UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE

“DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN COMPARATIVA DE COMPONENTE DE FACTURACIÓN ELECTRÓNICA”

LEONARDO LUIS SANTIS ARENAS

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: HERNAN ASTUDILLO

NOVIEMBRE- 2016

Agradecimientos

Mis agradecimientos van a mi madre y tía, que fueron incondicionalmente presentes y me brindaron siempre su apoyo, como a la empresa Arteformas por contribuir en la formulación de este trabajo. También a la UTFSM y a sus profesores, que brindaron gran parte de mi formación académica.

Finalmente a mis amigos y a Karla, que alentaron y se preocuparon por mí.

Muchísimas gracias a todos

Resumen

En este trabajo se propondrá un componente de facturación electrónica que pretende asistir el desarrollo de sistemas ERP a medida, poniendo énfasis en aquellos que por su rubro o procesos internos, la compra de un ERP comercial no se apega a su funcionamiento. El desarrollo del componente se encuentra basado en el ecosistema de gemas de Ruby: RubyGems, y es integrado posteriormente a un sistema web basado en el framework del mismo lenguaje Ruby on Rails.

Se propondrá además una rúbrica con la cual se evaluaron componentes de facturación electrónica. A partir de los resultados de la evaluación, se perfilan los beneficios y contras de cada solución, identificando escenarios que favorecen la elección de un componente por sobre el resto.

Palabras clave: Arquitectura basada en componentes, Componentes, Facturación Electrónica, Desarrollo web

Abstract

This component will propose a component of electronic invoicing that pretends to assist the development of tailored EPR systems, putting emphasis in those that because of their area of operation or internal processes, the bought of a commercial ERP system doesn’t fit to their functions. The development of the component is being based around the gem ecosystem of Ruby: RubyGems, and thereafter is integrated to a web system based on the same language framework: Ruby on Rails.

Additionally, a rubric will be proposed and used in the evaluation of components of electronic invoicing. From the results encountered at the evaluation, the benefits and cons of each solution will be profiled, identifying scenarios which favor the choice of one component above the rest.

Keywords: Component Based Architecture, Components, Electronic Invoice, Web Development

Glosario

- Application Program Interface (API): especificación que consiste en una serie de reglas que permite a distintos productos de software comunicarse entre sí.
- Black-Box: hace referencia a la naturaleza de los componentes, que son vistos como una serie de entradas y salidas que estos pueden proveer, sin conocimiento de su funcionamiento interno (como una caja negra).
- Certificado Digital: Documento electrónico generado por un prestador de servicios acreditado en la materia, que permite la autenticación de un contribuyente. Puede ser utilizado en el sitio web del SII o para la emisión y firmas de un DTE.
- Código de Autorización de Folios (CAF): documento entregado por el SII que autoriza al contribuyente la emisión de un rango de DTE de un tipo. La generación de los DTE requiere información del CAF para tener validez.
- Código de Autorización de Libros (CAL): Símil del CAF para los libros de compra y venta electrónicos.
- Contribuyente: Son las personas naturales o jurídicas, o los administradores o tenedores de bienes ajenos afectados por impuestos.
- Contribuyente electrónico: Contribuyente autorizado por el SII para la emisión (y recepción) de documentos tributarios electrónicos.
- Commercial Off The Shelf (COTS): componentes distribuidos comercialmente por un proveedor para su reuso en productos de propiedad de un cliente.
- Computer Line Interface (CLI): modo de interacción en que el usuario envía comando al programa en forma de sucesivas líneas de texto (comandos). Provee una forma de comunicación únicamente basada en input y output textual.
- Documento Tributario: documento emitido que respalda la entrega de bienes, o préstamo de servicios, y que está gravado con tributos.
- Documento Tributario Electrónico (DTE): documento generado y firmado electrónicamente por un emisor electrónico, que produce efectos tributarios y cuyo formato está establecido por el Servicio de Impuestos Internos.
- Enterprise Resources Planning (ERP): sistemas de información que tienen por finalidad el manejo de los recursos económicos de una compañía, en relación a procesos de planificación, producción, inventario, envío, finanzas, entre otros.
- Framework: estructura reusable que permite la construcción de un producto de software, mediante la extensión de un producto genérico diseñado para este propósito.
- In-House: conducir una actividad, operación o mantener un activo al interior de la empresa, sin apoyarse en una entidad externa para esto.
- Model View Controller (MVC): patrón de arquitectura de software que separa un producto de software en tres partes: modelo, vista y controlador.
- On-Demand: servicio disponible al momento que se requiere, y que es cobrado en relación a su frecuencia y/o intensidad de uso.

- ORM: técnica que permite la manipulación de información relacional de una base de datos utilizando un paradigma orientado a objetos.
- PDF417: formato de código de barras bidimensional. Es utilizado en las muestras impresas de los DTE para codificar algunos datos del documento, permitiendo validarlo fuera de línea.
- RUN: Rol Único Nacional (RUN). Identificador similar al RUT que se le otorga a personas naturales. Sirve como identificador ante el Servicio de Impuestos Internos y ante todo otro organismo del Estado.
- RUT: Rol Único Tributario (RUT). Es un identificador único que se asigna a entidades jurídicas una vez dado cuenta del inicio de actividades. Es utilizado universalmente por los organismos del Estado para identificar las entidades.
- Simple Object Access Protocol (SOAP): protocolo para el intercambio de información estructurada en servicios web. Utiliza el formato XML para la codificación de los mensajes.
- SII: Sigla de Servicio de Impuestos Internos. organismo público que realiza la aplicación y fiscalización del pago de impuestos internos en Chile.
- Stakeholder: Actor clave que tiene esta involucrado en el desarrollo de un producto de software, necesitando una solución óptima atingente a sus intereses.
- Universal Description, Discovery and Integration (UDDI): protocolo que permite el descubrimiento de servicios web WSDL mediante el envío de mensajes SOAP.
- Web Services Description Language (WSDL): Formato XML utilizado para la descripción de servicios web. Provee una definición abstracta de los tipos y las operaciones provistas por este, permitiendo ser ligado a un protocolo concreto de red. Puede ser utilizado en conjunción a SOAP o API’s HTTP para poner a disposición un servicio en particular.
- XML (eXtensive Markup Language): Metalenguaje extensible de etiquetas que permite la definición de lenguajes específicos a un dominio para organizar y etiquetar información. Es utilizado ampliamente por el SII para especificar el formato de los DTE, CAF, CAL y otros documentos electrónicos.

Índice

Agradecimientos i
Resumen ii
Abstract ii
Glosario iii
Índice v
Introducción viii
Capítulo 1: Contexto 1
1.1 Definición inicial del problema 1
1.2 Facturación electrónica en Chile 3
1.2.1 Normativa legal previa a obligatoriedad 3
1.2.2 Normativa legal vigente 5
1.2.3 Descripción del sistema 7
1.2.4 Certificado digital 8
1.2.5 Modelo de operación para contribuyentes 10
1.2.6 Proceso de postulación, certificación y autorización, Ambiente de pruebas 12
1.3 Objetivo General 16
1.4 Objetivos secundarios 16
Capítulo 2: Estado del arte 17
2.1 Soluciones de Facturación Electrónica en Sistemas ERP 17
2.2 Componentes de Facturación Electrónica 19
2.3 Otras soluciones existentes 22
Capítulo 3: Propuesta específica 24
3.1 Arquitectura basada en componentes 24
3.1.1 Problemas asociados a Component-Based Architecture 26
3.1.2 Reuso vs Component-Based-Architecture 27
3.2 Propuesta de componente y sistema ERP 28
Capítulo 4: Desarrollo de Componente 33
4.1 Exigencias impuestas por el SII 33
4.1.1 Generación de Documentos Tributarios Electrónicos (DTE) 33

4.1.2 Envío automático de DTE's al SII 39
4.1.3 Consulta de envío de los DTE 40
4.1.4 Intercambio de DTE entre contribuyentes 40
4.1.5 Generación de copias impresas de los DTE 41
4.1.6 Generación y envío de información de compras y ventas 44
4.1.7 Herramientas de generación de pruebas para certificación 45
4.2 Especificaciones de Sistema ERP 46
4.2.1 Descripción General del Sistema ERP 46
4.2.2 Modelo de datos 47
4.3 Implementación de componente y sistema ERP 47
4.3.1 Estructura de aplicación Rails 47
4.3.2 Estructura de aplicación a desarrollar 50
4.3.3 Gemas a utilizar 50
4.3.4 Uso de engines en la aplicación 51
4.3.5 Implementación del componente 55
Capitulo 5: Evaluación y análisis comparativo de componentes 57
5.1 Selección de Componentes 57
5.1.1 LibFacturista - FacturaElectronicaChile.com 57
5.1.2 LibreDTE-lib - LibreDTE 58
5.1.3 Facturación_Electronica 58
5.2 Modelos de calidad 58
5.2.1 Primeros modelos de calidad 58
5.2.2 Modelos de calidad para COTS 61
5.3 Criterios de Evaluación 65
5.3.1 Atributos de calidad 65
5.3.2 Cobertura funcional 73
5.4 Evaluación de componentes 74
5.4.1 Resultados de "Atributos de calidad" 74
5.4.2 Resultados de "Cobertura funcional" 85
Capítulo 6: Validación 87
6.1 Plan de transición y despliegue 88
6.1.1 Transición del sistema ERP 88

6.1.2 Despliegue de sistema ERP 88
6.2 Identificación de escenarios 91
6.2.1 Discusión de resultados 91
6.2.2 Caracterización de componentes 93
Capitulo 7: Conclusiones 95
Referencias 98
Anexo 100
A - Casos de uso de sistema ERP 100
Iniciar Sesión 100
Administrar Cuenta 100
Administrar Vendedores 100
Administrar Clientes 101
Administrar Láminas 102
Administrar Artículos 102
Emitir o actualizar una guía de venta 103
Emitir una factura a partir de una guía de venta 104
Emitir una factura electrónica a partir de una factura 105
Obtener historial de un cliente 105
Ingresar cheque abonado por el cliente 105
Administrar estados de guía de venta 106
Administrar recargos y descuentos 106
Administrar bancos y plazas 107
Imprimir o enviar por correo guía de venta 108
Imprimir lista de láminas 108
Imprimir factura electrónica 108
B - Esquemas de modelos de datos 110

Introducción

Debido a la creciente necesidad generalizada por parte de los contribuyentes a implementar una solución de facturación electrónica en la empresa, se han abierto una serie de problemáticas que estas deben enfrentar. Las empresas deberán adquirir nuevos productos de software que operen en concordancia con el modelo operacional del SII.

