INFORME DE PRÁCTICA PROFESIONAL

CC4901-1

PAPERLESS S.A.

ALFONSO JAVIER CORNEJO CASTILLO

Nº MATRÍCULA 28002721

INGENIERÍA EN CIVIL EN COMPUTACIÓN

ACORNEJO@ING.UCHILE.CL

OBSERVACIONES

SUMARIO

RESUMEN

Este informe trata del sistema desarrollado para la empresa Paperless S.A. durante los meses de diciembre de 2010 y enero y febrero de 2011 cuya función es la de generar de manera fácil definiciones para transformar archivos de texto plano con información tributaria en Documentos Tributarios Electrónicos (DTE) con el formato XML definido por el Servicio de Impuestos Internos (SII) con el fin de realizar declaraciones tributarias.

Paperless S.A. es una empresa que ofrece servicios informáticos a un grupo variado de clientes. Uno de los servicios más solicitados es el de las facturas y boletas electrónicas o DTEs. El servicio consiste en que los clientes se comprometen a enviar documentos tributarios en un formato determinado a los servidores de Paperless donde estos son procesados, validados y posteriormente enviados al Servicio de Impuestos Internos en el formato aceptado por este.

La problemática que se intentó solucionar en este proyecto de práctica fue la de evitar que el área de desarrollo de la empresa tuviese que participar del proceso de creación del sistema que procesa los documentos tributarios en los servidores de Paperless y que en cambio ese sistema se hiciera para cada nuevo cliente mediante una aplicación con interfaz gráfica que manejarían solamente los jefes de proyecto sin la necesidad de la intervención del área de desarrollo.

Para esto se desarrolló una aplicación web que permitía a un usuario hacer un mapeo entre el formato de los archivos de envío que se acuerdan con el cliente y los campos y el esquema definido por el SII para las declaraciones tributarias. Una vez hecho el mapeo, este es guardado en disco para ser ocupado por el sistema en los servidores para hacer la transformación y validación de los documentos tributarios enviados por el cliente.

El proyecto se desarrolló correctamente y la funcionalidad descrita en los párrafos anteriores se logró completar. Se creó una aplicación robusta que procesaba documentos tributarios de forma eficiente por una parte, y una interfaz gráfica lo suficientemente intuitiva y completa para hacer el mapeo y generar el sistema de procesamiento. Sin embargo, por motivos de tiempo, no se lograron atar pequeños cabos sueltos de la aplicación web que concluían el proceso de guardar en disco el archivo con las definiciones del mapeo por lo que al momento de conclusión de la práctica el sistema no quedó terminado completamente.

INTRODUCCIÓN

LUGAR DE TRABAJO

Paperless S.A. es una empresa Chilena que presta servicios y ofrece productos informáticos a un grupo variado de clientes. Se dedica principalmente a la generación, procesamiento y administración de documentos electrónicos. Entre sus clientes se encuentran desde grandes empresas de retail que funcionan en el país como París, Ripley y Wal-Mart, empresas de logística y control e incluso ha prestado servicios al Gobierno de Chile conforme se han emitido decretos que demandan el desarrollo de sistemas electrónicos. Algunos de sus clientes según sus áreas se pueden ver en la ilustración 1. La lista completa de clientes se puede encontrar en su sitio web http://www.paperless.cl.

Ilustración 1. Algunos de los clientes de Paperless según sus áreas.

Entre algunos de los servicios y productos propios de la empresa que ofrecen se encuentran las facturas y boletas electrónicas, el almacenamiento de documentos electrónicos, contratos electrónicos y correo electrónico certificado. El mayor objetivo de la empresa es ofrecer a sus clientes la facilidad de administrar sus empresas utilizando solamente documentos electrónicos y evitando así administrar archivos de papel impreso. Es además una de las empresas autorizadas por el SII para emitir Documentos Tributarios Electrónicos por cumplir con las normas adecuadamente.

Recientemente la empresa se ha expandido hacia otros países, logrando firmar acuerdos de trabajo con Petrobras en Brasil entre otras compañías de ese país y por otra parte participar de un programa piloto del Gobierno del Perú para que las empresas comiencen a emitir Documentos Tributarios Electrónicos.

En Chile la compañía tiene sus oficinas en Santiago, en la comuna de Las Condes en el Edificio del Pacífico en la dirección Av. Andrés Bello 2687 en el piso 25, además cuenta con una oficina adicional en un edificio en la misma área que es donde funciona el área de Operaciones de la empresa. Por su parte la compañía también tiene sus centros de trabajo correspondientes en Brasil y Perú.

Grupo de Trabajo

La compañía consiste en total de un grupo de aproximadamente 100 personas, considerando todas sus áreas. El grupo de personas que trabaja en Santiago es de alrededor de 60 personas. Los grupos de trabajo se gobiernan por áreas, existiendo las de Desarrollo, Operaciones, Administración y finanzas y Comercial. La empresa tiene una política de mantener un buen ambiente de trabajo entre los empleados y una comunicación fluida entre todas sus áreas.

En el grupo de trabajo en el que se desarrolló esta práctica y que es el área responsable y autora del proyecto fue el área de Desarrollo. El área estaba compuesto de un grupo de 10 personas y su responsable es el Gerente de TI el señor Aliosha Bertini Salas quién fue el supervisor de este proyecto de práctica junto con el Arquitecto Sénior Cristián Rosas y la Ingeniero de Desarrollo Sénior Melissa Pino. El proyecto fue llevado a cabo por los desarrolladores en práctica Juan F. Sanhueza, estudiante de Ingeniería Civil en Computación de la Universidad de Chile y el autor del presente informe.

La empresa proveyó a ambos desarrolladores en práctica un escritorio con un computador para trabajar en el proyecto ubicado en el área de desarrollo lo que facilitó una fluida comunicación entre los miembros del proyecto y permitió el rápido avance en él.

Equipo y Software

El equipo dispuesto para desarrollar consistió de un escritorio y un computador con el sistema operativo Ubuntu, mientras que las tecnologías, piezas de software y librerías utilizadas para el funcionamiento de la aplicación fueron las siguientes:

- IDE Eclipse: Entorno de desarrollo integrado para trabajar con diversos lenguajes de programación. Se configuró para entregar soporte para Java EE utilizando Apache Tomcat como servidor contenedor de las aplicaciones web.
- Java SE: Plataforma de desarrollo de software portable con compatibilidad para múltiples sistemas operativos. Cómo lenguaje de programación resultó cómodo para desarrollar la funcionalidad pedida, en particular por la facilidad con la que permite tratar el manejo de objetos, arreglos, archivos, memoria, errores y archivos XML. Se debió trabajar procurando compatibilidad con la versión 1.4.
- Apache Tomcat 5: Aplicación que hace de servidor y contenedor de aplicaciones web según el estándar definido por Java EE. Se utilizó como servidor predeterminado para

ejecutar la aplicación web con la interfaz de usuario del proyecto. Una de sus ventajas en comparación con otros servidores de aplicaciones es su eficiencia y bajo consumo de recursos.

- HTML, CSS y JavaScript: A pesar de que el estándar Java EE ofrece una amplia gama de utilidades para desarrollar aplicaciones web robustas y complejas, para lograr los objetivos de este proyecto se debió entrar a trabajar no solo en el diseño HTML de las páginas sino también en las hojas de estilo CSS y programar varias funciones JavaScript que utilizando la librería jQuery proveyeron una interfaz intuitiva y eficiente para el sistema.
- PostgreSQL 8: Base de datos relacional de código abierto, una de las más conocidas y utilizadas en el mercado. Se utilizó en el proyecto para la aplicación web.
- Subversion (SVN): Sistema de control de versiones. Se utilizó para llevar un ordenado control del código del proyecto y sincronizar que el código del proyecto encontrado en el equipo de cada uno de los miembros del proyecto se encontrara actualizado.
- Apache Ant: Aplicación utilitaria que permite la fácil compilación y publicación de código y aplicaciones web utilizando definiciones hechas en un archivo XML con ciertas reglas. En este proyecto se utilizó para compilar y generar librerías de los distintos paquetes que se desarrollaron.
- Apache POI: Librería que permite la lectura y escritura de archivos con el formato de Microsoft Office como “doc” y “xls”. Se utilizó para interpretar un archivo de reglas de definiciones manejado por los jefes de proyecto de Paperless en formato “xls”.
- Apache Tiles: Librería utilizada en la aplicación web que diagrama las páginas separando por secciones cada parte de la página de forma modular, de modo que la edición de la página se realiza por sección lo que simplifica el proceso de editar una página al hacerlo modularmente.
- Lenguaje de esquemas XSD recomendado por el W3C, XML Schema. Debió estudiarse este lenguaje de definición XML para comprender el formato válido definido por el SII para el envío de Documentos Tributarios Electrónicos. Además se utilizó para el mismo proceso de validación de los DTE generados.
- Variadas librerías utilitarias de Paperless para el manejo de archivos, acceso a base de datos y manejo de usuarios. Mencionamos estas librerías en esta sección con el objetivo de indicar todas las piezas de software que no correspondiesen a las desarrolladas exclusivamente en y para el proyecto de práctica, sin embargo la funcionalidad que ofrecen es menor comparado con el resto del sistema y por ende no las describiremos específicamente.

Veremos que la mayoría de estas librerías facilitaron algunos procesos al punto que de no haberlas tenido a disposición no se hubiese podido completar el proyecto ni construir el sistema que se hizo.

SITUACIÓN PREVIA

La metodología de Paperless para implementar el servicio de boletas y facturas electrónicas para un nuevo cliente consistía hasta el momento de realizar la práctica en el siguiente proceso:

1. El cliente con un Jefe de Proyecto (JP) de Paperless acuerdan un formato en el que el cliente enviará a Paperless archivos con información tributaria. A pesar de que para cada cliente el formato puede definirse de una forma particular, este formato muy comúnmente se acordará que será de un formato de texto plano en el que los campos son delimitados por un caracter especial. Describiremos este formato en detalle más adelante.
2. El JP recurre al área de Desarrollo de Paperless para crear un programa que haga la transformación y validación de los archivos enviados por el cliente a archivos XML con el formato aceptado por el SII. Estos programas son llamados Filtros.
3. El filtro es cargado en los servidores de Paperless para recibir los archivos enviados por el cliente y enviarlos al SII.

