UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO

FACULTAD DE INGENIERIA ELÉCTRICA

“REINGENIERÍA DE PROCESOS APLICADA A UN GENERADOR, VALIDADOR DE PÓLIZAS CONTABLES”

TESIS

PARA OBTENER EL GRADO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

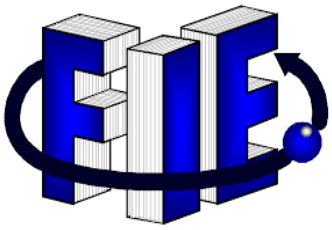
JESUS EDUARDO LEMUS VILLICAÑA

ASESOR:

ISC. Bertha Georgina Flores Díaz

MORELIA, MICHOACÁN

ENERO 2020



Agradecimientos

Cuando se inicia una carrera se inicia con temor y muchas preguntas que conforme avanza el tiempo estas se van disolviendo gracias a los profesores quienes te transmiten su conocimiento y a veces que tomes la iniciativa para hacer cosas interesantes y así, cuando salgas al ambiente laboral te defiendas ante la jungla de asfalto.

De la mano van los padres son los motores de ignición que necesitamos para despegar, ya que todo el tiempo te dan el apoyo que necesitas para que tengas todo lo necesario y salgas adelante. Nunca habrá palabras suficientes para agradecerles el haberme dado los estudios que ahora tengo.

Cuando uno es estudiante se conocen muchos compañeros y amigos con los cuales se viven experiencias y al igual entre nosotros nos ayudamos para que nos sea de una manera más amena el día a día de la escuela es por esto que me atrevo a decir que una carrera no se termina solo sino acompañado por todos aquellos que conocí en el transcurso de la misma.

Gracias a todos.

Dedicatoria

A mi esposa quien todo el tiempo a pesar de todas las cosas que se presentaron siempre tuvo paciencia y mucho apoyo, ya que sin ella no habría podido completar mi carrera y mucho menos esta tesis. Gracias por todo guapa.

.

Resumen

Esta tesis presenta el rediseño e implementación de un generador-validador de pólizas contables, para la empresa INOWEBS bajo los estándares establecidos por el Servicio de Administración Tributaria (SAT), haciendo uso de la metodología de reingeniería de procesos adoptada por **Michael Hammer y James Champy**.

Este sistema consiste en tomar un archivo de texto que se valida y pasa a un proceso llamado serialización, que consiste, en tomar un esquema XSD (XML Schema Definition) establecido por el SAT y a partir de este generar archivos de tipo XML (Lenguaje de Marcas Extensible), por lo que dicho sistema es capaz de leer, validar y generar un archivo XML.

Palabras clave: Reingeniería de procesos, serialización, validación, tiempo de ejecución.,XML

Abstract

This thesis takes as a practical case the local software development company called INOWEBS. This company is responsible for developing technological solutions to meet the requirements related to electronic documents with official validity before the SAT and within these developments is electronic accounting. The generator-validator system of accounting policies is analyzed and redesigned, under the reengineering methodology process adopted by Michael Hammer and James Champy (Champy, 2001).

The system takes a text file that is validated and goes to a process called serialization, which consists of taking a scheme (XSD) established by the SAT and generates the XML (Extensible Markup Language) files using the Python programming language and the serialization of Visual Basic .NET. The system is able to read, validate and generate an XML file easily and quickly.

Key Words: Process reengineering, serialization, validation, execution time.

Contenido

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

1.1 Identificación del problema

Todas las empresas en México necesitan declarar ante el SAT (Servicio de Administración Tributaria) de manera mensual las operaciones financieras a través de pólizas contables, de ingresos y de egresos de manera electrónica, a través de un archivo XML, para cumplir con sus obligaciones fiscales, como lo indica el Anexo 24 publicadas en la Segunda Modificación a la Resolución Miscelánea Fiscal 2014 (RMF-14). Para facilitar la declaración fiscal, se puede automatizar este proceso, recurriendo al desarrollo de software.

En la actualidad existe una gran diversidad de sistemas informáticos cuya tarea es cumplir con un proceso determinado, por lo general estos ya están implementados en las empresas, pero la pregunta es, ¿en cuánto tiempo lo realiza? ¿se pueden mejorar?,¿es la mejor forma de implementarlo?

En la presente tesis se toma como caso de estudio la empresa INOWEBS, la cual se dedica a crear sistemas contables y de facturación. Esta empresa cuenta con un sistema cuya función es la generación de pólizas contables, el cual es muy tardado. El objetivo del presente trabajo es implementar la reingeniería de procesos adoptada por Michael Hammer y James Champy (Champy, 2001).

Al aplicar la reingeniería de procesos se propone analizar el sistema que se encuentra en funcionamiento actualmente, el cual se le llamará durante la presente tesis como “sistema anterior”.

A partir de este análisis desarrollar un nuevo sistema el cual se le nombrará como “sistema actual”, este sistema, tendrá que tener cambios significativos en el tiempo de ejecución para mejorar la producción de pólizas contables a partir de archivo CSV en más de un 50%.

1.2 Antecedentes

La empresa Moreliana INOWEBS inicia su operación en el año 2014 dedicada al desarrollo de sistemas computarizados. Dicha empresa se encarga de desarrollar soluciones tecnológicas para cubrir los requerimientos relacionados a documentos electrónicos con validez oficial ante el SAT y dentro de estos desarrollos se encuentra la contabilidad electrónica.

Durante el año 2014 en las reglas i.2.8.6., i.2.8.7. y i.2.8.8. y del Anexo 24 publicadas en la Segunda Modificación a la Resolución Miscelánea Fiscal 2014 (RMF-14), obliga a ciertos clientes de dicha empresa a declarar su contabilidad electrónica y entre estos requerimientos se encuentran las pólizas contables, por lo cual surge una petición para crear un nuevo desarrollo de un validador y generador de pólizas en base al reglamento que establece el SAT.

Este sistema procesa pólizas con una gran cantidad de datos, lo cual causa que la plataforma web se vea afectada en el servidor ya que las peticiones consumen una gran cantidad de recursos, afectado el funcionamiento del sistema para otros clientes que también hacen uso del servidor.

La empresa decide desarrollar un nuevo software que sea más rápido, para procesar estos archivos que contienen las pólizas y así satisfacer las necesidades de las empresas solicitantes, al nuevo sistema lo llamaremos sistema actual.

1.3 Objetivos

1.3.1 Objetivo general

Analizar, rediseñar y desarrollar un sistema de información (validación y generación de archivos XML) de la empresa INOWEBS, con el fin de reducir los tiempos de ejecución, así como rediseñar e implementar nuevos procedimientos y en consecuencia incrementar la producción de pólizas con respecto al sistema actual.

1.3.2 Objetivos específicos

- Analizar los procesos de validación y generación de pólizas.
- Rediseñar los procesos del sistema implementando la metodología de reingeniería de procesos propuesta por Michael Hammer y James Champy en su libro "Reengineering the corporation", utilizando los lenguajes de programación Visual Basic y Python.
- Desarrollar el sistema utilizando la reingeniería propuesta
- Verificar que efectivamente el tiempo de ejecución del nuevo sistema se vea reducido por cada póliza generada aumentando su producción.

1.4 Descripción de los capítulos

Capítulo 1 Introducción: En este capítulo se da una breve introducción del tema de tesis a desarrollar, sus antecedentes, el problema de la tesis, se hace mención también a los objetivos a los que pretende se pretende llegar tanto generales, como específicos.

Capítulo 2 Marco teórico: Se aborda el concepto de la reingeniería de procesos describiendo cada una de sus fases, el concepto de contabilidad electrónica, así como las definiciones de las tecnologías utilizadas para el desarrollo y el entendimiento de las tecnologías utilizadas y a utilizar durante el desarrollo de la presente tesis.

Capítulo 3 Diseño e implementación: Durante el desarrollo del capítulo se expone, el análisis del problema a través del análisis del sistema anterior, para posteriormente tener un panorama de las fases del proyecto nuevo y así llegar a su implementación, aplicando la reingeniería de procesos propuesta por Michael Hammer y James Champy.

Capítulo 4 Pruebas y resultados: Se analiza el nuevo sistema con respecto al anterior dentro de un ambiente de pruebas en donde se analizan los tiempos de validación y tiempos de generación de archivos XML en donde se llega a un resultado después de varias pruebas.

Capítulo 5 Conclusiones: se presentan las conclusiones a las cuales se ha podido llegar después de realizar las diferentes pruebas con diferentes cantidades de pólizas

Capítulo 2

Marco teórico

2.1 Contabilidad electrónica

La contabilidad electrónica es el envío de archivos en formato XML de las transacciones registradas en medios electrónicos que realiza una empresa o una persona y envía a través del buzón tributario del SAT (SAT, s.f.).

El SAT, en las reglas i.2.8.6, i.2.8.7. y i.2.8.8. del Anexo 24, publicadas en la Segunda Modificación a la Resolución Miscelánea Fiscal 2014 (RMF-14), para dar cumplimiento a las fracciones iii y iv del artículo 28 del Código Fiscal de la Federación (CFF), se describe lo que los contribuyentes deberían reportar en relación con la contabilidad electrónica. El 19 de agosto de 2014 se publicaron modificaciones a dichas reglas en la Tercera Modificación a la RMF-14 (León , 2014).

Actualmente, el SAT ha liberado la especificación técnica para lo que será la nueva versión 1.3.

A partir del 1 de junio 2017 las empresas deben llevar su contabilidad fiscal por medios electrónicos, por lo que deberán actualizar su software contable y adaptar su proceso de registro de la contabilidad, ya que la forma de capturar o registrar los movimientos en la contabilidad no cumplirán con los cambios establecidos en la Ley (Contalinea, 2017).

El presente documento tomará la información de pólizas, ya que la empresa solicitante requiere que en caso de que el SAT solicite información contable, pueda enviar dichas pólizas mensualmente, como lo describen los paso en la Figura 2.1.

Figura 2. 1 Pasos para enviar las pólizas contables en el buzón tributario

Las pólizas contables son un documento físico o digital en el que se registran las operaciones contables desarrolladas por una persona o una empresa. Se deben registrar de preferencia a más tardar dentro de los cinco días siguientes a la realización de la operación; acto o actividad. Estas se envían a través del Buzón Tributario, en el apartado Contabilidad electrónica, con firma electrónica.

Los tipos de pólizas contables que existen se clasifican en: diario, ingresos y egresos.

Dichas características existen dentro de una póliza capturada manualmente ya que dentro de una póliza generada electrónicamente está relacionada con su respectivo comprobante fiscal digital (CFDI).

Póliza de diario: Una póliza de diario sirve para registrar las operaciones que afectan la economía de la empresa; pero que no representan flujo de efectivo alguno.

Es decir, es la que se elabora cuando la operación que se está registrando no implica una entrada de dinero al banco a través de una ficha de depósito ni una salida por la cual se deba elaborar un cheque, es importante resaltar que este tipo de pólizas siempre debe de tener anexo el comprobante que está dando origen a su elaboración. En la figura 2.2 se

PÓLIZA DE DIARIO

CUENTA	SUBCUENTA	NOMBRE	PARCIAL	DEBE	HABER
204		Acreedores Diversos		$ 30,000.00	
	204.01	Rubén Altamirano	$ 30,000.00		
210		Anticipo de clientes		5,000.00	
	210.01	Joaquín Estraza	$ 5,000.00		
104		Clientes			$ 35,000.00
	104.01	Joaquín Estraza	$ 5,000.00		
	104.02	Rubén Altamirano	30,000.00		
		SUMAS IGUALES		$ 35,000.00	$ 35,000.00

CONCEPTO

Compensación de saldos de los clientes Joaquín Estraza y Rubén Altamirano.

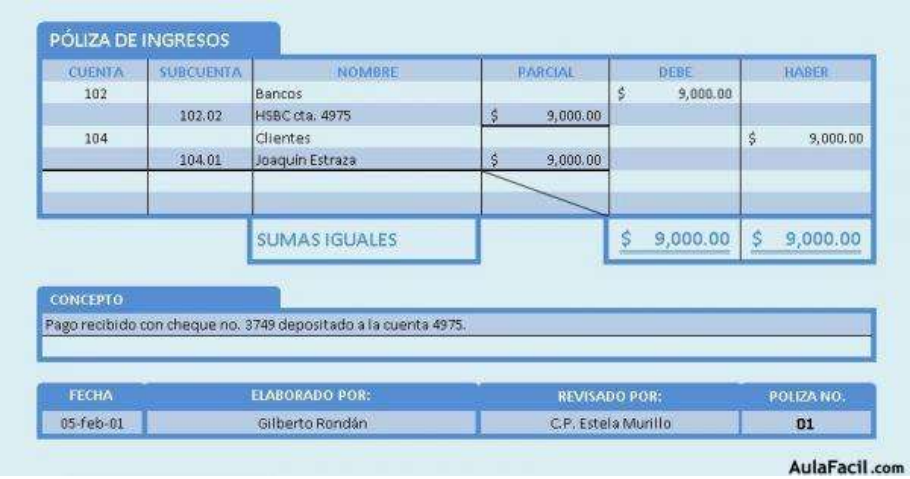
FECHA	ELABORADO POR:	REVISADO POR:	POLIZA NO.
22-feb-01	Gilberto Rondán	C.P. Estela Murillo	02

AulaFacil.com

muestra la captura de una póliza de diario realizada manualmente

Póliza de ingresos: Las pólizas de ingresos son las que contienen los registros contables de todo aquello que implique entrada de dinero a la empresa ya sea en efectivo, transferencia o cheque y del cual tienes que expedir una factura, por ejemplo, el pago que hace un cliente.