Dependiendo del rubro, los sistemas ERP con facturación electrónica propuestas en el mercado podrían no ajustarse demasiado a las necesidades del cliente, y en este caso, una solución ajustada a medida deberá ser utilizada.

Este trabajo tiene como finalidad presentar una alternativa de facturación electrónica a los contribuyentes y desarrolladores de soluciones, así como otorgar una guía de las obligaciones a cumplir por parte de un software de facturación electrónica, y los procesos que el contribuyente deberá integrar en su empresa. Como información adicional, este informe entregará un criterio de evaluación de un tipo específico de soluciones de facturación electrónica, con el fin de asistir a usuarios desarrolladores que necesiten integrar estas funcionalidades a un sistema ERP completo, o un símil.

El presente informe comienza con una definición del contexto de la problemática específica a tratar junto la normativa legal vigente en la materia en el capítulo 1, seguido de los objetivos a cumplir en este trabajo. El capítulo 2 cubre las propuestas existentes en el mercado, de índole general y las específicas a tratar. El capítulo 3 da el diseño de la propuesta específica a desarrollar, la cual es elaborada en el capitulo 4, capitulo que otorga las bases técnicas de la solución. El capitulo 5 contiene la rúbrica de evaluación propuesta, junto con la evaluación de los candidatos. Luego en el capítulo 6 se discuten los resultados encontrados, y se establecen algunas conclusiones a partir de estos. Finalmente, se da paso a las conclusiones generales de este trabajo, junto a una síntesis de los puntos relevantes encontrados en la realización de este.

Capítulo 1: Contexto

A modo de facilitar la comprensión de este trabajo, esta sección se propone entregar una definición inicial del problema, la cual es detallada posteriormente con la explicación de las bases legales que rigen la materia y los procesos que el SII demanda a los contribuyentes. Esta base teórica permitirá entender el dominio del problema y posteriormente, los detalles técnicos de la solución propuesta.

1.1 Definición inicial del problema

Las tecnologías de información (TI) ejercen roles bastante importantes en la sociedad actual. Aplicaciones y servicios permiten agilizar algunos aspectos de nuestras vidas y resolver problemas que antes no eran posibles o viables. Hoy en día podemos observar diversas aplicaciones de las TI en salud, entretenimiento, comunicación, análisis científico, administración de recursos, entre otros.

La organización moderna también hace uso de estas tecnologías. Pudiendo controlar de forma automatizada como se llevan a cabo sus procesos de manufacturación, inventario, ventas, finanzas, etc. El uso apropiado de estas otorga una base competitiva, con la cual una organización puede destacar respecto a la competencia y mantenerse en el tiempo.

Organizaciones gubernamentales están empujando a que las empresas adopten tecnologías de información: el Servicio de Impuestos Internos (SII), con fin de agilizar sus procesos, disminuir la evasión e impulsar el comercio utilizando tecnologías de información, definió un modelo operacional para el uso de documentos tributarios electrónicos (DTE) [1]. En el sistema propuesto, los contribuyentes pueden generar, firmar, timbrar y transmitir distintos documentos tributarios, los cuales anteriormente debían ser manejados en folios y manualmente inspeccionados por el SII. El 31 de enero del 2014 entró en vigencia la ley 20.727, la cual obliga a organizaciones contribuyentes a utilizar ciertos DTE (facturas de venta, facturas de venta y compra, liquidaciones de factura y notas de debito y crédito), remplazando sus contrapartes en papel, según el tamaño de las empresas (medido en ingresos anuales), la ley define plazos límite para apegarse a la normativa.

Para la integración con los modelos de operación basados en DTE, SII provee diversas alternativas para los contribuyentes, estas son:

- Utilización del portal web MiPyme, que permite a empresas generar DTE's
- Utilización de sistema de facturación del mercado
- Utilización de sistema de facturación de elaboración propia.

En relación a las últimas dos soluciones, el SII define un proceso de postulación y certificación en donde los contribuyentes deben cumplir una serie de requisitos para obtener la autorización por parte del SII para emitir los DTE.

El problema surge a partir de la adopción de una solución indicada de facturación electrónica. Existen varias razones por las cuales las dos primeras alternativas no son las más apropiadas: El tamaño de la empresa obliga a utilización de un software de mercado/propio, el software del mercado resulta caro y disminuye las utilidades de la empresa, es difícil integrar los software existentes en el mercado a las operaciones de venta de una empresa, o ya se cuenta con un software de ventas, que puede ser adaptado para cumplir la normativa.

Para estas situaciones, una construcción (o adaptación) de software a medida permitiría: abaratar los costos (requiriendo el producto mínimo para operar) y apegarse mejor a las operaciones de una empresa.

Excluyendo las lógicas de cada organización en solución en particular, todas tienen que cumplir con el estándar impuesto por el SII para la emisión de DTE de manera uniforme. Sería deseable de este modo, la existencia de un software que se dedicara exclusivamente a dicho propósito, y que pueda ser integrado con facilidad a un sistema de ventas, encargándose éste exclusivamente de aquellas responsabilidades.

El desarrollo de software basado en componentes propone que un software informático puede ser construido mediante reutilización de piezas de software que realizan un grupo reducido de tareas del total que se desea construir [2]. Estas piezas se denominan componentes y en especial, si estos se encuentran disponibles en el mercado para su reuso, son componentes COTS (sigla en inglés de “Comercial off the shelf”).

Existiendo algunos componentes que prometen cumplir la funcionalidad deseada, se producen las siguientes interrogantes a la hora de elegir una alternativa: ¿Cuál componente de facturación conviene utilizar para mi empresa?, ¿Si un cliente requiere facturación electrónica en su sistema, cuál sería el más apropiado acorde a sus necesidades?, ¿Cuál sería el más acorde a mi como empresa de desarrollo del software, de integrar a un producto para cumplir con la calidad, tiempo y costos asignados?

De esta forma, el trabajo de esta memoria define sus objetivos en la creación de un componente de facturación electrónica y otros DTE’s enfocándose en su reusabilidad, y la realización de una evaluación comparativa con otros componentes existentes en el mercado, que permita a empresas de desarrollo de software y clientes, tomar la mejor decisión a la hora de adoptar la facturación electrónica en un sistema a medida. Adicionalmente se propone la integración del componente a diseñar con un sistema de ventas de prueba, dando especificaciones de cómo se llevaría a cabo su implementación y puesta en marcha.

1.2 Facturación electrónica en Chile

La facturación electrónica en Chile se gestó de manera concreta el año 2003, cuando el SII inició el proceso de masificación del sistema de facturación electrónica para todos los contribuyentes que solicitaron voluntariamente su uso mediante una postulación. Esta postulación define requisitos a cumplir, de tal manera que un contribuyente puede obtener una resolución por parte del SII, que le permita ser habilitado como emisor y receptor de documentos tributarios electrónicos. En adición a este proceso de postulación, el SII también define un sistema de facturación electrónica gratuito, el cual queda de libre uso para los contribuyentes que lo deseen.

Esta sección define sus contenidos en los subcapítulos 1 y 2 como un resumen de la normativa legal previa y las modificaciones de la vigente actualmente. Luego se pretende otorgar una descripción de la operación básica del sistema propuesto por el SII, el cual los contribuyentes deben adoptar. El subcapítulo 4 explicará lo que dentro del modelo de funcionamiento se entiende por certificado digital. El subcapítulo 5 se dedicará a explicar el modelo para los contribuyentes de emisión y recepción de documentos tributarios electrónicos propuestos por el SII. Finalmente, el subcapítulo 6 explicará las fases del proceso de postulación para certificarse como contribuyente de documentos tributarios electrónicos.

1.2.1 Normativa legal previa a obligatoriedad

Anterior a la ley actual de facturación electrónica (ley N° 20.727), la ley sobre impuesto a las ventas y servicios definió las obligaciones de los contribuyentes afectos a IVA, los cuales debían emitir ciertos documentos tributarios de manera física, consisten en formularios previamente timbrados.

Este timbrado proveniente del SII autoriza a los contribuyentes a emitir el documento tributario que respalda cierta operación comercial, otorgando validez a los documentos. En caso de su ausencia, o de omitir la emisión de los documentos (realizar operaciones comerciales que requieran documento tributario sin emitir este), el contribuyente se arriesga a grandes multas o su cancelación del permiso para operar [3].

Los documentos tributarios que deben ser emitidos físicamente, son nombrados principalmente del artículo 53 de la misma ley, y son los siguientes:

- Facturas o boletas por ventas o servicios afectos a IVA
- Facturas o boletas por operaciones no afectas o exentas que realicen con otros contribuyentes de IVA
- Factura de compra por operaciones afecta, en el caso de que el vendedor no tenga la calidad para emitir facturas de ventas.