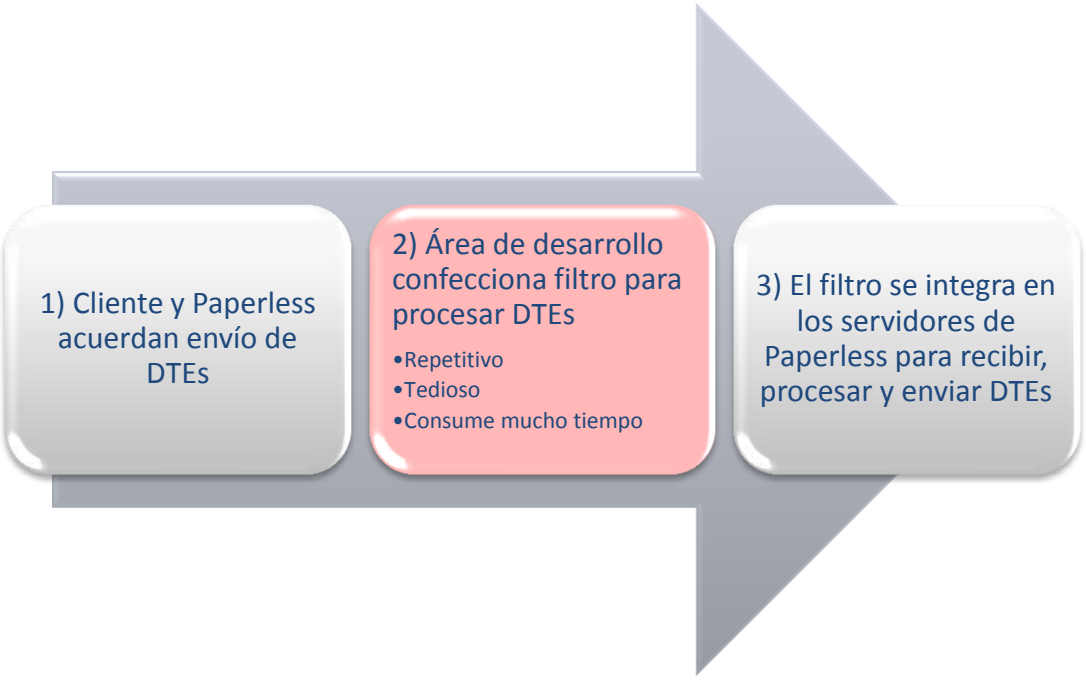


Ilustración 2. Proceso de integración de un cliente al servicio de Boletas y Facturas Electrónicas.

Es precisamente el punto 2 el proceso que toma más tiempo y produce más ineficiencia en la compañía, ya que el área de Desarrollo necesita los espacios para trabajar en otros productos y sistemas que Paperless quiere completar y debido a que el formato de los archivos enviados por los clientes suele repetirse con la mayoría de los clientes, surge la clara visión de que el punto 2 se podría realizar evitando la participación del área de Desarrollo y en cambio desarrollar una herramienta simple de ocupar que permita a los JP generar los Filtros por si mismos liberando a los desarrolladores para trabajar en otros proyectos.

Esta fue la situación en la que se encontraba la empresa al momento de realizar la práctica. Hubo anteriormente sin embargo intentos de desarrollar herramientas que hicieran la funcionalidad descrita en el párrafo previo, pero no fueron completados exitosamente por lo que la empresa continuó usando esta metodología de trabajo.

RESUMEN DEL TRABAJO REALIZADO

El objetivo principal de este proyecto de práctica consistió en eliminar el paso 2 de la metodología de integración del servicio de boletas y facturas electrónicas descrito en la sección anterior, es decir, evitar la participación del área de desarrollo en la integración de un nuevo cliente a este servicio.

Para esto se desarrolló por partes un sistema pensado para los JP para que por su propia cuenta y considerando lo acordado con el cliente y el formato definido por el SII pudiesen desarrollar el filtro que procesará los documentos enviados por el cliente.

Las partes del proyecto final consistieron en dos componentes:

- F-Box

  El nombre viene de Filtro-Box. Esta componente es el motor central del sistema; su funcionalidad consiste en procesar los archivos de texto enviados por los clientes y generar Documentos Tributarios Electronicos en formato XML y Archivos de reportes de errores en caso de ser necesarios. Para hacer el procesamiento de los archivos de texto, el motor necesita leer un archivo que contiene las especificaciones para mapear los campos del archivo de texto con los del archivo XML, dicho archivo de especificaciones es llamado Archivo de Definiciones de Reglas y es generado mediante la componente GEFA del sistema.

- GEFA

  Corresponde a la aplicación web (Interfaz gráfica) diseñada para los JP para crear el Filtro a ser integrado en los servidores de Paperless para procesar DTEs. Consiste en una aplicación de fácil manipulación que permite la creación y edición de Archivos de Definiciones de Reglas para el uso de la componente F-Box. Esta aplicación hace uso de una Base de Datos y sistema de archivos para crear y almacenar en disco los Archivos de Definiciones de Reglas trabajados y permitir su posterior edición.

Las dos componentes interactúan constantemente en el proceso de creación de un filtro.

Para crear un Filtro, el JP debe entrar usando un explorador de internet a la aplicación web de GEFA y una vez identificado dentro del sistema de usuarios de la aplicación seguir el siguiente proceso.

1. Entrar a la sección de la aplicación web de creación o edición de filtros.
2. Seleccionar el tipo de documento tributario al que corresponde el filtro, los cuales son definidos por el SII y que en este proyecto de práctica son soportados solamente los tipos de documentos DTE y Libro, siendo DTE el documento tributario correspondiente a las boletas o facturas y Libro un documento tributario correspondiente a las compras-ventas realizadas en un mes.
3. Entrar a la interfaz de edición de Filtro y agregar y editar reglas de mapeo entre los campos existentes en los documentos tributarios enviados por el cliente y el XML resultante que será enviado al SII. Esta interfaz es la pieza fundamental de la aplicación web y su funcionalidad será descrita en detalle en las secciones siguientes.
4. Generar un Archivo de Definiciones de Reglas que queda guardado en disco y que es el archivo utilizado por el motor F-Box para procesar los DTEs. Posteriormente este archivo de definiciones puede ser editado utilizando la interfaz de edición de Filtro.

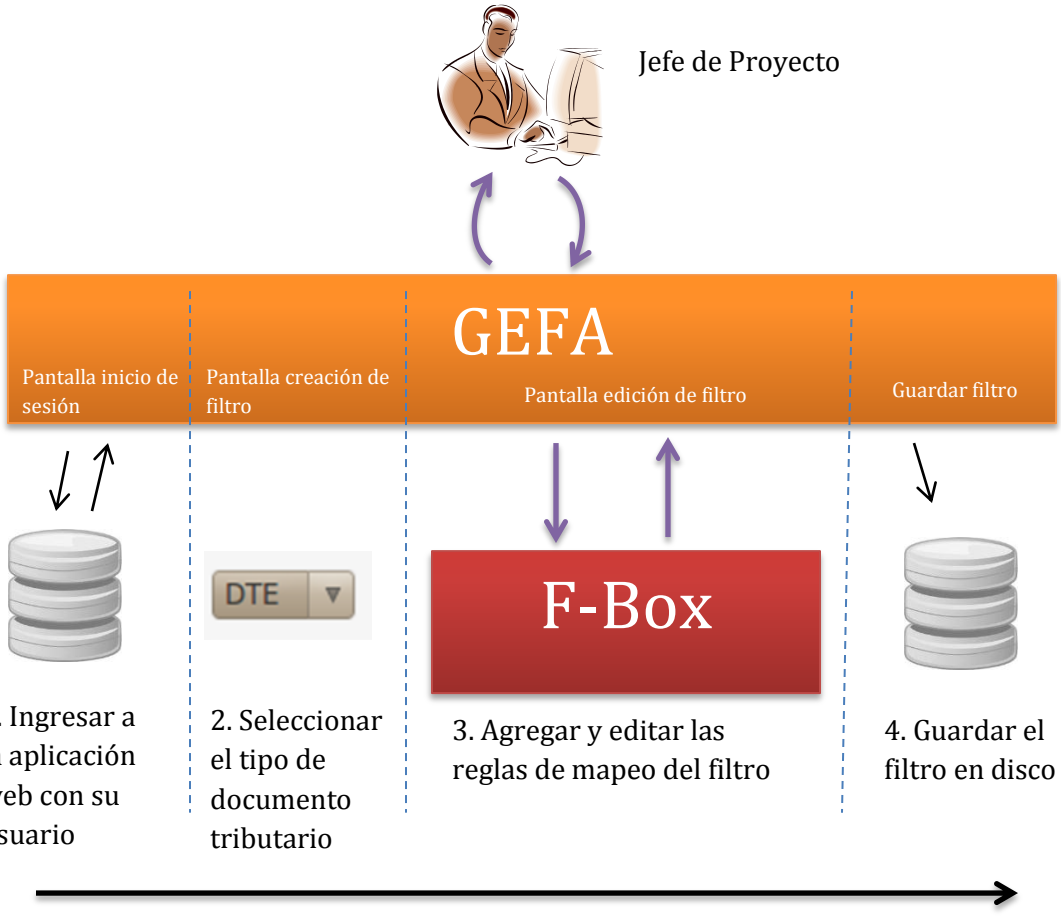


Ilustración 3. Diagrama del proceso de creación de un filtro

De este modo el filtro queda terminado y se procede al paso 3 del proceso de integración en el que se habilita un servidor de Paperless para recibir los archivos enviados por el cliente utilizando el Archivo de Definiciones de Reglas guardado y la componente F-Box para procesarlos.

Veremos que la componente F-Box por su parte cumplirá con la funcionalidad necesaria para procesar y validar DTEs, mientras que la componente GEFA no se completó a tiempo lo que finalmente llevó a la conclusión de que el proyecto no cumplió con su propósito.

El practicante desarrollador Juan F. Sanhueza fue el encargado de la construcción de la componente GEFA mientras que al autor de este informe fue el encargado de la construcción de la componente F-Box. Al ser la componente GEFA en estricto rigor parte del proyecto en el que se desarrolló esta práctica pero no parte de la práctica misma, no entraremos a definir su funcionalidad sino que nos quedaremos con la definición general de su utilidad.

TRABAJO REALIZADO

Cada componente desarrollada fue construida como un proyecto eclipse distinto con el mismo nombre de la componente. La componente F-Box corresponde a un proyecto Java simple mientras que la componente GEFA corresponde a un proyecto de Web Dinámica basada en Java EE.

A pesar de que no fueron mencionados en el resumen del trabajo realizado, también se desarrollaron proyectos para probar algunas de las componentes, cuyo propósito fue el de comprobar tanto funcionalidad de las aplicaciones cómo eficiencia. Dichos proyectos de prueba no serán descritos en este informe por la simplicidad de los mismos pero si explicaremos algunas de las pruebas que se realizaron y sus resultados.

F-BOX

DESCRIPCIÓN DE FUNCIONALIDAD GENERAL

Esta es la componente principal del sistema. Es el motor que procesa los documentos tributarios y genera los archivos XML.

En esta componente se procesan dos tipos de archivos: Los Archivos de Definiciones de Reglas y los documentos tributarios.

Los Archivos de Definiciones de reglas son unos archivos especiales con un formato definido internamente en Paperless que definen cómo se deben procesar los documentos tributarios y son generados mediante la componente GEFA.

Los documentos tributarios son los archivos que son enviados por los clientes, que se asume que pueden contener errores o que pueden venir mal formados pero que el formato en el que se espera que se encuentren sea el definido por el Archivo de Definiciones de Reglas. Los documentos tributarios pueden ser de tipo DTE o Libro.

DEFINICIONES PREVIAS

Antes de pasar a explicar el funcionamiento del motor, debemos abordar la definición de Documentos Tributarios Electrónicos según el SII y cómo son especificados mediante el lenguaje de definición de esquema XSD.

El lenguaje de definición de esquemas XSD es un lenguaje recomendado por la World Wide Web Consortium (W3C) cómo una estandarización para la definición de aplicaciones XML. Entendemos como aplicación XML a cualquier lenguaje para documentos que utiliza XML, de modo que por ejemplo el estándar XHTML es una aplicación XML ya que utiliza la sintaxis y cumple con las condiciones definidas para un lenguaje XML mientras que el estándar HTML no es una aplicación XML ya que no cumple con algunas restricciones necesarias para el estándar XML. El lenguaje de definición de esquemas XSD no es más que una aplicación XML utilizada para definir aplicaciones XML.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="orderperson" type="xs:string"/>
      <xs:element name="shipto">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="item" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="xs:positiveInteger"/>
            <xs:element name="price" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Ilustración 4. Ejemplo de definición de esquema XSD.

