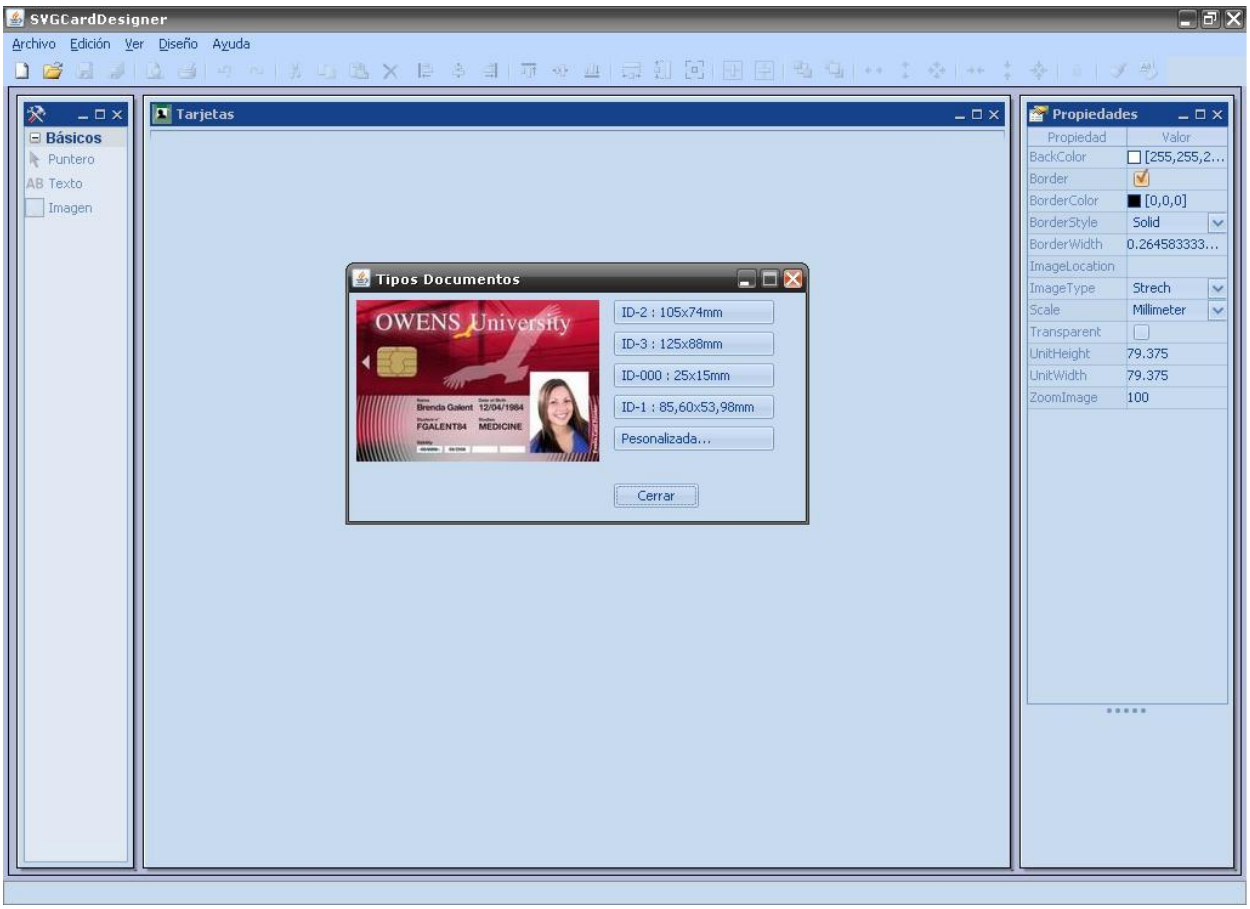


Figura 30: Vista principal del diseñador de plantillas en la creación de una nueva plantilla.

3.1.10. Tratamiento de errores

El tratamiento de errores se lleva a cabo a través del uso de las excepciones lo cual hace que el manejo de los eventos que puedan ocurrir por errores de programación o por acciones provocadas por el usuario sea mucho más transparente para este.

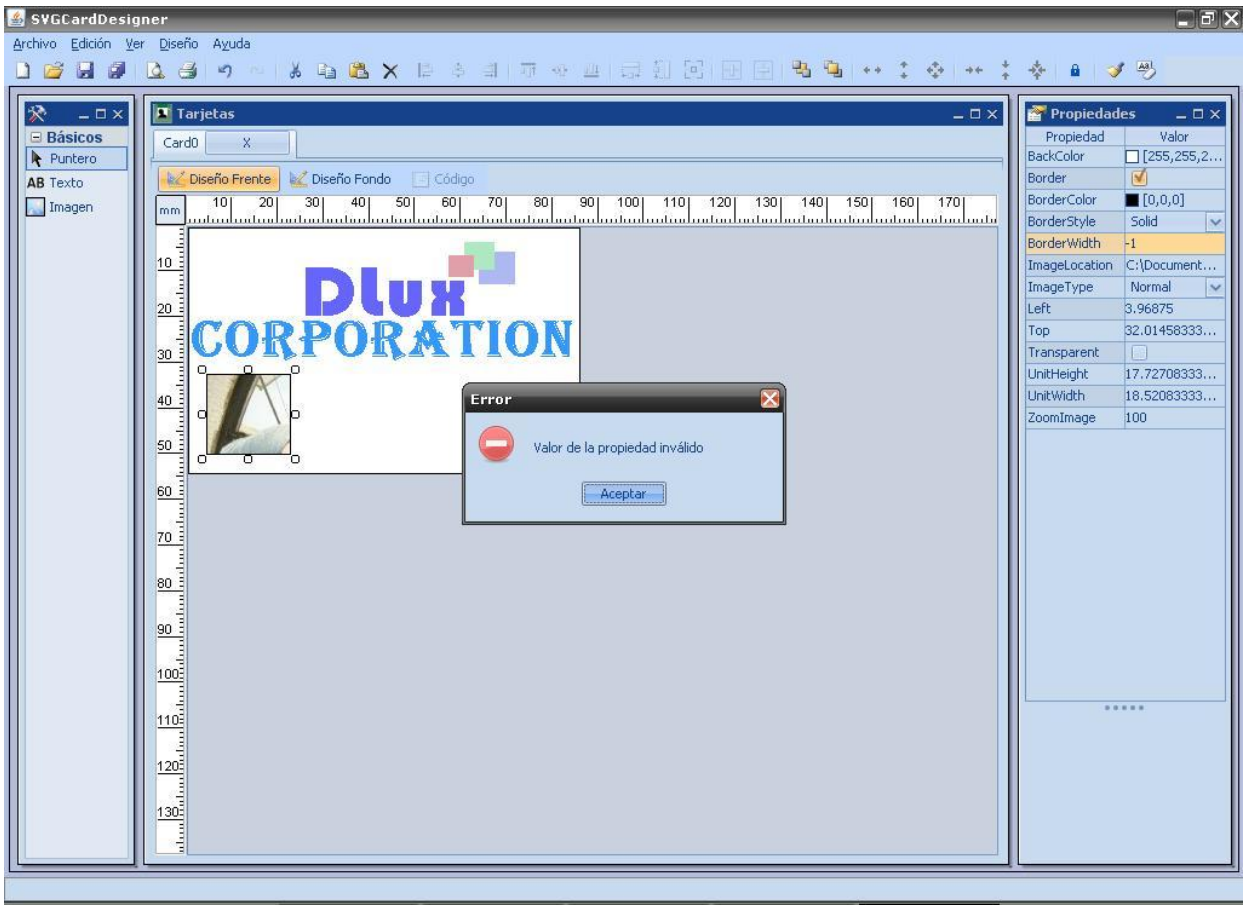


Figura 31: Representación del tratamiento de errores en la aplicación.

3.1.11. Concepción de la ayuda

La solución propuesta, al cumplir las pautas de accesibilidad y usabilidad, se convierte en una herramienta de trabajo bastante fácil e intuitiva. Además, esta se asemeja en organización y funcionalidades a otros editores gráficos que se utilizan comúnmente como el Microsoft Paint. Esto hace más fácil la creación de la ayuda pues se pondría más énfasis a la parte de la explicación técnica y haría de la ayuda algo más útil y fácil de consultar.

3.2.Conclusiones Parciales del Capítulo

En este capítulo se ha representado la información referente a las etapas de Análisis y Diseño, donde se obtuvo el Modelo de Análisis como uno de los artefactos del Análisis, refinándose toda la información obtenida de las etapas anteriores de la construcción del software, se decidió la arquitectura a utilizar, lo cual permitirá a los programadores tener la visión futura de la implementación. Se obtuvo en el Diseño, los Diagramas de Interacción y de Clases del Diseño, describiéndose cada una de las clases principales utilizadas lo que dará una visión detallada con vistas a la implementación. Se presenta, además, una breve explicación de cómo funcionará el sistema mostrándose interfaces de cómo deberá lucir la aplicación tras su culminación.

Conclusiones Generales

Con la realización de este trabajo se realizó el análisis y diseño de una herramienta de confección de plantillas para la impresión de documentos de identificación que logrará una mayor eficiencia en el proceso de personalización de estos. Como complemento y adición a este objetivo principal, se dio cumplimiento a otros objetivos más específicos:

1. Se demostró que los sistemas que existen en el ámbito internacional no son los más óptimos para modelar el proceso de confección de plantillas para la impresión de documentos de identificación en Cuba.

2. La investigación arrojó como resultado que los sistemas con los que cuenta Cuba comprenden muchas más funcionalidades como para utilizarlos con este fin, y utilizan una tecnología que no permitiría su uso comercial en Cuba. En el caso particular de la UCI, no se cuenta con aplicaciones lo suficientemente desarrolladas como para encargarse de estos procesos.

3. Se especificaron las funcionalidades para la realización de una herramienta de confección de plantillas para la impresión de documentos de identificación.

4. Se realizó un prototipo de la herramienta diseñada.

5. La herramienta de confección de plantillas diseñada será capaz de darle solución a la problemática existente y agilizará el proceso de creación de los documentos de identificación.

6. El sistema ha sido diseñado para ser desarrollado totalmente con Software Libre, por lo que no es necesaria la adquisición de licencias.

Recomendaciones

Al concluir la presente tesis se realizan una serie de recomendaciones que podrán tenerse en cuenta para el desarrollo futuro del sistema:

- Se propone que se le de seguimiento a este trabajo para lograr un producto de mayor calidad.
- Se recomienda realizar la implementación y pruebas de la propuesta que se presenta en este trabajo, con el fin de obtener al menos una versión del producto.
- Se recomienda realizar otras iteraciones para optimizar el proceso de levantamiento de requisitos.
- Proseguir con el estudio realizado con el propósito de añadir nuevas funcionalidades al sistema.
- Se propone recoger las diversas opiniones de estudiantes y profesores con el objetivo de enriquecer y perfeccionar la aplicación.
- Se recomienda una vez terminado el producto, sea puesto en explotación, para comprobar su rendimiento y la facilidad de familiarización y uso del usuario con la aplicación.

Bibliografía

2001. 6.2.2 Características de XML. *6.2.2 Características de XML.* [En línea] Jose Manuel Lopez Franco, 15 de 10 de 2001. [Citado el: 31 de 01 de 2009.] http://trevinca.ei.uvigo.es/~txapi/espanol/proyecto/superior/memoria/node156.html.

Almenares, Kiosmy y de León, Rubén. 2007.*Diseño e implementación del proceso de inscripción del Módulo de Mercantil en las Oficinas Registrales de la República Bolivariana de Venezuela.* Caracas, Venezuela : s.n., 2007.

Ariza Rojas, Maribel y Molina García, Juan Carlos. 2004.*Introducción y Principios Básicos del Desarrollo de Software Basado en Componentes.* Bogotá : s.n., 2004.

Beck, K. 1999.*Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999.

—. 2000.*Una explicación de la programación extrema. Aceptar el cambio.* s.l. : Addison Wesley, 2000.

Cano Ossa, Mauricio. 2006.*EL DIOS DE LOS MONOS. Una guía para el desarrollador de GUIs en windows con el mono.* Medellín : s.n., 2006. Agosto 12, 2006.

DataCard. DataCard. *DataCard.* [En línea] http://www.datacard.es/pages/ssc_subpage.jhtml?contentId=545873094uz2AUqr9G.

—. DataCard. *DataCard.* [En línea] http://www.datacard.es/pages/ssc_subpage.jhtml?contentId=09707dK0s8BQa.