- Guía de despacho en caso de traslado de bienes importados asociados a una factura, la cual es postergada
- Notas de Crédito y/o Débito por operaciones afectas
- Liquidación y Liquidación factura

La legislación en el artículo 56, otorgaba la facultad al Servicio de Impuestos Internos el intercambio de mensajes mediante el uso de diferentes sistemas tecnológicos, en remplazo de la emisión de los documentos físicos, exigiendo como requisito que los intereses fiscales sean debidamente resguardados [4].

Con la facultad legal que se le otorga al SII, y habiendo iniciado pruebas piloto el año 2002 con 8 empresas (Agrosuper, Embotelladora Andina, Entel PCS, Ideal, Montecarlo, NIC Chile de la Universidad de Chile, Sodimac y Telefónica Móvil), el SII abre públicamente el 2 de septiembre del 2003 el proceso de masificación de la facturación electrónica, poniendo a disposición del proceso de postulación de manera pública todos los contribuyentes que deseen adoptar el modelo de facturación electrónica[1].

En este estado, el SII puede facultar a los contribuyentes a facturar electrónicamente sus documentos tributarios, sin embargo, estos quedan en libertad de seguir emitiendo físicamente trabajando con dos series de timbrado: uno para documentos físicos y otro para los electrónicos, a modo que la numeración de los documentos nunca se intercepta. Esto no los excluye de la recepción de documentos tributarios electrónicos, la cual pasa a ser una obligatoriedad por parte de los contribuyentes.

Postular al proceso de facturación electrónica supone contar con un software de mercado o propio con las facultades necesarias para operar, por lo que genera una barrera de entrada importante a los contribuyentes de tamaño pequeño, los cuales no cuentan con los recursos económicos necesarios para implementar una solución completa.

En vista a este problema, el SII abre el año 2005 el portal MiPyme, el cual es un portal web diseñado pensando en las pequeñas y medianas empresas que desean facturar electrónicamente sin la necesidad de certificarse comprando un software de mercado o desarrollando una solución personalizada. La legislación hace posible su uso el 1 de septiembre del 2005, en la resolución exenta SII N°86, en el cual, dado que "el sistema de facturación electrónica ha demostrado ser eficiente, seguro, facilitador del cumplimiento tributario y redundar en la obtención de importantes beneficios económicos por parte de los contribuyentes usuarios de este sistema", otorga la autorización a contribuyentes que se inscriban en el portal MiPyme para ser emisores de documentos tributarios electrónicos, a aquellos que cumplan con una serie de requisitos.

---

[1] Servicio de Impuestos Internos. SII inicia masificación de la factura electrónica [en línea] <http://www.sii.cl/pagina/actualizada/noticias/2003/020903noti01jo.htm> [consulta: 26 octubre 2016].

La resolución ha sido modificada posteriormente en el año 2006 (Resolución exenta N°93 y N°124) las cuales modifican la barrera de requerimientos en las cuales la utilización del portal MiPyme se enmarca. Estas barreras son posteriormente removidas en el año 2014 en la Resolución exenta N°99 en vista de la obligatoriedad propuesta por la Ley 20.727, derogando las resoluciones anteriores y dejando solo aquellas condiciones que verifiquen el inicio de actividades por parte del contribuyente y que este quede clasificado como contribuyente de primera categoría.

1.2.2 Normativa legal vigente

La facturación electrónica trae numerosos beneficios en costos para los contribuyentes, un mejor control de la evasión y procesos tributarios para el SII. Para generalizar y masificar el uso de facturación electrónica, el 31 de enero de 2014, es publicada la ley 20.727, la cual introduce modificaciones en la legislación tributaria. Uno de sus cambios es la exigencia a los contribuyentes el uso de sistemas tecnológicos para la emisión y recepción de DTE de manera obligatoria, en remplazo del documento físico.

En particular, los documentos afectos por la obligatoriedad de la normativa son: facturas, factura de compra, liquidaciones facturas, notas de débito y notas de crédito. Respecto a las guías de despacho y boletas de ventas y servicios, la ley no obliga a los contribuyentes a emitirse electrónicamente, sin embargo, ellos podrán optar a emitirlos mediante una solicitud al SII. Actualmente, el SII solo entrega la autorización para emitir boletas en caso de contribuyentes que sean proveedores de servicios periódicos.

La ley define un plazo antes de que la medida opere efectivamente de dos años, estableciendo un periodo de transición que incluye la aplicación gradual de la obligación legal. Durante dicho periodo se espera que en el mediano plazo la totalidad de los contribuyentes estén operando con documentos tributarios electrónicos, lo cual irá aparejado de los importantes beneficios anteriormente mencionados para los contribuyentes y el mismo SII. La disposición transitoria propuesta por la ley obliga a empresas según su tamaño y ubicación, a adoptar la facturación electrónica en distintos plazos. Los tiempos propios de cada tipo de empresa quedan detallados en la figura 1.1.

Calendario según tamaño de empresas			
Tamaño	Ingresos anuales por ventas y servicios en el último año calendario	Ubicación	Fecha
Grandes	Mayor a 100.000 UF	Todas	1 de noviembre de 2014
Medianas y Pequeñas	Mayor a 2.400 UF y menor o igual a 100.000 UF	Urbana	1 de agosto de 2016
		Rural	1 de febrero de 2017
Microempresas	Menor a 2.400 UF	Urbana	1 de febrero de 2017
		Rural	1 de febrero de 2018

Figura 1.1: Calendario de obligatoriedad según tamaño y ubicación de empresas definido por el SII[2].

La obligatoriedad de la normativa afecta a todos los contribuyentes, sin embargo, existen casos excepcionales en los que se permite seguir emitiendo documentos tributarios con timbrado físicos, estos casos son:

- Contribuyente opera en una zona sin cobertura de datos móviles o fijos de operadores de telecomunicaciones que tienen infraestructura.
- Contribuyente opera en una zona sin cobertura de suministro eléctrico.
- Contribuyente opera en zona declarada como catástrofe conforme a la ley Nº 16.282.
- Otras causales que el SII estime.

En dichos casos, el SII debe enviar una resolución identificando el grupo de contribuyentes afectados por dicha situación, previo estudio de los organismos técnicos autorizados en la materia. La resolución debe indicar además los plazos en la medida opere en vigencia.

También los contribuyentes pueden solicitar al SII una petición de excepción a la emisión de DTE. En este caso se otorga al SII 30 días para estudiar la resolución y dar una respuesta, entendiéndose transcurrido ese plazo sin respuesta, como solución aceptada. Durante el plazo de espera, el SII deberá otorgar al contribuyente los documentos timbrados físicamente necesarios para operar.

[2] Servicio de Impuestos Internos. Aspectos Generales de la nueva Ley de Factura Electrónica (Ley 20.727 de 2014) [en línea] <http://www.sii.cl/factura_electronica/ley/ley_fe_20727.htm> [consulta: 26 octubre 2016].

1.2.3 Descripción del sistema

El sistema propuesto por el SII, permite a los contribuyentes que se encuentren autorizados para emitir DTE mediante la certificación que lo avala, y poder realizar transacciones comerciales que requieran el apoyo tributario con la utilización de estos documentos.

Similar a su contraparte física, DTE deben ser firmados y asignados a un contribuyente por parte del SII. El contribuyente podrá solicitar una serie de folios a través del sitio web del SII, de manera que con la utilización de estos folios (código de autorización de folios), podrá generar DTE válidos que cuenta con la autorización del SII. La generación por parte del emisor requiere ser firmada por medio de su certificado digital.

Al momento de realizar una transacción comercial que requiera la emisión de un DTE, este deberá ser generado en conformidad con su estructura y transmitido al SII antes de la recepción de bienes por parte del destinatario o su transporte. El emisor adicionalmente, deberá hacer envío de este DTE al receptor vía correo electrónico, en caso de que esté igualmente sea un contribuyente electrónicamente autorizado, si este no fuese el caso, el emisor deberá hacer envío de una copia en papel.

De manera similar, el contribuyente deberá ser capaz también de recibir DTE de otros contribuyentes a su casilla electrónica, los cuales deberán ser revisados y respondidos, enviando mensajes de “acuse de recibo” al emisor del envío.

1.2.3.1 Formato de los DTE

Cada DTE define su información mediante el metalenguaje XML, información que debe estar correctamente estructurada de acuerdo a las especificaciones propuestas por el SII. Estas se encargan de definir la obligatoriedad de que campos deben estar anexados en el, dependiendo del tipo de DTE que se está emitiendo.

Todos los documentos deben contener un campo con un timbre electrónico para seguimiento de su estado y que los fiscalizadores verifiquen la validez del documento fuera de línea, la cual también puede ser verificada por los contribuyentes a través de una opción en el sitio web del SII. El timbre electrónico consiste en una firma con los datos relevantes del DTE junto con el código de autorización de folios (CAF) el cual debe ser solicitado previamente por el contribuyente en el sitio web del SII. En el caso de documentos físicos, el timbre electrónico es expresado mediante un código de barras bidimensional.

Adicional al timbre, todos los documentos deben ser firmados electrónicamente. Las firmas son generadas por la llave privada de un certificado digital que una de las empresas autorizadas concede al contribuyente. El detalle de este certificado será explicado en detalle en la sección correspondiente a “Certificado digital”.

1.2.3.2 Almacenamiento de los DTE

Se requiere que los contribuyentes electrónicos almacenen los DTE de forma segura para revisiones que el SII pueda exigir a los contribuyentes. El documento físico deja de tener importancia para las revisiones y no es necesario su almacenamiento. En el caso del contribuyente manual, la obligación de almacenar físicamente todos los documentos tributarios que este recibe queda sin cambios.