En la figura 2.3 se visualiza una póliza de ingresos capturada manualmente en donde se visualizan las características anteriormente mencionadas.



PÓLIZA DE INGRESOS

CUENTA	SUBCUENTA	NOMBRE	PARCIAL	DEBE	HABER
102		Bancos		$ 9,000.00	
	102.02	HSBC cta. 4975	$ 9,000.00		
104		Clientes			$ 9,000.00
	104.01	Joaquín Estraza	$ 9,000.00		
		SUMAS IGUALES		$ 9,000.00	$ 9,000.00

CONCEPTO

Pago recibido con cheque no. 3749 depositado a la cuenta 4975.

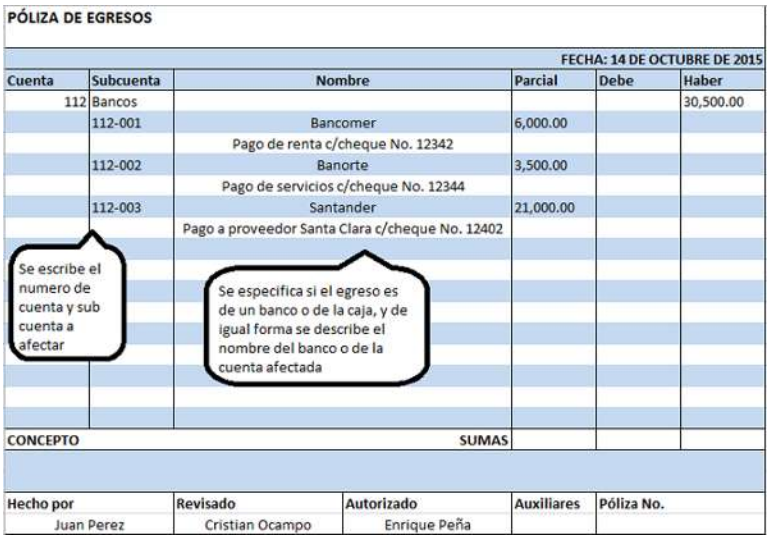
FECHA	ELABORADO POR:	REVISADO POR:	POLIZA NO.
05-feb-01	Gilberto Rondán	C.P. Estela Murillo	01

AulaFacil.com

Figura 2. 3 Póliza de ingresos captura manual de una póliza de ingresos

Póliza de egresos: Las pólizas de egresos sirven para registrar las operaciones contables que impliquen erogaciones (egresos, pagos) o salidas de dinero para la empresa. Pero debemos tener en cuenta que, si la erogación (Se denomina erogación a los gastos que realiza un determinado agente económico) se realiza por medio de un cheque, la póliza contable generada se conocerá como póliza de cheque, tal como se muestra en la figura 2.4.

Por lo cual cada póliza debe contener los folios fiscales (UUID) de los comprobantes fiscales que avalen cada una de las operaciones, permitiendo identificar la forma de pago, las distintas contribuciones, tasas y cuotas, incluyendo aquellas operaciones, actos o



PÓLIZA DE EGRESOS

FECHA: 14 DE OCTUBRE DE 2015

Cuenta	Subcuenta	Nombre	Parcial	Debe	Haber
112	Bancos				30,500.00
	112-001	Bancomer	6,000.00		
		Pago de renta c/cheque No. 12342			
	112-002	Banorte	3,500.00		
		Pago de servicios c/cheque No. 12344			
	112-003	Santander	21,000.00		
		Pago a proveedor Santa Clara c/cheque No. 12402			

Se escribe el numero de cuenta y sub cuenta a afectar

Se especifica si el egreso es de un banco o de la caja, y de igual forma se describe el nombre del banco o de la cuenta afectada

CONCEPTO	SUMAS			
Hecho por	Revisado	Autorizado	Auxiliares	Póliza No.
Juan Perez	Cristian Ocampo	Enrique Peña		

Figura 2. 4 Póliza de egresos con descripción de captura

actividades por las que no se deban pagar contribuciones.

2.2 Definición de conceptos

Tiempo de ejecución

Se denomina tiempo de ejecución al intervalo de tiempo en el que un programa de computadora se ejecuta en un sistema operativo.

Este tiempo se inicia con la puesta en memoria principal del programa, por lo que el sistema operativo comienza a ejecutar sus instrucciones. El intervalo de tiempo finaliza en el momento en que el programa envía al sistema operativo la señal de terminación, sea ésta una terminación normal, en que el programa tuvo la posibilidad de concluir sus instrucciones satisfactoriamente, o una terminación anormal, en el que el programa produjo algún error y el sistema debió forzar su finalización.

Lenguaje de programación Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multi - paradigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma (Sum, s.f.).

Python es un lenguaje de scripting independiente de plataforma, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- La sencillez y velocidad con la que se crean los programas.
- La cantidad de plataformas en las que podemos desarrollar (Alvarez, 2003).

Lenguaje de Marcas Extensible (XML)

XML proviene de extensible Markup Language ("Lenguaje de Marcas Extensible"). Se trata de un metalenguaje (un lenguaje que se utiliza para decir algo acerca de otro) extensible de etiquetas que fue desarrollado por el Word Wide Web Consortium (W3C), una sociedad mercantil internacional que elabora recomendaciones para la World Wide Web.

El XML es una adaptación del SGML (Standard Generalized Markup Language), un lenguaje de etiquetado de documentos. Esto quiere decir que el XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades (Europe, 2003).

Partes de un documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento, así como texto de etiquetas que contiene una gran variedad de efectos positivos o negativos en la referencia opcional a la que se refiere el documento, hay que tener mucho cuidado de esa parte de la gramática léxica para que se componga de manera uniforme (Gonzalo, 2010).

Prólogo:

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

El prólogo de un documento XML contiene:

- Declaración: Es la sentencia que declara al documento como un documento XML.
- Tipo de documento: Enlaza el documento con su definición de tipo de documento (DTD), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
- Uno o más comentarios e instrucciones de procesamiento.

A continuación, se muestra un prólogo dentro de un archivo XML en donde se puede visualizar la declaración, el tipo de documento y sus comentarios tal y como se muestra en

Figura 2. 5 Prólogo de un archivo XML

el ejemplo de la Figura 2.5:

Cuerpo:

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener solo un elemento raíz, característica indispensable también para que el documento esté bien formado. Sin embargo, es necesaria la adquisición de datos para su buen funcionamiento.

Lo conforma una etiqueta de inicio y una etiqueta de terminación, la cual se distingue por la anteposición de una diagonal "/".

Ejemplo:

```
<Edit_Mensaje> (…) </Edit_Mensaje>
```

Elementos:

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

Atributos:

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.

Por ejemplo, un elemento "fechaPublicacion" puede tener un atributo *"año"*. Como se muestra en la Figura 2.6

Figura 2. 6 Atributos en un archivo XML

Comentarios:

Comentarios a modo informativo para el programador que han de ser ignorados por el procesador. Los comentarios en XML tienen el siguiente formato:

Un ejemplo de un comentario en un archivo XML se escribe como se muestra a continuación:

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

Ahora si ejemplificamos la mayoría de estos conceptos en un ejemplo, como lo muestra la figura 2.7:

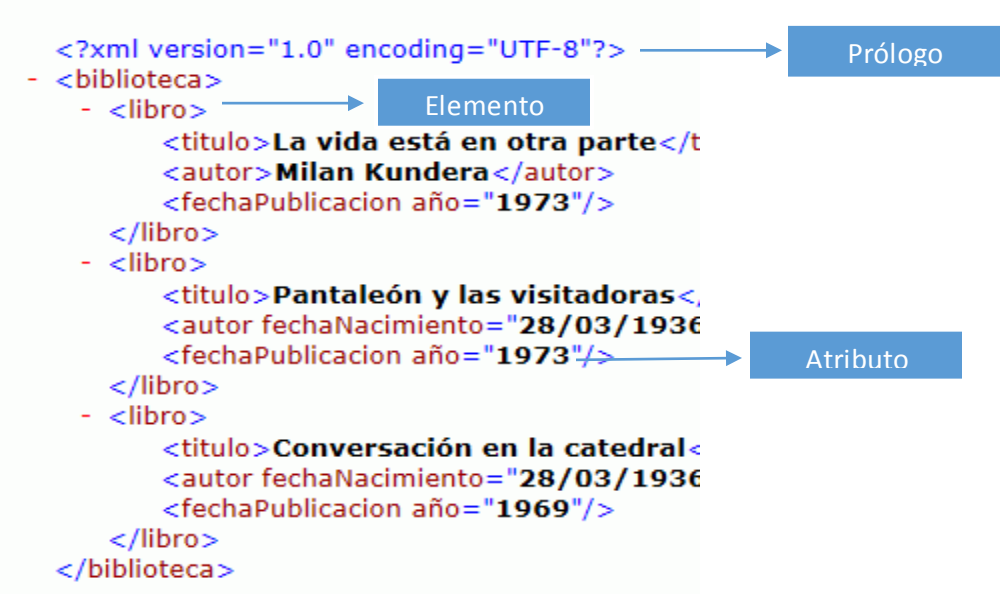


Figura 2. 7 Ejemplo de un archivo en formato XML

XML Schema Definition (XSD)

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML.

Se consigue así una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el World Wide Web Consortium (W3C) y alcanzó el nivel de recomendación en mayo de 2001.

Los documentos XSD (usalmente con extensión .xsd de XML Schema Definition) se concibieron como una alternativa a las DTD, más complejas, intentando superar sus puntos débiles y buscar nuevas capacidades a la hora de definir estructuras para documentos XML.

El principal aporte de XML Schema es el gran número de tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings (Morales, 2014).

Los componentes imprescindibles para construir un esquema XSD son los siguientes:

- xs:element: Este componente permite declarar los elementos del documento XML
- xs:attribute: Este componente permite declarar los atributos de los elementos del documento XML.

Tipos de datos:

Existen dos grandes grupos de tipos de datos que se pueden utilizar en los esquemas XSD:

Tipos de datos simples (xs:simpleType): Se dividen en los siguientes.

- Tipos de datos predefinidos.
- Tipos de datos construidos con nuestras propias restricciones y basados en los tipos de datos predefinidos.

Tipos de datos complejos (xs:complexType): Se dividen en los siguientes.

- Elementos dentro de otros elementos.
- Elementos que tienen atributos.

- Elementos mixtos que tienen datos y otros elementos.
- Elementos vacíos.

Tipos de datos predefinidos:

Son 44 los tipos de datos que define el esquema XSD, clasificados de una manera jerárquica, en el que los elementos hijos poseen las mismas características del padre más alguna particularidad añadida que los distinga

El tipos de datos predefinido principal es xs:anyType , el más genérico y del cual derivan el resto de tipo de dato predefinidos se muestra en la Figura 2.8.

Los tipos de datos predefinidos los podemos dividir en:

- Primitivos: Descienden directamente de xs:anySimpleType
- No primitivos: Descienden de alguno de los primitivos.