Fernandez Vilas, Ana. 2001. Diagrama de Componentes. *Diagrama de Componentes.* [En línea] 20 de Marzo de 2001. [Citado el: 16 de Abril de 2009.] http://tvdi.det.uvigo.es/~avilas/UML/node49.html.

—. 2001. Diagrama de Despliegue. *Diagrama de Despliegue.* [En línea] 20 de Marzo de 2001. [Citado el: 16 de Abril de 2009.] http://tvdi.det.uvigo.es/~avilas/UML/node50.html.

—. 2001. Diagramas de Interacción. *Diagramas de Interacción.* [En línea] 20 de Marzo de 2001. [Citado el: 16 de Abril de 2009.] http://tvdi.det.uvigo.es/~avilas/UML/node41.html.

Fowler, M., Beck, K. y Brant, J. 1999.*Refactoring: Improving the Design of Existing Code.* s.l. : Addison-Wesley, 1999.

Francisco, Mendoza. 1997. Monografias.com. *Monografias.com.* [En línea] 1997. [Citado el: 31 de 01 de 2009.] http://www.monografias.com/trabajos6/ixml/ixml.shtml#xml.

Fuentes, Lidia, Troya, José M. y Vallecillo, Antonio. 2003.*Desarrollo de Software Basado en Componentes.* 2003.

García Carballeira, Dr. Félix. 2000. Revista Digital Universitaria. *Revista Digital Universitaria.* [En línea] 1 de Octubre de 2000. http://www.revista.unam.mx/vol.1/num2/art4/.

Group, Gartner.

ibiblio.org. Modelado de Sistemas com UML - Capítulo 4. Un estudio a fondo de UML. *Modelado de Sistemas com UML - Capítulo 4. Un estudio a fondo de UML.* [En línea] ibiblio.org. [Citado el: 16 de Abril de 2009.] http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.

2009. It's About Common Sense. *It's About Common Sense.* [En línea] Control Chaos, 2009. [Citado el: 30 de 01 de 2009.] www.controlchaos.com.

Juristo, Natalia, Moreno, Ana M. y Vegas, Sira. 2006.*Técnicas de Evaluación de Software.* 2006.

Krutchen, Philippe.*Rational Software.*

Kynetia. 2007. Kynetia. *Kynetia.* [En línea] Kynetia, 2007. [Citado el: 16 de Abril de 2009.] http://www.kynetia.es/calidad/metodologia-de-pruebas.html.

Larman, Craig. 2008. Modelo de Dominio. *Modelo de Dominio.* [En línea] Ed Prentice Hall, 27 de Abril de 2008. [Citado el: 13 de Abril de 2009.] http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio?tmpl=component&print=1&page=.

Maria. 2008. A desalambrar. A desalambrar. *A desalambrar. A desalambrar.* [En línea] 2008. [Citado el: 03 de 12 de 2008.] http://maria.lamatriz.org/historia-del-dni.

—. 2008. A desambalar. A desambalar. *A desambalar. A desambalar.* [En línea] 2008. [Citado el: 03 de 12 de 2008.] http://maria.lamatriz.org/historia-del-dni.

MeRinde. 2009. MeRinde. *MeRinde.* [En línea] Centro Nacional de Tecnologías de Información, 2009. [Citado el: 16 de Abril de 2009.] http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=92&Itemid=296.

Microsoft Corporation. 2008. .NET Framework Developer Center. *.NET Framework Developer Center.* [En línea] 2008. http://msdn.microsoft.com/netframework/.

—. 2008. Microsoft Encarta. *Microsoft Encarta.* [En línea] Microsoft Corporation, 2008. [Citado el: 30 de 01 de 2009.] http://encarta.msn.com/encyclopedia_761562769/Printing.html .

Montilva C., Jonás A., Arapé, Nelson y Colmenares, Juan Andrés. 2003.*Desarrollo Basado en Componentes.* 2003.

Mühlbauer Group. 2008. Mühlbauer AG High Tech International. *Mühlbauer AG High Tech International.* [En línea] Mühlbauer AG High Tech International, 30 de 11 de 2008. [Citado el: 30 de 11 de 2008.] http://www.muhlbauer.com/docs/index_home.asp?id=21572&sp=S&m1=21564&m2=21572&m3=&domid=1016&addminprt=1.

—. Mühlbauer Group. *Mühlbauer Group.* [En línea] http://www.muhlbauer.com/docs/index.asp?id=21607&sp=S&m1=21564&m2=21568&m3=21607&m4=&domid=1016.

Policía, Grupo Nacional de. DNI Electrónico. *DNI Electrónico.* [En línea] Grupo Nacional de Policía. http://www.dnielectronico.es/imagenes_dni/dnie_descrip_p.jpg&imgrefurl=http://www.dnielectronico.es/Asi_es_el_dni_electronico/descripcion.html.

Pozo, Pedro. Domitienda. *Domitienda.* [En línea] [Citado el: 16 de Abril de 2009.] http://www.clikear.com/manuales/uml/diagramasinteraccion.aspx.

Rational Software Corporation. 2002.*Product.* s.l. : Rational Software Corporation, 2002.

Sadín Odio, Abdel y Labrada Nueva, Yainier. 2008.*Tesis de Curso: Componentes y utilitarios para la impresión de documentos de identificación en software libre "IDSVG".* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

Santos, Victor Manuel Rivas. 2001. Tutorial de XPath . *Tutorial de XPath .* [En línea] (C) GeNeura Team, 01 de 2001. [Citado el: 31 de 01 de 2009.] http://geneura.ugr.es/~victor/cursillos/xml/XPath/.

Seoane, Fabian. 2004. LUGCIX - Linux User Group Chiclayo. *LUGCIX - Linux User Group Chiclayo.* [En línea] Diciembre de 2004. http://www.lugcix.org/tutoriales/mono/queesmono.php.

Sosa Cano, Oliver y Real Gómez, Roldán. 2008.*Tesis de Curso: Sistema de Información para la toma de decisiones del cierre de operaciones postales y telegráficas.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

Szyperski, Clemens. 1998.*Component Software Beyond Object–Oriented Programming.* 1998. págs. 20-22.

Tomado de las Conferencias de Ingeniería de Software I.

UCI, ISW I -.*Tomado de las Conferencias de Ingeniería de Software I.*

UCI, Universidad de las Ciencias Informáticas. 2009.*Conferencias de Ingeniería de Software II.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2009.

Vilas, Ana Fernandez. 2001. Ana Fernandez Vilas. *Ana Fernandez Vilas.* [En línea] Ana Fernandez Vilas, 20 de Marzo de 2001. [Citado el: 13 de Abril de 2009.] http://tvdi.det.uvigo.es/~avilas/UML/node25.html.

W3C. 2000. Sidar.org. *Sidar.org.* [En línea] 07 de 08 de 2000. http://www.sidar.org/recur/desdi/traduc/es/notas/svg-acces/SVG-access.html.

Wikipedia®. 2008. Wikipedia®. *Wikipedia®.* [En línea] Wikimedia Foundation, Inc., 30 de 11 de 2008. [Citado el: 3 de 12 de 2008.] http://es.wikipedia.org/wiki/Biometria.

Anexos

Anexo 1

Representación Gráfica de los Diagramas de Clases del Análisis

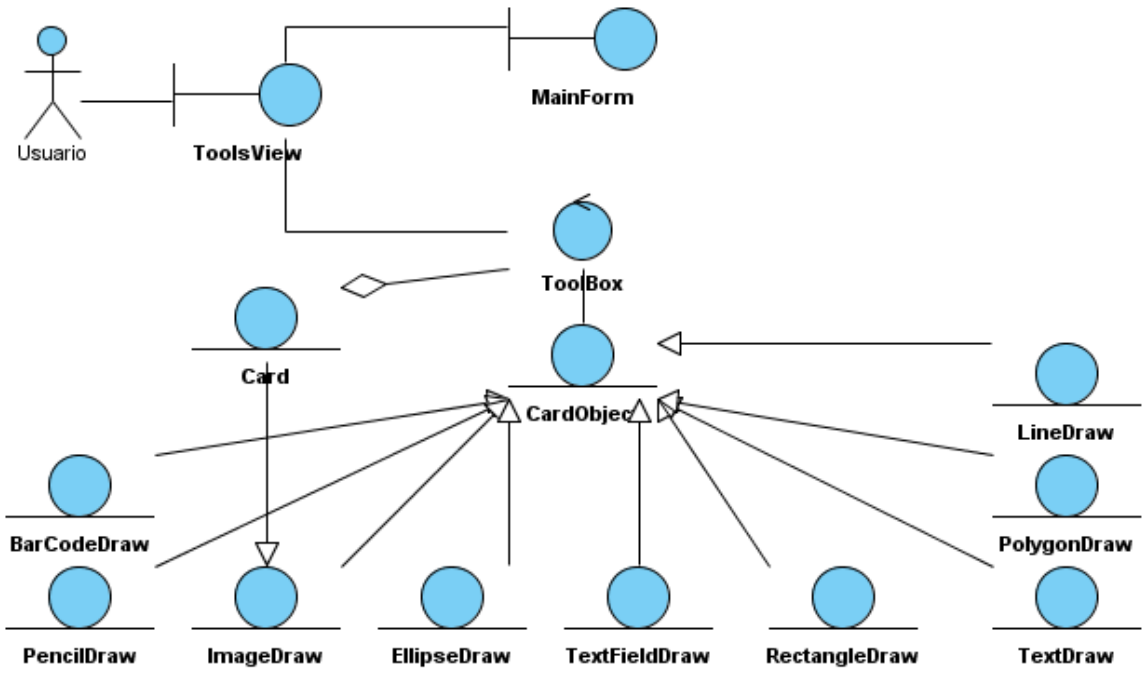


Figura 32: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Crear Objeto.

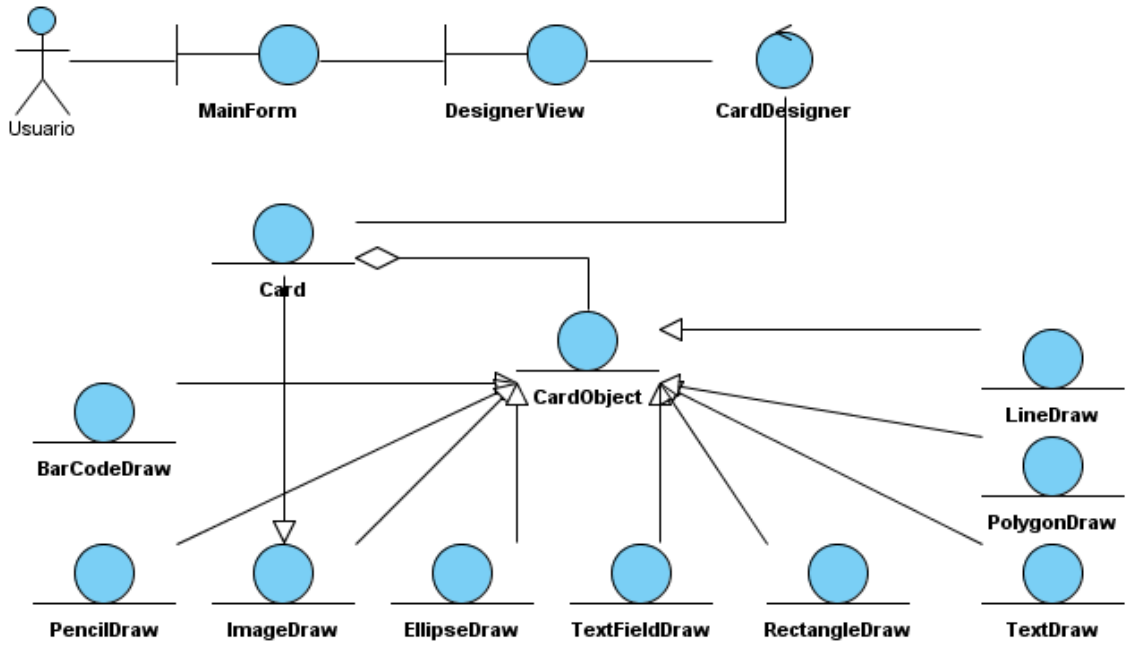


Figura 33: Diagrama de Clases del Modelo de Análisis. Casos de Uso: Copiar Objeto, Cortar Objeto, Pegar Objeto y Eliminar Objeto.