1.2.4 Certificado digital

El certificado digital es utilizado en la generación de un DTE, logrando identificar una persona natural o jurídica como tal. Es obligatorio para emitir un DTE válido, y permite generar confianza de que el documento fue firmado electrónicamente en conformidad por su emisor.

Para entender el comportamiento de un certificado digital, es necesario comprender el estado de la legislación chilena sobre la materia. La norma N°19.799 "Ley sobre documentos electrónicos, firma electrónica y servicios de certificación de dicha firma", define jurídicamente la firma electrónica como: "Cualquier sonido, símbolo o proceso electrónico, que permite al receptor de un documento electrónico identificar al menos formalmente a su autor", siendo el DTE una subcategoría de documento electrónico.

La jurisdicción también define la firma electrónica avanzada, la cual es "Aquella certificada por un prestador acreditado, que ha sido creada usando medios que el titular mantiene bajo su exclusivo control, de manera que se vincule únicamente al mismo y a los datos a los que se refiere, permitiendo la detección posterior de cualquier modificación, verificando la identidad del titular e impidiendo que desconozca la integridad del documento y su autoría". Para el modelo operacional propuesto por el SII, ambas firmas electrónicas aplican, siendo dispuestas por los prestadores acreditados en la materia.

El prestador acreditado se encarga de almacenar las identificaciones de los titulares de manera segura, a modo que las identificaciones de estos quedan a acceso de las partes que lo requieran y de esta forma verificar su autenticidad. Para impedir que el documento sea modificado (conservar su integridad), verificar la identidad del titular de manera inequívoca e irrenunciable (autentificación sin repudio) y garantizar la confidencialidad del envío de los DTE, es necesario acudir a sistemas de encriptación de mensajes.

Un sistema de encriptación permite la codificación (ofuscación) de información mediante un algoritmo de encriptación a modo de garantizar el envío confidencial de la información. El algoritmo se dedica a transformar la información con el propósito de codificar o descodificar el mensaje. Para codificar un mensaje, es necesario contar con una llave. Dependiendo si el

algoritmo es asimétrico o simétrico, se define si el receptor debería utilizar la misma llave para decodificar el mensaje.

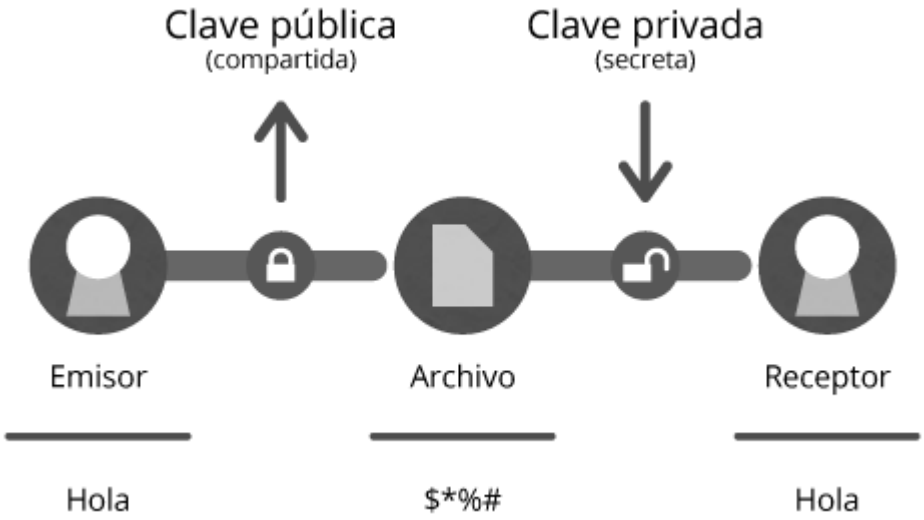

Figura 1.2: Esquema de encriptación de mensaje con clave pública en criptografía con llaves asimétricas[3].

En particular, para el caso de los DTE, el algoritmo de encriptación necesario a utilizar es de tipo asimétrico. Los algoritmos asimétricos consisten en la generación vinculada de dos llaves, la llave pública que el receptor adquiere, y la llave privada, la cual es resguardada por el emisor. Este tipo de encriptación cumple con la propiedad de que los mensajes encriptados con la llave pública sólo pueden ser encriptados con la privada y viceversa, además de que un mensaje encriptado con la llave pública no puede ser desencriptado con esta. La figura 1.2 muestra como un mensaje puede ser encriptado y desencriptado utilizando llaves asimétricas.

De esta forma los DTE son enviados utilizando encriptación asimétrica. A modo de garantizar su integridad, es necesario generar un "fingerprint" del documento, el cual es un hash del documento completo cifrado con la llave privada del emisor. Las llaves públicas de los contribuyentes son almacenadas en los certificados digitales, las cuales quedan anexas y suscritas a estos, a modo que se pueda probar la autenticidad de los DTE que estos emiten de forma innegable y sin repudio. Los repositorios en los cuales se almacenan, y se generan los certificados junto con las llaves privadas corresponden a los prestadores acreditados.

En la actualidad, los prestadores acreditados corresponden a empresas particulares los cuales generan y almacenan los certificados públicos de las firmas electrónicas avanzadas. El

[3] GENBETA DEV. Tipos de criptografía: simétrica, asimétrica e hibrida [en línea] <http://www.genbetadev.com/seguridad-informatica/tipos-de-criptografia-simetrica-asimetrica-e-hibrida> [consulta: 26 octubre 2016].

organismo el cual certifica la calidad de los prestadores es la Subsecretaría de Economía. Las empresas que actualmente se encuentran acreditadas para prestar sus servicios según el sitio web de SII son:

- Acepta.com
- E-CertChile
- E-Sign
- Certinet

1.2.5 Modelo de operación para contribuyentes

El Modelo de operación propuesto es que las empresas contribuyentes deben adoptar para emitir DTE y lograr pasar las pruebas del proceso de postulación. Los pasos propuestos quedan descritos en el "Instructivo Técnico" y son los siguientes:

- Enrolamiento y Autorización de los Firmantes: El SII en su sitio, permite que empresas se enrolen para se contribuyentes electrónicos. Este proceso designará el usuario administrador de la empresa, además de definir el correo electrónico bajo el cual el contribuyente recibirá DTE de otros contribuyentes electrónicos. El usuario administrador debe ser un representante legal del contribuyente, el cual podrá designar los firmantes autorizados de la empresa. Los firmantes autorizados en conjunto con el usuario administrador, serán los encargados realizar tareas en representación del contribuyente de forma exclusiva:

    - Solicitar folios, los cuales permiten a la empresa generar un número de DTE válidos.
    - Anulación de folios previamente solicitados
    - Firmar DTE
    - Enviar DTE emitidos al sitio del SII

El usuario administrador asignará a los firmantes autorizados y podrá otorgar o restringir el acceso a las acciones previamente mencionadas, es fundamental que la empresa adquiera un certificado digital para cada firmante autorizado que designe.

- Obtención de rango de folios y código de autorización de folios (CAF): Un firmante autorizado por la empresa podrá solicitar un rango de folios para emitir DTE válidos, el SII responderá con el CAF, el cual corresponde a un fichero XML que debe ser insertado en cada DTE del rango de folios, y que además es utilizado para obtener el timbre electrónico a la hora de emitir un DTE.

- Alimentar sistema con rango de folios y asignación de identificación a cada DTE: El contribuyente es el encargado de llevar la numeración que le da a sus documentos a partir de

los rangos de folio que ha solicitado, de modo que cada DTE utilice solo un identificador y que cada identificador sea anexado al CAF que lo generó, por lo que el sistema del contribuyente debe ser capaz de realizar dichos procesos.

- Generar DTE en formato XML en conformidad con lo exigido por el SII: Cada DTE deberá cumplir según su tipo con una estructura obligatoria de campos que lo representan, bajo un esquema XML.

- Calcular timbre electrónico de cada documento: El timbre electrónico consiste en una representación de datos relevantes del documento junto con el CAF obtenido por el SII. La firma del timbre electrónico debe ser efectuada por la llave privada asociada al CAF bajo el cual se desea numerar el documento, junto con el rango de folios correspondiente.

- Firmar documento completo: El DTE luego de ser completado y timbrado electrónicamente debe ser firmado por un certificado digital vigente dado por una de las empresas acreditadoras en la materia.

- Establecer un proceso de impresión de documentos en conformidad con lo exigido por el SII: Para aquellos contribuyentes que operen de forma manual, los documentos deberán ser enviados en forma física mediante una representación impresa de estos. La representación impresa deberá considerar cierto formato, el cual incluye la representación del timbre electrónico mediante un código de barras bidimensional, bajo el formato estándar PDF417.

- Implementar un proceso de intercambio de DTE con otros contribuyentes adecuado: Se deberá habilitar como mínimo una casilla electrónica en la cual otros contribuyentes hagan el envío de una copia de los DTE emitidos, así como los comprobantes de recepción o rechazo de los DTE que el contribuyente envía. Los comprobantes de recepción o rechazo son obligatorios de enviar al contribuyente receptor de un DTE y cumplen un formato XML similar al de los DTE. Se debe considerar también la opción de implementar la generación y envío automatizado de estos comprobantes.

De forma adicional a los propuestos, se recomienda que el contribuyente se haga cargo de las siguientes responsabilidades:

- Definir un procedimiento de respaldo y recuperación de DTE: Los DTE enviados al SII quedan únicamente a disposición interna del SII. En caso que un inspector lo solicite o para realizar una consulta, el contribuyente puede considerar pertinente la forma de respaldar de forma segura los DTE emitidos y recibidos.

- Diseñar la generación de la información mensual de compra y venta: Es obligación de los contribuyentes electrónicos enviar mensualmente la información de compra y venta

(IECV) del periodo, en un formato similar al de los DTE. Se debe considerar un proceso automatizado que permita con la información disponible, generar los IECV.