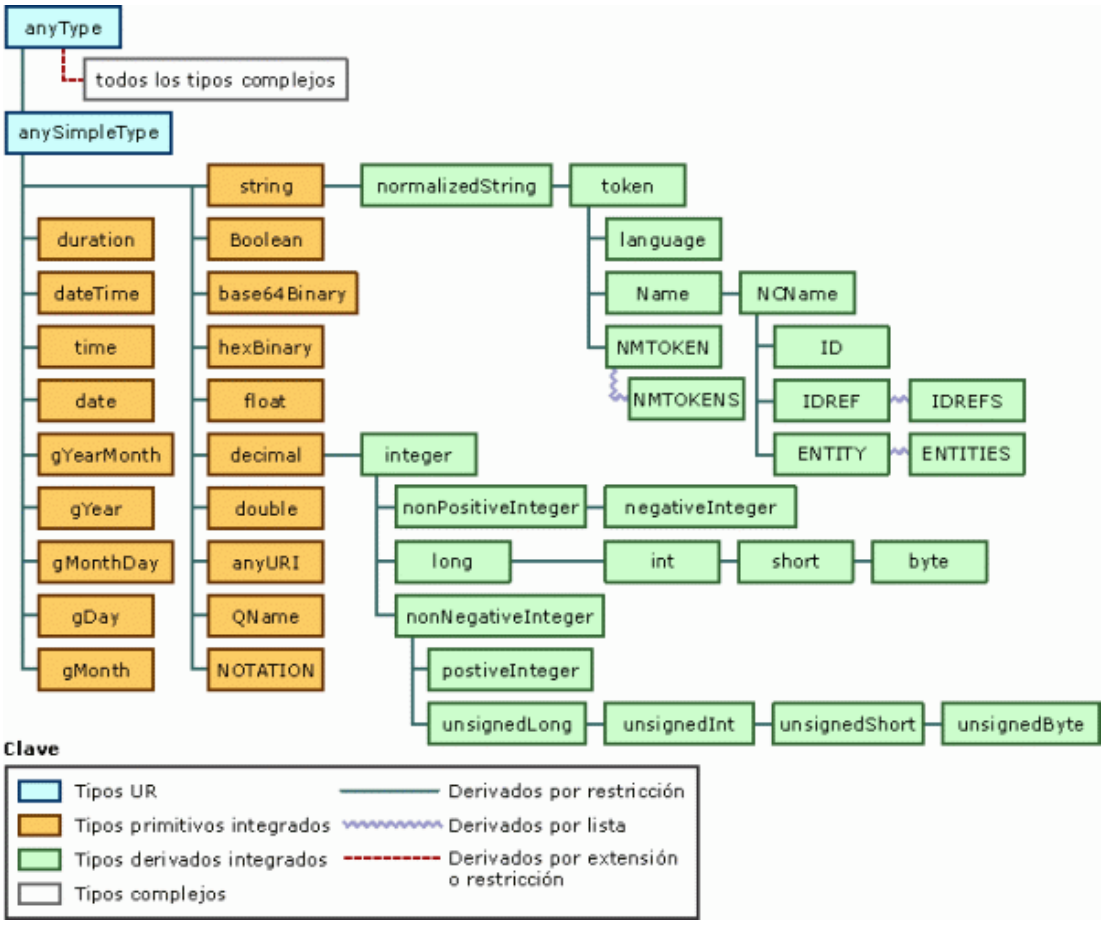


Figura 2. 8 Tipo de datos utilizados dentro de un formato XSD

En la figura 2.9 se muestra el ejemplo de un archivo XSD el cual se define por el elemento "note" que a su vez contiene los siguientes elementos de tipo string: to, from, heading, body.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Figura 2. 9 Ejemplo de un archivo en formato XSD para la estandarización de un XML

JavaScript Object Notation JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

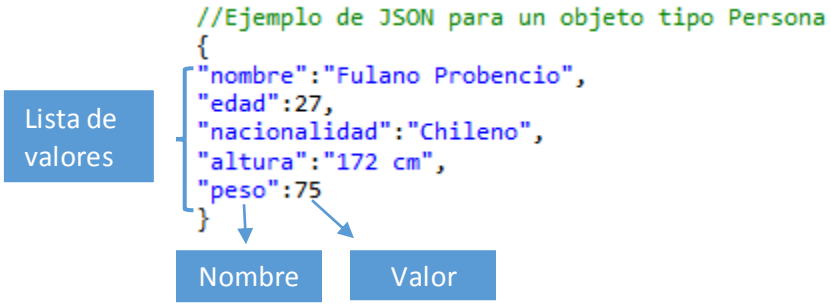
JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos (json.org, s.f.).

JSON está constituido por dos estructuras:

1. *Una colección de pares de nombre/valor*. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
2. *Una lista ordenada de valores*. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

El principio básico es con pares atributo-valor, éstos deben estar encerrados entre llaves { } que es lo que definen el inicio y el fin del objeto, como se muestra en el siguiente ejemplo en la Figura 2.10:



Valores separados por coma CSV

Un archivo CSV (Comma Separated Values, Valores Separados por Coma) contiene datos del tipo de los que se encuentran en bases de datos en formato de texto simple. Como el

Figura 2. 10 Ejemplo de un archivo en formato JSON

nombre indica cada valor está separado del siguiente por una coma.

El formato CSV es muy sencillo y no indica un juego de caracteres concreto, ni cómo van situados los bytes, ni el formato para el salto de línea. Estos puntos deben indicarse muchas veces al abrir el archivo, por ejemplo, con una hoja de cálculo o como formato para insertar datos en una base de datos.

El formato CSV no está estandarizado. La idea básica de separar los campos con una coma es muy clara, pero se vuelve complicada cuando los valores del campo también contienen comillas dobles o saltos de línea. Las implementaciones de CSV pueden no manejar esos datos, o usar comillas de otra clase para envolver el campo. Pero esto no resuelve el problema: algunos campos también necesitan embeber estas comillas, así que las implementaciones de CSV pueden incluir caracteres o secuencias de escape.

Además, el término "CSV" también denota otros formatos de valores separados por delimitadores que usan delimitadores diferentes a la coma (como los valores separados por tabuladores). Un delimitador que no está presente en los valores de los campos (como un tabulador) mantiene el formato simple. Estos archivos separados por delimitadores alternativos reciben en algunas ocasiones la extensión, aunque este uso sea incorrecto. Esto puede causar problemas en el intercambio de datos, por ello muchas aplicaciones que usan archivos CSV tienen opciones para cambiar el carácter delimitador. (Files, 2005). La figura 2.11, muestra un ejemplo de un archivo en formato CSV delimitado por comas.

```
Año,Marca,Modelo,Descripción,Precio
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

Figura 2. 11 Ejemplo de un archivo en formato CSV delimitado por comas

En la Figura 2.12, muestra un ejemplo de un archivo en formato CSV delimitado por pipes