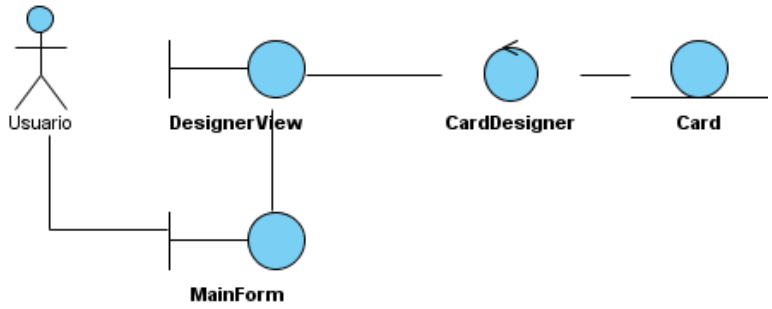


Figura 34: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Editar Codificación.

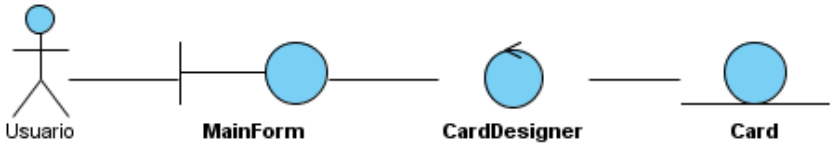


Figura 35: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Mostrar Vista Previa.

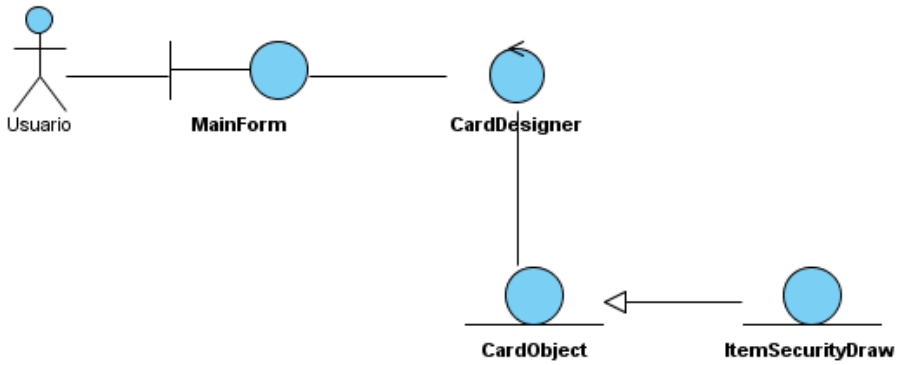


Figura 36: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Especificar Elemento de Seguridad.

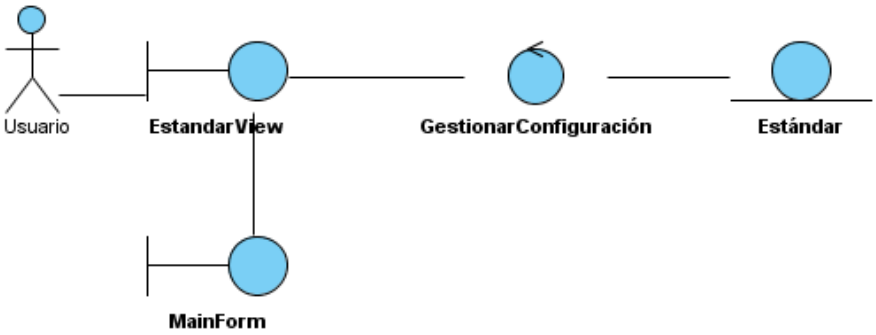


Figura 37: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Gestionar Estándar.

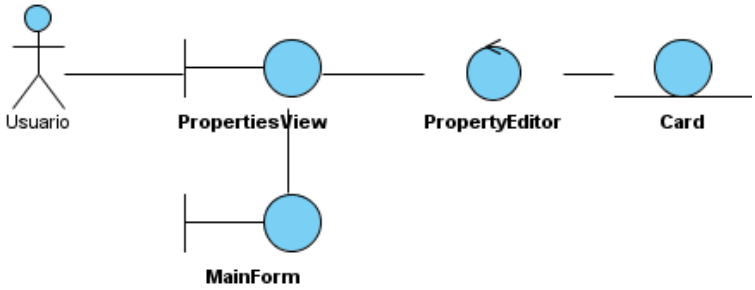


Figura 38: Diagrama de Clases del Modelo de Análisis. Caso de Uso: Asociar Propiedades.

Anexo 2

Diagramas de Secuencia

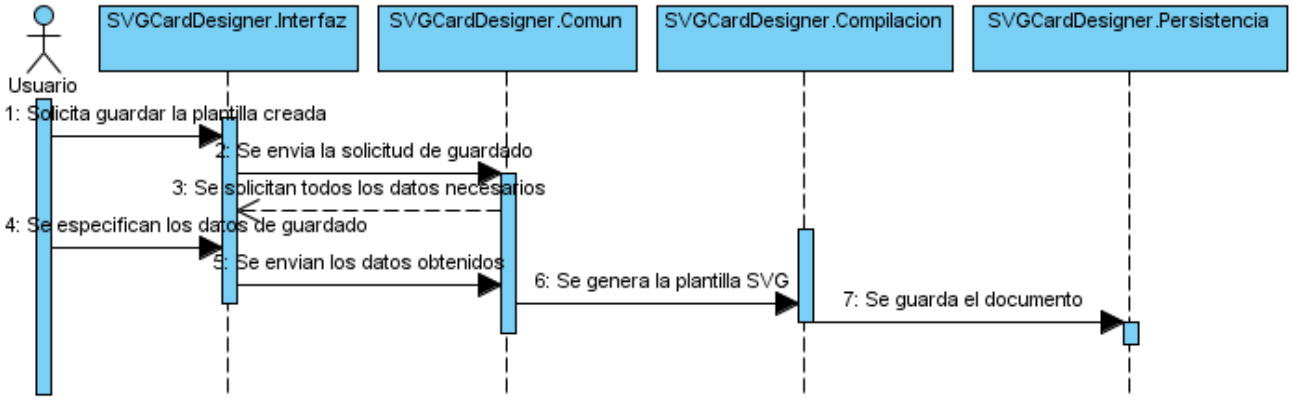


Figura 39: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Guardar Plantilla.

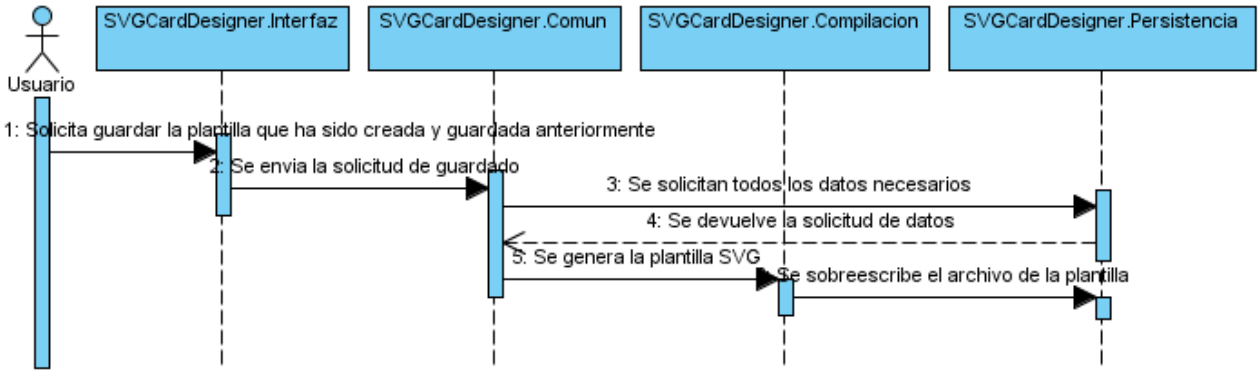


Figura 40: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Guardar Plantilla. (Curso Alterno 1)

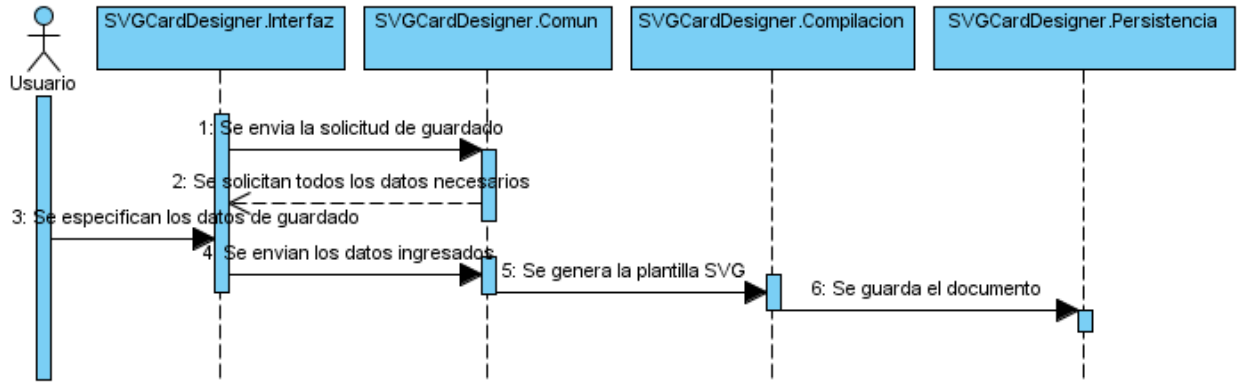


Figura 41: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Guardar Plantilla. (Curso Alterno 2)

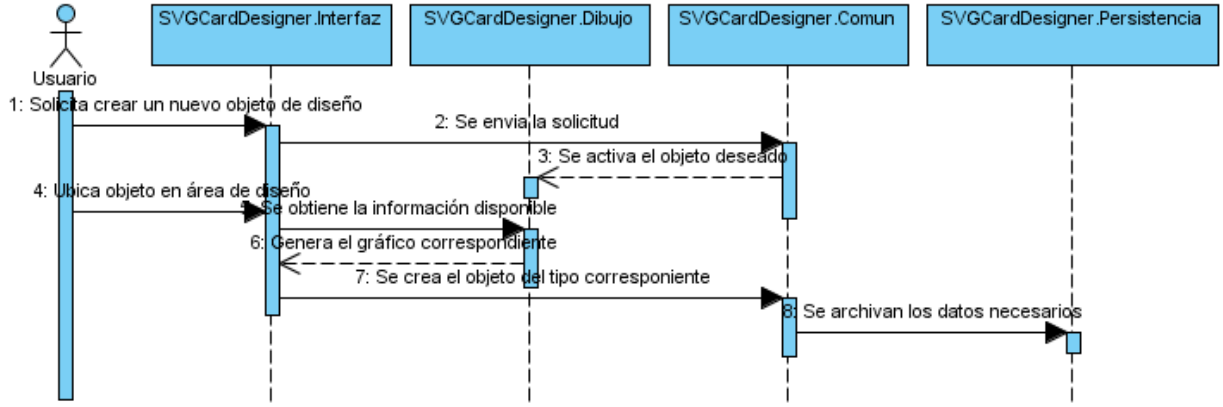


Figura 42: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Crear Objeto.

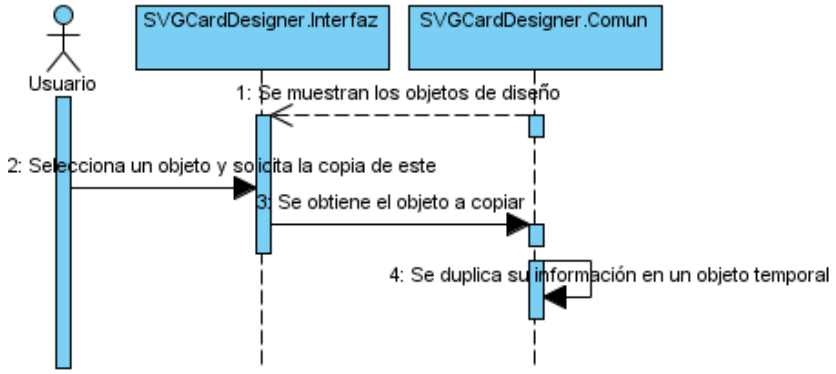


Figura 43: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Copiar Objeto.