- Llevar un control apropiado de las respuestas de los documentos enviados: El envío de documentos al SII, dependiendo de la conformancia del documento puede ser aceptado, rechazado o aceptado parcialmente (con reparos). Para administrar las contingencias, es necesario llevar un control apropiado de cada uno de estos casos.

1.2.6 Proceso de postulación, certificación y autorización, Ambiente de pruebas

El contribuyente puede postular en el sitio web del SII para ser aprobado como contribuyente electrónico con la capacidad de enviar DTE. Los requisitos que un contribuyente debe cumplir para poder postular son: haber dado inicio de actividades y que este quede clasificado como contribuyente de primera categoría, en el caso de documentos afectos a IVA (como en el caso de la factura electrónica), se requiere adicionalmente estar catalogado como contribuyente de IVA y validar que mantiene actividades en terreno, o estar en proceso de validación de estas.

El proceso de postulación puede ser iniciado por un representante legal de la empresa contribuyente que cumpla con las condiciones anteriormente señaladas y que el representante cuente con un certificado digital válido. El SII revisa la postulación y sitúa al contribuyente en el ambiente de pruebas, designando al representante legal como el usuario administrador del contribuyente[4].

En este ambiente de pruebas, el contribuyente puede certificarse para ser admitido por el SII para operar como contribuyente electrónico, para esto, el proceso de certificación considera 4 fases de pruebas que verifican la capacidad para llevar a cabo las tareas descritas en el modelo de operación para contribuyentes[5], estas fases son:

1. Envío de set de pruebas
2. Envío de set de simulación
3. Intercambio de información
4. Envío de documentos impresos

---

[4] Servicio de Impuestos Internos. PROCEDIMIENTO DE POSTULACIÓN, CERTIFICACIÓN Y AUTORIZACIÓN [en línea] <http://www.sii.cl/factura_electronica/factura_mercado/proc_postulacion.htm> [consulta: 26 octubre 2016].

[5] Servicio de Impuestos Internos. MANUAL PARA EMPRESAS USUARIAS [en línea] <http://www.sii.cl/factura_electronica/factura_mercado/manual_certificacion.pdf> [consulta: 26 octubre 2016].

El contribuyente que desee completar el proceso de certificación debe completar cada una de las pruebas sin objeciones por parte del SII. En adición a estas, para finalizar el proceso es necesario emitir una declaración en la cual el contribuyente declara que cumple con los requisitos mediante un soporte computacional adecuado. Cabe destacar que el proceso de postulación permite al contribuyente operar bajo un ambiente de pruebas, por lo que la implementación de los procesos que son exigidos al contribuyente en cada una de las fases, puede ser llevado de manera incremental e incluso posterior al inicio de la postulación.

A continuación se presenta un resumen de lo requerido en cada una de las fases de pruebas, y la posterior declaración y autorización del contribuyente, a partir de lo que el SII expone en el "Manual de certificación"[6]:

1.2.6.1 Envío de set de pruebas

Una vez aceptada la postulación, el SII otorga al contribuyente la posibilidad de generar un set de pruebas desde el ambiente de postulación, el cual consiste en una serie de datos ficticios que el contribuyente debe emitir en DTE's al SII. El set de pruebas considera un set de documentos tributarios a enviar que por lo menos incluye factura electrónica, nota de crédito electrónica y nota de débito electrónica. Como requerimiento adicional, se deben enviar libros de compra y venta, los cuales deben ser construidos con la información dada en los DTE (en el caso del libro de venta), y con información provista por el SII (en el caso del libro de compra),

Los datos del set de pruebas deben ser enviados al SII exactamente como se le fueron presentados y siguiendo las consideraciones de confección que el SII dispone, en un plazo máximo de 2 meses posterior a la generación del set de pruebas.

Los documentos pueden ser comprobados dentro de ese plazo en el ambiente de postulación las veces que se estime conveniente, hasta que estos se encuentren correctamente formulados sin reparos, cuando eso se cumpla, el contribuyente podrá declarar su avance de la postulación desde la web del ambiente de certificación.

El SII revisará los DTE generados a partir del set de pruebas, verificando la validez de estos respecto al último set de pruebas generado.

---

[6] Servicio de Impuestos Internos. MANUAL PARA EMPRESAS USUARIAS [en línea] <http://www.sii.cl/factura_electronica/factura_mercado/manual_certificacion.pdf> [consulta: 26 octubre 2016].

En caso de rechazo, el contribuyente podrá reintentar el proceso, siempre que el envío esté dentro del plazo de 2 meses posterior a la generación del set de pruebas, en caso contrario, el contribuyente deberá generar un nuevo set de pruebas y confeccionar los DTE a partir de este.

Si el SII acepta el avance de postulación, el contribuyente podrá iniciar la fase 2 del proceso.

1.2.6.2 Envío de set de simulación

En esta fase se exige al contribuyente a emitir DTE al SII de facturas electrónicas con datos representativos y reales de su rubro. El contribuyente deberá generar DTE respectivos a datos de su facturación de los dos últimos meses, con un máximo de 100 documentos. En caso de empresas con mayor flujo de información, los documentos pueden corresponder solo a un mes, y en el caso de empresas con bajo volumen, los documentos pueden ser de más de dos meses, pero cumpliendo un mínimo de 10 documentos. El SII verificará que el volumen de DTE enviados sea similar al volumen histórico de documentos emitidos por el contribuyente.

Cuando el contribuyente esté conforme con su envío y este haya sido aceptado sin reparos, podrá declarar su avance de la postulación, el cual será revisado por el SII al igual que la fase anterior. Una vez que este sea aprobado, el contribuyente podrá iniciar la siguiente fase del proceso.

1.2.6.3 Intercambio de información

Esta fase verifica que el contribuyente posea la capacidad de hacer recepción de DTE recibidos por otros contribuyentes y que responda correctamente con acuse de recibo o rechazo de acuerdo a las definiciones dadas por el SII.

El SII desde una casilla de correo especialmente habilitada para el proceso, enviará DTE al contribuyente a la casilla de correo que esté asignó al momento de la postulación. Los DTE deberán ser validados respecto al esquema XML específico de cada tipo de documento. Este verificará la respuesta que el contribuyente realiza para cada uno de los DTE enviados en esta fase, analizando su consistencia y correctitud en cada caso. Una vez que estas pruebas ocurran y sea su correctitud verificada, el contribuyente podrá hacer inicio de la siguiente fase del proceso.

1.2.6.4 Envío de documentos impresos

Esta fase requiere al contribuyente la capacidad de generar documentos para ser impresos en el formato acordado en el SII, el cual considera la impresión de un código de barras bidimensional con la información del timbre electrónico (en el estándar PDF417).

Se necesita que el contribuyente envíe, a una casilla de correo especialmente designada para el envío de estos, las imágenes generadas a partir de cada uno de los documentos generados desde el set de pruebas, además de 10, los cuales deberán ser representativos de los tipos de documentos que el contribuyente será autorizado a emitir.

Las imágenes deben ser enviadas a la casilla en un único documento PDF, el cual será revisado por el SII. Una vez que este sea verificado, el SII declara la completitud del contribuyente en todas las fases de pruebas del proceso de certificación, y que este se encuentra listo para dar la declaración de cumplimiento de requisitos y finalizar el proceso.

1.2.6.5 Declaración de cumplimiento de requisitos

El SII habilita la opción de otorgar una declaración en el cual el contribuyente, a través del representante legal el cual está llevando a cabo el proceso de postulación. Dicha declaración señala obliga a cumplir las declaraciones juradas que el SII emita respecto a la facturación electrónica, que conoce las obligaciones que emanan de la autorización como contribuyente electrónico y que además cuenta, con procesos establecidos y controlados, formas de tratar procedimientos que el SII considera críticos. Estos son los siguientes[7]:

- Gestión de CAF's (almacenamiento y control de acceso).
- Foliación controlada (asignación única de cada folio autorizado por el SII).
- Respaldo de los documentos e información generada.
- Envío de documentos al SII
- Intercambio (envío y recepción) de documentos con otros contribuyentes
- Cuadratura de envíos aceptados, rechazados y aceptados con reparos por el SII
- Administración de contingencias.

1.2.6.6 Autorización al contribuyente

Una vez que el contribuyente emita la declaración de cumplimiento de requisitos, y con todas las fases de certificación aprobadas, el SII finalizara el proceso de postulación, emitiendo una resolución que autoriza al contribuyente a emitir DTE y este pasará al ambiente de producción, en el cual podrá comenzar a generar DTE legalmente válidos a partir del periodo tributario indicado en la resolución.

---

[7] Servicio de Impuestos Internos. MANUAL PARA EMPRESAS USUARIAS [en línea] <http://www.sii.cl/factura_electronica/factura_mercado/manual_certificacion.pdf> [consulta: 26 octubre 2016].

1.3 Objetivo General

Diseñar e Implementar un componente reusable de facturación electrónica, en conformancia con el modelo de operación del SII, a fin de poder integrarlo a un sistema ERP a medida.

1.4 Objetivos secundarios

- Realizar una evaluación comparativa de componentes de facturación electrónica existentes en el mercado con el componente a implementar, utilizando modelos de calidad para componentes.

- Integrar componente de facturación electrónica a sistema web ERP.

- Diseñar un plan de transición y despliegue, que permita poner el sistema web ERP en producción.

- Identificar escenarios que favorezcan la elección de un componente sobre el otro, basados en los resultados obtenidos por la evaluación comparativa de componentes.