```
1|JUAN|11/12/2013|Culiacan|CONTADOR|20829
2|MARIA|07/08/2010|Los Mochis|GERENTE GENERAL|22348
3|PEDRO|25/02/2011|Guadalajara|VENTAS 1|14858
4|JOSEFA|14/03/2014|Tijuana|VENTAS 2|18518
5|JORGE|03/03/2011|Culiacan|GERENTE OPERACIONES|17090
6|MARIO|24/03/2011|Guadalajara|COMPRAS|20174
7|ARTURO|19/09/2013|Los Mochis|MANTENIMIENTO|23490
8|MARTHA|10/01/2012|Culiacan|GERENTE DE VENTAS|22919
9|MANUEL|17/03/2013|Guadalajara|ADMINISTRADOR|14625
10|PABLO|19/07/2012|Tijuana|AUDITOR|19687
```

Figura 2. 12 Ejemplo de un archivo en formato CSV delimitado por pipes (|)

2.3 IDE Visual Studio 2017

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas, entre otros.

En el editor de código de Visual Studio 2017 como se muestra en la figura 2.13, encontramos una nueva característica llamada Visualizador Estructurado. Esta novedad muestra guías verticales entre áreas anidadas de código, facilitándonos la visualización y navegación a través del código. Esta característica está disponible para todos los lenguajes basados en TextMate como Visual C#, Visual Basic y XML (Gualix, 2017).

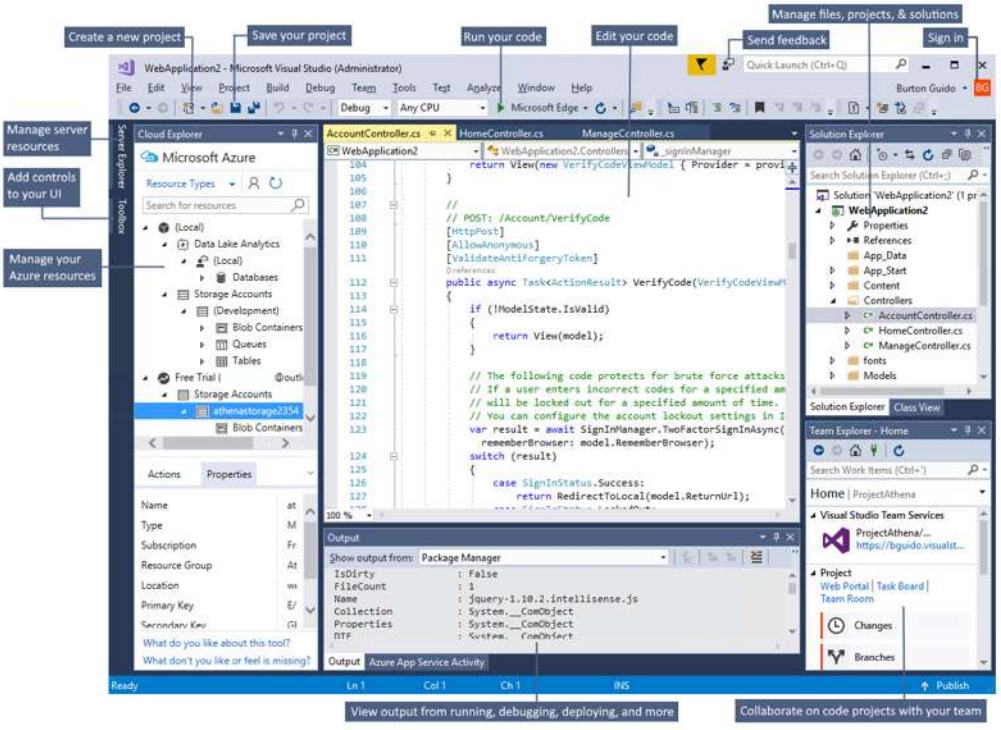


Figura 2. 13 Captura de pantalla de Visual Studio 2015

2.4 IDE Spyder

Es un entorno de desarrollo integrado (*IDE*) y multiplataforma de código abierto, para programación científica en el lenguaje Python (Amoedo, 2017). Posee funciones avanzadas de edición, pruebas interactivas y depuración, a continuación, se listan algunas ventajas, El editor que integra este IDE es multilenguaje.

- Consola interactiva. Las consolas Python o IPython son un espacio de trabajo y soporte de depuración para evaluar al instante el código escrito en el editor.
- Posee un visor de documentación. El programa puede mostrar documentación para cualquier llamada de clase o función realizada en el editor o en una consola.
- Explorar las variables creadas durante la ejecución de un archivo.
- La posibilidad de buscar en archivos. También ofrece soporte de expresiones regulares.
- Se dispone de un explorador de archivos para una mayor comodidad. Se tiene acceso al registro del historial.
- Registro de historial de comandos.

En la figura 2.14 se muestra la interfaz del IDE Spyder para Python:

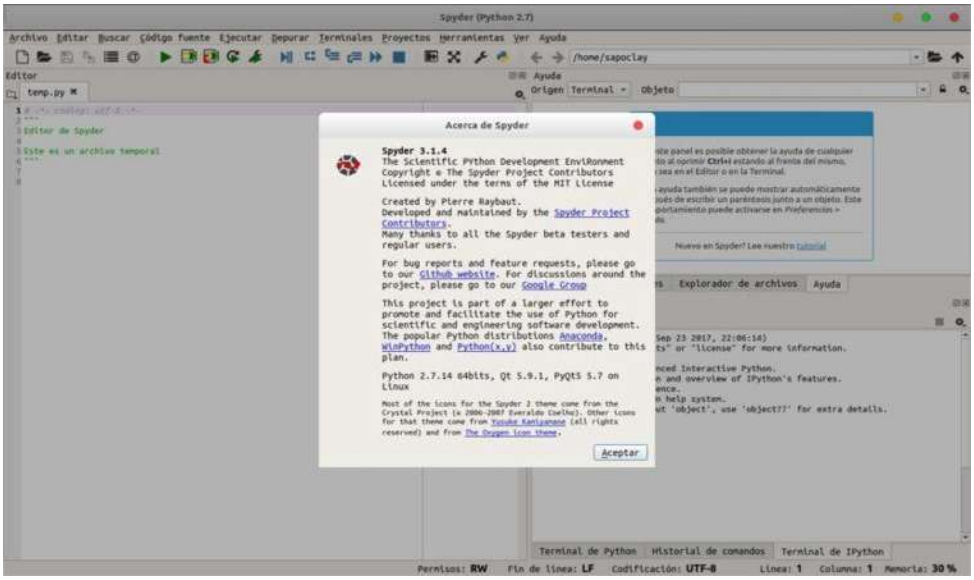


Figura 2. 14 Captura de pantalla del IDE Spyder para Python

2.5 Serialización

La serialización es el proceso de convertir un objeto en un flujo de bytes para almacenar el objeto o transmitirlo a la memoria, una base de datos o un archivo. Su propósito principal es guardar el estado de un objeto para poder recrearlo cuando sea necesario. El proceso

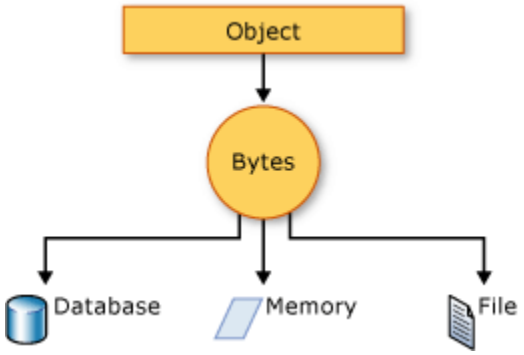


Figura 2. 15 Diagrama de funcionamiento de la serialización

inverso se llama deserialización.

El objeto se serializa a una secuencia, que transporta no solo los datos, sino también información sobre el tipo del objeto, como su versión, cultura y nombre de conjunto. Desde esa secuencia, se puede almacenar en una base de datos, un archivo o una memoria, como se muestra en la Figura 2.15 en donde se describe un diagrama de este proceso (Microsoft , 2015).

2.6 Reingeniería de procesos

La expresión Reingeniería de Procesos o BPR (*Business Process Reeingeniering*) fue adoptada por **Michael Hammer y James Champy** en el ya clásico volumen, publicado en (Champy, 2001), la cual consiste en la revisión fundamental y el rediseño radical de procesos para alcanzar mejoras espectaculares en medidas críticas y contemporáneas de rendimiento, tales como costes, calidad, servicio y rapidez, tal y como lo dicen los descubrimientos de Adam Smith "de que el trabajo industrial debía dividirse en sus tareas más simples y básicas" (gestión, 2016).

En el trabajo de (Davenport, 1990) apunta que la reingeniería de procesos es solo una parte de lo que es necesario en un cambio radical de procesos, por cuanto se refiere específicamente al diseño del nuevo proceso. La innovación de procesos involucra la visión y estrategias del nuevo trabajo, el diseño del proceso y la ejecución del cambio en sus complejas dimensiones; tecnológica, humana y organizacional (C., 2005).

Para algunos autores como Lowenthal "la Reingeniería de procesos o modelo... proporciona un enfoque global al rediseño y reconstrucción de una organización. Es más amplio que el modelo de Reingeniería, que es sólo un componente del modelo. Este modelo proporciona pasos de acción para los aspectos técnicos, culturales y estratégicos de reingeniería en una organización" (C., 2005).

Estos autores ponen énfasis en el carácter radical de las mejoras en el rendimiento que una organización puede obtener a través del rediseño radical de sus procesos.

Los procesos de la empresa INOWEBS surgieron de forma improvisada o apresurada. Es decir, a lo largo del tiempo se fueron añadiendo actividades y usos como consecuencia de decisiones agregadas que fueron combinándose hasta configurar el proceso tal y como se encuentra en un momento determinado.

Durante el proceso, en algunas ocasiones se contaba con una planificación para responder a una situación dada, en la que estaban presentes una o más premisas sobre el modo de realizar el trabajo. Pero estas premisas perdieron validez y, consiguientemente, las actuaciones derivadas de ellas se tornaron ineficaces e ineficientes (pymes, 2018).

Ante la ausencia de una tecnología concreta aplicada para el sistema anterior, se propone aplicar los pasos de reingeniería de procesos para reestructurar los procedimientos y después aplicar tecnologías actualizadas para la automatizarlos.

2.7 Reingeniería de software

Son muchas y variadas las referencias que se pueden encontrar del concepto de reingeniería de software. Algunos, como Arnold, R.S, la definen como una actividad que mejora la comprensión del software, o bien, lo prepara o mejora para incrementar su facilidad de mantenimiento, reutilización o evolución. Para otros la reingeniería de software, es el examen y alteración de un sistema para reconstruirlo en una nueva forma y la subsiguiente implementación de esa forma. Otros lo ven como el proceso de ingeniería inversa seguida de una ingeniería directa. El concepto de reingeniería de software, está muy relacionado con el concepto reutilización, y así se puede comprobar en donde Briggerstaff (Biggerstaff, 1990), se refiere a la reutilización como la reaplicación de una variedad de tipos de conocimientos de un sistema a otro para reducir el esfuerzo de desarrollo y mantenimiento de ese otro sistema; es decir, la reutilización está enfocada a mejorar la calidad y reducir el esfuerzo haciendo uso de parte de un sistema en un nuevo contexto (Álvarez G, 2004).

2.8 Reingeniería de procesos vs reingeniería de software

Al comparar el concepto de reingeniería de procesos con el de reingeniería de software, ambos implican mejoras dentro de sus propios conceptos, pero en lo que estos difieren, es en la manera de aplicar ambos conceptos, tal y como lo dice en la reingeniería de software en donde, se parte de algo ya implementado para realizar mejoras, lo cual implica reutilizar partes del software para optimizarlo para reducir tiempos de mantenimiento, esfuerzo de desarrollo.

En la reingeniería de procesos se tiene que analizar, replantear e implementar, desde cómo se está realizando el proceso y en qué aspectos se podría mejorar, seguido de una planeación meticulosa, haciéndolo mucho más rápido, eficiente, y a mucho menor costo.

Es por esto que en la presente tesis se plantea seguir la reingeniería de procesos propuesta por **Michael Hammer y James Champy**, por lo cual no es un proyecto de reingeniería de software al contemplar cada uno de los aspectos que podrían mejorar y aplicando las metodologías correspondientes a la reingeniería de procesos.

2.9 Metodología de la reingeniería de procesos

La reingeniería de procesos es una perspectiva radical al traducirse en cambios dramáticos y mejoras espectaculares. Debe aplicarse en un marco flexible que asegure la transición entre el entorno actual y la situación futura (Champy, 2001).

Por esta razón, se presenta un enfoque metodológico determinado que capte la secuencia de los procesos más habituales. Ello considerando que los procedimientos no son lo suficientemente óptimos para alcanzar un objetivo, persiguiendo la máxima eficacia y la mayor eficiencia.

El rediseño de los procesos está basado en las metas estratégicas. Es la planificación estratégica la referencia obligada de la reingeniería de procesos (gestión, 2016).

El planeamiento estratégico provee un conjunto de metas que han de expresarse en términos de necesidades de los clientes. El plan estratégico debe definir el objetivo principal, el cual consiste en tener una mejora significativa con respecto al proceso anterior. Igualmente, dónde ha de estar situada en el futuro.

A continuación, se describen las fases esenciales de la metodología de la reingeniería de procesos, basado en la metodología de (Champy, 2001).

2.10 Fases de un proyecto de reingeniería de procesos

Según la metodología de la reingeniería de procesos de (Champy, 2001), está compuesto por 4 pasos esenciales, los cuales se muestran en la figura 2.16:

- Definición del proyecto
- Comprensión del Proceso Actual
- Innovación del Proceso
- Implementación del Nuevo Proceso

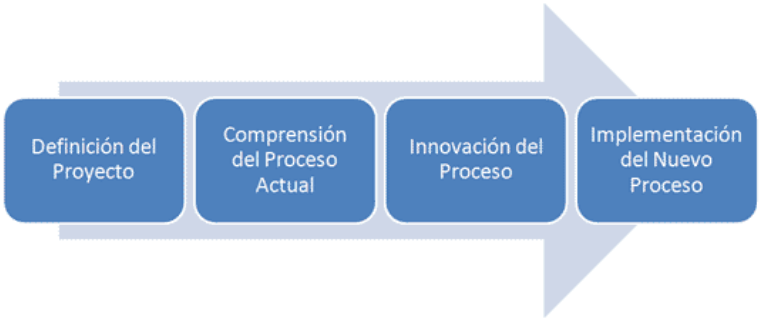


Figura 2. 16 Fases de la reingeniería de procesos

A continuación se describe cada uno de estos pasos:.

1. **Definición del proyecto**. Se sitúa el proyecto de reingeniería con relación a la estrategia de la organización, decidiendo qué hay que cambiar. Es el momento de planificar el proyecto y llevar a cabo tres actividades añadidas.

- Crear un **mapa de procesos**, donde se muestre el flujo de los distintos procesos que operan en el programa y las funcionalidades implicadas. El objetivo es alcanzar una visión de conjunto que permita tomar decisiones sobre qué procesos serán objeto de la reingeniería.
- Seleccionar los procesos objeto de la actuación.
- Iniciar la campaña de comunicación interna. Lo normal es que aparezcan fuertes resistencias desde el principio. Por esta razón es crucial llevar a cabo una campaña de comunicación. Se centraría en mensajes fuerza que superen esas resistencias y dispongan a la organización para los cambios que se producirán.

2. Comprender el estado actual del proceso. Una vez seleccionado el proceso y subprocesos, el equipo de reingeniería comienza a trabajar sobre ellos. Los procesos implicados son examinados para determinar sus objetivos y quiénes intervienen en sus actividades. Los elementos críticos de esta fase, son:

- Definición de los componentes clave del proceso.
- Comprensión de las necesidades del cliente y de sus requerimientos para lograr el resultado del proceso.

- Identificación de debilidades y de posibles **puntos de ruptura** que constituirán oportunidades de mejora radical.
- Establecimiento de objetivos de rendimiento.

Serán varias las tareas a llevar a cabo. Por una parte, se modelará el proceso existente. Es decir, se describirá completamente identificando las distintas actividades y quién las ejecuta. Se trata de describir el proceso ***tal como es***, descomponiéndolo paso a paso.

De otro lado, se modelarán los datos. Esto quiere decir que se describirá exactamente la información y documentación necesarias para llevar a cabo todas y cada una de las actividades comprometidas en el proceso.

3. Innovación del proceso. Se rediseñará el proceso, pasando **del** *tal como es* **al** *tal como debe ser*. En realidad, este trabajo habrá comenzado durante la fase anterior, en la que el hecho de modelar el proceso habrá puesto de manifiesto posibles puntos de ruptura y alternativas de rediseño al quedar al descubierto las causas-raíz de las debilidades del proceso existente.

Una idea básica es organizar en función de los resultados del proceso y de sus salidas, y no sobre sus actividades. En este caso, se estaría reproduciendo el tipo de estructura funcional.

Los elementos clave de esta fase, son:

- Identificar innovaciones potenciales.
- Desarrollar una perspectiva inicial del nuevo proceso.
- Identificar posibles mejoras incrementales.
- Asegurar el compromiso de la dirección con la óptica del nuevo proceso.

4. **Implementación del nuevo proceso**. Una vez implementado el nuevo proceso pueden, pasar varios meses hasta empezar a percibir los resultados. Que el proceso nuevo funcione con toda su potencia puede ser cuestión de más tiempo al requerir un cambio cultural que siempre es lento. Por esta razón, su implantación y desarrollo han de ser objeto de un plan de transición en que se tendrán en cuenta los cambios de normas, sistemas de evaluación y compensación, formación, etc.

Los puntos clave de esta etapa, son:

- Prueba del proceso y evaluación de sus resultados.
- Elaborar el plan de transición.
- Plan de mejora permanente.

Este último punto se relacionaría con la gestión de la mejora continua. Una vez que se ha rediseñado el proceso puede ponerse en marcha un programa de control y mejora de procesos para el ajuste permanente a las necesidades y expectativas de los clientes.

Un método de gran utilidad, y muy frecuente en la reingeniería de procesos, es el ***benchmarking***. Puede ser definido como la investigación de las mejores prácticas. Supone compararse con otras organizaciones que hayan resuelto con éxito los problemas que quiere resolver la propia organización. Para ello se identifican las organizaciones pertinentes y se inicia un proceso de colaboración con ellas.

Capítulo 3

Diseño e implementación

3.1 Definición del proyecto

El sistema descrito en la sección 2.8, al cual se le denominará "sistema anterior", pasa por los siguientes pasos para lograr generar un archivo XML a partir de un archivo CSV.

En la Figura 3.1 se visualizan cada uno de los 5 pasos para lograr la generación de un archivo XML para ser entregado al cliente.

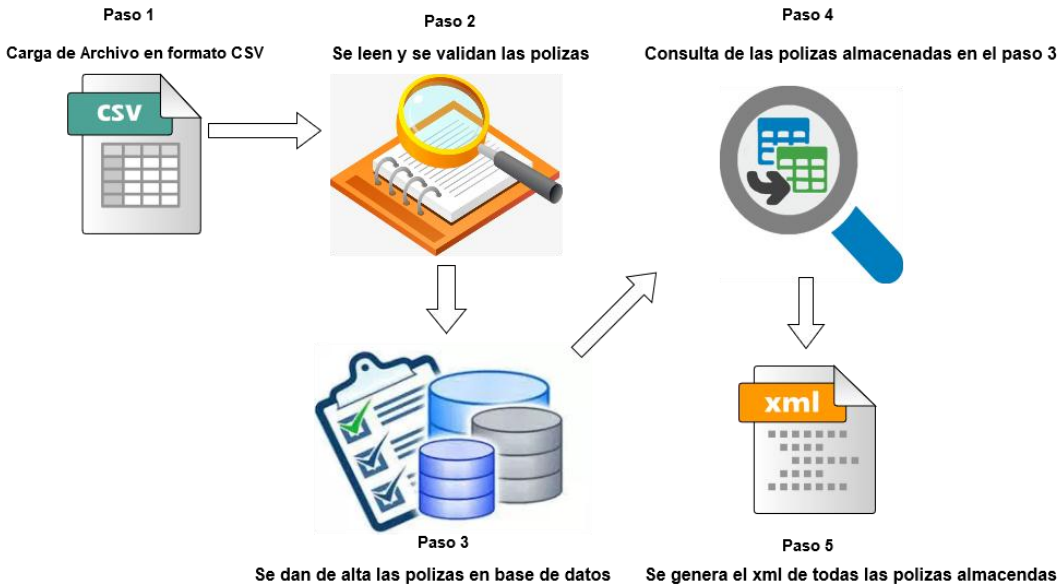


Figura 3. 1 Diagrama generalizado del sistema de generación de pólizas actual

A continuación se describe cada paso del diagrama:

Paso1 Carga de archivo:

Se carga el archivo en formato CSV, este formato está definido dentro de la Tabla 3.1, cada parámetro separado por pipes (|), los elementos obligatorios están marcados con un asterisco.

Tabla 3. 1 Definición de los datos por renglón de cada póliza en el archivo CSV

Conjunto de datos	Parámetros	Conjunto de datos	Parámetros
1.Póliza	tipo de póliza*	4.CHEQUE	No. de cheque
	No. de póliza*		Banco nacional
	Fecha*		Banco extranjero*
	ConceptoPoliza*		Cuenta origen
2.TRANSACCION O PARTIDA	No. de cuenta*		Fecha
	ConceptoTransaccion		Beneficiario
	Debe		RFC
	Haber		Monto
3.CFDI	UUID		Moneda*
	RFC		Tipo de cambio*
	Monto		
	Moneda*		
	Tipo de cambio*		
5.MÉTODO DE PAGO TRANSFERENCIA	Cuenta origen*	6.OTRO METODO DE PAGO	Método de pago
	Banco origen		fecha
	Banco origen extranjero*		Beneficiario
	Cuenta destino		RFC
	Banco destino		Monto
	Banco destino extranjero*		Moneda*
	Fecha		Tipo de cambio*
	Beneficiario		
	RFC		
	Monto		
	Moneda*		
	Tipo de cambio*		

Como se puede observar en la Tabla 3.1 los datos de una póliza capturada manualmente como se describe en la sección 2.1. cambian ya se capturan adicionalmente datos de los comprobantes fiscales digitales (CFDIs) los cuales amparan las transacciones de la póliza, es por esto que se adicionan datos de cuentas bancarias, tipo de moneda en que fue elaborada la transacción, etc. En la figura 3.2 se muestra un ejemplo de una cadena de texto en formato CSV utilizando la Tabla 3.2:

```
I|I706710|02/05/2016|SEGURO DE AUTOMOVILES|26070|I.V.A. POR
DEVENGAR | 214.96| 0.00|c24a6f3e-6bc6-4d72-8949-
4be94014ac9c|XAXX010101000|1558.50|||||||||||||||||||||||99
|02/05/2016|GKSEGUROS|XAXX010101000| 1558.50||
```

Figura 3. 2 Ejemplo de una póliza en formato CSV

Paso 2 Lectura y validación:

El segundo paso comienza con la lectura de cada renglón, separando los parámetros antes mencionados y ubicándolos dentro de un arreglo de datos. Posteriormente se valida cada dato conforme el archivo XSD que proporciona el SAT, que es una guía para la creación correcta de un archivo XML para la generación de pólizas.

Paso3 Registro de los datos en la base de datos:

Una vez que se lee la póliza, se establece la conexión a la base de datos para dar de alta el resultado de la lectura, si pasó el proceso de validación. En el caso contrario, los errores se guardarán en un archivo de texto indicando en que línea y el tipo de error que se capturó. Una vez capturados todos los datos, se notifica que la validación ha terminado y que puede proceder a la generación del archivo XML.

Un ejemplo de error común es que el formato de fecha de alguna de las líneas del archivo CSV no sea válida, como se muestra en la Figura 3.3

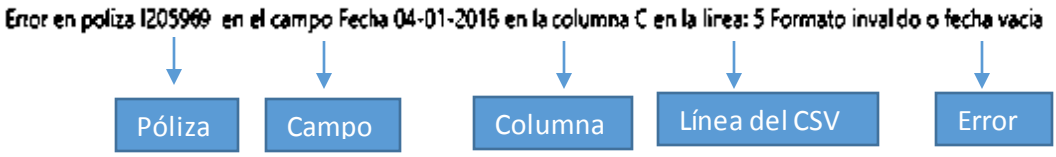


Figura 3. 3 Ejemplo de error en CSV

Paso 4 Lectura de datos:

Este paso es intermedio, el cual se realiza durante la ejecución del código que genera los archivos XML, ya que durante éste se establece la conexión a la base de datos y se realiza la consulta de la póliza a la cual será agregada al archivo XML.

Paso 5 El archivo XML:

Una vez que se obtuvo la información de la consulta que se hizo durante el paso 4, se procede a crear el formato que será agregado al archivo XML, que será entregado como producto final al cliente.

En la figura 3.4 se muestra un ejemplo de un archivo XML con polizas contables que se entrega a la empresa solicitante.