En la ilustración 4 vemos un ejemplo de una definición de esquema bastante simple de comprender, que nos define una cierta aplicación XML cuyos documentos tienen una estructura tal que:

- El primer elemento del documento y que encierra a todos los demás se llama “shiporder”.
- Dentro de este elemento se encontrarán en orden los siguientes elementos:
    - “orderperson” de tipo string.
    - “shipto” de tipo complejo.
    - Uno o más de los elementos “item” de tipo complejo.
    - Y además un atributo que es obligatorio y cuyo nombre es “orderid” de tipo string.

- Además cada uno de los elementos de tipo complejo tienen sus propios elementos hijos que también están definidos.

De esta forma, un XML de ejemplo que corresponde al definido por el XSD de la ilustración 4 se muestra en la ilustración 5.

```
<shiporder orderid="order1">
        <orderperson>Alfonso</orderperson>
        <shipto>
                <name>Alfonso</name>
                <address>Italia 01</address>
                <city>Santiago</city>
                <country>Chile</country>
        </shipto>
        <item>
                <title>Poster</title>
                <quantity>1</quantity>
                <price>9.99</price>
        </item>
</shiporder>
```

Ilustración 5. Ejemplo de un archivo XML que cumple con el esquema definido en la ilustración 4.

El lenguaje de definición de esquemas XSD es la especificación utilizada por el SII para definir el formato de los archivos DTE que se envían, luego los documentos tributarios que procesemos deben convertirse en archivos XML con este formato y es esta la función de la componente F-Box del proyecto. Las definiciones de esquema para los documentos tributarios DTE y Libro desarrolladas por el SII se pueden encontrar en su sitio web en http://www.sii.cl/factura_electronica/formato_xml.htm.

No revisaremos la definición de los elementos en estas definiciones de esquema ya que tienen que ver con conceptos tributarios que escapan a la intención de este informe de práctica, sin embargo, debemos tener en mente en todo momento que son estos documentos los que definen el producto final del proceso que debemos realizar en la componente F-Box.

QUÉ SE HACE

Como mencionamos anteriormente, la función de esta componente es el de procesar documentos tributarios transformándolos en XML. Un documento tributario es un archivo de texto plano que es enviado por el cliente a los servidores de Paperless, y su formato es acordado por el cliente y el JP previamente.

```
FH|61|20101111_NC1|2010-11-11|||||1||78921690|Centennial Cayman Corp Chile
FHIR||||
DH|1||001.001.0001|EQ, MOTOROLA I205|99.90009|UND|1111|99.90009|UND|1111|||||…
DHCD|||||||
SR|||||||
RE|1|55|610|2010-11-11|3||
PE||||||||||2312||6842||
```

Ilustración 6. Ejemplo de documento tributario en texto plano.

En la ilustración 6 podemos ver un documento tributario de ejemplo, este es un DTE y su formato es el siguiente:

- Cada línea corresponde a una sección de información que debe ser enviada al SII, estas secciones son nodos particulares que encontramos en la definición XSD de los XMLs tributarios que en general tienen poca altura con respecto al nodo raíz y contienen una gran cantidad de elementos hijos. Veremos ejemplos de secciones al ver un ejemplo de XML resultante en los párrafos siguientes.
- Las primeras letras de cada línea hasta antes del primer delimitador "|" indican a qué sección corresponde la línea. Existe un mapeo en el Archivo de Definiciones de Reglas que identifica la sección en el documento tributario con la sección a la que corresponde en el XML.
- A partir del primer delimitador "|" en adelante vienen en orden, hasta el final de la línea, datos que corresponden a un elemento específico del XML resultante y se sabe gracias al Archivo de Definiciones de Reglas qué dato corresponde a qué elemento del XML.

La componente F-Box utiliza un Archivo de Definiciones de Reglas para interpretar un cierto documento tributario de texto plano y utilizando sus definiciones procesa, mapea y genera internamente una estructura de árbol en memoria con los datos y elementos del XML que debe ser generado para finalmente utilizando este árbol generar el archivo XML.

```
TIPO|DTE
SECCION|Encabezado|FH
SECCION|Detalle|DH
SECCION|CdgItem|DHCD
⋮
SECCION|Personalizados|PE

FH|1|TipoDTE|1||NUMERO|2|
FH|2|Folio|1||NUMERO|10|
⋮
FH|35|MntTotal|1||NUMERO|18|FN1

DH|1|NroLinDet|1||NUMERO|4|
DH|2|IndExe|0||NUMERO|1|
⋮
DH|12|MontoItem|1||NUMERO|18|
```

Ilustración 7. Ejemplo de Archivo de Definiciones de Reglas.

En la ilustración 7 podemos ver un ejemplo de un Archivo de Definiciones de Reglas. Este archivo recibe este nombre porque está compuesto de Reglas de mapeo, que son indicaciones para la componente F-Box y su formato es el siguiente:

- La primera indica el tipo de documento tributario que se procesará, de modo que esta línea puede tener los valores "TIPO|DTE" o "TIPO|Libro" solamente.
- Desde la línea segunda hasta la primera línea vacía encontramos mapeos de secciones. Estos son indicadores que identifican las primeras letras de una línea del documento tributario con la sección a la que corresponde en el XML resultante.
- En adelante, para cada sección, separadas por líneas en blanco, encontramos un número finito de líneas que corresponden a las reglas de mapeo. Esto es, cada línea en este conjunto de líneas de una sección indica sobre el documento tributario:
    - La posición de un dato dentro de la línea de la sección.
    - El elemento XML al que corresponde ese dato.
    - Si el dato puede venir vacío o no.
    - El valor por defecto del dato.
    - El tipo del dato (Número, texto, fecha, decimal, etc.).
    - El largo máximo que puede tener el dato (como palabra).
    - El formato del dato si es que viniese en un formato especial.

Así tenemos que por ejemplo para el Archivo de Definiciones de Reglas de la ilustración 7, en un documento tributario que procesemos según estas reglas de mapeo nos debiésemos encontrar con cosas como que:

- La línea que venga con las letras "FH" corresponde a la sección "Encabezado" del XML.
- La línea de la sección encabezado debiese venir con 35 datos.
- El último dato de la sección encabezado corresponde al elemento "MntTotal" de la sección y debe cumplir con lo siguiente:
    - El dato no puede venir vacío (Lo indica el número 1 a la derecha del tercer delimitador "|").
    - El dato no tiene valor por defecto (Lo indica el espacio vacío a la derecha del cuarto delimitador "|").
    - El tipo del dato corresponde a un número (Lo indica la palabra "NUMERO").
    - El largo máximo del dato puede ser 18 (Lo indica el número 18).
    - El dato viene en un formato especial llamado "FN1".

Utilizando Archivos de Definiciones de Reglas cómo el descrito en el ejemplo, la componente F-Box procesa documentos tributarios y genera documentos XML con el formato pedido por el SII.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DTE version="1.0" xmlns="http://www.sii.cl/SiiDte">
   <Documento ID="DOCT39F1390">
      <Encabezado>
         <TipoDTE>39</TipoDTE>
         <Folio>1390</Folio>
         <FchEmis>2010-11-18</FchEmis>
         <MntTotal>2502</MntTotal>
      </Encabezado>
      <Detalle>
         <NroLinDet>1</NroLinDet>
         <PrcItem>6230</PrcItem>
         <MontoItem>6230</MontoItem>
         <CdgItem>
            <TpoCodigo>1</TpoCodigo>
            <VlrCodigo>24531</VlrCodigo>
         </CdgItem>
      </Detalle>
   </Documento>
</DTE>
```

Ilustración 8. Ejemplo de archivo XML DTE.

En la ilustración 8 podemos ver un ejemplo de un archivo XML con el formato definido por el SII. Para los efectos de este informe vamos a asumir que un archivo como el de este ejemplo es un archivo válido según el esquema definido por el SII para el envío de documentos tributarios, sin embargo, en estricto rigor este archivo se encuentra incompleto según la definición XSD ya que le falta entre otras cosas una firma digital y un timestamp que son necesarios según un cierto protocolo de seguridad del SII. Estos elementos del documento XML son extensos por lo que de agregarlos a la ilustración la dejaría poco legible. Además, la firma digital y el timestamp no son impresas en el documento por la componente F-Box sino que son agregadas por un proceso posterior implementado previamente por Paperless el cual no se utilizó durante el desarrollo de esta práctica.

En esta ilustración podemos identificar las secciones del documento XML las cuales en este caso serían "Encabezado" y "Detalle" que cómo indicamos anteriormente corresponden a elementos XML que contienen a un gran número de elementos dentro y en general se encuentran a una distancia pequeña de la raíz del documento (En una altura pequeña en el árbol). Además también se definen lo que identificamos como subsecciones que corresponden a conjuntos de datos que vienen en una línea en los documentos tributarios de texto plano pero que en el XML van ubicados dentro de una sección. En este ejemplo tenemos la presencia de la subsección "CdgItem" la cual si utilizamos las reglas de mapeo que vimos en el Archivo de Definiciones de Reglas de la ilustración 7 vendría en el documento tributario en una línea con el identificador "DHCD".

Toda la funcionalidad que hemos descrito hasta ahora es útil sólo en casos ideales, en los que el Archivo de Definiciones de Reglas contiene exactamente las reglas necesarias para procesar documentos tributarios y cuando los XML resultantes cumplen con el formato definido en el XSD por el SII. Pero no debemos asumir que esto será siempre de este modo, de hecho es muy probable que hayan errores en los datos contenidos en un documento tributario. Es por esto

que se implementó un sistema de generación de reporte de errores, que genera archivos con información sobre todos los errores que se produzcan al momento de procesar un documento tributario.

```
Gravedad: Minima -> El campo sobrepasa el largo maximo permitido (8)
                    En linea 1: FH|39|1390|2010-11-18||||3|||898072002|(…)
                    En campo con tag FchEmis en la posicion 3 con valor: "2010-11-18"

Gravedad: Grave -> El campo es obligatorio y se encuentra ausente en el archivo
                   En linea 1: FH|39|1390|2010-11-18||||3|||898072002|FARMAC(…)
                   En campo con tag Acteco en la posicion 13 con valor: ""