Capítulo 2: Estado del arte

Debido a la problemática, esta sección se enmarca en el estudio de distintas soluciones de facturación electrónica existentes en el mercado, siendo la vasta mayoría sistemas que integran funcionalidades esperadas en un ERP ("Enterprise Resources Planning", o planificación de recursos empresariales).

La minoría restante, otorga componentes de facturación electrónica, los cuales permiten ser integrados a sistemas ya existentes. Se dará un vistazo a estos dos tipos de soluciones, siendo nuestro foco de estudio las soluciones basadas en componentes.

2.1 Soluciones de Facturación Electrónica en Sistemas ERP

Un gran número de soluciones de facturación electrónica son entregadas como parte de un Sistema ERP. Estas soluciones proponen llevar a cabo una gestión extensiva de las actividades de la empresa, involucrándose en áreas típicamente abordadas por estas: control de compras, inventario, clientes, registro de ventas, cobranza, entre otras. La generación de documentos tributarios electrónicos (facturación electrónica) esta integrada en el sistema, usualmente siendo fuertemente ligada a los otros módulos de estos sistemas. Las soluciones de este tipo, varían principalmente en la plataforma, como son ofrecidas a sus clientes, tipo de funcionalidades ERP que realizan, y la estructura de costos de las soluciones que ofrecen. A continuación se muestran algunas de estas:

Nubox: Empresa orientada a ofrecer aplicaciones web que son distribuidos desde su infraestructura y accesibles desde cualquier equipo con conexión a internet. Ofrecen distintas sub-aplicaciones para el manejo de distintas necesidades de una empresa: contabilidad, pago de remuneraciones, administración comercial, control de existencias e inventario, control de activos fijos, gestión y facturación electrónica. Cada una de estas sub-aplicaciones puede ser contratada en base a un plan mensual que dependerá de las necesidades de usuarios en el sistema, cantidad de documentos de venta y en el caso de facturación electrónica, por la cantidad de DTE emitidos[8].

SoftLand: Ofrece dos líneas de soluciones para distintos tamaños de empresa. SoftLand ERP y SoftLand PYME. La primera provee en un solo servicio todos los módulos ERP disponibles, la solución PYME divide sus soluciones en tres aplicaciones, para contabilidad, remuneración de sueldos, y gestión comercial/POS (Point of sale). Siendo esta última la responsable de

---

[8] Nubox. Factura electrónica Nubox | Software de facturación [en línea] <http://www.nubox.com/software-de-factura-electronica.html> [consulta: 26 octubre 2016].

facturar electrónicamente. Ambas gamas de soluciones son provistas en aplicación de escritorio que debe ser apoyada con infraestructura de servidores provista por empresa cliente[9].

SoftRam: Empresa provee dos soluciones de escritorio para apoyar empresas según su tamaño: PymeXSys, orientada a la pequeña y mediana empresa, y SoftRam-ERP, para grandes empresas. Las diferencias de las dos soluciones que proveen consisten en la cantidad de módulos funcionales (suites) que poseen, siendo el módulo de facturación electrónica integrado en ambas soluciones[10]. Softram además dispone de soluciones móviles para necesidades de un POS (Point of Sale, o punto de venta), y de una solución de portal web para manejo del área comercial, calidad de servicio y pago de liquidaciones de sueldo. Estas dos últimas soluciones no consideran funcionalidades de facturación electrónica.

FacturaMovil: Ofrece una solución de facturación electrónica unificada a un servicio de POS (Point of Sale), permite gestionar clientes, productos, reportes y ventas desde plataforma web y móvil (en dispositivos Android o iOS). Ofrecen distintos planes que varían según la cantidad de transacciones, número de usuarios simultáneos, cobertura de asistencia técnica y omisión de publicidad (el plan más barato contiene publicidad embebida en sus aplicaciones). El plan más caro ("Huemul") otorga la posibilidad de mantener la infraestructura de manera "In-House", así como también poseer un número ilimitado de transacciones y usuarios[11].

Defontana: Proveen un software ERP disponible para una plataforma web, el sistema promete adaptarse a distintos rubros para cumplir sus necesidades específicas (producción, minería, salud e inmobiliaria). En todas sus versiones, cuenta con integración a servicio de facturación electrónica. Los planes están definidos de forma mensual y difieren en la cantidad de módulos disponibles, numero de usuarios habilitados, y número de trabajadores que se pueden gestionar dentro del software[12].

Lanix: Empresa posee solución LanixERP bajo plataforma web, la cual consiste de módulos que apoyan a tareas comunes de las organizaciones: Modulo de ventas, compras, inventario, gestión de producción, contabilidad, tesorería y pago de remuneraciones. Cada módulo interactúa con los otros para realizar sus funciones[13].

---

[9] Softland. Softland – Lo hacemos fácil [en línea] <http://www.softland.cl/> [consulta: 26 octubre 2016].

[10] SOFTRAM. Productos y Servicios [en línea] <http://www.softram.cl/productos-servicios.html?value=n0fot6k7h7x80nfkdpwbfn53894q%27%20id%20%27&detail=rgpb813580z%27%20iddet%20%27> [consulta: 26 octubre 2016].

[11] Factura Movíl SpA. Factura Electrónica [en línea] <http://facturamovil.cl> [consulta: 26 octubre 2016].

[12] Defontana Chile. Software ERP: Valores y Funcionalidades [en línea] <http://www.defontana.com/cl/precios/> [consulta: 26 octubre 2016].

[13] LanixERP. Soluciones para las empresas de hoy [en línea] <http://www.lanixerp.cl/soluciones/> [consulta: 26 octubre 2016].

Informat: Proveen un servicio INET ERP que cumple funcionalidades básicas de un software ERP (gestión financiero, gestión de recursos humanos, logística y área comercial) en plataforma de escritorio para sistemas Windows. Añade un módulo adicional llamado IFACTURE INET que se comunica única y exclusivamente con INET ERP (no se puede utilizar de forma independiente) y se encarga de hacer la facturación electrónica. Adicionalmente poseen otra solución de plataforma web, que solo permite realizar la gestión de ventas y facturación electrónica (IFACTURE WEB)[14].

Transtecnia: Entregan una gama de soluciones para empresas: Contabilidad, pago de remuneraciones, herramientas para automatizar procesos tributarios, y de facturación electrónica. Algunas soluciones entregadas a medida para cubrir las necesidades de ciertos rubros especializados (empresas de venta de combustibles). Las aplicaciones son de escritorio para ser usadas en ambientes Windows, y suponen que el cliente debe mantener la infraestructura donde se apoyarán los sistemas[15].

2.2 Componentes de Facturación Electrónica

Este tipo de soluciones está enfocado a empresas de desarrollo que deseen integrar su sistema ERP con el componente ofrecido, a modo de mantener la lógica de sus procesos intacta e integrar la facturación electrónica.

Las soluciones difieren en qué funcionalidades son capaces de entregar al momento de integrarlas, como es el tipo de comunicación que se debe realizar con el componente para la emisión de solicitudes, y en quien se apropia del componente (en infraestructura cliente o mantenido por proveedor). Las empresas encontradas que proveen este tipo de soluciones son: FacturaEnLinea.cl, Softnet ERP, Azurian, Posland, Paperless, FacturaElectronicaChile.com, NDTEChile. Esta sección presentará la propuesta ofrecida por cada una de las empresas anteriormente mencionadas.

FacturaEnLinea.cl provee soluciones web de software ERP, permitiendo la facturación electrónica y la generación de otros DTE's. Además provee soluciones para empresas que mantengan un E-commerce, y que requieren integrar de manera automatizada la facturación electrónica a sus procesos, proponiendo el uso de un componente Web Service. Es un servicio descubrible posible de acceder desde un repositorio UDDI, el cual envía un WSDL que otorga

---

[14] Informat. Informat.cl - IFACTURE INET [en línea] <http://www.informat.cl/ifacture-inet/> [consulta: 26 octubre 2016].
[15] Transtecnia. Catálogo de Soluciones [en línea] <http://www.transtecnia.cl/catalogo_soluciones.htm> [consulta: 26 octubre 2016].

la información del servicio, el cual es posteriormente solicitado con mensajes XML utilizando el protocolo SOAP.

Figura 2.1: Diagrama de interacción de componente FacturaEnLinea.cl

El esquema anterior detalla el funcionamiento del componente: El consumidor debe comunicarse a través de su sistema con el servidor UDDI y el web service para realizar peticiones relacionadas con la generación de DTE's, y otros servicios provistos por la implementación de FacturaEnLinea.cl. FacturaEnLinea también propone otros tipos de soluciones para la integración con sistemas ya existentes, basados en el envío de documentos de texto (.txt), .csv o planillas de Excel .xls en un formato previamente establecido. Estos documentos pueden ser digitados a mano o generados por un sistema y contienen peticiones de generación de DTE's para ser enviados al SII.cl. El envío de esos documentos puede ser a través de transferencia FTP, o desde su portal web. El coste de las soluciones entregadas por FacturaEnLinea.cl va determinado por la cantidad de documentos electrónicos emitidos mensualmente, teniendo planes desde 1 UF mensual, y además considera un coste de instalación de 40 UF[16].

Otra empresa que además de ofrecer un sistema ERP (Softnet ERP) provee servicios de facturación electrónicas vía componentes es Softnet. La empresa entrega dos planes para trabajo con un componente de generación de DTE, una solución de pago mensual, en la que la empresa proveedora mantiene el componente accesible como web service para la empresa cliente, y un plan "In-House", en el cual el cliente dispone del componente para instalarlo en su arquitectura y modificarlo a gusto.