```
<?xml version="1.0" encoding="UTF-8"?>
- <PLZ:Polizas TipoEnvio="AF" Anio="2015" Mes="10" RFC="SASC8310248X9" Version="1.1"
xmlns:PLZ="http://www.sat.gob.mx/esquemas/ContabilidadE/1_1/PolizasPeriodo"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="www.sat.gob.mx/esquemas/ContabilidadE/1_1/PolizasPeriodo
http://www.sat.gob.mx/esquemas/ContabilidadE/1_1/PolizasPeriodo/PolizasPeriodo_1_1.xsd">
  - <PLZ:Poliza Fecha="2015-10-01" Concepto="Poliza de cobro" NumUnIdenPol="1334703_CO">
      <PLZ:Transaccion Concepto="Abono a Banorte / Banorte" Haber="0.00" Debe="186.00"
        DesCta="Banorte / Banorte" NumCta="21001879"/>
      <PLZ:Transaccion Concepto="Cargo a Iva Trasladado No Cobrado" Haber="0.00" Debe="25.66"
        DesCta="Iva Trasladado No Cobrado" NumCta="20901"/>
      <PLZ:Transaccion Concepto="Abono a Iva Trasladado Cobrado" Haber="25.66" Debe="0.00"
        DesCta="Iva Trasladado Cobrado" NumCta="20801"/>
      <PLZ:Transaccion Concepto="Cargo a Cliente Alberto Vazquez Ramos" Haber="186.00" Debe="0.00"
        DesCta="Cliente Alberto Vazquez Ramos" NumCta="105010149600"/>
    </PLZ:Poliza>
  - <PLZ:Poliza Fecha="2015-10-01" Concepto="Poliza de ingreso" NumUnIdenPol="1334703_IN">
      <PLZ:Transaccion Concepto="Abono a Cliente Alberto Vazquez Ramos" Haber="0.00" Debe="186.00"
        DesCta="Cliente Alberto Vazquez Ramos" NumCta="105010149600"/>
      <PLZ:Transaccion Concepto="Abono a Iva Trasladado No Cobrado" Haber="25.66" Debe="0.00"
        DesCta="Iva Trasladado No Cobrado" NumCta="20901"/>
      <PLZ:Transaccion Concepto="Abono a Ventas Y/o Servicios Gravados A La Tasa General De Contado"
        Haber="160.34" Debe="0.00" DesCta="Ventas Y/o Servicios Gravados A La Tasa General De
        Contado" NumCta="40102"/>
    </PLZ:Poliza>
```

Figura 3. 4 Ejemplo de póliza contable en formato XML

3.2 Análisis del estado actual del proceso

Los pasos que se mencionan en la definición del proyecto consta de las siguientes desventajas:

- La validación del archivo CSV tarda debido a que la lectura que realiza, es más lenta al usar métodos de lectura de las librerías estándar del lenguaje Visual Basic.
- Genera una gran cantidad de peticiones de procesamiento, saturando la red y el servidor, el cual almacena las pólizas.
- El uso de catálogos en formato Excel genera un retraso en la lectura de los mismos, a través del lenguaje Visual Basic para leer archivos en este formato.
- En la generación del archivo XML, hace consultas a la base de datos retrasando el tiempo de generación del archivo.

El proyecto debe cumplir las siguientes características una vez rediseñados los procesos:

- Acelerar el proceso de validación y lectura de cada póliza en el archivo CSV utilizando un nuevo procedimiento aplicando nuevas tecnologías disponibles como lo es Pandas de Python y la serialización de Visual Basic.
- Transformar los catálogos a formato JSON para una lectura más rápida y ayudar a la validación
- Eliminar el almacenamiento de cada póliza en un servidor de base de datos, genera mucho menos tiempo de procesamiento al prescindir de conexiones y consultas a la misma.
- Aplicar el nuevo método para la generación de los archivos con las pólizas en formato XML a partir del archivo CSV.
- Modularizar los procesos para facilitar el desarrollo de futuros cambios.

3.3 Innovación del proceso

El proceso descrito en la sección 3.1, se conforma por 5 pasos para la generación del archivo XML en donde se hace énfasis en la lectura y escritura en una base de datos. Ahora considerando el orden que lleva el archivo CSV, al estar organizado se precede de esta base de datos y se toma este archivo de texto. Con lo anterior se ahorraría tiempo y a su vez mejoraría la producción de archivos XML. Con la implementación de estos cambios, en lugar de realizar cinco pasos con el rediseño de los procedimientos, se reduce a tres pasos. A la implementación del rediseño de estos pasos se le denominará como "sistema actual" y

se muestra en la Figura 3.5.

Figura 3.5 Diagrama generalizado del generador de pólizas aplicando la reingeniería de procesos

Paso 1 Carga de archivo: Se carga el archivo CSV el cual ya tiene definidos sus campos, esto facilita la lectura de la cadena generada sin modificar la manera en que el usuario captura los datos con respecto al sistema anterior.

Paso 2 Lectura y validación del archivo: Utilizando nuevas tecnologías para la lectura del archivo, permite revisar la información contenida en el documento. Si dicha validación no cumple con las reglas, entonces se escribe un archivo al terminar la revisión el cual contiene el número de la línea y la columna en donde se registró el error.

Paso 3 Generación del archivo XML: En este paso se simplifica la generación del XML al utilizar un procedimiento llamado serialización, de lo cual se hablará más adelante en la sección 3.4.3.

Una vez validados los datos correctamente se pueden ingresar los datos en la clase que estos, serán serializados y así escribir de una manera más rápida el archivo XML.

3.3.1 Selección de las nuevas tecnologías para el nuevo proceso.

Dentro del punto 3.2 se menciona la utilización de nuevas tecnologías de desarrollo para la validación y lectura de cada póliza, así como en la generación del archivo XML, todo esto sin ser necesario un motor de base de datos, por lo cual se toma la decisión de separar en tres pasos el proyecto, haciendo hincapié en la validación y lectura de las pólizas y en la generación del XML.

Para cumplir con las nuevas definiciones del proyecto se tienen que elegir nuevas tecnologías de desarrollo. Al separar el proyecto en tres partes principales, hace que el proyecto sea mucho más fácil de modificar en un futuro.

La lectura del archivo CSV y la validación de cada póliza requiere que se genere un script, el cual esta implementado en el lenguaje de programación Python. Las librerías de Python que se usan para este paso hacen que el proceso sea más sencillo y rápido.

En cuanto a la creación del archivo XML se pretende simplificar utilizando el lenguaje de programación Visual Basic, y utilizar las librerías de serialización a partir de un esquema XSD para mejorar notablemente el tiempo de generación del archivo.

A continuación, se mencionan las librerías, para cubrir las necesidades del nuevo proyecto.

Pandas para Python.

Pandas es un módulo de análisis de datos. Proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fácil de usar (Moya, 2015).

Contiene herramientas de análisis de datos para el lenguaje de programación Python, facilita la lectura del archivo en formato CSV generando un diccionario para acceder a sus valores y validarlos más fácilmente.

Newtonsoft para Visual Basic

Es una librería para realizar lectura y escritura de archivos en formato JSON. Una de las ventajas de hacer uso de esta librería es ordenar los catálogos definidos por el usuario y los del SAT sin necesidad de que estos estén en formato Excel (Newtonsoft, 2019).

Serializer para Visual Basic

Es una librería que permite crear clases a partir de un archivo XSD para su correcta serialización. Genera un archivo en formato XML de manera mucho más rápida debido al manejo de los objetos creados por Visual Studio (Microsoft, 2019).

3.3.2 Diseño de la interfaz de usuario

La interfaz del usuario del sistema anterior, la cual está desarrollada en Windows Forms (Formulario de Windows), dicha interfaz la cual presenta los siguientes campos del sistema

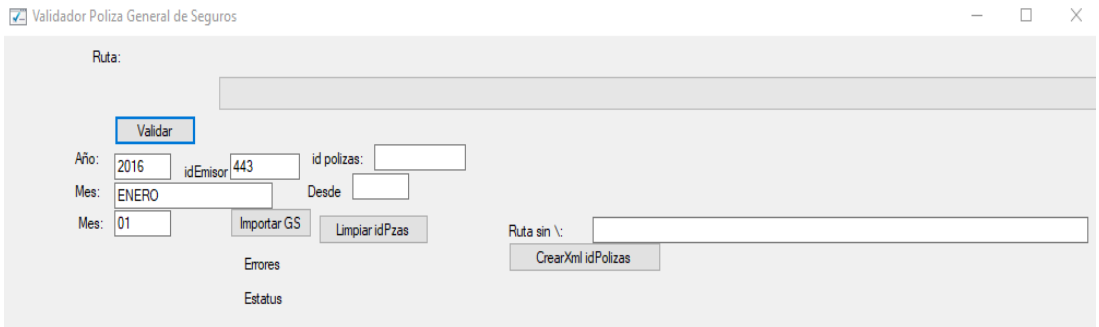


Figura 3. 6 Interfaz de usuario del sistema anterior

anterior se muestran en la Figura 3.4:

En la Figura 3.6, los campos no se encuentran ordenados ni tienen un diseño intuitivo, lo cual puede causar confusión para futuros usuarios. Se optó por rediseñar la interfaz para que fuera más amigable en donde se cambia el formulario común de Windows y se utiliza un tipo de formulario moderno llamado WPF(Windows Presentation Foundation), con la capacidad de agregar detalles más estéticos e intuitivos, mejorando el uso del nuevo sistema.