Gravedad: Nula -> Se encontraron mas campos que los esperados
                  En linea 2: DH|1||DORMILAN COM. 5MG.20|1|6230||||||6230|
```

Ilustración 9. Ejemplo de un reporte de errores.

En la ilustración 9 podemos ver un ejemplo de un reporte de errores. Se generaron dos modalidades de reportes de errores; los reportes extensos y los reportes de errores cortos, el del ejemplo corresponde a un reporte de errores extenso y lo que hace es indicar qué cosas se esperaban al procesar el documento tributario y qué se encontró que no corresponde. Además cada error encontrado cuenta con una gravedad definida y el reporte de errores se genera solamente si los errores encontrados tienen gravedad por lo menos mínima.

La idea general de esta componente es la de inicializar una instancia de procesador de documentos tributarios a la cual una única vez se le asigna un Archivo de Definiciones de Reglas con el que la componente se dispondrá a procesar constantemente documentos tributarios en el servidor. En caso de que se produzcan errores al procesar un documento, el motor generará un reporte de errores para ese documento y no un XML y en caso de que el documento sea procesado sin errores se generará el XML correspondiente para ser enviado al SII.

CÓMO SE HACE

Primero explicaremos el modelo utilizado para formar los XML resultantes de procesar documentos tributarios y luego explicaremos el proceso detallado que se ejecuta.

MODELO

Se modeló la estructura del árbol XML utilizando la convención de representación Document Object Model (DOM) recomendada por la W3C para documentos XML De modo que diseñamos clases Java con estructura de datos de nodos de árboles de la forma más abstracta posible con utilidad para el proyecto.

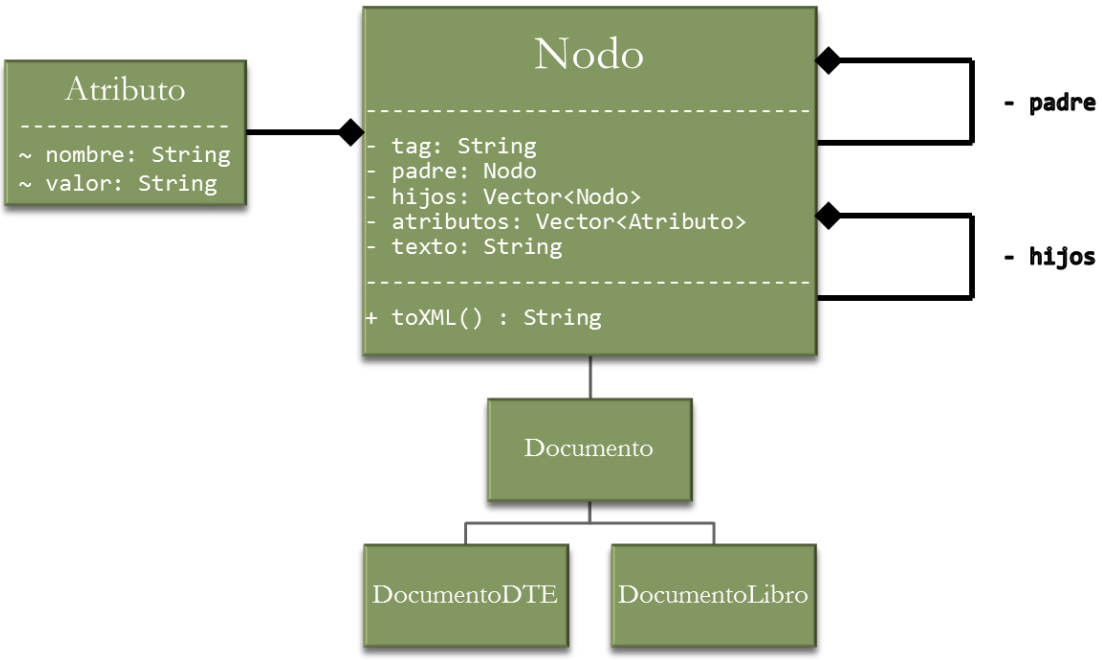


Ilustración 10. Diagrama de clases del modelo de datos

En la ilustración 10 podemos ver que la base de nuestro modelo es la clase Nodo, la cual contiene una referencia a su Nodo padre, una lista a sus Nodos hijos, una lista de Atributos (que se refiere a los atributos XML del elemento y no a atributos del objeto Java), un atributo con su tag (Nombre de elemento XML) y un atributo texto. En estricto rigor, la W3C recomienda modelar el texto como un Nodo hijo del Nodo que lo contiene, pero para efectos prácticos preferimos incluirlo como un atributo del Nodo. Utilizando este modelo de datos, realizar la convención a archivo XML teniendo el árbol construido correctamente resulta un problema trivial y el método toXML() es el encargado de esto.

Además creamos las clases Documento, DocumentoDTE y DocumentoLibro que son, de acuerdo con el modelo DOM, los Nodos raíces del documento final. Estas clases contienen ciertos métodos específicos que son utilizados en un paso casi al final del parseo de documentos tributarios y que tienen que ver con recuperación de campos específicos que Paperless necesita para la identificación del documento.

Esta estructura de datos es utilizada en el proceso de procesamiento de documentos tributarios para estructurar en memoria el documento XML que resulta de procesar el documento tributario con los datos correspondientes.

Parseo de Archivo de Definiciones de Reglas

En esta componente se asume la existencia de los Archivos de Definiciones de Reglas, la definición de estos, su formato y la procedencia viene especificada desde antes, la componente F-Box solamente se debe preocupar de procesarlos y utilizar las reglas de mapeo que se encuentren en ellos para procesar documentos tributarios.

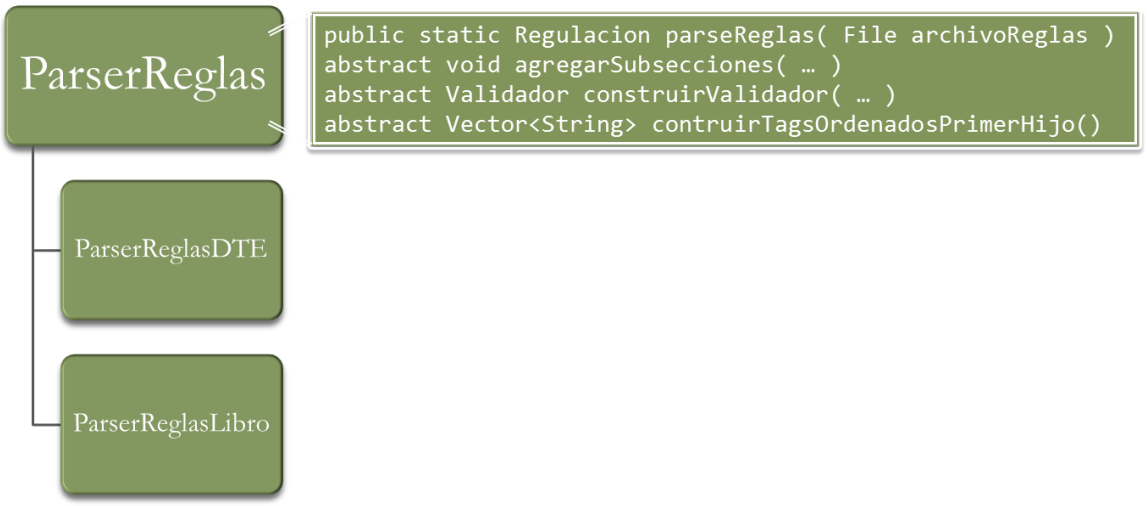


Ilustración 11. Diagrama de clases procesadoras de Archivos de Definiciones de Reglas.

La clase que se encarga de procesar los Archivos de Definiciones de Reglas es la clase abstracta ParserReglas, que posee un método estático con firma:

- public static Regulacion parseReglas(File archivoReglas)

este método recibe una referencia a un Archivo de Definiciones de Reglas y retorna un objeto llamado Regulacion. Un objeto de clase Regulacion es un objeto que contiene todas las reglas y definiciones de mapeo y validación de un documento tributario.

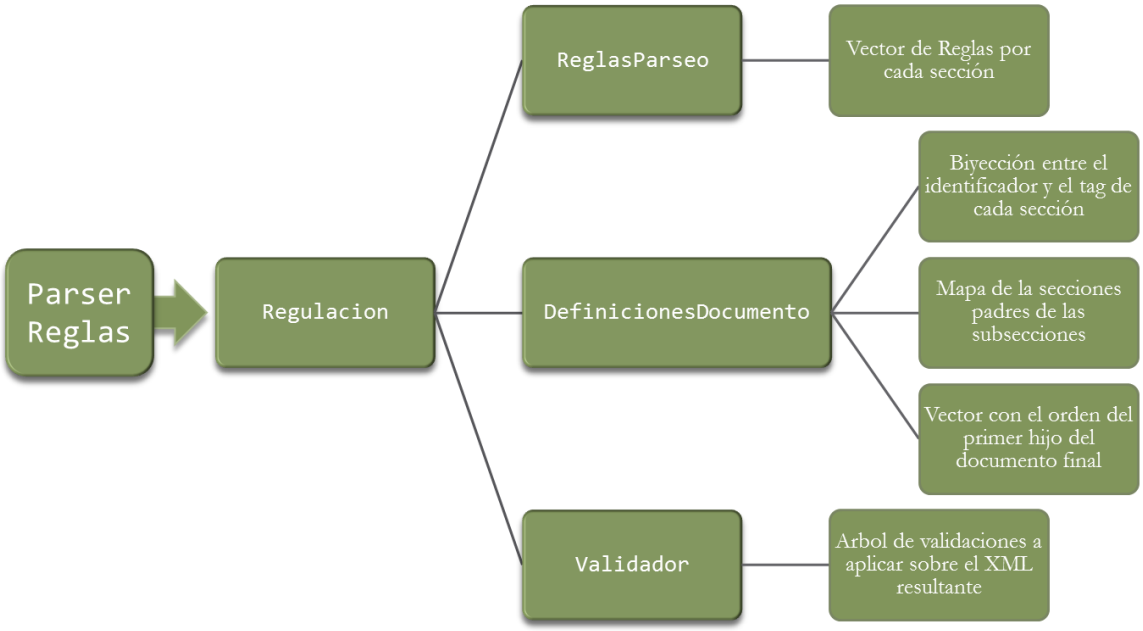


Ilustración 12. Diagrama de la composición de objetos de clase Regulacion.

Veremos qué función cumple cada componente de los objetos de la clase Regulacion a medida que veamos el proceso de parseo del Archivo de Definiciones de Reglas.

Paso 1. La clase ParserReglas es abstracta y declara los métodos abstractos:

- abstract void agregarSubsecciones(…)
- abstract Validador construirValidador (…)
- abstract Vector<String> construirTagsOrdenadosPrimerHijo()

que son definidos en sus clases extendidas ParserReglasDTE y ParserReglasLibro. Al utilizar el método estático parseReglas(…) de la clase ParserReglas, lo primero que se hace es leer la primera línea del Archivo de Definiciones de Reglas y determinar el tipo de documento tributario que define el archivo, dependiendo de este, dentro del mismo método se instancia a un objeto ParserReglasDTE o un objeto ParserReglasLibro para realizar el resto del procesamiento del Archivo de Definiciones de Reglas. Los métodos abstractos definidos en las clases específicas de cada tipo serán explicados más adelante.

Paso 2. Luego de haber determinado el tipo de documento tributario al que corresponde el Archivo de Definiciones de Reglas, el siguiente paso es leer las definiciones de las secciones y realizar el mapeo entre los identificadores de sección y el nombre del elemento al que corresponden en el XML resultante. Esto se implementa utilizando dos Hashtables que funcionan como una biyección entre los identificadores de sección y los nombres de los elementos y esta biyección es almacenada como un atributo del componente DefinicionesDocumento de la clase Regulacion.

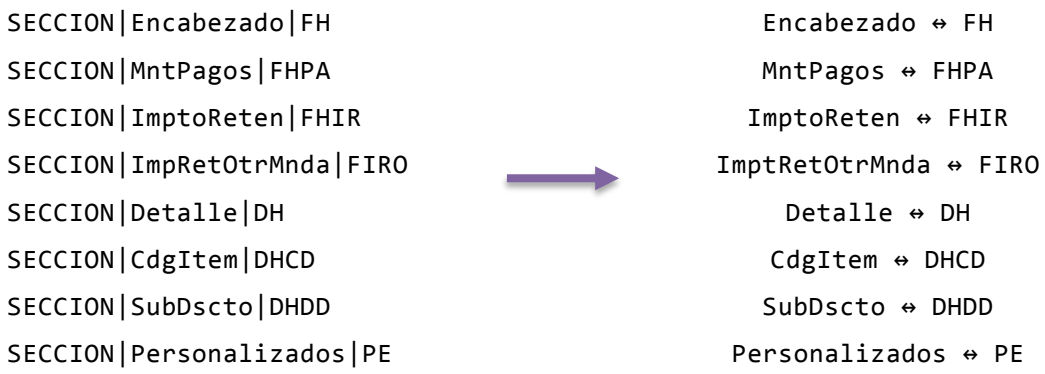


Ilustración 13. Proceso de creación de biyección entre identificadores de secciones y nombre de elemento.

Paso 3. El siguiente paso una vez leídas las definiciones de secciones es comenzar a procesar todas las reglas de mapeo de cada una de las secciones. Cada regla de mapeo es almacenada en un objeto de la clase Regla que tiene atributos análogos a los de una línea de regla de mapeo. Luego se hace una lista de reglas de mapeo por cada sección y se ingresa a una Hashtable que tiene por llaves el nombre de las secciones y como objetos la lista de reglas de mapeo. Esta Hashtable queda en la componente ReglasParseo de la clase Regulación. Cada regla de mapeo es ingresada a la lista en el orden el que se encuentra en el Archivo de Definiciones de Reglas.