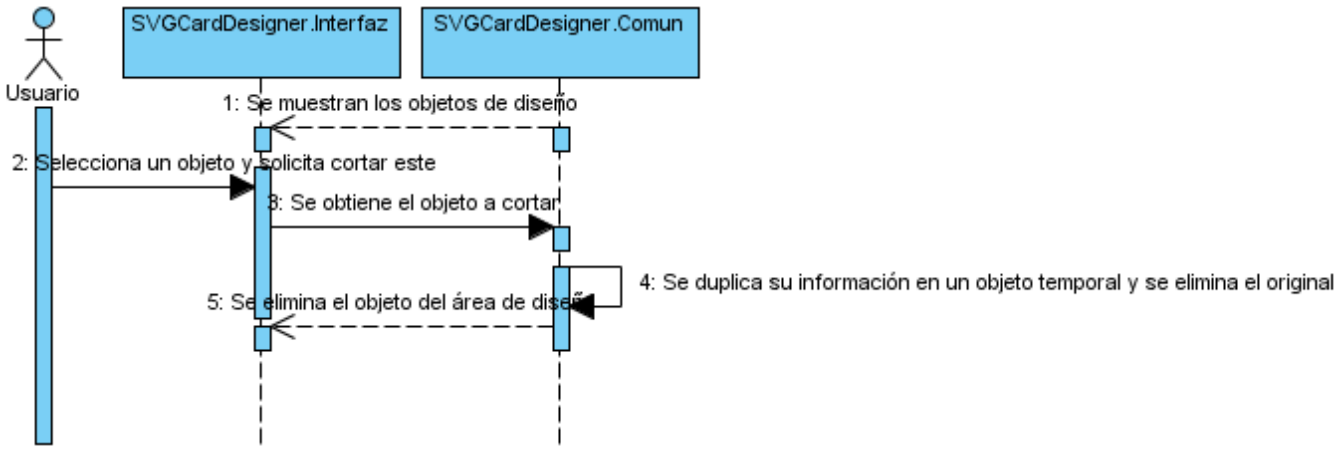


Figura 44: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Cortar Objeto.

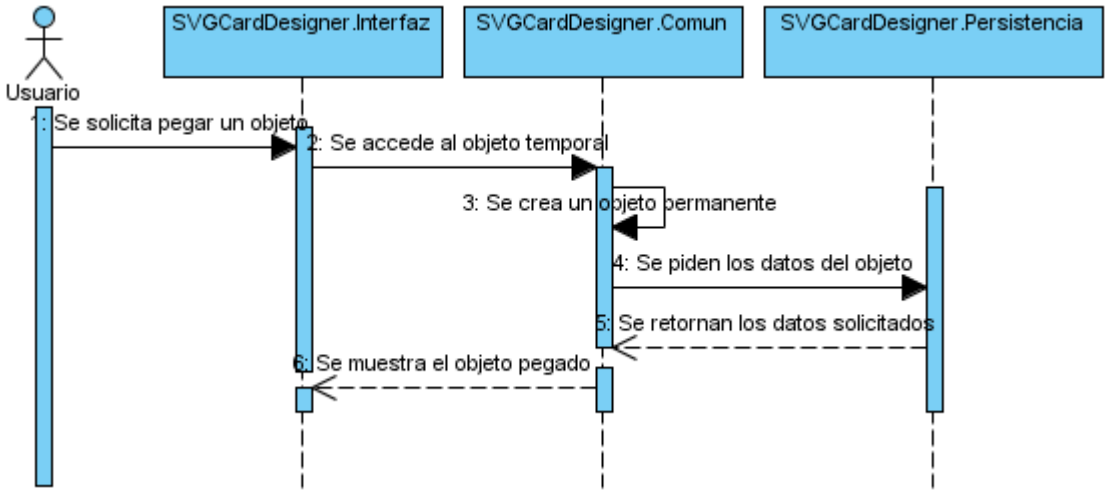


Figura 45: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Pegar Objeto.

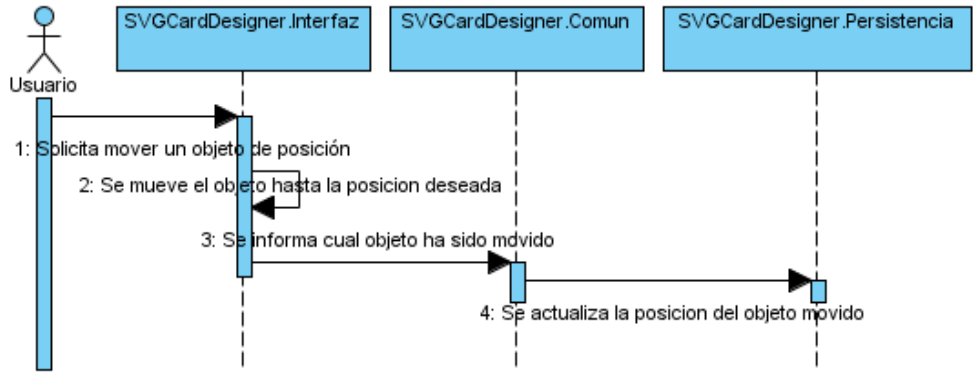


Figura 46: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Mover Objeto.

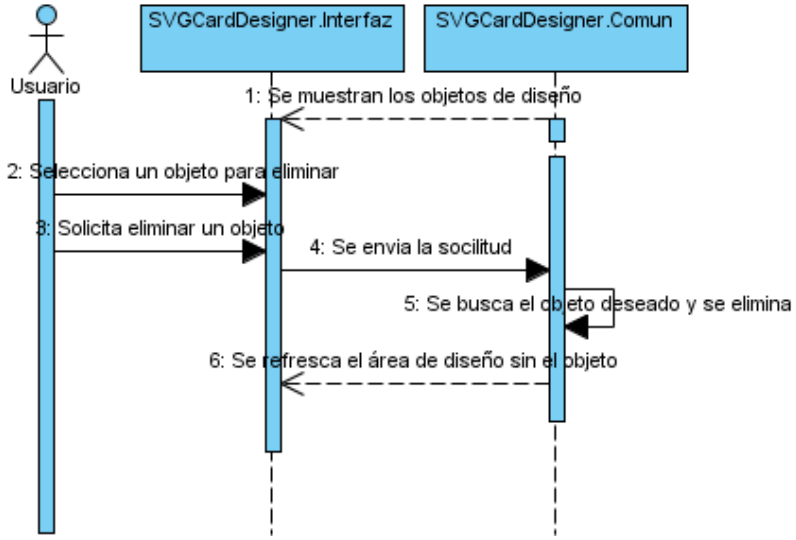


Figura 47: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Eliminar Objeto.

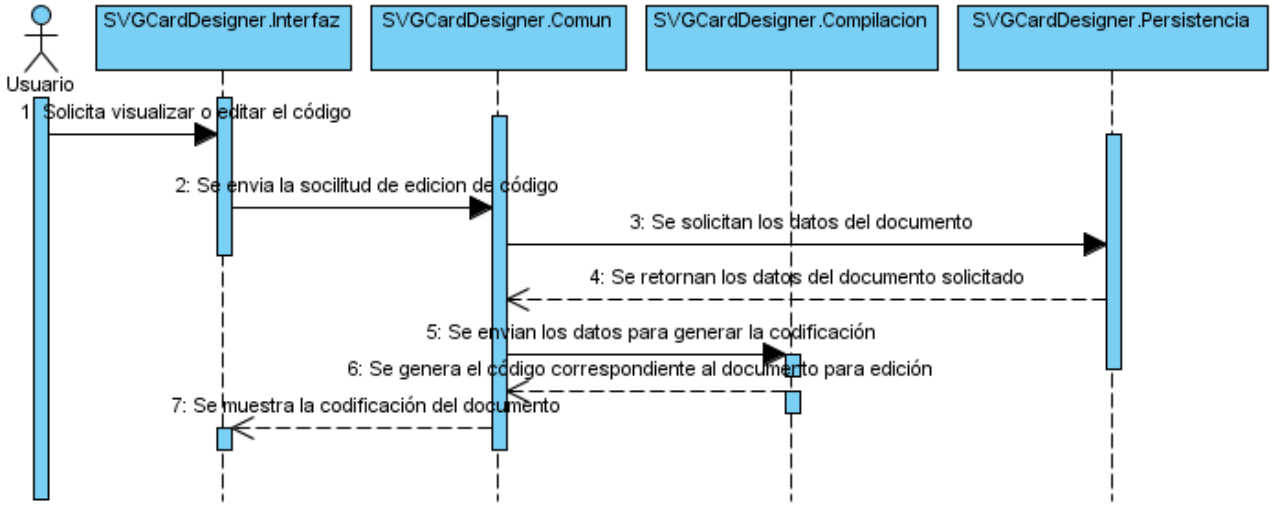


Figura 48: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Editar Configuración.

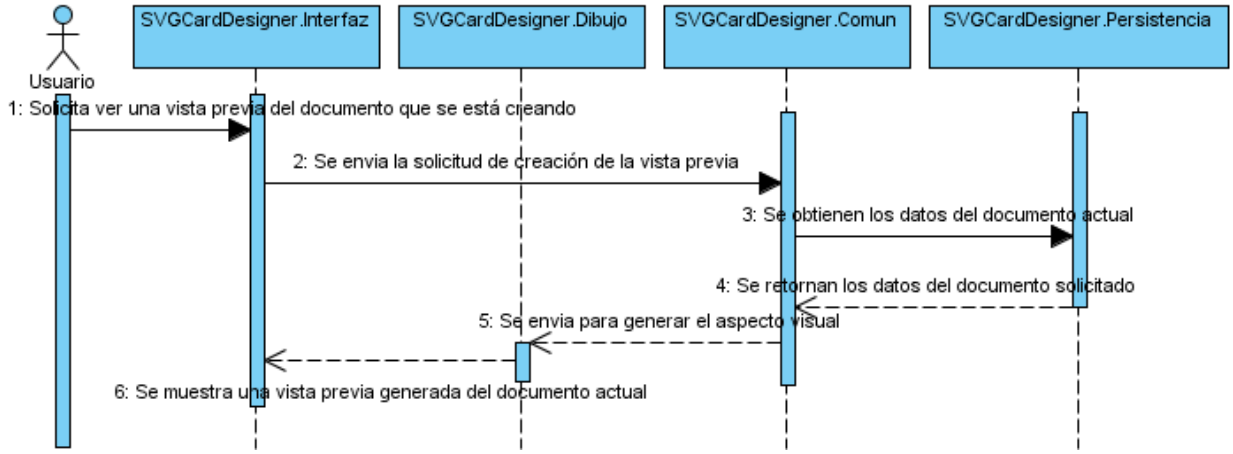


Figura 49: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Mostrar Vista Previa.

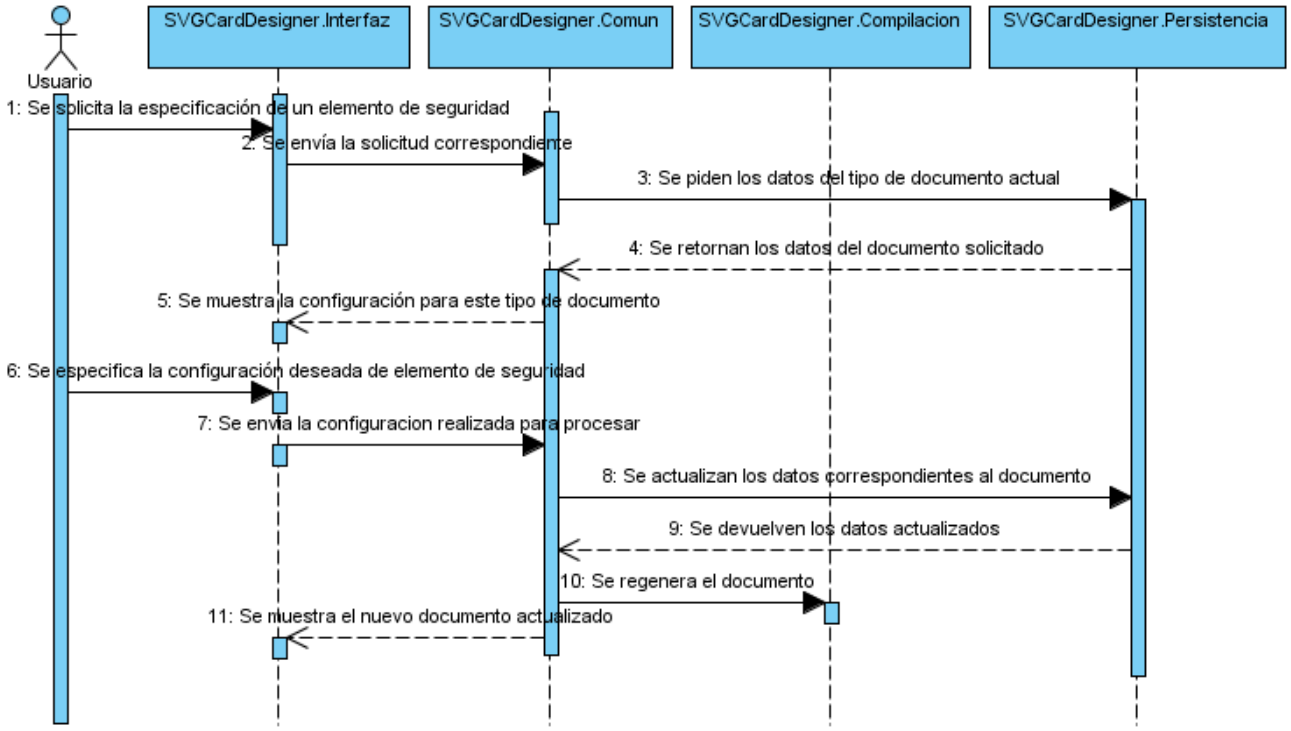


Figura 50: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Especificar Elemento de Seguridad.