Tabla 3. 2 Controles removidos y agregados en la nueva interfaz de usuario

Controles removidos	**Controles agregados**
idPolizas	Barra de menús (agregado)
Desde	Subir archivo (agregado)
Limpiar idPolizas	Control de fecha (agregado)
idEmisor	Selector de RFC
Validar	Botón ruta de almacenamiento
Mes	Barra de progreso
Año	Indicadores de validación
Mes	
Errores	

En la Tabla 3.2, quitando y actualizando los controles y pasando al nuevo formato de formulario WPF, se logra el objetivo de facilitar la experiencia de usuario y haciendo que el diseño del formulario sea más intuitivo como se visualiza en la Figura 3.7

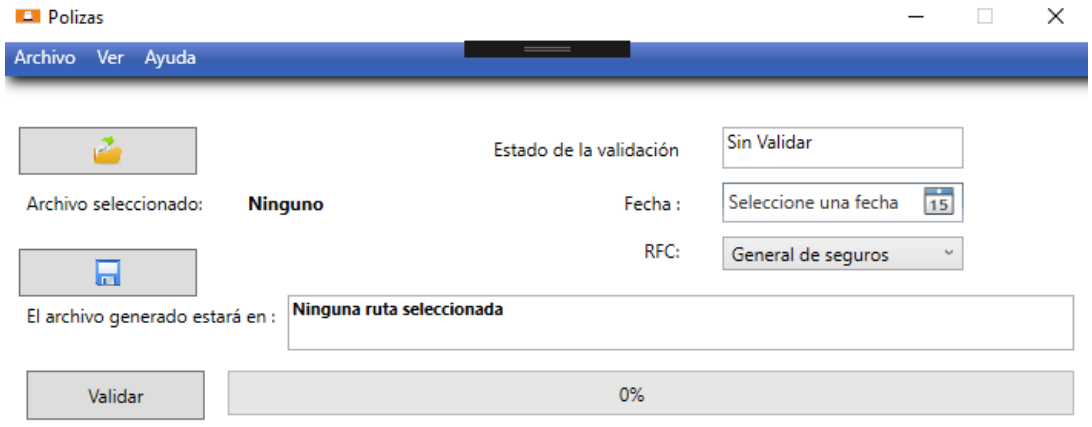


Figura 3. 7 Nueva interfaz de usuario rediseñada

Interfaz para visualización de errores en el archivo XML

Para la visualización de errores en la aplicación anterior, se busca la ubicación de donde está el archivo de errores, por lo que se rediseñara la interfaz. En caso de que la validación no haya sido exitosa, se mostrará el menú "Ver" en la barra superior. En la Figura 3.8, se muestra la nueva interfaz de visualización de errores, la cual se compone por la barra superior y por un campo de texto, en donde se indica la especificación de los errores.

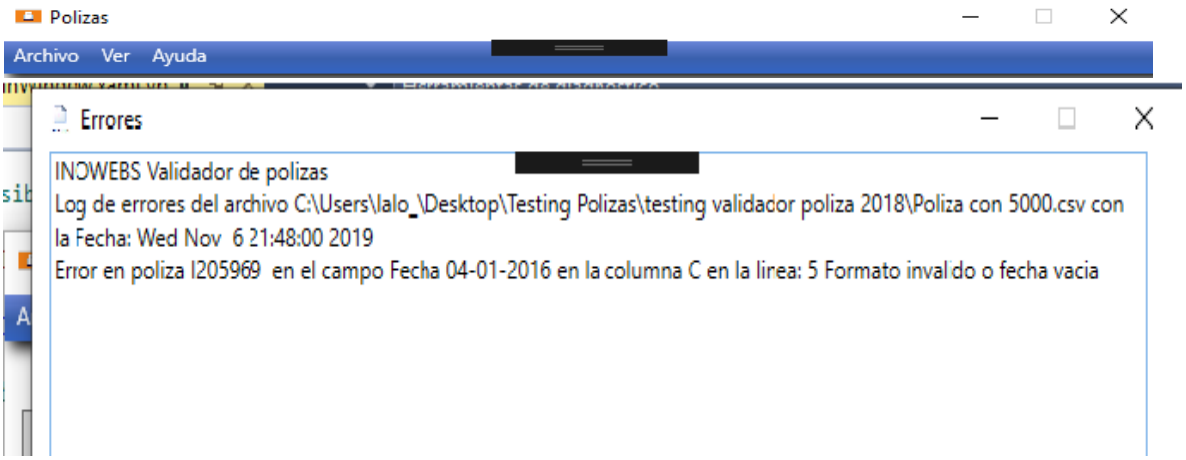


Figura 3. 8 Interfaz de visualización de errores

Interfaz de ayuda para el usuario y configuración previa

En caso de errores o dudas de cómo utilizar el sistema se añade una interfaz para que el usuario pueda acceder al archivo de ayuda, así como de posibles soluciones para que el

Figura 3. 9 Interfaz de ayuda al usuario

programa funcione correctamente tal y como se muestra en la Figura 3.9

3.4 Implementación

3.4.1 Estructura del proyecto

El proyecto al ser modular se dividirá en 2 partes principales, la primera parte utiliza un script desarrollado en el lenguaje de programación Python, ya que es muy efectivo para el manejo de una gran cantidad de datos.

Una vez desarrollado el script validador, el programa desarrollado en Visual Basic se encargará de manejar la llamada al script de Python, así como la serialización de la clase creada a partir del esquema XSD proporcionado por el SAT creando el archivo XML finalizado.

3.4.2 Lectura y validación de pólizas

Como se mencionó en la sección 3.3, se precede de un motor de base de datos y se utilizará el mismo archivo de texto en formato CSV lo cual ayudará a optimizar el proceso ya que al no depender de dicho motor de base de datos el proceso de lectura se hará de una manera mucho más rápida.

La manera en que el script de lectura y validación funciona se describe a continuación:

Una vez que se carga el archivo de texto en formato CSV, se presiona el botón de validar. Éste pasa a través de un script desarrollado en Python el cual esta descrito por el diagrama de flujo del proceso de la Figura 3.10, haciendo uso de la librería PANDAS.

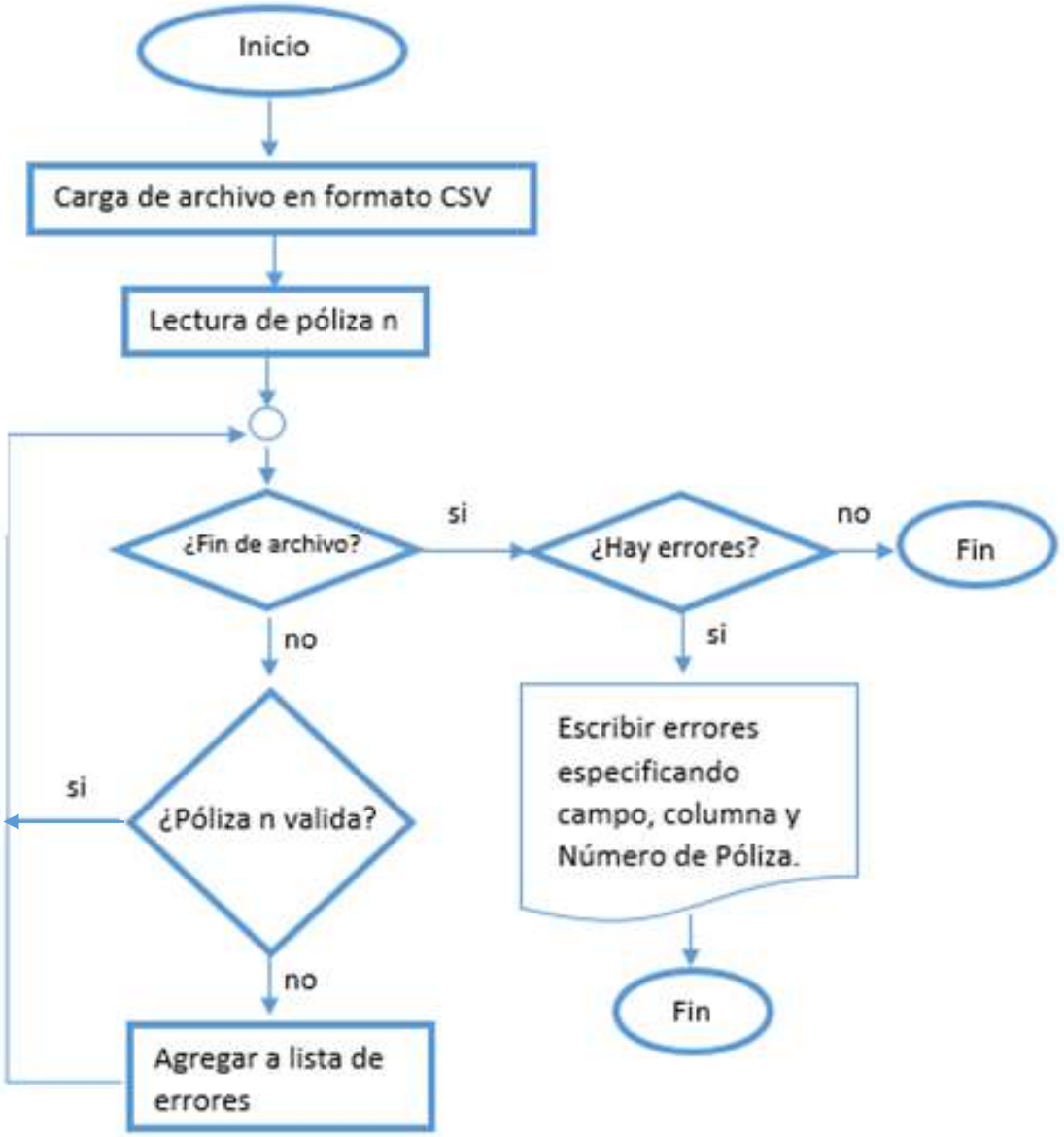


Figura 3. 10 Diagrama de flujo del script de lectura y validación

3.4.3 Serialización del archivo validado en Visual Basic

Para lograr generar el archivo XML, a partir de los datos contenidos en el archivo CSV es necesario utilizar una clase creada con una herramienta contenida en el IDE de Visual Studio.

La herramienta que se utiliza para crear una clase a partir de un esquema XSD se llama "Símbolo de sistema para desarrolladores".

En la imagen siguiente se muestra un ejemplo de cómo crear la clase a partir de un archivo XSD con el comando que lleva su nombre (**xsd**).

El comando toma los elementos y atributos del esquema para crear una clase de Visual Basic, adicionalmente se le especifican los siguientes parámetros que son:

- l: Especifica en que lenguaje de programación se creará la clase, puede ser en los lenguajes Visual Basic, C# y JavaScipt.
- c: Indica que el archivo de salida será una clase.
- El tercer parámetro es el nombre del archivo en formato XSD

En la Figura 3.11, se muestra un ejemplo utilizando el símbolo de sistema para desarrolladores ejecutando el comando xsd para generar una clase a partir de un esquema XSD.

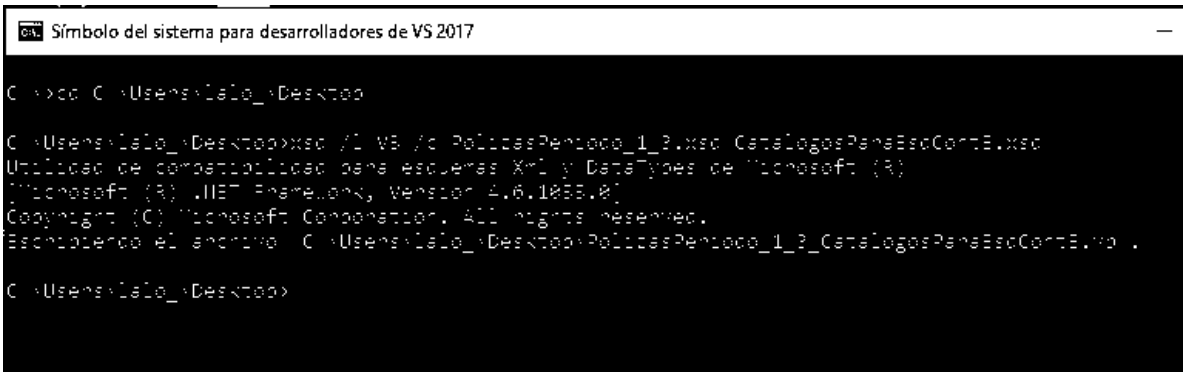


Figura 3. 11 Creación de una clase a partir de un archivo XSD

En el programa una vez que se han validado los datos, estos se comienzan a llenar a través de programación dentro de la clase generada.

Una vez capturados los datos, se serializan con la librería **serializer**, esto consiste en convertir en bytes, los datos proporcionados por la clase, para finalmente generar un archivo XML.

En la figura 3.12 se puede observar un diagrama generalizado del funcionamiento descrito anteriormente.

1. Creación de la clase a través del esquema en formato XSD con la herramienta "Símbolo de sistema para desarrolladores".
2. Llenado de los datos validados, desde el archivo CSV hacia la clase generada.
3. Serialización de los datos obtenidos desde la clase, creando un archivo XML.

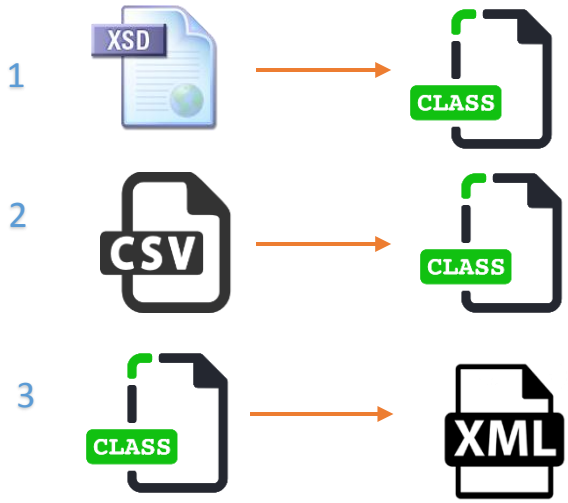


Figura 3. 12 Proceso de serialización para generar un archivo XML

Capítulo 4

Pruebas y resultados

Las pruebas que se llevan a cabo para comparar ambos sistemas son las siguientes:

Usando el sistema Anterior:

1. Prueba de rendimiento de memoria RAM, al insertar pólizas en la base de datos.
2. Prueba de rendimiento de memoria RAM, al generar el archivo en formato XML.
3. Tiempo de validación del archivo en formato CSV.
4. Tiempo de generación de archivo en formato XML.

Usando el sistema actual:

1. Prueba de rendimiento de memoria RAM, al validar las pólizas.
2. Prueba de rendimiento de memoria RAM, al generar el archivo en formato XML.
3. Tiempo de validación del archivo en formato CSV.
4. Tiempo de generación de archivo en formato XML.

Se usaron 20,000 pólizas, del periodo de enero del 2019, como muestra para realizar las pruebas de rendimiento de memoria RAM. Se hizo uso de la herramienta “Uso de CPU” de Visual Studio, para obtener datos de rendimiento. Una vez realizadas las pruebas, se hacen comparaciones para obtener los resultados de los cuales se obtendrán las conclusiones del presente trabajo.

4.1 Ambiente de pruebas

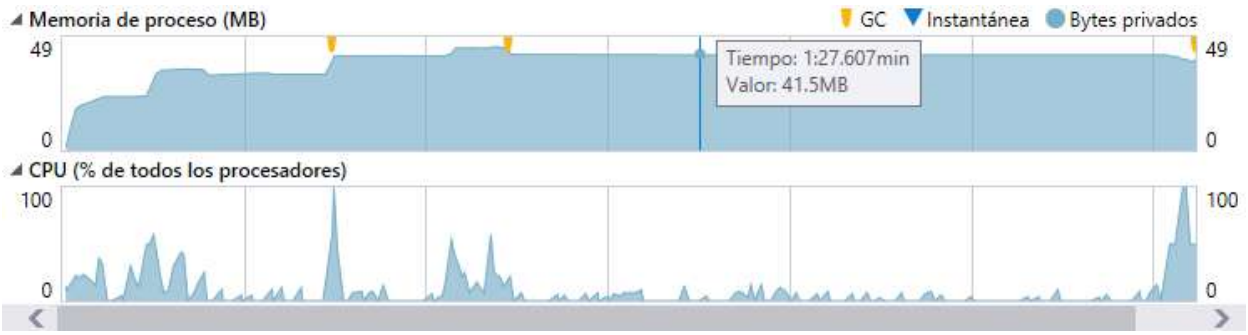
Las pruebas de ambos sistemas se realizaron en un equipo con las siguientes características, descritas en la Tabla 4.1:

Tabla 4. 1 Especificaciones técnicas del sistema

Especificaciones técnicas del sistema	
Sistema operativo	Windows 10 pro 64bits
Modelo	Asus K42f
Memoria RAM	8Gb
Procesador	Intel(R) Pentium(R) CPU P6100 2GHz 2 Núcleos

4.2 Prueba de rendimiento de memoria RAM en el sistema anterior

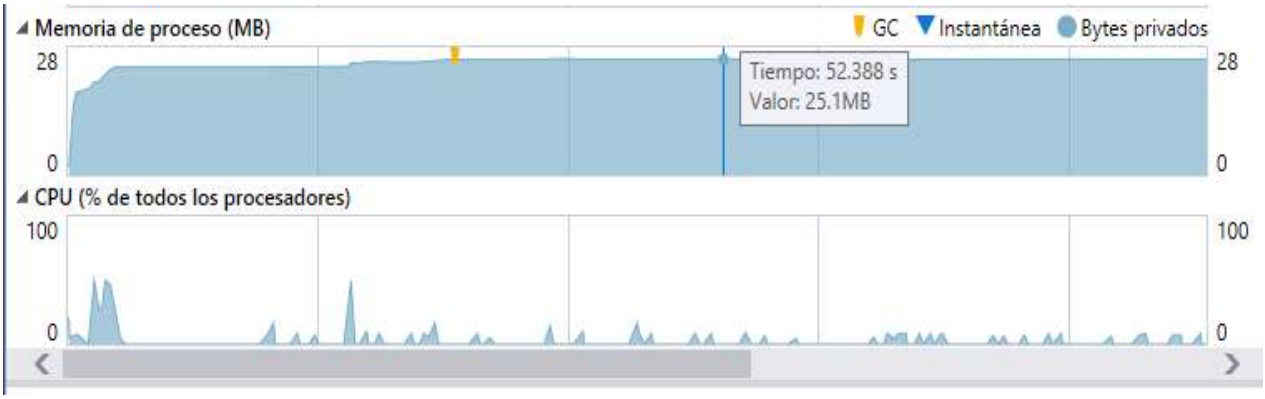
En la figura 4.1, se presentan los resultados obtenidos de las mediciones de rendimiento de



la memoria RAM durante el proceso de inserción de las 20,000 pólizas en la base de datos.

Figura 4. 1 Consumo de memoria RAM durante la inserción de pólizas en el sistema anterior

En la figura 4.2 se muestra el consumo de memoria RAM registrado al generar el archivo



en formato XML de las 20,000 pólizas de la base de datos, usando el sistema anterior.

Figura 4. 2 Consumo de memoria RAM durante la creación del archivo XML en el sistema anterior

4.3 Prueba de rendimiento de memoria RAM en el sistema actual

En la figura 4.3, se presentan los resultados obtenidos de las mediciones de rendimiento de la memoria RAM durante el proceso de generación del archivo XML.

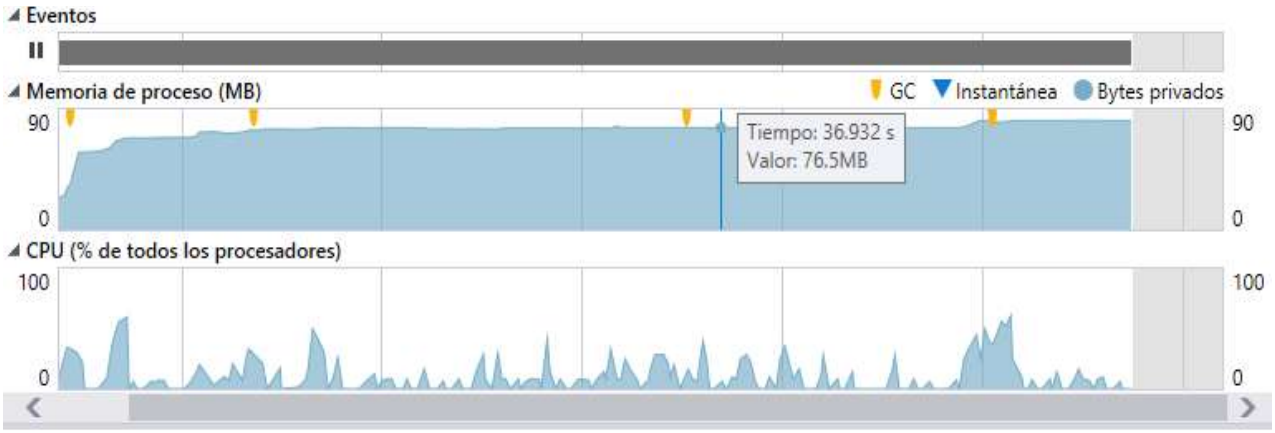


Figura 4. 3 Consumo de memoria RAM durante la generación del archivo XML en el sistema actual.

4.4 Registro de tiempos de validación de archivo CSV y generación de archivo XML en el sistema anterior