Una Regla de mapeo entre los tributos que contiene está el Formato en el que se espera que se encuentre el dato. Este formato es asignado por defecto al leer el tipo de dato al que

corresponde el campo y sobrescrito posteriormente si se especifica un formato especial. Veremos la definición de la clase Formato más adelante.

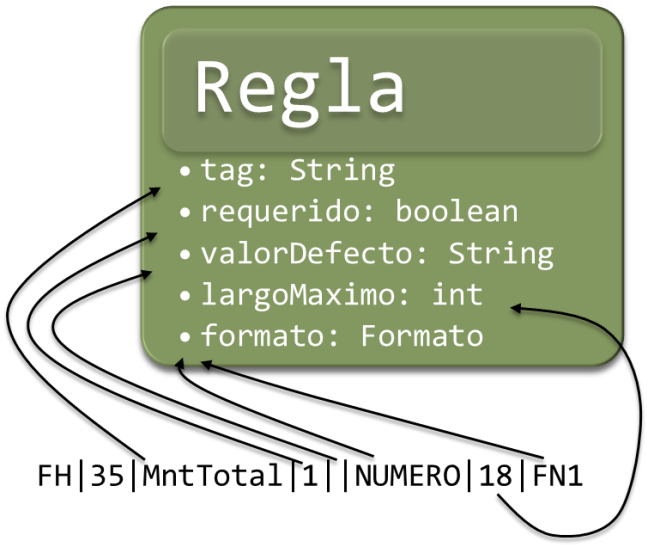
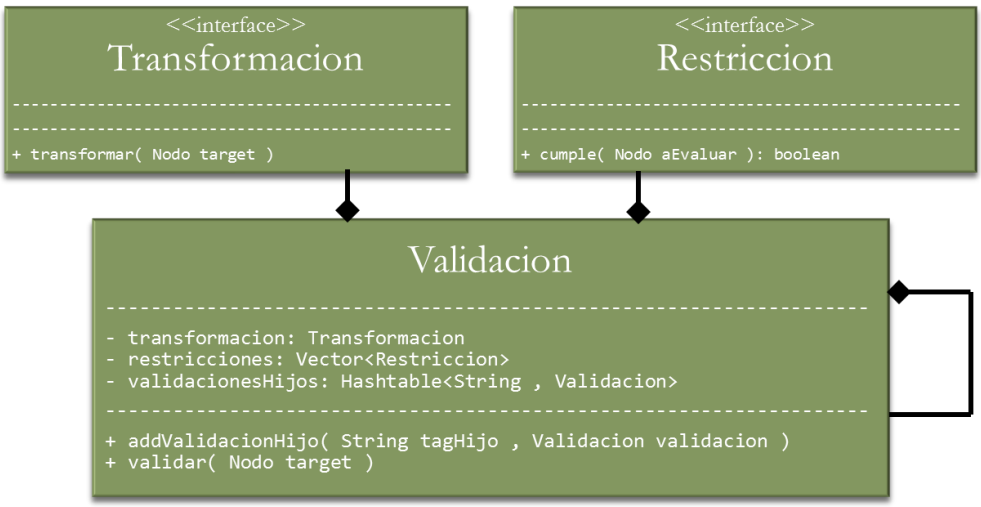


Ilustración 14. Proceso de parseo de una regla de mapeo.

Hasta este punto el procesamiento de un Archivo de Definiciones de Reglas es el mismo para los dos tipos de documentos tributarios. Los siguientes pasos utilizan los métodos definidos en las clases especializadas para definir operaciones que deben realizarse al procesar el documento tributario para construir el árbol XML.



Paso 4. La realización del siguiente paso tiene que ver con generar un árbol análogo al árbol XML que resulta del procesamiento de documentos tributarios para realizar validaciones y transformaciones sobre los nodos que se especifiquen. Para esto se utiliza el método abstracto construirValidador(…) el cuál retorna un objeto de clase Validador que contiene un árbol de Validaciones, cada nodo de Validacion tiene asociado un nombre de elemento, una Transformacion y una Restriccion. El nombre de elemento corresponde al elemento del

documento XML al que se le aplicará la Validacion y con aplicar nos referimos a ejecutar los métodos de las interfaces Transformacion y Restriccion sobre el Nodo.

La intención de la interfaz Transformacion es realizar alguna operación sobre el Nodo que altere la estructura del árbol o algún atributo del Nodo, cómo por ejemplo, subir un Nodo de altura en el árbol dejándolo como hermano de su Nodo padre. Un nodo de Validacion puede aplicar solo una Transformacion sobre un Nodo.

Por su parte la interfaz Restriccion define el método boolean cumple(Nodo aEvaluar) cuya intención es evaluar si el Nodo de la Validacion cumple con una condición específica. Esta interfaz es utilizada para validar que el árbol XML resultante sea válido según la especificación de esquema del SII. Un ejemplo de Restriccion aplicada a los documentos DTE es que el Nodo raíz debe tener un y solamente un nodo hijo llamado Encabezado. Un nodo de Validacion puede tener varias Restricciones que evaluar sobre un Nodo.

El método construirValidador(…) definido en las clases especializadas ParserReglasDTE y ParserReglasLibro crea este árbol de Validaciones utilizando Hashtables en cada nodo de Validacion de modo que se agrega a la Hashtable un nodo de Validacion como hijo utilizando de llave el nombre del elemento XML sobre el que se ejecuta la Validacion.

Paso 5. La siguiente operación que se lleva a cabo en el proceso de parseo del Archivo de Definiciones de Reglas es la que también es definida en las clases especializadas en el método construirTagsOrdenadosPrimerHijo(). Esta operación es bastante sencilla y consiste en retornar un Vector (java.util.Vector) de tipo String que contiene en orden el nombre en el que se debieran encontrar los elementos hijos del primer nodo del documento después de la raíz (Es decir, los hijos del primer nodo de altura 1). Este Vector es utilizado en el proceso de parseo de documentos tributarios para ordenar las secciones en el orden definido por el esquema de definición XSD del SII.

Paso 6. Y el último paso del proceso es el de identificar y mapear las secciones del documento tributario que son subsecciones de otra sección.

Este paso resuelve la siguiente problemática: En la definición del Archivo de Definiciones de Reglas tenemos una sección de líneas que corresponde al mapeo de identificadores de secciones con el nombre del elemento XML al que corresponde la sección. Ocurre que en estas líneas se pueden incluir secciones que no corresponden a lo que entendemos por secciones del documento XML resultante, sino a nodos que son hijos de secciones pero que también contienen una gran cantidad de elementos hijos. Sin la posibilidad de modificar el Archivo de Definiciones de Reglas para especificar en este cuales son secciones y cuales son subsecciones, debemos hacer esto dentro del ParserReglas.

Para esto, el método abstracto agregarSubsecciones(…) configura el componente DefinicionesDocumento de la Regulación con un mapa de los nombres de secciones que corresponden a subsecciones y su respectiva sección padre. Esto es implementado con una Hashtable y esta es utilizada posteriormente en el parseo de documentos tributarios para insertar las subsecciones en el Nodo correspondiente.

PARSEO DE DOCUMENTOS TRIBUTARIOS

Este es el proceso principal de la componente, es un proceso que debe ser eficiente ya que debe procesar una gran cantidad de documentos tributarios por segundo. El proceso consiste en utilizar un Archivo de Definiciones de Reglas para basado en esas definiciones leer documentos tributarios de formato de texto plano y generar archivos XML.

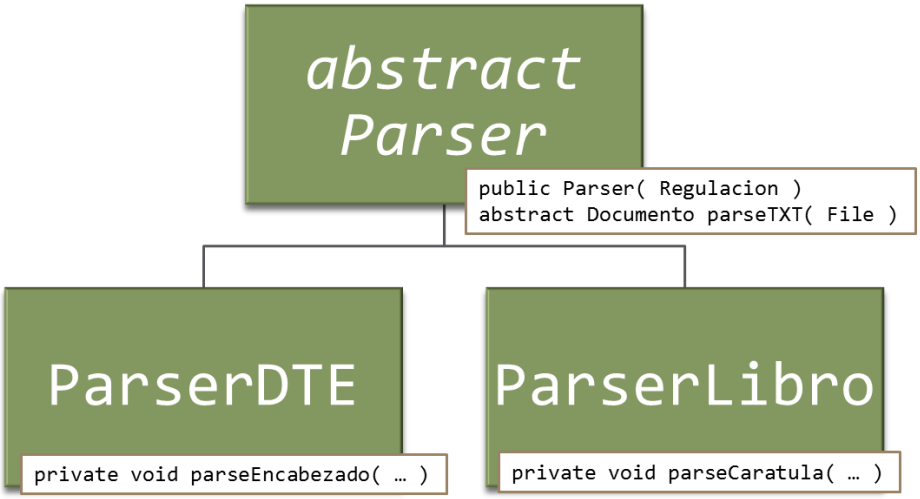


Ilustración 15. Diagrama de clases Parsers de documentos tributarios.

Para esto se define la clase abstracta Parser, la cual define que debe ser inicializada pasándole como argumento un objeto de clase Regulacion (Obtenido del parseo de un Archivo de Definiciones de Reglas) y define el método abstracto parseTXT que recibe como argumento una referencia a un archivo o un InputStream y entrega un Documento (Recordemos que según nuestro modelo un Documento es un Nodo raíz de un árbol con elementos).

El motivo de que el método parseTXT sea abstracto es que cada Documento generado tiene algunas operaciones específicas que se deben realizar al procesar el documento tributario para generar el XML final. Sin embargo, la mayor parte de la funcionalidad está implementada en métodos protegidos (Protected) de la clase Parser, de modo que en las clases ParserDTE y ParserLibro se define un conjunto pequeño de instrucciones y el resto se reutiliza de la superclase.

Luego el proceso de parseo de un documento tributario, asumiendo que tenemos en nuestra memoria un objeto de la clase Regulacion, es como sigue.

Paso 1. Usando el modelo DOM que describimos previamente, creamos el Nodo raíz del documento el cual es un objeto de la clase Documento. Puede ser DocumentoDTE o DocumentoLibro según el Parser que se esté ocupando. Además, a este Nodo raíz, le agregamos su primer hijo. En la ilustración 16 podemos ver el XML resultante que se mostraría si se transformara a XML en árbol existente hasta este momento.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DTE version="1.0" xmlns="http://www.sii.cl/SiiDte">
   <Documento>
   </Documento>
</DTE>
```