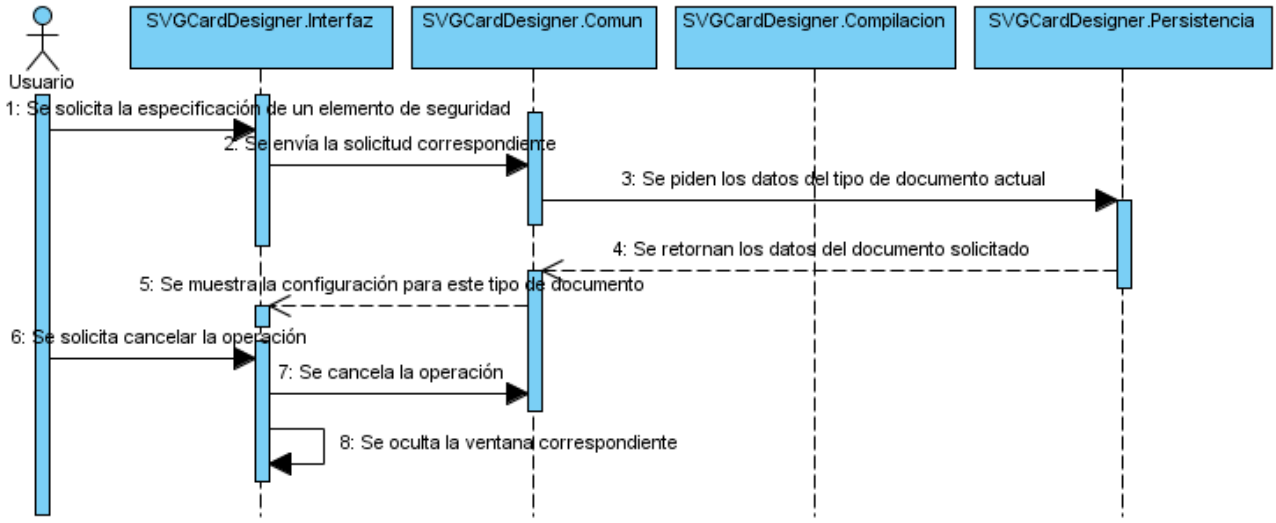


Figura 51: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Especificar Elemento de Seguridad. (Curso Alterno)

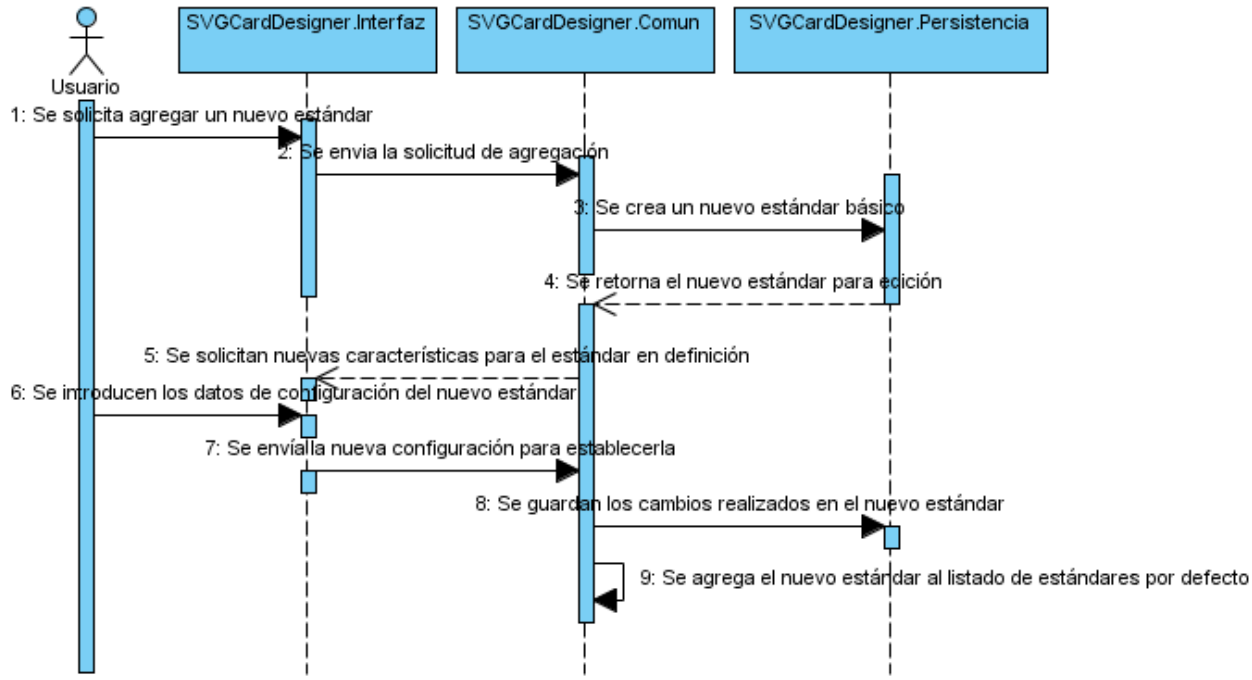


Figura 52: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Agregar).

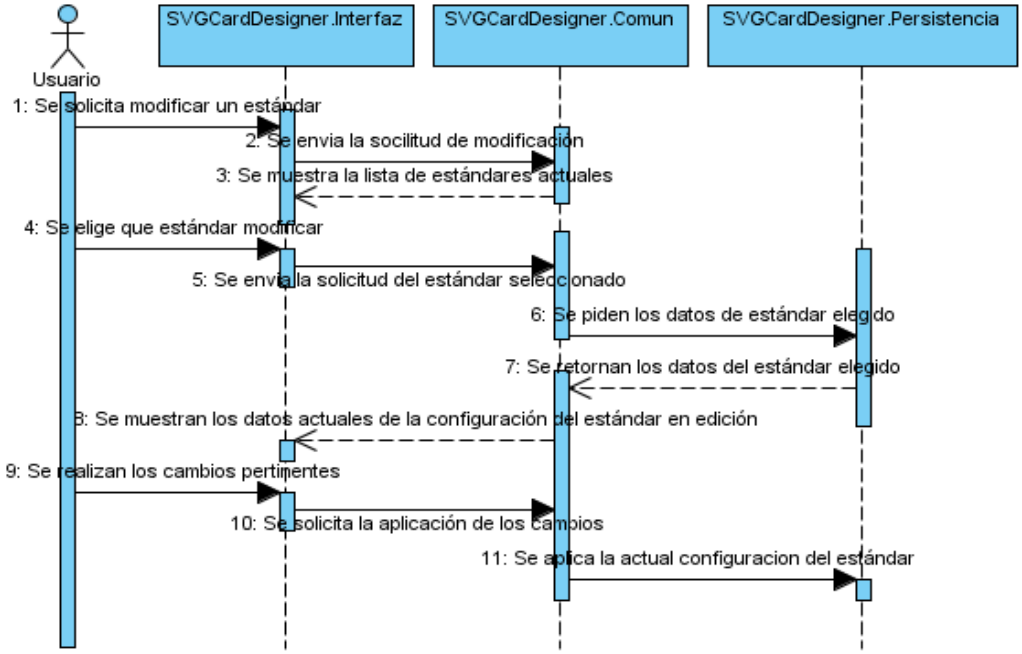


Figura 53: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Modificar).

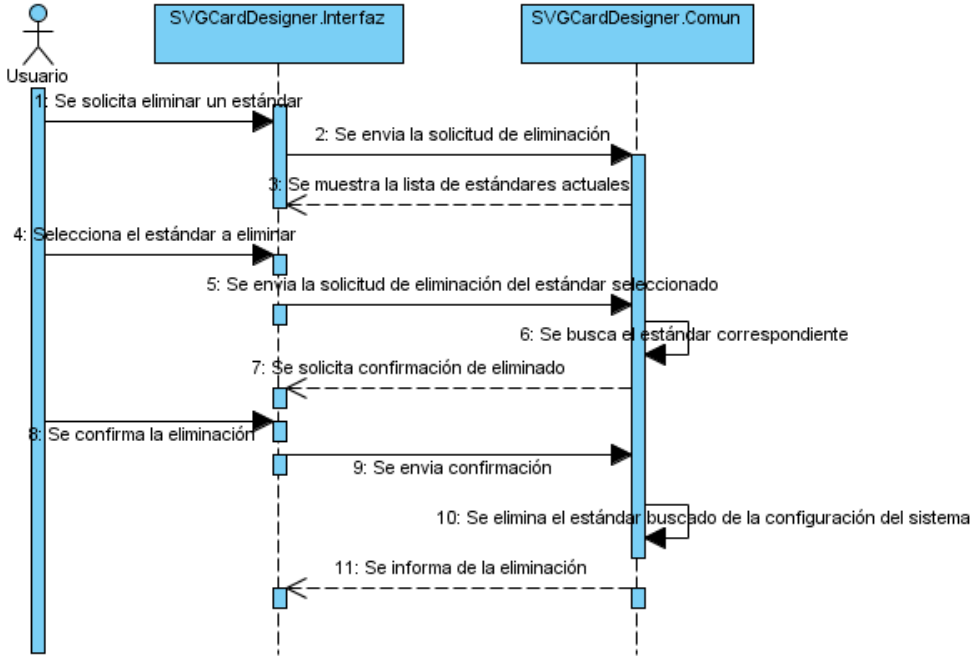


Figura 54: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Eliminar).

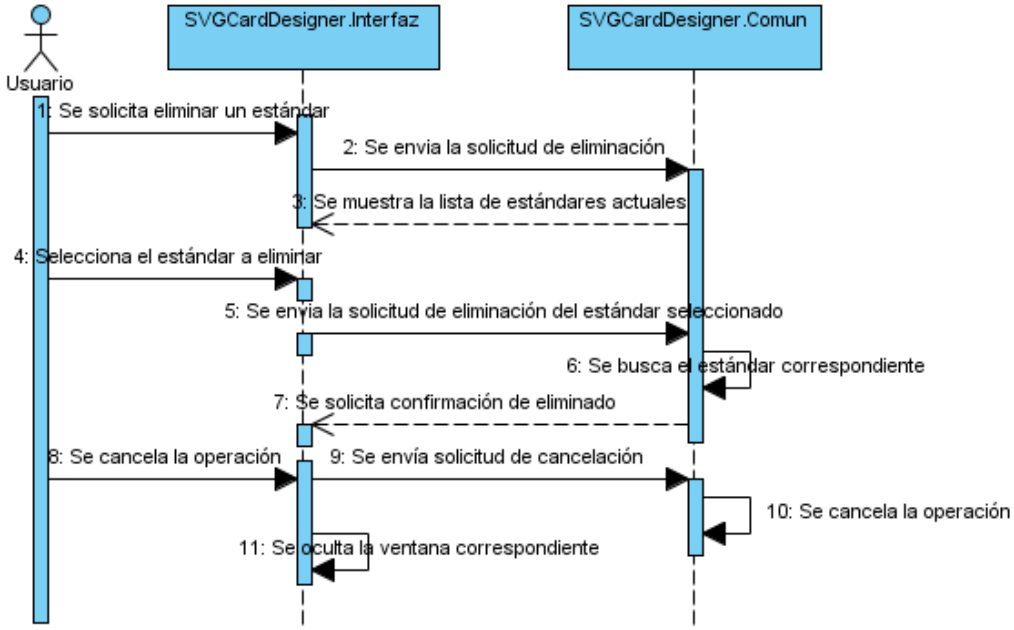


Figura 55: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Gestionar Estándar (Eliminar). (Curso Alterno)