Las soluciones garantizan la generación de todos los DTE necesarios y generación de documentos PDF para la impresión de facturas en caso de receptores que aún no adopten la factura electrónica. La comunicación con el componente es a través de protocolo SOAP

[16] FacturaEnLinea.cl. Cotiza [en línea] <http://www.facturaenlinea.cl/cotiza.php> [consulta: 26 octubre 2016].

utilizando XML para enviar/recibir solicitudes, la misma empresa además provee conectores para integración con componente en distintos lenguajes: VB.NET, VB6, Java, PHP y Javascript[17].

Azurian, posee cuatro planes para la emisión de DTE's y a diferencia de las anteriores, también integra procesos de recepción de DTE's. Los planes entregas por Azurian son: "DTE plus In House", en el cual el componente es instalada en infraestructura del cliente, "DTE plus Mixto", en el que parte de la infraestructura es instalada en infraestructura cliente, permitiendo facturación fuera de línea y sincronización con aplicación su nube (mantenida en infraestructura de Azurian), "DTE plus Cloud" otorga todos los servicios de emisión y recepción de DTE como componente en la nube (tipo web service) y "DTE plus Cloud Facturador", que integra la posibilidad de facturar en la nube, trabajando en solitario sin ser un componente [18].

Posland entrega una solución de componente de facturación electrónica, el cual es posible integrar con OdooERP (sistema ERP provisto por la misma empresa) o un sistema propio. La comunicación con el componente es a través de mensajes en formato texto plano o XML. El componente es entregado al cliente para montarlo en su propia infraestructura. El pago del servicio se divide en una habilitación única de 150.000 CLP y mensualmente un pago de 39.000 CLP por 100 DTE's mensuales a emitir[19].

NDTEChile ofrece un componente el cual el usuario cliente es el propietario (se mantiene bajo infraestructura del cliente). Este componente tiene funcionalidades de emisión y recepción de DTE's necesarios para pasar el proceso de certificación (esto incluye todos los flujos necesarios para la facturación electrónica). El componente permite ser fácilmente integrado a través de conectores nativos para los lenguajes de C#, VB.net, Python y Java[20].

FacturaElectronicaChile.com es otra empresa que provee una solución basada en componente que reside en arquitectura del cliente. Ofrecen un componente que puede ser integrado como librería DLL, o como programa ejecutable para arquitecturas Windows o Unix. De esta forma, el componente puede ser integrado con facilidad a lenguajes como C++, FoxPro, Pascal/Delphi, Java, .Net (C# o Sharp y VB), PHP, ASP, Python, entre otros. Las

---

[17] Softnet. Softnet | Facturaciónn Electrónica, Erp Softnet [en línea] <http://www.softnet.cl/aplicaciones_ti-soluciones-emision_DTE.php> [consulta: 26 octubre 2016].

[18] Azurian. Modalidades de Implementación de Factura Electrónica en Chile [en línea] <http://www.azurian.com/fichas/factura-electronica-chile> [consulta: 26 octubre 2016].

[19] Posland. Factura Electrónica desde su propio software [en línea] <http://posland.cl/odoo/servidor-factura-electronica-conector/> [consulta: 26 octubre 2016].

[20] NDTE Chile. ¿Qué ofrece NDTE Chile? [en línea] <http://www.ndtechile.cl/que-ofrece-nte-chile/> [consulta: 26 octubre 2016].

funcionalidades del componente comprende la emisión y firmado de los DTE's, así como la generación del documento PDF para impresión a contribuyentes no electrónicos[21].

Por último encontramos LibreDTE-lib[22], librería PHP que contiene funcionalidades de facturación electrónica, y de interacción con el SII. La librería puede generar y enviar DTE's al servicio de impuestos internos, consultar el estado de envío de un DTE en particular, recibir DTE's de otros contribuyentes y generación de documentos PDF aceptados por el SII, entre otras funcionalidades.

La librería puede ser usada en conjunto a la plataforma LibreDTE, la cual es accesible desde su sitio web (https://libredte.sasco.cl), además de contar con una API en la cual puede ser usada a modo de componente web service para realizar los procesos de facturación electrónica. LibreDTE está construida utilizando el framework SowerPHP[23], el cual es un framework minimalista basado en principios MVC.

2.3 Otras soluciones existentes

Adicionalmente a las soluciones anteriormente presentadas, existen otras que no corresponden a aplicaciones comerciales ERP con posibilidad de facturación electrónica ni componentes de emisión de DTE. En esta sección se presentan algunas alternativas que pueden satisfacer las necesidades de facturación electrónica, o facilitar el proceso.

2.1.3.1 Portal MiPyme

El servicio de impuestos internos, pone a disposición el portal MiPyme[24], en el cual los contribuyentes pueden emitir y recibir documentos tributarios electrónicos para cumplir con las normativas vigentes. El sitio se pone a disposición para cualquier contribuyente de primera categoría con un certificado digital vigente y que haya iniciado actividades, y a diferencia de los sistemas bajo compra o de manufacturación propia, el portal MiPyme no requiere certificación para ser usado. Algunas de las desventajas del portal son: la nula integración con ERP existentes (teniendo que doble ingresar datos, generando inconsistencias y un mayor

---

[21] FacturaElectronicaChile.com. Facturación Electrónica (SII) [en línea] <http://facturaelectronicachile.com> [consulta: 26 octubre 2016].

[22] LibreDTE. Biblioteca PHP para facturación electrónica en Chile [en línea] <https://github.com/LibreDTE/libredte-lib> [consulta: 26 octubre 2016].

[23] LibreDTE. Arquitectura de LibreDTE [en línea] <https://wiki.libredte.cl/doku.php/dev/arquitectura> [consulta: 26 octubre 2016].

[24] Servicio de Impuestos Internos. Conozca de factura eléctronica y de los sistemas disponibles [en línea] <http://www.sii.cl/factura_electronica/como_fact_elect.htm> [consulta: 26 octubre 2016].

tiempo requerido), no poder emitir todos los documentos tributarios electrónicos posibles (facturas de exportación, notas de crédito y débito de exportación, liquidación de factura, boletas afectas, etc.) y que cada DTE tiene que ser emitido uno por uno, eliminando la posibilidad de grandes empresas de facturar por lotes mediante la administración de folios.

2.1.3.2 Librerías de desarrollo

Existen disponibles en repositorios de manera pública, algunas librerías que facilitan el desarrollo de aplicaciones de facturación electrónica: NICLabs[25] mantiene unas librerías de facturación electrónica, que están escritas en Java, manejan muchos flujos típicos necesarios para manejar DTE (CAF, firmado de documentos y emisión), además de presentar algunos ejemplos de su uso. Talonario[26] es una gema de Ruby on Rails que facilita el proceso de facturación electrónica, aunque su desarrollo esta descontinuado y sus funciones son escasas (las últimas actualizaciones del repositorio son del 2012).

---

[25] niclabs. Bibliotecas de factura electrónica (DTE), desarrollada y mantenida por Jose Urzua y NIC Chile Research Labs [en línea] <https://github.com/niclabs/DTE> [consulta: 26 octubre 2016].

[26] pbruna. A gem to work with Chile SII Factura Electrónica [en línea] <https://github.com/pbruna/talonario> [consulta: 26 octubre 2016].

Capítulo 3: Propuesta específica

Existiendo una ausencia de soluciones de facturación electrónica, a modo de componentes que puedan ser utilizadas desde infraestructura propietaria, a disposición gratuita y sin problemas de licenciamiento para uso comercial, se presenta una propuesta de componente que pretende responder las necesidades de facturación electrónica requeridas por el SII para operar en el modelo de contribuyente electrónico.

Esta sección pretende explicar la arquitectura basada en componentes, y las tecnologías base a utilizar para implementar la propuesta.

3.1 Arquitectura basada en componentes

A medida que la complejidad de un software aumenta, se vuelve imperante tratar de disminuir el esfuerzo necesario para implementar una nueva funcionalidad, o para reducir los costos de mantención.

Situándonos en el caso de un sistema de gran tamaño implementa funcionalidades para casos de uso de distintos tipos de usuarios: Aun cuando los roles no posean una relación sustancial, su desarrollo formará una red de dependencias bastante compleja y unida, lo que resulta que pequeños cambios en una de sus funcionalidades sean propagados a otras partes del sistema de formas inesperadas, provocando posibles defectos que requerirán ser reparados.

Estos defectos, pueden aparecer en partes del código que están asignados a otra persona del equipo, requiriendo una colaboración estrecha los miembros del equipo involucrados para resolver las consecuencias de los cambios, aun cuando estos eran pertinentes sólo a una zona acotada del código.

Esto provoca que la delegación de tareas sea pobre, lo cual en equipos grandes puede impactar bastante en la productividad del equipo, aún más si estos están distribuidos geográficamente.

El desarrollo de software basado en componentes propone que un software puede ser construido mediante utilización de piezas de software que realizan un grupo reducido de tareas del total que se desea construir [2].

En particular, estas piezas denominadas componentes, se definen como un “trozo de software, pensado para interactuar con otros componentes, encapsulando cierta funcionalidad o una serie de funcionalidades. Un componente tiene una interfaz claramente definida y está suscrito

a un comportamiento prescrito, común a todos los componentes en una arquitectura”[27]. La arquitectura que comprende la utilización de componentes de software para componer así las funcionalidades de un software, se entiende como arquitectura basada en componentes.

El uso de componentes permite modularizar partes de una aplicación, llevándola a un dominio discreto en el cual ejercerá sus funciones, facilitando la división de equipos en torno a tareas específicas. El uso de interfaces permite a otras partes de la aplicación, requerir de las funciones de un componente sin tener nociones del funcionamiento interno de este.

El uso de componentes permite entender nuestro sistema como un grafo de módulos que realizan funcionalidades, unidos mediante conexiones unidireccionales que denotan las dependencias entre estos. De esa forma permite expresar con mayor claridad el funcionamiento de un sistema, tanto a terceros como a miembros del equipo.