Ilustración 16.Ejemplo de Documento XML resultante después del primer paso.

Paso 2. Procesar la primera línea del documento tributario. La primera línea contiene información clave para el documento tributario, de ella se obtiene un identificador único para el documento y se debe comprobar siempre que esté bien formada. En la ilustración 17 podemos ver un ejemplo de cómo se vería un XML de un documento DTE después de procesar su primera línea, la sección del encabezado.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DTE version="1.0" xmlns="http://www.sii.cl/SiiDte">
   <Documento ID="DOCT39F1390">
      <Encabezado>
         <TipoDTE>39</TipoDTE>
         <Folio>1390</Folio>
         <FchEmis>2010-11-18</FchEmis>
         <MntTotal>2502</MntTotal>
      </Encabezado>
   </Documento>
</DTE>
```

Ilustración 17. Ejemplo de Documento XML resultante después del segundo paso.

Paso3. Procesar el resto de las líneas del documento. Cómo ya hemos visto, cada línea representa una sección del documento tributario, y por ende, todas estas líneas contienen información sobre un Nodo con muchos elementos hijos y cada elemento hijo es un dato que viene en la línea.

El parseo se hace utilizando la lista de Reglas que se encuentra en el componente ReglasParseo de la clase Regulación. Para cada línea existe un Vector de tipo Regla que tiene en orden las Reglas que le corresponden a cada campo. De este modo, se parte del primer delimitador "|" de la línea en adelante usando para cada campo separado por delimitador la Regla de parseo encontrada en la lista en la posición de la línea en la que estamos, y es aquí donde se realiza la transformación de un campo a un Nodo donde su atributo texto corresponde al dato del campo. Dependiendo de si se especifica que el dato viene en algún formato en particular, se hace una transformación del dato al formato aceptado por el SII antes de ser asignado como texto del elemento.

Hasta este punto no hemos usado ninguna de las Validaciones, ordenaciones ni inserciones de subsecciones que discutimos en el Parseo de Archivos de Definiciones de Reglas. En la ilustración 18 podemos ver un ejemplo de cómo se vería un XML de un documento DTE después de este paso.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DTE version="1.0" xmlns="http://www.sii.cl/SiiDte">
   <Documento ID="DOCT39F1390">
      <Encabezado>
         <TipoDTE>39</TipoDTE>
         <Folio>1390</Folio>
         <FchEmis>2010-11-18</FchEmis>
         <MntTotal>2502</MntTotal>
      </Encabezado>
      <Detalle>
         <NroLinDet>1</NroLinDet>
         <PrcItem>6230</PrcItem>
         <MontoItem>6230</MontoItem>
      </Detalle>
      <DscRcgGlobal>
         <NroLinDR>1</NroLinDR>
         <TpoMov>D</TpoMov>
         <ValorDR>879</ValorDR>
      </DscRcgGlobal>
      <Personalizados>
         <CantProd>3</CantProd>
         <CantUnid>3</CantUnid>
      </Personalizados>
   </Documento>
</DTE>
```

Ilustración 18. Ejemplo de Documento DTE después del paso 3.

Paso 4. Ejecutar las Validaciones sobre los Nodos del árbol. Recordemos que en la Regulacion del parser teníamos un árbol de Validaciones con Transformaciones y Restricciones que debían realizarse sobre los nodos. Es en este paso en el que se recorre este árbol de Validaciones y se aplican las Transformaciones sobre los Nodos que las tengan y se verifican las Restricciones que existan.

El algoritmo para recorrer el árbol de Validaciones es bastante simple. El árbol de Validaciones a diferencia del árbol de Nodos está estructurado mediante Hashtables y no Vectores, de modo que la idea es que si para algún elemento que se use de llave no se encuentra Validacion dentro del Hashtable, es porque ese nodo no contiene Validaciones que se le deban hacer. De este modo, para recorrer el árbol de Validaciones en realidad se recorre el árbol de Nodos del documento y para cada elemento se verifica en el Hashtable la existencia o no de Validacion y se aplica de ser necesaria. En la ilustración 19 podemos ver un ejemplo de la aplicación de algunas Restricciones y Transformaciones del árbol de Validaciones a un documento DTE; algunas Validaciones solamente comprueban Restricciones y otras solo realizan transformaciones, depende del tipo de documento tributario.

Ilustración 19. Ejemplo de aplicación de árbol de Validaciones sobre un DTE.

Paso 5. Utilizando el Vector de orden de los elementos del primer nodo de altura 1 (En nuestro ejemplo, el nodo con nombre "Documento") que se encuentra en la componente DefinicionesDocumento de la Regulación, ordenamos los elementos hijos del Nodo para que se encuentren en el orden especificado en el XSD del SII. Esto se realiza en caso de que en el documento tributario recibido se encuentren líneas de secciones desordenadas. En la ilustración 20 podemos ver un ejemplo de un intercambio de orden de dos nodos realizados en este paso.

```
definicionesDocumento.getOrdenHijos() =
{ "Encabezado"  , "Detalle" , "SubTotInfo" , "DscRcgGlobal" , "Referencia" , "Comisiones" }
```

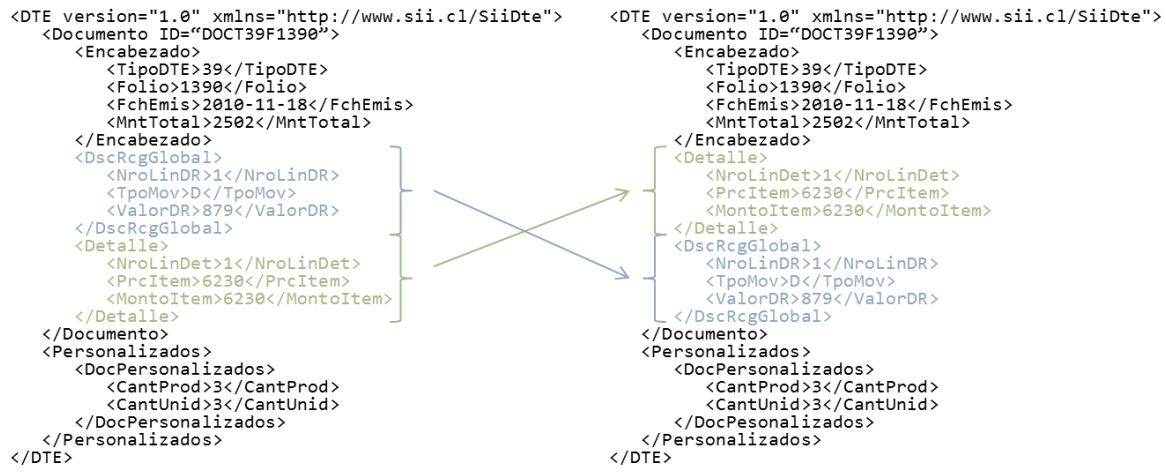


Ilustración 20. Ejemplo de ordenación de Nodos en un Documento DTE.

Paso 6. Remover Nodos vacíos. A pesar de que no está definido en el esquema de definición XSD, el SII especifica que los nodos sin información no deben ser enviados en el XML, es decir, los elementos XML de la forma "<elemento></elemento>" o "<elemento/>" que no contienen texto, no deben estar presentes en el Documento. Luego en este paso hacemos un simple recorrido por todos los nodos del árbol y eliminamos los nodos que no contengan texto, atributos ni elementos hijos. En la ilustración 21 podemos ver un ejemplo de un proceso de eliminación de nodos.

Paso intermedio. Mencionamos anteriormente la problemática que se presentaba cuando nos encontrábamos con secciones que realmente eran subsecciones de secciones. Esto se resuelve en el paso 3 al utilizar el mapa de las subsecciones con sus secciones padres. Básicamente lo que ocurre es que para cada nueva línea que se va procesando, se revisa previamente si es que está incluida dentro de este mapa, y si es así, en vez de insertarse en el Nodo en el que se encuentran las secciones, se inserta en la sección en la que corresponde según el mapa.

Y de esta forma concluye el proceso de parseo de documentos tributarios. Una vez obtenido el objeto Documento que retorna el método parseTXT, basta con llamar al método toXML definido en la clase Nodo para obtener el XML resultante y escribirlo en un archivo o enviarlo por un OutputStream según se necesite.

```
<DTE version="1.0“ xlns="http://www.sii.cl/SiiDte">
   <Documento ID=“DOCT39F1390”>
      <Encabezado>
         <TipoDTE>39</TipoDTE>
         <Folio>1390</Folio>
         <FchEmis>2010-11-18</FchEmis>
         <MntTotal>2502</MntTotal>
      </Encabezado>
      <Detalle>
         <NroLinDet>1</NroLinDet>
         <PrcItem>6230</PrcItem>
         <MontoItem></MontoItem>
      </Detalle>
      <DscRcgGlobal>
         <NroLinDR>1</NroLinDR>
         <TpoMov></TpoMov>
         <ValorDR>879</ValorDR>
      </DscRcgGlobal>
   </Documento>
   <Personalizados>
      <DocPersonalizados>
         <CantProd></CantProd>
         <CantUnid></CantUnid>
      </DocPersonalizados>
   </Personalizados>
</DTE>
```

```
<DTE version="1.0“ xlns="http://www.sii.cl/SiiDte">
   <Documento ID=“DOCT39F1390”>
      <Encabezado>
         <TipoDTE>39</TipoDTE>
         <Folio>1390</Folio>
         <FchEmis>2010-11-18</FchEmis>
         <MntTotal>2502</MntTotal>
      </Encabezado>
      <Detalle>
         <NroLinDet>1</NroLinDet>
         <PrcItem>6230</PrcItem>
      </Detalle>
      <DscRcgGlobal>
         <NroLinDR>1</NroLinDR>
         <ValorDR>879</ValorDR>
      </DscRcgGlobal>
   </Documento>
</DTE>
```

Ilustración 21. Ejemplo de proceso de eliminación de Nodos vacíos en un documento DTE.

La ilustración 22 es un diagrama de resumen de lo que es el proceso de procesamiento de archivos, tanto Archivos de Definiciones de Reglas como documentos tributarios.

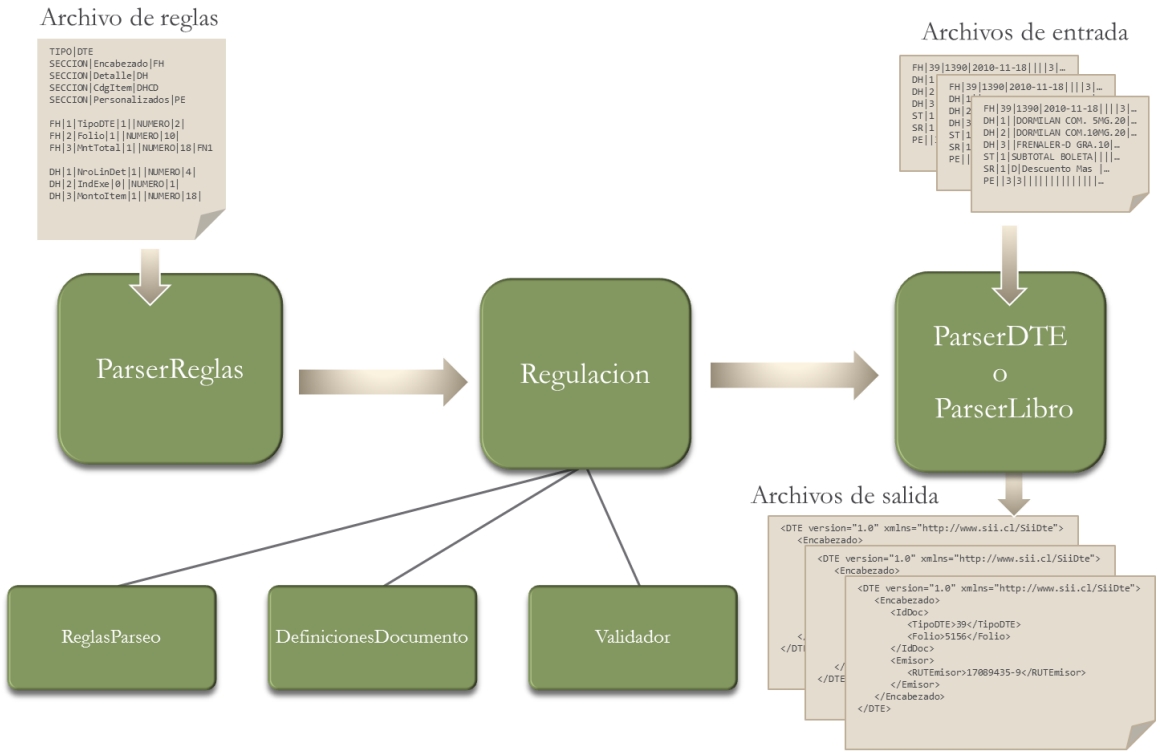


Ilustración 22. Diagrama de resumen del procesamiento de documentos tributarios.