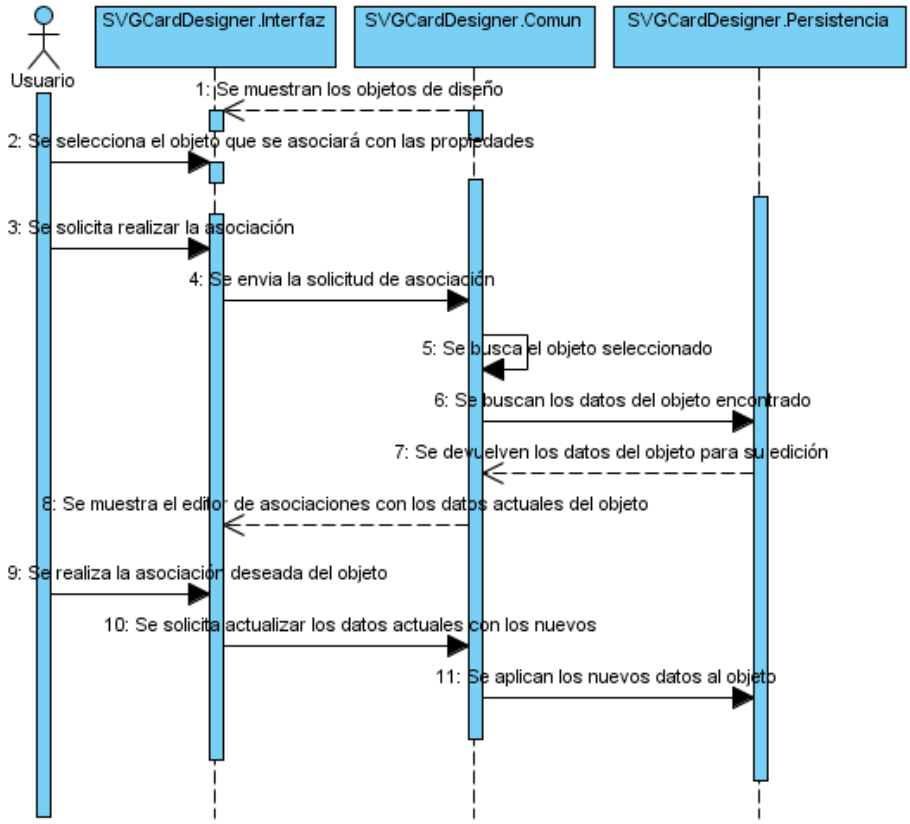


Figura 56: Representación gráfica del Diagrama de Secuencia del Caso de Uso: Asociar Propiedades.

Anexo 3

Representación Gráfica de las Clases del Diseño

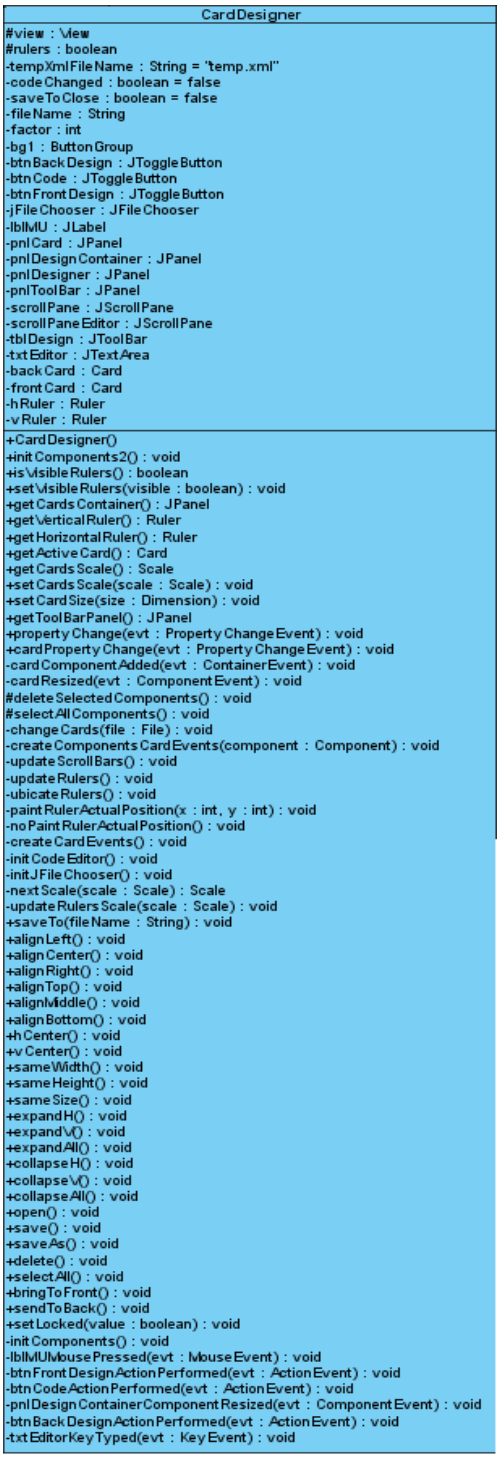


Figura 57: Representación gráfica de la clase CardDesigner.

Figura 58: Representación gráfica de la clase DesignerView.

Figura 59: Representación gráfica de la clase MainForm.

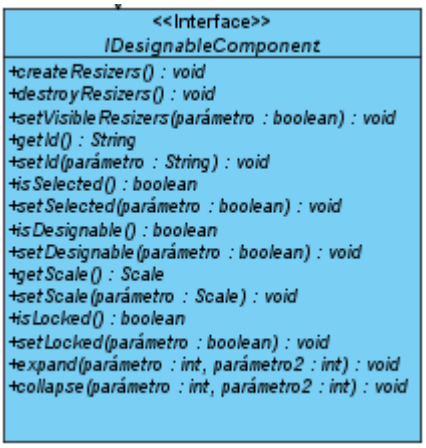


Figura 60: Representación gráfica de la clase IDesignableComponent.

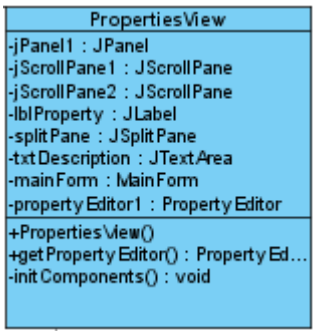


Figura 61: Representación gráfica de la clase PropertiesView.

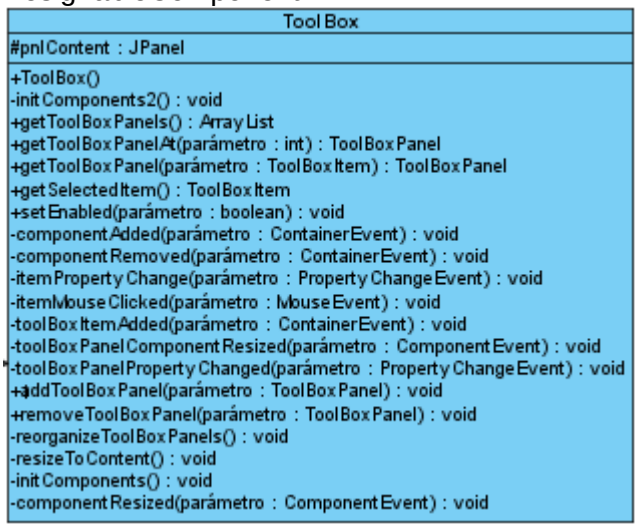


Figura 62: Representación gráfica de la clase ToolBox.

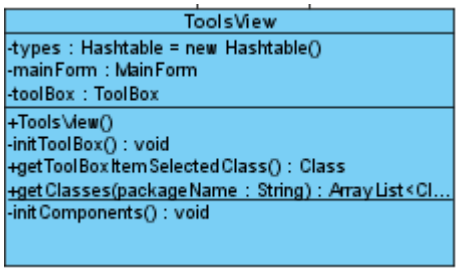


Figura 63: Representación gráfica de la clase ToolsView.

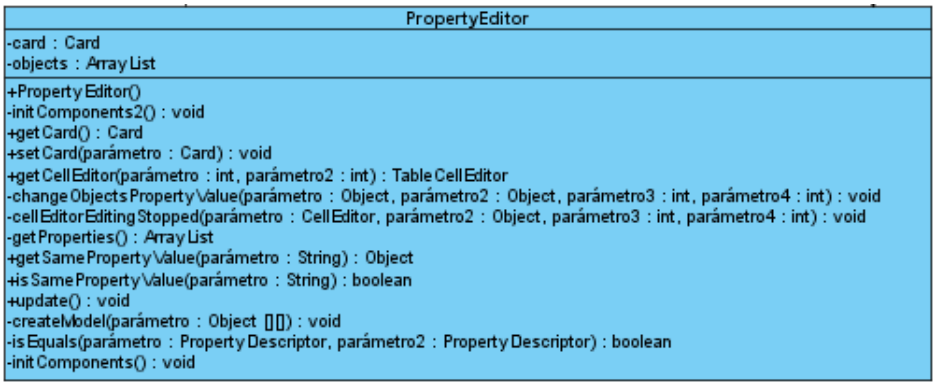


Figura 64: Representación gráfica de la clase PropertyEditor.

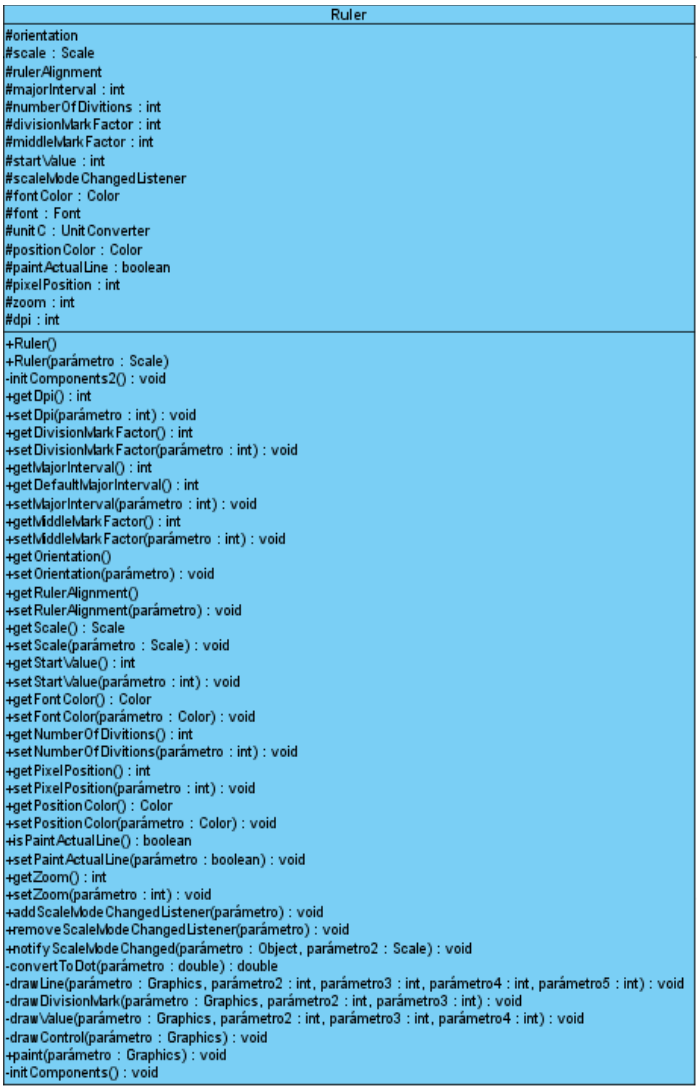
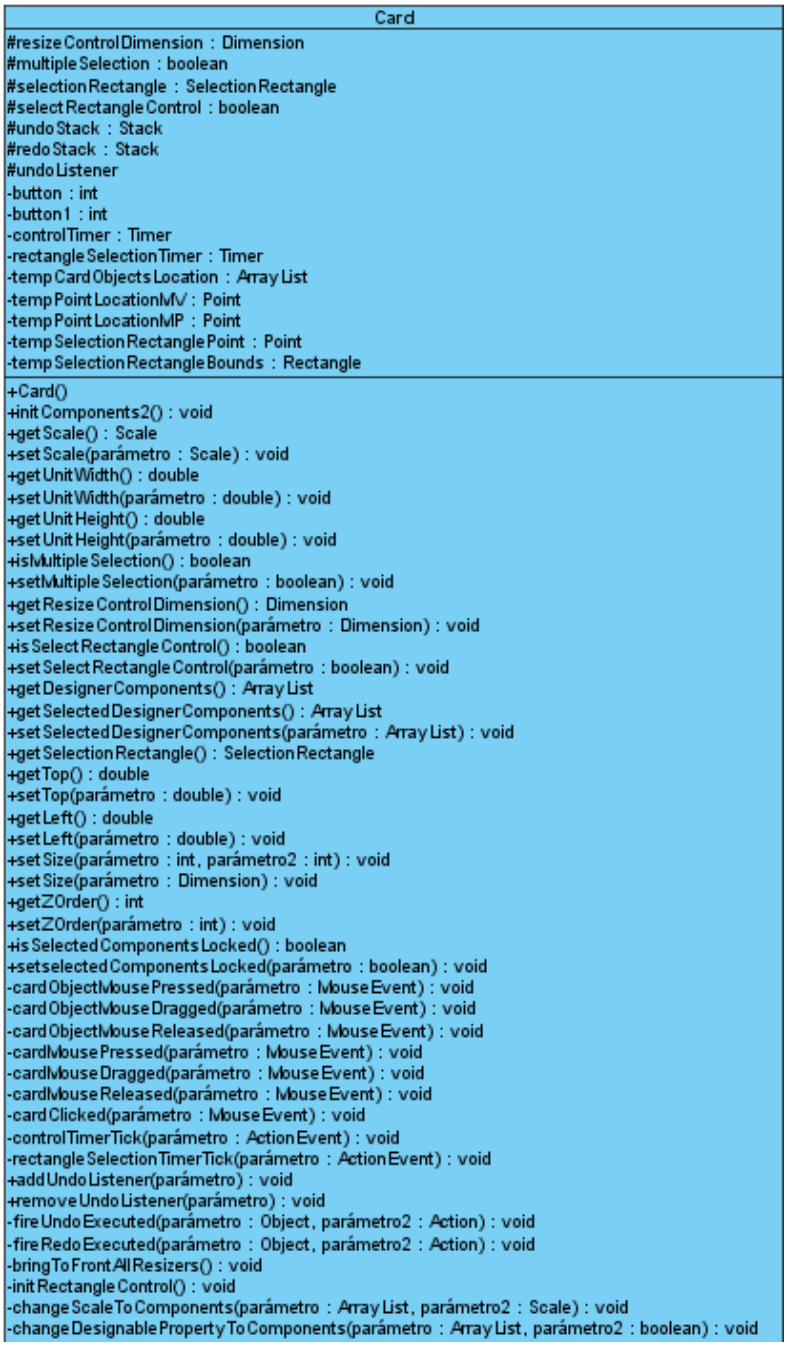