En la Tabla 4.2 se muestran la cantidad de pólizas utilizadas, tiempo de validación y el tiempo que tarda en generar el archivo XML en el sistema anterior:

Tabla 4. 2 Registro de tiempos de validación CSV y generación de XML usando el sistema anterior

Cantidad de pólizas	Tiempo de validación CSV	Tiempo de generación XML
100,000	0:28 segundos	14 horas 09min
50,000	0:34 segundos	6 horas 1 min
30,000	0:24 segundos	3 horas 43 min
20,000	0:16 segundos	2 horas 07 min

10,000	0:14 segundos	1 horas 15 min
5,000	0:13 segundos	0 horas 41 min

En la Figura 4.4 Se observa que, el sistema anterior crece de manera exponencial conforme el número de pólizas aumenta, y por lo tanto el tiempo en que tarda en generar un archivo XML es demasiado.

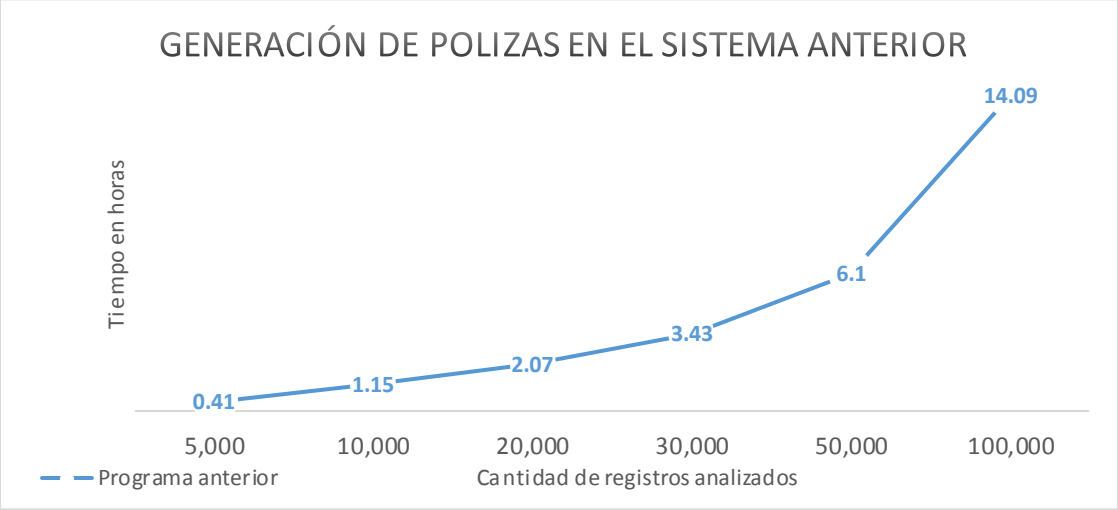


Figura 4. 4 Registros de tiempo de generación del archivo XML en el sistema anterior

4.5 Registro de tiempos de validación y generación de archivo XML en el sistema actual

En la Tabla 4.3 se muestran los tiempos de validación y creación dentro del sistema actual. Se incrementa la cantidad de pólizas para probar la capacidad de procesamiento de la implementación de los nuevos algoritmos, obteniendo como resultado los siguientes datos:

Tabla 4. 3 Registro de tiempo de creación y validación del sistema actual

Cantidad de pólizas	Tiempo de validación	Tiempo de generación XML
500,000	3:41 minutos	5:51 minutos
300,000	2:37 minutos	3:10 minutos
100,000	0:33 segundos	0 min 55 segundos
50,000	0:17 segundos	0 min 28 segundos
30,000	0:11 segundos	0 min 15 segundos
20,000	0:08 segundos	0 min 12 segundos

10,000	0:05 segundos	0 min 08 segundos
5,000	0:03 segundos	0 min 05 segundos

Al ver los resultados de la figura 4.5, se logra observar que de la misma manera que en el programa anterior el tiempo de generación de las pólizas crece de manera exponencial, lo cual indica que, conforme se aumenta la cantidad de pólizas el tiempo, se ve notablemente reducido ya que se reduce el tiempo de generación, de horas a minutos.

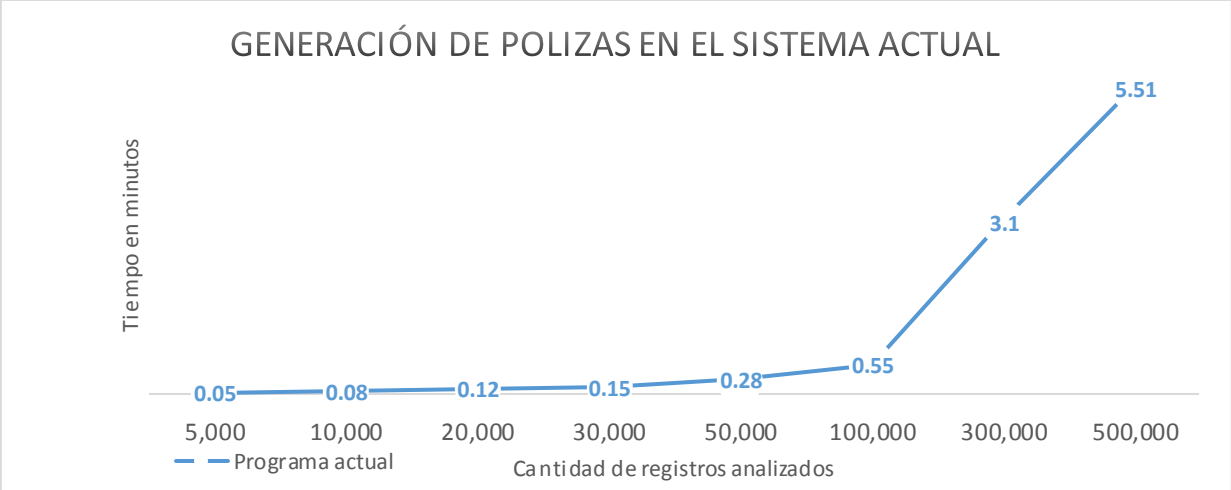


Figura 4. 5 Registro de tiempo de creación del archivo XML en el sistema actual

4.6 Comparación entre el sistema anterior vs el sistema actual

A continuación, en la figura 4.6 se muestra la comparativa del tiempo de generación de XML entre el sistema anterior y el nuevo sistema. Se ve reducido el tiempo de generación de estos archivos en gran escala, además de que se visualiza que ambos sistemas crecen de manera exponencial conforme la cantidad de pólizas aumentan.

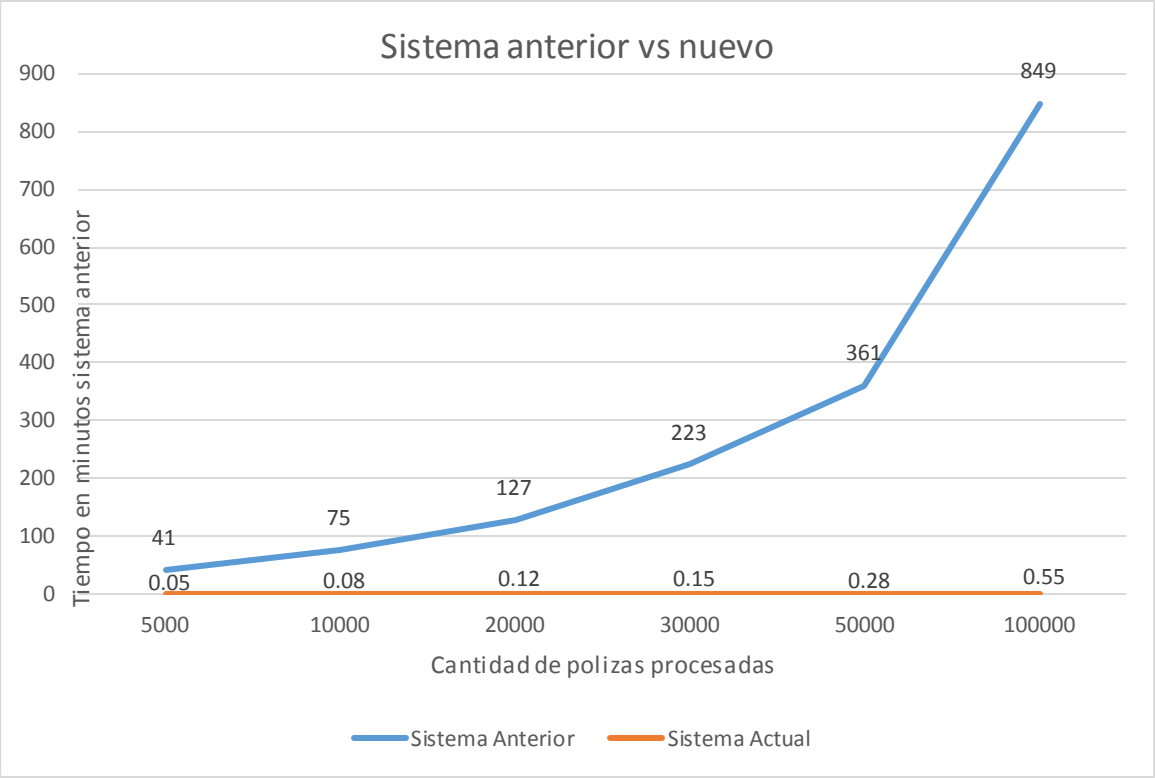


Figura 4. 6 Tiempo de generación de XML del sistema anterior vs el sistema actual

Capítulo 5

Conclusiones

Al comparar el funcionamiento de ambos sistemas en sus respectivas faces de generación, se pueden dividir en 2 fases principales para determinar el resultado de la reingeniería de procesos aplicada al sistema anterior estas son:

El tiempo de validación del archivo CSV:

Al hacer el análisis de ambos sistemas durante el proceso de validación se puede visualizar que ambos validadores manejan aproximadamente los mismos tiempos, siendo el nuevo sistema ligeramente más rápido, este ligero retraso de validación, sucede debido a la ejecución del módulo elaborado en Python, con la compatibilidad con Visual Basic. Lo cual nos dice que la validación dentro del rediseño del proceso, con respecto al sistema anterior cumple con su propósito.

El tiempo de generación de archivos XML:

Durante la generación del XML se puede concluir que al quitar el almacenamiento en una base de datos y dejar de lado su uso y al utilizar la serialización a través de la librería **Serializer** de Visual Basic se determina, en base a los resultados obtenidos durante las pruebas entre el sistema actual y el sistema anterior, el tiempo de ejecución del nuevo sistema con respecto al tiempo de generación de archivos XML, se ve reducido desde 800 a 1000 veces más.

De manera generalizada al aplicar la reingeniería de procesos en el sistema actual tal y como se esperaba, al analizar y replantear cada una de sus fases, permitió rediseñar el funcionamiento del sistema anterior mejorando notablemente en los siguientes aspectos:

- Experiencia de usuario, al integrar la nueva interfaz, más intuitiva y amigable.
- Tiempo de generación y validación, reducido notablemente en la generación de pólizas en XML.
- Mejor manejo de errores, al notificar específicamente la posición del mismo acelerando las correcciones.
- Actualizaciones sencillas de implementar, por la modularización del nuevo sistema.

La implementación reingeniería de procesos para la empresa INOWEBS fue de gran utilidad ya que se comprobó que los tiempos se vieron reducidos hasta en un 50% para la validación de archivos CSV y hasta en un 1000% para la generación de archivos XML. Con estas mejoras la empresa puede atender a más clientes, hacienda uso de menos recursos en una menor cantidad de tiempo, reflejándose en mayores ingresos para la empresa.

Bibliografía

Alvarez, M. A. (19 de Noviembre de 2003). *Qué es Python*. Obtenido de https://desarrolloweb.com/articulos/1325.php

Amoedo, D. (2017). *Unblog*. Obtenido de https://ubunlog.com/spyder-entorno-desarrollo-python/

Biggerstaff. (1990). Software Reusability. Addison-Wesley. November,.

C., M. (12 de 10 de 2005). *"Reingeniería". Revista Enlaces de Recursos Humanos*. Obtenido de http://www.losrecursoshumanos.com/reingenieria.htm

Champy, M. H. (2001). *Reengieneering the corporation.* New York: Harper collings.

Contalinea, E. (26 de 05 de 2017). *http://blog.contalinea.mx/contabilidad-electronica-version-13*.

Davenport. (1990). The new industrial enginnering.

Europe, X. (2003). *XML*. Obtenido de http://www.hipertexto.info/documentos/xml.htm

Files, C. F.-S. (October de 2005). Obtenido de https://tools.ietf.org/html/rfc4180

gestión, A. c. (14 de 11 de 2016). *aiteco.com*. Obtenido de https://www.aiteco.com/reingenieria-de-procesos/

Gonzalo. (07 de Enero de 2010). *Educación y nuevas tecnologías*. Obtenido de https://blogs.ua.es/gonzalo/2010/01/07/partes-de-un-documento-xml/

Gualix, J. (03 de marzo de 2017). *Visual Studio 2017, el IDE perfecto para crear apps en Windows*. Obtenido de https://www.microsoftinsider.es/121003/asi-visual-studio-2017-ide-perfecto-crear-apps-windows-ios-android-la-nube/

json.org. (s.f.). *Introducción a JSON*. Obtenido de https://www.json.org/json-es.html

Juan Carlos Álvarez García, M. M. (15 de Julio de 2004). METODOLOGÍA DE REINGENIERÍA DEL SOFTWARE PARA LA REMODELACIÓN DE APLICACIONES CIENTÍFICAS HEREDADAS. Obtenido de https://pdfs.semanticscholar.org/bf41/16b03e46b28a0903d64f74b298f945a29543.pdf

León . (octubre de 2014). *Contabilidad electrónica - Colegio de Contadores Públicos de México*. Obtenido de https://www.ccpm.org.mx/avisos/PAF%20601%202da%20octubre%2014_2.pdf

Microsoft . (19 de 07 de 2015). *Microsoft docs*. Obtenido de https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/concepts/serialization/

Microsoft. (2019). *Serializer .NET*. Obtenido de https://docs.microsoft.com/en-us/dotnet/api/system.xml.serialization.xmlserializer?view=netframework-4.8

Morales, R. (15 de Mayo de 2014). *ticarte*. Obtenido de http://www.ticarte.com/contenido/que-son-los-esquemas-xsd

Moya, R. (30 de 10 de 2015). *Pandas en Python*. Obtenido de Pandas en Python

Newtonsoft. (2019). *Json.Net*. Obtenido de https://www.newtonsoft.com/json

pymes, G. (23 de 06 de 2018). *GrandesPymes*. Obtenido de https://www.grandespymes.com.ar/2018/06/23/reingenieria-de-procesos-concepto-y-metodologia/

SAT. (s.f.). *Contabilidad electronica* . Obtenido de http://omawww.sat.gob.mx/contabilidadelectronica/Paginas/01-1.htm

Silberschatz, A. (2006). *Fundamentos de bases de datos.* McGRAW-HILL.

Sum, P. E. (s.f.). *Programo Ergo Sum*. Obtenido de https://www.programoergosum.com/cursos-online/raspberry-pi/244-iniciacion-a-python-en-raspberry-pi/que-es-python