REPORTE DE ERRORES

Uno de los objetivos de la componente F-Box es generar también información lo más detallada y comprensible posible en caso de que un documento tributario no se pudiese procesar o no cumpliera con lo necesario para generar un documento XML válido. Para esto se utilizan reportes de errores, los cuales informan de toda las situaciones encontradas al procesar un documento tributario que no permiten generar un XML válido.

La clase Parser que procesa documentos tributarios, cada vez que procesa un nuevo documento instancia un objeto de la clase GeneradorReporteErrores. Este objeto tiene visibilidad estática para todos los métodos que realizan procesamientos sobre el archivo y es a quién se le debe informar todas las situaciones que provocaron que un cierto dato no se haya podido procesar. Cualquier problema que surja durante el procesamiento es reportado aquí utilizando un método que permite indicar la posición en el documento tributario en la que se encontró el error, el motivo del error, la gravedad del error, y que era lo que se esperaba en cambio.

Las gravedades reportables son:

- Nula: Eventos inesperados que no afectan al proceso ni producen un XML inválido. (Ej: Encontrar más campos que los esperados en una línea).
- Mínima: Errores que impiden que el XML resultante sea válido según su definición de esquema.
- Grave: Errores que impiden que el XML resultante esté completo, tenga validez semántica o cumpla con su definición de esquema.

Una vez finalizado por completo todo el proceso de parseo del documento tributario, se le pide al GeneradorReporteErrores que genere el reporte de errores. Si el GeneradorReporteErrores tiene registrado por lo menos un problema de gravedad mínima o grave, el reporte de errores es generado quedando guardado un archivo en una carpeta determinada para su posterior revisión, si en cambio solo encuentra errores de gravedad nula o no encuentra errores, no se genera reporte de errores y se permite la generación del archivo XML resultante.

En la ilustración 9 veíamos un ejemplo de un reporte de errores generado para un documento DTE.

PRUEBAS

Se realizaron algunas pruebas para comprobar el funcionamiento correcto y eficiente de la componente F-Box.

Las pruebas más básicas consistieron en comprobar que los archivos XML generados al procesar documentos tributarios utilizando un cierto Archivo de Definiciones de Reglas estuviesen bien formados y correspondieran al XML correcto resultante del proceso del documento tributario.

Para esto el área de QA de Paperless proveyó algunos archivos de prueba los cuales fueron procesados exitosamente sin errores y fueron validados utilizando herramientas Java para comprobar que cumplían con el formato definido por el XSD del SII.

Otro tipo de pruebas que se realizó fue el de generar documentos tributarios al azar utilizando generadores de palabras, números, fechas y Rut aleatorios, y comprobar que el sistema no se caía al procesarlos a pesar de la aleatoriedad de los archivos. Esta prueba fue pasad con éxito y se generaron los reportes de errores correspondientes al procesar los archivos.

```
FH|1733276587|6865712-5|1916-6-15|176|6659715-0|801||443|6957569-8|22515975-K|||15271261-4||7724645-0
SR|14805754-2|308|4778715-5|187|313|1969-3-5|19267582-6|624|1975-4-20||22815019-7|2008-2-25|22936326-7
RE|164|TE4182|685|1923-6-20|1940-6-26||541|6454097-1|
ST|622|709|598|||914|1901-10-4|71|84
FHPA|1921-7-11|NAFTRKJN354||1943-11-6|14545446-9|14302209-2|17082573-8|20282365-0|652|349|1962-6-21
SR|1912-2-20|314|SGNFHGX1752115|1922-12-16|21530254-8|898|IFCBM377758|15543804-1|12614029-7|1932-1-7
FIRO|1942-11-8|2010-3-10|LXIPP58261677|176|1932-2-9|23640916-2|2009-11-3|801|647|22558413-5|176|330
FHIR|1913-2-6|1983-4-24||21335633-5|1939-2-8|1924-2-28||22616086-4|12623412-K|1974-8-10|||1923-1-16
DHOM||1993-6-8|241|5575897-5|850|827|729||809|1926-2-28|20617232-8|1968-4-4|810|919|10986906-1||
SR|1988-9-23||6966708-0|23986968-2|15558974-2|
DHDD|||396|864|1966-4-4|10829046-4|19661554-5|5145254-9|1948-11-26|SYC241671|14646057-5|1945-5-4||85||
DHCD|11543260-8|1926-11-22|XWXI431||7380495-2|1956-5-2|RTSFENFWYM16386764|GR3885452|18997218-3|96976
SR||435|6992507-3|1952-7-17|17438281-4|21362052-7|1994-9-3||1903-8-8|855|17188621-6|1974-11-28|2008
DHDD|1965-12-13|KM61537576|||1935-2-20|1957-5-2|573|11211814-K|21894137-6|18247763-2|MNCMFLYK554|
IM|914|552|1907-3-7|9766776-5|4234445-8|209|||1941-5-4
DHDD|13577313-1|2004-8-13|DL5317|1909-4-22|6243395-6|1952-3-23|18493107-K|17359169-8|199||1929-10-15
DHCD|382|520|21649525-5|871|5274648-3||13460732-5|9967524-1|SH6534|23896238-8|6728296-3|OBHK1862313
DHDR|SLTLX41648|1940-12-26|17508465-K|10246816-4|600|15528369-7|302|15144034-2|318|2008-9-4|2||1919
DHDC|20471471-4|1944-7-22|831|15585461-7|RYS6626621|488|23936406-3|1976-8-11|UHVO35883|B63731||AKNB71
FHIR|849||1938-6-19||11984342-K|AHWFW86682||27||14825918-6|9035627-K|10925525-2|4141487-1|1991-1-19
DHDC||1980-5-21|471|980|1946-6-10|1951-7-19|102|176|989|17497934-6|892|287|LGGCQA32733|1934-9-11|
```

Ilustración 23. Ejemplo de documento tributario generado al azar.

Y el último tipo de prueba que se realizó fue el de eficiencia, en el que se midieron tiempos de procesamiento de documentos tributarios al intentar procesar una gran cantidad de ellos. Los resultados fueron los siguientes.

- Se procesaron 100.000 DTEs aleatorios; 630.1 Mb.
- Tiempo total: 5:32 minutos
- Velocidad promedio: 300.65 DTEs por segundo

- Se procesaron 100.000 Libros aleatorios; 1.2 Gb.
- Tiempo total: 10:43 minutos
- Velocidad promedio: 155.28 Libros por segundo

Por lo que se pudo concluir que la componente F-Box cumplió con su propósito satisfactoriamente y de forma eficiente.

INTEGRACIÓN CON GEFA

La integración de la componente F-Box con la aplicación web GEFA fue uno de las funcionalidades más útiles al momento de construir filtros utilizando esta interfaz, ya que permite en el mismo momento de construcción del filtro evaluar cuál será el resultado obtenido de procesar un cierto documento tributario de ejemplo.

Esta integración ocurre en la pantalla de edición de filtro de la aplicación web, la cual tiene por objetivo generar un Archivo de Definiciones de Reglas. La interactividad se da al utilizar las reglas de mapeo que el JP va agregando en la página de edición de filtro para procesar ahí mismo en tiempo real el documento tributario de ejemplo, indicando qué errores se encuentran en el proceso y una vista preliminar del XML generado con esas reglas.

En la ilustración 24 podemos ver la pantalla de edición de filtro. Esta pantalla está dividida en 4 secciones. El archivo de ejemplo, el mapa de parseo, el XML resultante y los errores. En la sección de mapa de parseo se modifican las reglas de parseo del filtro, pudiéndose agregar nuevas reglas, y eliminar o editar reglas existentes. Al mismo tiempo que una regla de parseo es modificada, se realiza el procesamiento del documento tributario de ejemplo y se arroja en las secciones de XML resultante y errores lo correspondiente.

Esto se logra haciendo que la aplicación web interactúe con la componente F-Box generando un Archivo de Definiciones de Reglas usando sus reglas de mapeo y entregándoselo al motor para generar Parsers de documentos dinámicamente y actualizar sus vistas.

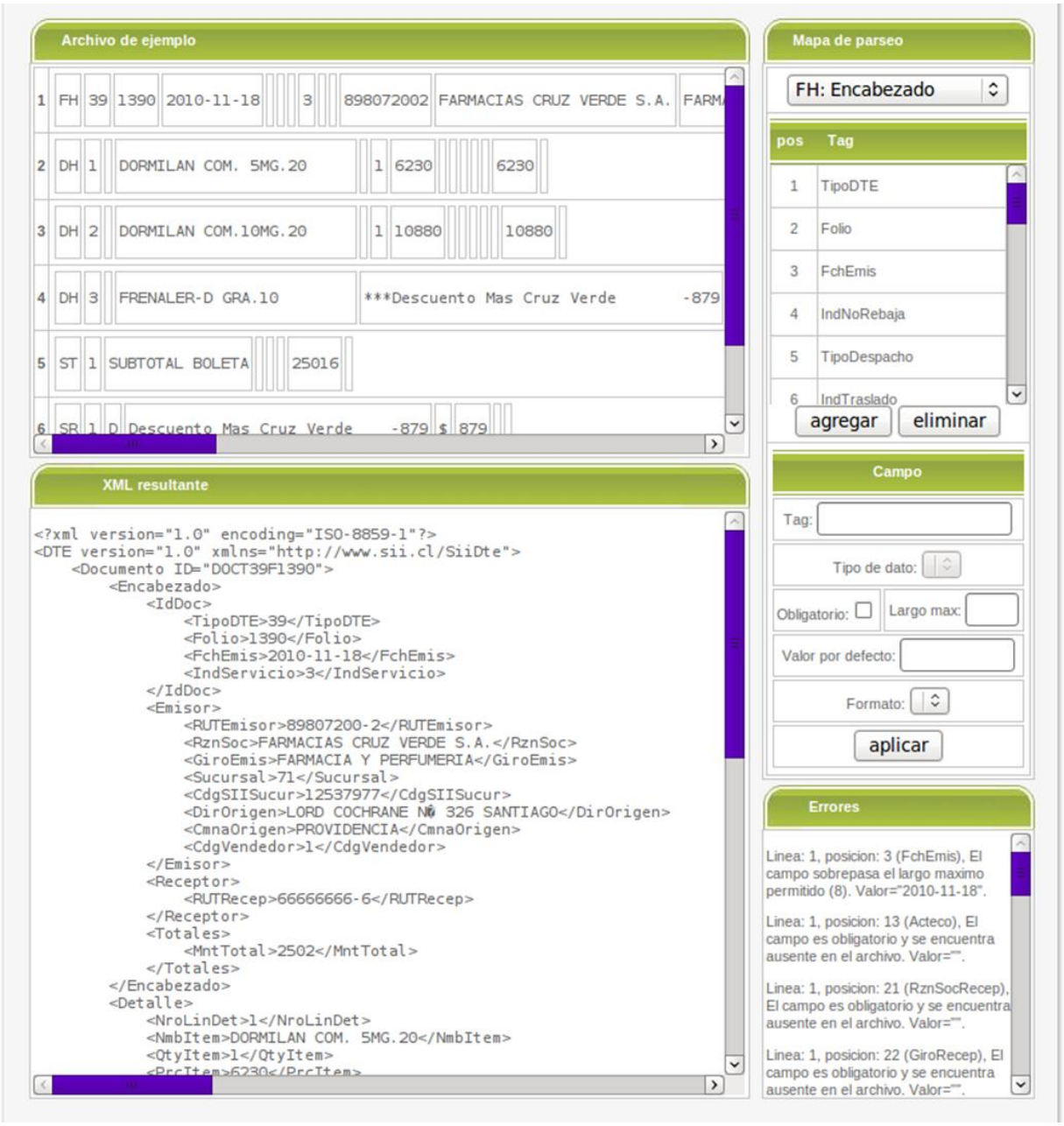


Ilustración 24. Pantalla de edición de filtro de la aplicación web GEFA.