Figura 65: Representación gráfica de la clase Ruler.


Card
#resize Control Dimension : Dimension
#multiple Selection : boolean
#selection Rectangle : Selection Rectangle
#select Rectangle Control : boolean
#undo Stack : Stack
#redo Stack : Stack
#undo Listener
-button : int
-button1 : int
-control Timer : Timer
-rectangle Selection Timer : Timer
-temp Card Objects Location : Array List
-temp Point Location MV : Point
-temp Point Location MP : Point
-temp Selection Rectangle Point : Point
-temp Selection Rectangle Bounds : Rectangle
+Card()
+init Components2() : void
+get Scale() : Scale
+set Scale(parámetro : Scale) : void
+get Unit Width() : double
+set Unit Width(parámetro : double) : void
+get Unit Height() : double
+set Unit Height(parámetro : double) : void
+is Multiple Selection() : boolean
+set Multiple Selection(parámetro : boolean) : void
+get Resize Control Dimension() : Dimension
+set Resize Control Dimension(parámetro : Dimension) : void
+is Select Rectangle Control() : boolean
+set Select Rectangle Control(parámetro : boolean) : void
+get Designer Components() : Array List
+get Selected Designer Components() : Array List
+set Selected Designer Components(parámetro : Array List) : void
+get Selection Rectangle() : Selection Rectangle
+get Top() : double
+set Top(parámetro : double) : void
+get Left() : double
+set Left(parámetro : double) : void
+set Size(parámetro : int, parámetro2 : int) : void
+set Size(parámetro : Dimension) : void
+get ZOrder() : int
+set ZOrder(parámetro : int) : void
+is Selected Components Locked() : boolean
+set selected Components Locked(parámetro : boolean) : void
-card Object Mouse Pressed(parámetro : Mouse Event) : void
-card Object Mouse Dragged(parámetro : Mouse Event) : void
-card Object Mouse Released(parámetro : Mouse Event) : void
-card Mouse Pressed(parámetro : Mouse Event) : void
-card Mouse Dragged(parámetro : Mouse Event) : void
-card Mouse Released(parámetro : Mouse Event) : void
-card Clicked(parámetro : Mouse Event) : void
-control Timer Tick(parámetro : Action Event) : void
-rectangle Selection Timer Tick(parámetro : Action Event) : void
+add Undo Listener(parámetro) : void
+remove Undo Listener(parámetro) : void
-fire Undo Executed(parámetro : Object, parámetro2 : Action) : void
-fire Redo Executed(parámetro : Object, parámetro2 : Action) : void
-bring To Front All Resizers() : void
-init Rectangle Control() : void
-change Scale To Components(parámetro : Array List, parámetro2 : Scale) : void
-change Designable Property To Components(parámetro : Array List, parámetro2 : boolean) : void

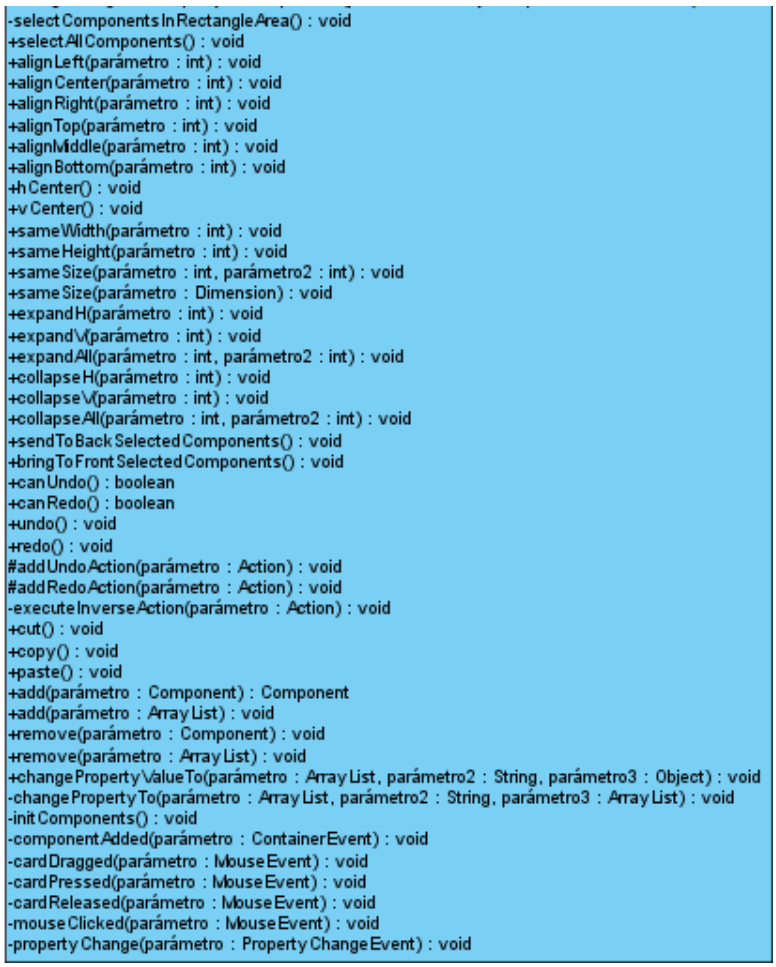
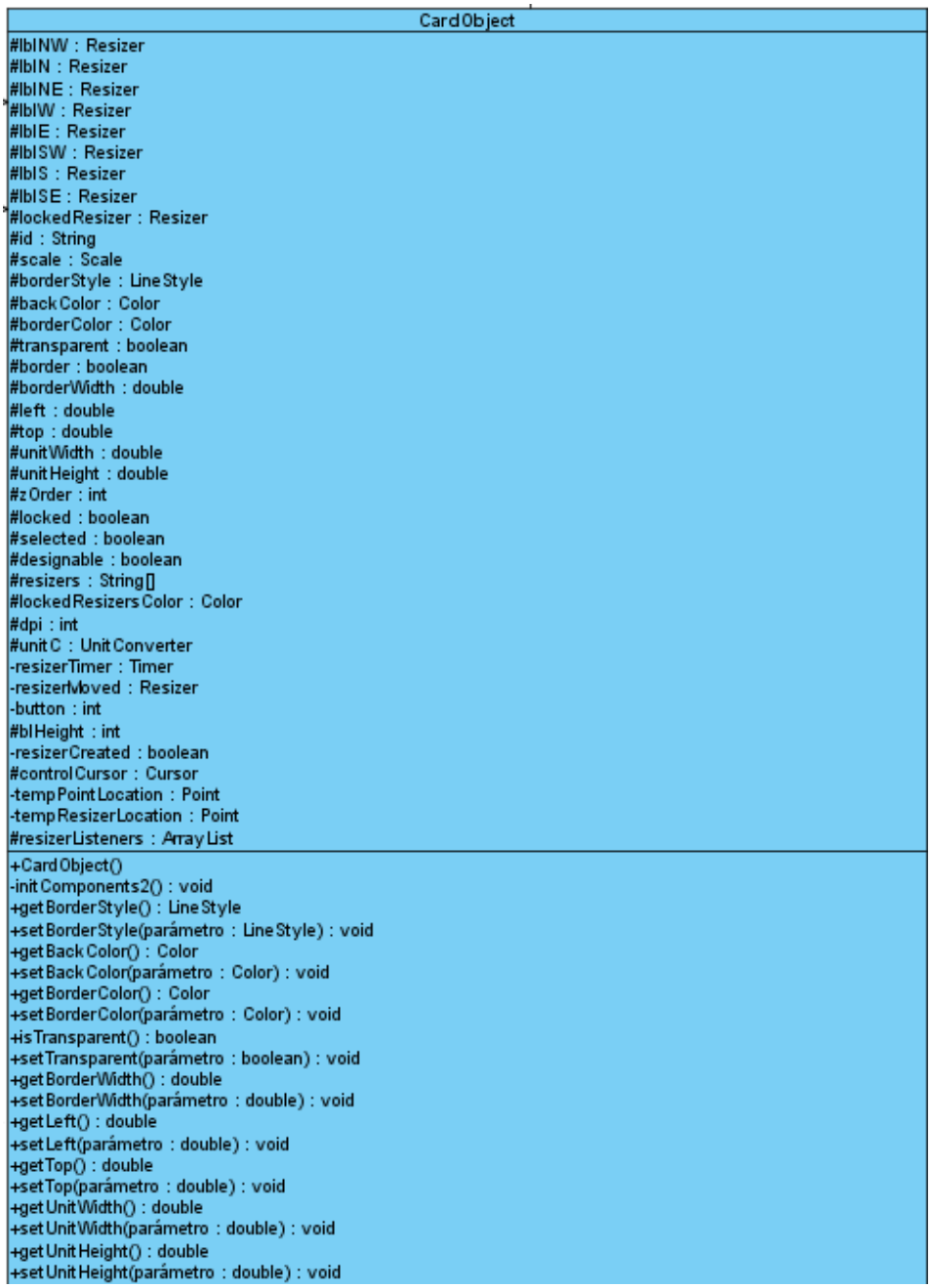


Figura 66: Representación gráfica de la clase Card.