LISTA DE ARCHIVOS

A continuación daremos una breve descripción de todos los archivos desarrollados en la componente F-Box ordenándolos por el paquete Java en el que se encuentran.

Paquete cl.paperless.fbox

- DefinicionesDocumento.java
  - Clase de las DefinicionesDocumento que son parte de una Regulacion.
- DocumentoDTE.java
  - Clase que extiende de Documento y que contiene métodos para acceder a cierta información importante sobre un DTE.
- DocumentoLibro.java
  - Clase que extiende de Documento y que contiene métodos para acceder a cierta información importante sobre un Libro.
- Parser.java
  - La clase abstracta Parser.
- ParserDTE.java
  - La clase para parsear documentos tipo DTE.
- ParserLibro.java
  - La clase para parsear documentos tipo Libro.

Paquete cl.paperless.fbox.dom

- Atributo.java
  - Bean que contiene los atributos “nombre” y “valor”. Utilizado para agregar atributos a un Nodo.
- Documento.java
  - Clase que extiende a Nodo cuyo propósito es ser utilizado como una abstracción del Nodo raíz de todo árbol. En esta clase se sobrescribe el método toXML() para agregar la línea “<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>” al comienzo del archivo.
- Nodo.java
  - Clase Nodo, base para la estructura del modelo DOM.

Paquete cl.paperless.fbox.regulacion

- ParserReglas.java
  - Clase abstracta del parser de reglas.
- ParserReglasDTE.java
  - Clase con los métodos para procesar Archivos de Definiciones de Reglas de tipo DTE.
- ParserReglasLibro.java
  - Clase con los métodos para procesar Archivos de Definiciones de Reglas de tipo Libro.
- Regla.java

    - Clase de una regla de parseo.
- ReglasParseo.java
    - Clase que contiene las listas de las Reglas de parseo separadas para cada sección. Este es uno de los objetos que se encuentran dentro de un objeto Regulacion.
- Regulacion.java
    - La clase obtenida de procesar un Archivo de Definiciones de Reglas, que contiene objetos ReglasParseo, DefinicionesDocumento y el Validador.

Paquete cl.paperless.fbox.reporte

- GeneradorReporteErrores.java
    - Clase para generar reportes de errores.
- ReporteErrores.java
    - Clase que contiene una lista de Problemas. Un nuevo objeto de esta clase es instanciado por el GeneradorReporteErrores cada vez que se procesa un documento tributario y cada Problema que se encuentra es agregado a esta lista.
- Gravedad.java
    - Clase que define los tipos de gravedad de los Problemas.
- Problema.java
    - Clase utilizada para definir un problema encontrado. Cada vez que se encuentra un error al procesar un documento tributario, el GeneradorReporteErrores crea una nueva instancia de esta clase con la descripción del error encontrado y lo agrega a la lista de la clase ReporteErrores.

Paquete cl.paperless.fbox.validacion

- Restriccion.java
    - Interfaz que define cómo definir una cierta restricción con la que debe cumplir un Nodo
- RestriccionCantidadHijos.java
    - Implementación de interfaz Restriccion. Esta clase agrega una restricción sobre un Nodo que indica un mínimo y un máximo de apariciones que puede tener un elemento con cierto nombre como Nodos hijos. Esta clase es usada por los ParserReglas especializados para agregar restricciones sobre ciertos Nodos.
- Validacion.java
    - Clase con la implementación del árbol de validaciones definido con Hashtables.
- Validador.java
    - Clase que contiene el árbol de Validaciones y un método para realizar la validación completa del árbol de Nodos XML.

Paquete cl.paperless.fbox.estructuracion

- Ascencion.java

    - Clase que define una Transformacion consistente en subir de altura un Nodo convirtiéndolo en hermano de su Nodo padre.
- Descenscion.java
    - Clase que define una Transformacion consistente en bajar de altura un Nodo ingresándolo como hijo de otro nodo.
- Renombre.java
    - Clase que define una Transformacion consistente en cambiar el nombre del Nodo.
- Transformacion.java
    - Interfaz que define cómo implementar una transformación que se quiera realizar sobre un Nodo.
- TransformacionNula.java
    - Clase que define una Transformacion aplicada sobre un Nodo que no produce ningún cambio.

Paquete cl.paperless.fbox.formatos

- Formato.java
    - Clase abstracta que define los métodos que se deben implementar para soportar la conversión de un dato en un cierto formato al formato aceptado por el SII.

A continuación veremos la lista de las clases con todos los formatos soportados que se implementaron. Indicaremos solamente el formato al que corresponden ya que todos implementan la misma funcionalidad, la cual es tomar un String que se asume que se encuentra en el formato indicado y transformarlo al formato aceptado por el SII. Los formatos aceptados por el SII se definen como los formatos por defectos y son asignados automáticamente un campo si en su regla de parseo no se especifica un formato especial.

Paquete cl.paperless.fbox.formatos.decimal

- FormatoDecimal1.java: 9999,9999
- FormatoDecimalDefecto.java: 9999.9999

Paquete cl.paperless.fbox.formatos.fecha

- FormatoFecha1.java: AAAAMMDD
- FormatoFecha2.java: DD-MM-AAAA
- FormatoFecha3.java: DDMMAAAA
- FormatoFecha4.java: DD/MM/AA
- FormatoFecha5.java: DD/MM/AAAA
- FormatoFecha6.java: AAAA/MM/DD
- FormatoFechaDefecto.java: AAAA-MM-DD

Paquete cl.paperless.fbox.formatos.numero

- FormatoNumero1.java: ###.###,00
- FormatoNumero2.java ###,###.00

- FormatoNumeroDefecto.java: 9999

Paquete cl.paperless.fbox.formatos.periodo

- FormatoPeriodo1.java: AAAAMM
- FormatoPeriodo2.java: AAAA/MM
- FormatoPeriodo3.java: MM/AAAA
- FormatoPeriodo4.java: MMAAAA
- FormatoPeriodo5.java: MM-AAAA
- FormatoPeriodoDefecto.java: AAAA-MM

Paquete cl.paperless.fbox.formatos.rut

- FormatoRut1.java: 999999999
- FormatoRut2.java: 99.999.999-9
- FormatoRut3.java: 11,111,111-1
- FormatoRutDefecto.java: 99999999-9

Paquete cl.paperless.fbox.formatos.texto

- FormatoTextoDefecto.java: *

CONCLUSIONES

El trabajo fue enriquecedor particularmente en el acercamiento al ambiente de desarrollo laboral que tanto difiere del ambiente académico. En la experiencia de este practicante, los mayores conocimientos adquiridos fueron los del funcionamiento interno de una gran compañía de software, la organización de las áreas de la empresa y las responsabilidades de cada uno.

También se ganó experiencia en áreas de la computación que al momento de realizar la práctica no se tenía conocimiento, como es el desarrollo de aplicaciones web, trabajar bajo los estándares Java EE y todo lo que implica la implementación técnica, en este caso reflejada en el uso del servidor Tomcat. Y entre otros conocimientos que se adquirieron en los que no se tenía conocimiento muy profundo fueron en XML, XHTML, CSS2, JavaScript y las aplicaciones de gran utilidad SVN y Ant.

Los mayores problemas que se presentaron fueron los del cambio de requerimientos de las funcionalidades de la aplicación. En el sentido de que con la intención de que el desarrollo del proyecto fuese gradual, los supervisores del proyecto fueron agregando a medida que se iban viendo avances más funcionalidades a la componente F-Box; funcionalidades que no eran mencionadas inicialmente más que de haberlo sido hubiese evitado tener que reestructurar la lógica de los procesos o la arquitectura general del proyecto. Esto generó que el producto final no fuese el más eficiente, sino lo que resulta cuando se agregan piezas de código a un modelo pensado inicialmente para resolver un problema particular.

Por lo demás el proyecto es considerado un éxito por la parte de este practicante, ya que la componente en la cual se trabajó quedó terminada correctamente. Sin embargo no deja de ser lamentable que el proyecto final, la integración completa entre F-Box y GEFA, no se haya consolidado ya que de haberse hecho se hubiese dejado un producto de inmensa utilidad para Paperless. Las razones de que esto no se haya logrado fueron principalmente la falta de tiempo y coordinación del trabajo entre ambos practicantes desarrolladores.

Esto no quita que el proyecto pueda ser terminado posteriormente, ya que la funcionalidad faltante de la componente GEFA es un problema simple de implementar para cualquier desarrollador experimentado, por lo que bastaría con que el área de desarrollo dedicara un corto tiempo en la conclusión de la aplicación web y se cumpliría con el objetivo inicial de este proyecto que era el de evitar la participación del área de desarrollo en la creación de filtros.

BIBLIOGRAFÍA Y ENLACES

- Paperless S.A. Chile: http://www.paperless.cl
- Servicio de Impuestos Internos: http://www.sii.cl
    - Definiciones de esquema de documentos tributarios: http://www.sii.cl/factura_electronica/formato_xml.htm
- Lista de empresas autorizadas por el SII a emitir Documentos Tributarios Electrónicos: http://www.sii.cl/factura_electronica/prov/emp_prov_fe.htm
- Tecnologías ocupadas:
    - Eclipse: http://www.eclipse.org/
    - Java SE: http://www.oracle.com/technetwork/java/javase/overview/index.html
    - Tomcat: http://tomcat.apache.org/
    - HTML: http://www.w3schools.com/tags/default.asp
    - CSS: http://www.w3schools.com/cssref/default.asp
    - JavaScript: http://www.w3schools.com/jsref/default.asp
    - PostgreSQL: http://www.postgresql.org/
    - Subversion: http://subversion.apache.org/
    - Apache ANT: http://ant.apache.org/
    - Apache POI: http://poi.apache.org/
    - Apache Tiles: http://tiles.apache.org/
    - XSD: http://www.w3schools.com/Schema/schema_intro.asp, http://www.w3.org/XML/Schema
    - DOM: http://www.w3.org/DOM/