CardObject
#lblNW : Resizer
#lblN : Resizer
#lblNE : Resizer
#lblW : Resizer
#lblE : Resizer
#lblSW : Resizer
#lblS : Resizer
#lblSE : Resizer
#lockedResizer : Resizer
#id : String
#scale : Scale
#borderStyle : LineStyle
#backColor : Color
#borderColor : Color
#transparent : boolean
#border : boolean
#borderWidth : double
#left : double
#top : double
#unitWidth : double
#unitHeight : double
#zOrder : int
#locked : boolean
#selected : boolean
#designable : boolean
#resizers : String[]
#lockedResizersColor : Color
#dpi : int
#unitC : UnitConverter
-resizerTimer : Timer
-resizerMoved : Resizer
-button : int
#blHeight : int
-resizerCreated : boolean
#controlCursor : Cursor
-tempPointLocation : Point
-tempResizerLocation : Point
#resizerListeners : ArrayList
+CardObject()
-initComponents2() : void
+getBorderStyle() : LineStyle
+setBorderStyle(parámetro : LineStyle) : void
+getBackColor() : Color
+setBackColor(parámetro : Color) : void
+getBorderColor() : Color
+setBorderColor(parámetro : Color) : void
+isTransparent() : boolean
+setTransparent(parámetro : boolean) : void
+getBorderWidth() : double
+setBorderWidth(parámetro : double) : void
+getLeft() : double
+setLeft(parámetro : double) : void
+getTop() : double
+setTop(parámetro : double) : void
+getUnitWidth() : double
+setUnitWidth(parámetro : double) : void
+getUnitHeight() : double
+setUnitHeight(parámetro : double) : void

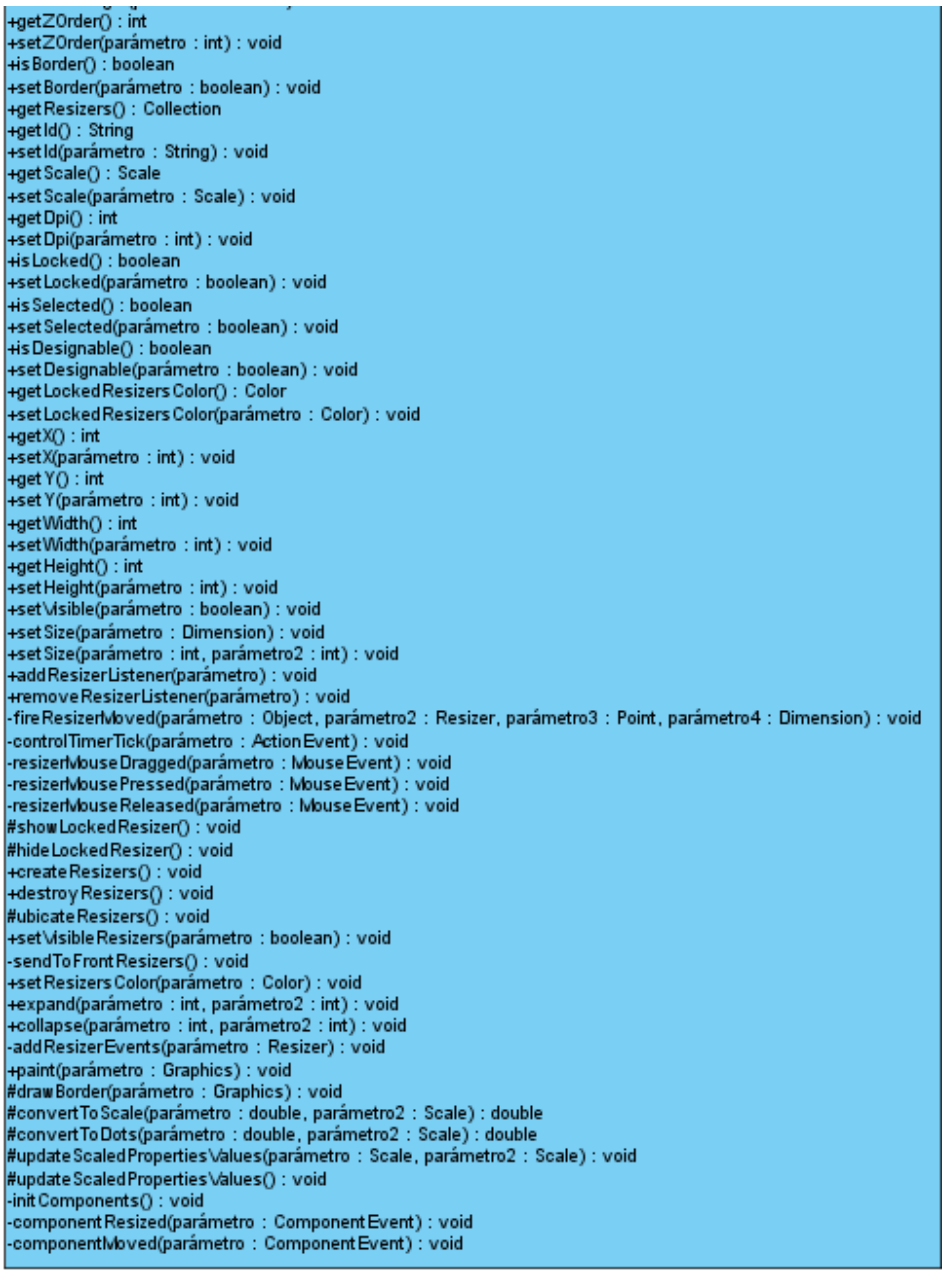


Figura 67: Representación gráfica de la clase CardObject.

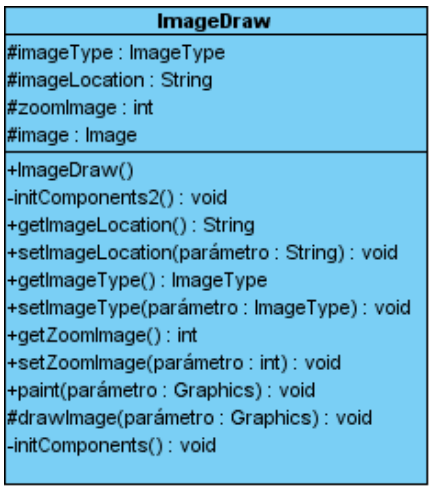


Figura 68: Representación gráfica de la clase ImageDraw.

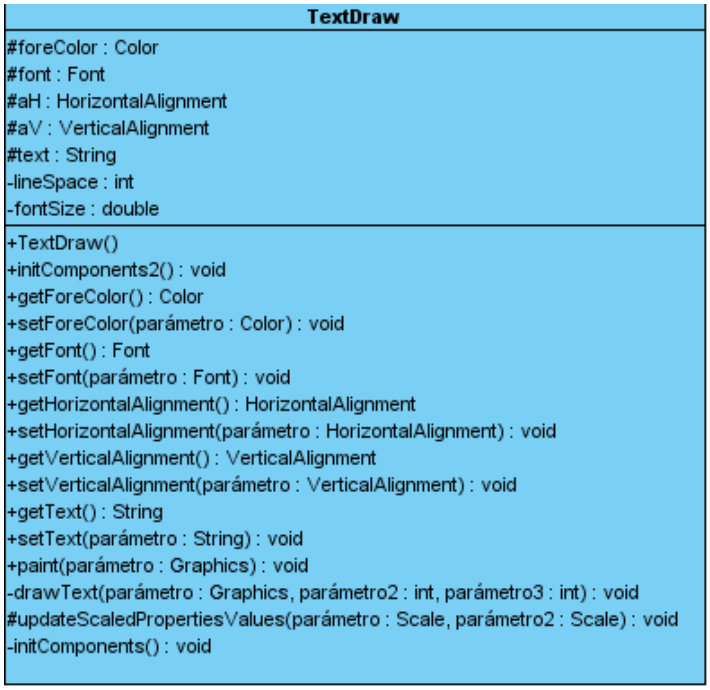


Figura 69: Representación gráfica de la clase TextDraw.

Figura 70: Representación gráfica de la clase LineDraw.

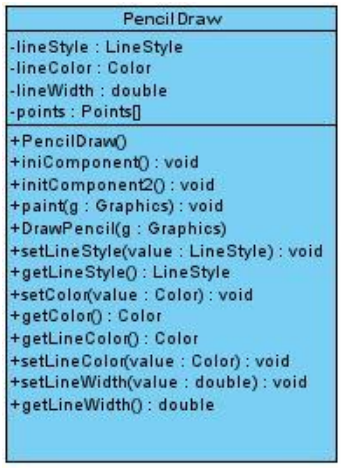


Figura 71: Representación gráfica de la clase PencilDraw.

Figura 72: Representación gráfica de la clase PolygonDraw.

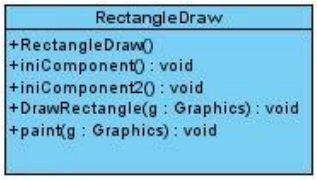


Figura 73: Representación gráfica de la clase RectangleDraw.

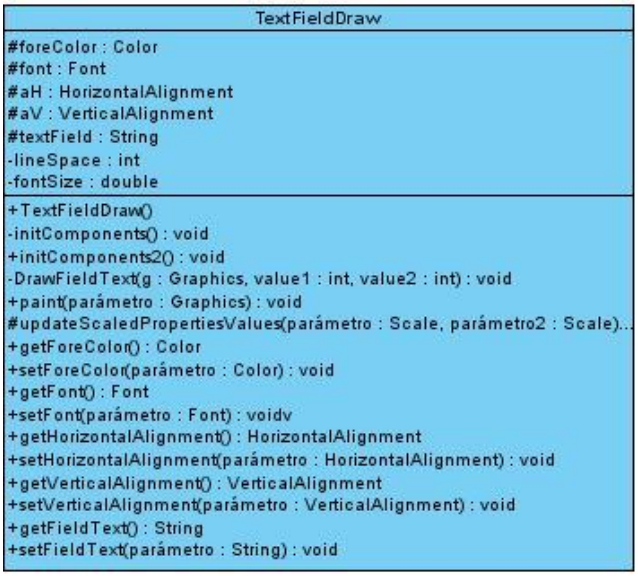


Figura 74: Representación gráfica de la clase TextFieldDraw.

Figura 75: Representación gráfica de la clase EllipseDraw.

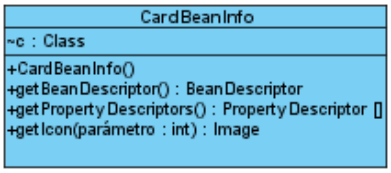


Figura 76: Representación gráfica de la clase CardBeanInfo.

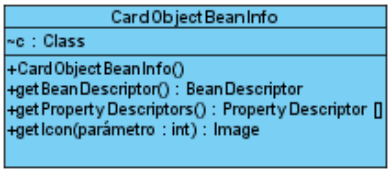


Figura 77: Representación gráfica de la clase CardObjectBeanInfo.

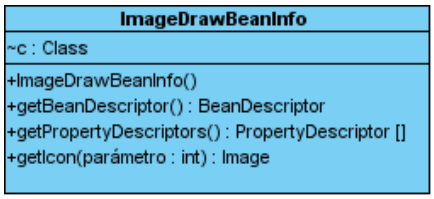


Figura 78: Representación gráfica de la clase ImageDrawBeanInfo.

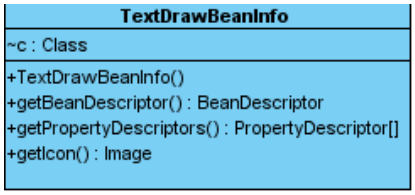


Figura 79: Representación gráfica de la clase TextDrawBeanInfo.

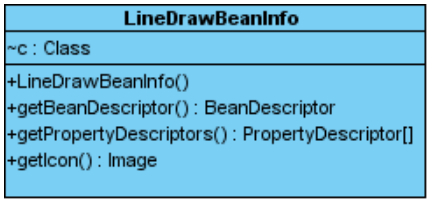


Figura 80: Representación gráfica de la clase LineDrawBeanInfo.

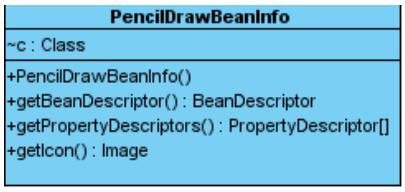


Figura 81: Representación gráfica de la clase PencilDrawBeanInfo.

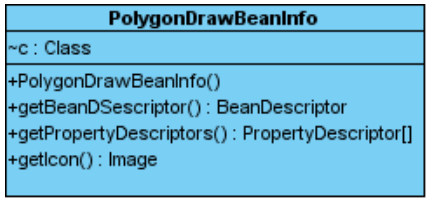


Figura 82: Representación gráfica de la clase PolygonDrawBeanInfo.

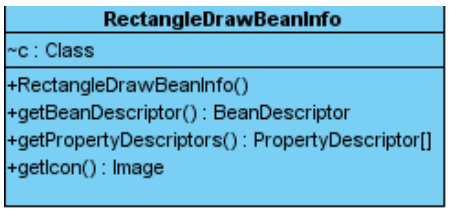


Figura 83: Representación gráfica de la clase RectangleDrawBeanInfo.

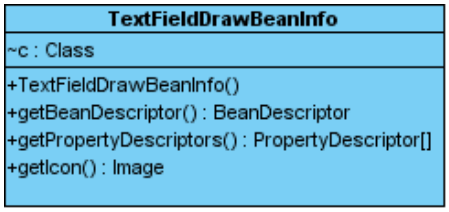


Figura 84: Representación gráfica de la clase TextFieldDrawBeanInfo.

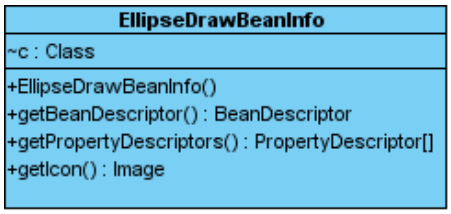


Figura 85: Representación gráfica de la clase EllipseDrawBeanInfo.