La red de dependencias además favorece a dar una noción precisa de como un componente afecta al resto del sistema, ya que solo los dependientes de este serán afectados, reduciendo los posibles defectos descontrolados en otras partes del sistema.

Desde otra arista, un componente que posee una interfaz claramente definida y que realiza una función concreta puede ser fácilmente extraído del proyecto y reutilizado en otros que también requieren su funcionalidad. La capacidad de reuso de los componentes permite a empresas mantener un ecosistema de componentes frecuentemente solicitados en proyectos y de esa forma, reducir el costo que requiere la realización de nuevos proyectos.

En particular, si estos repositorios de componentes son puestos a disposición por terceros de modo comercial, se dice que el componente es un COTS (“Component Of The Shelf”) [5]. La utilización de COTS permite a empresas evitar el costo de un desarrollo “In-House”, comprando componentes que realizan la funcionalidad requerida a una calidad que podría incluso superar el costo por el esfuerzo de ser desarrollado “In-House”.

La modularidad de la arquitectura basada en componentes y la reducción de costos que produce resultan particularmente atractivas para el desarrollo de una solución de facturación electrónica reusable, por lo que su uso fundamenta y motiva la implementación de una solución basada en este concepto.

---

[27] W3C. Web Services Glossary [en línea] <https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/> [consulta: 26 octubre 2016].

3.1.1 Problemas asociados a Component-Based Architecture

La arquitectura basada en componentes permite a proyectos de software elaborar soluciones a partir de COTS, componentes comerciales existentes en el mercado, y de esa forma, obtener una serie de beneficios económicos, entre otros como mejorar la distribución del trabajo de manera flexible mediante la separación de funcionalidades que usar componentes provee [6], pero también trae consigo nuevos problemas a aquellos que deseen seleccionar y utilizar estos en sus sistemas, problemas que han sido estudiados e intentados de mitigar por la CBSE (Component-Based Software Engineering).

Algunos de estos son:

- Confiabilidad de los componentes: Algunos requerimientos son de carácter crítico para el funcionamiento del sistema, y por lo tanto la selección de un COTS para cubrir la funcionalidad debe considerar en gran parte la confiabilidad del componente. Dado que el acceso al funcionamiento interno de un componente queda ofuscado, es difícil predecir aspectos claves de la funcionalidad que un componente puede entregar, por lo que deben existir mecanismos para mitigar la falta de información y de esa manera poder efectuar la selección de componentes.

- Dificultad de testing y medición de atributos del componente: La ofuscación del funcionamiento interno en COTS sitúa una caja negra en el funcionamiento del sistema que lo desee integrar, creando problemas al intentar realizar testing en el componente [7], pérdida de efectividad a la hora de detectar defectos, desconocimiento de funcionalidades complejas donde deberían ser situado los esfuerzos de testing, y un desafío a la hora de realizar testing funcional son algunos de los problemas.

Esta misma caja negra provoca que medir las características internas que definen la calidad de un componente sea imposible de realizar, propagando una incertidumbre en la calidad total del sistema [8]. Esto resulta especialmente problemático en sistemas en los cuales se requiere un QoS determinado.

- Manejo de requerimientos y selección de componentes: La selección de componentes supone una elección de componentes para cubrir una serie de requerimientos considerando aspectos como el costo total de estos, o la calidad relativa. Seleccionar componentes puede ser un problema cuando los posibles candidatos a resolver un requerimiento fallan al momento de cubrir la totalidad de los requerimientos necesitados por el sistema. Otro problema puede surgir de que aunque los componentes funcionen de forma individual, la composición de estos no sea factible o no funcione de manera óptima. Es necesario hacer un análisis de factibilidad en conjunto con la selección de componentes a modo de minimizar los riesgos, riesgos que de todos modos, deben ser contabilizados en el proyecto.

− Integración dificultosa y falta de predictibilidad: Aun habiendo seleccionado un componente que cumpla los requerimientos funcionales necesarios, la integración puede resultar dificultosa por diferencias entre la arquitectura actual y la propuesta por el componente [9], supuestos erróneos sobre la utilización del componente por falta de información en la documentación, composición errónea de componentes, entre otros.

− Mantención a largo plazo de sistemas basados en componentes: La dependencia en COTS puede otorgar problemas a la hora de mantener soluciones basadas en estos: La variación de las interfaces dado a actualizaciones de los componentes puede suponer un quiebre en el sistema existente y costos en adaptarse al cambio, la disponibilidad de los componentes (en caso que estos dependan de un servicio en la web) puede no estar asegurada en el tiempo, o el retiro de soporte técnico responsable del buen funcionamiento del componente.

Las principales causas de los problemas ligados al diseño de software con componentes provienen de una falta de técnicas apropiadas que permitan resolver efectivamente la selección de componentes y de la poca información que se posee de estos para realizar el proceso. Es difícil determinar la factibilidad de la integración entre varios componentes, o si estos componentes logran cubrir aquellos aspectos que son críticos para los stakeholders.

Para resolver este problema, la literatura provee técnicas que prestan especial atención en técnicas de selección de componentes, los cuales proveen métodos cuantitativos para evaluar los componentes en base a propiedades medibles de estos, y de esa manera seleccionar los componentes a utilizar. Los criterios que los modelos utilizan son variables, y por lo general van relacionados con métricas funcionalidades propias de algunos estándares de calidad de software, como la ISO/IEC 25010, o la ISO/IEC 9126-1.

Una revisión de los modelos de calidad se llevará a cabo en el estado del arte. A partir de los requisitos funcionales y el estudio de estos modelos, se definirá una rúbrica para componentes de facturación electrónica, que será utilizada para llevar a cabo la evaluación comparativa de componentes candidatos.

3.1.2 Reuso vs Component-Based-Architecture

Aunque el uso de una arquitectura basada en componentes fomentar el reuso, es necesario destacar que un componente es más que una pieza reusable de software. Un componente deberá cumplir con las siguientes características [10]:

− Interfaz definida: El acceso a las funcionalidades del componente es a través de una interfaz claramente definida. Los detalles de la implementación permanecen ocultos para el exterior.

- Arquitectura definida: Los componentes son diseñados pensando su integración en una arquitectura predefinida, a modo que puedan operar con otros componentes o frameworks.

- Estandarización: La interfaz y canales de comunicación por los cuales el repositorio dispone sus servicios deben ser estandarizados a modo de facilitar la creación de los componentes, y que la cobertura de empresas que puedan reusar el componente sea lo más amplia posible.

- Distribución por mercado: Componentes pueden ser adquiridos de forma comercial a mano de terceros (COTS). dando un incentivo a las empresas a la compra de estos sobre la manufacturación ad hoc.

De este modo, el diseño de un componente de software puede ocurrir como un proceso paralelo, o totalmente fuera de contexto con la construcción de un proyecto que utilice el componente. La construcción de componente reusable supone la abstracción de una funcionalidad común, llevándola a un contexto suficientemente genérico para facilitar su reuso. Esta generalidad requerida frecuentemente implica llevar a cabo más funcionalidades de las deseadas y requiere mayores esfuerzos de diseño y desarrollo [11].

Si la funcionalidad que provee el componente es demasiada específica al dominio de la aplicación que se está construyendo, dicho componente probablemente no será reusable. Aun así, su concepción como componente puede ser útil para obtener otros de los beneficios anteriormente mencionados.

3.2 Propuesta de componente y sistema ERP

Para la elección de la implementación del componente y sistema ERP, se consideró maximizar la disponibilidad y crear una solución que fuera independiente de la plataforma del usuario.

Considerando que los sistemas operativos más utilizados cuentan con ejecutables para Chrome, o Firefox (siendo navegadores bastante populares, cubriendo más de un 70% de los usuarios)[28], y que los dispositivos móviles con iOS o Android cuentan con estos navegadores, la construcción de un sistema web accesible desde estos navegadores permite que los usuarios puedan hacer uso de este desde una amplia gama de dispositivos.

---

[28] W3Counter. Browser & Platform Market Share [en línea] <https://www.w3counter.com/globalstats.php> [consulta: 26 octubre 2016].

Diversos lenguajes ofrecen herramientas para la generación dinámica de sitios web que es necesaria para el sistema. Las tecnologías más rudimentarias permiten el desarrollo de cada parte del sistema de manera entrelazada y son ejecutados en el lado del servidor, como es el caso de PHP, un lenguaje interpretado que permite la generación de páginas HTML con contenido dinámico, al momento que este es requerido por los clientes[29].

Si bien utilizando tecnologías como PHP, ASP o CGI junto a un motor de bases de datos relacional, se pueden satisfacer las necesidades del sistema, existen problemas debido a que no hay convención en cómo estructurar las partes, lo cual provoca código de bastante complejidad, dificultad para la colaboración y un grado escaso de reuso (lo cual se desea evitar). Se han presentado patrones de diseño que buscan resolver estos problemas de manera estandarizada, definiendo una estructura organizada para la construcción de sistemas web.

En particular a nuestro foco de interés, surgen los denominados Framework Web, los cuales están pensados para soportar la implementación y diseño de sistemas web. Estos son montados sobre las bases de un lenguaje definiendo una estructura molde a utilizar, e integran utilidades comunes y necesarias para el desarrollo web[30].

Los beneficios desarrollar bajo un Framework Web son varios:

- Permiten despreocuparse de tareas comunes y enfocarse en el dominio de la aplicación a construir, ya que las tareas comunes ya están implementadas de manera robusta por los framework.

- Mejor organización de las partes de la aplicación por los patrones de diseño propuestos por el framework, lo que permite mejorar la comprensión de la implementación, y disminuir los efectos colaterales causados por cambios en una parte del código.

- Facilidad en la colaboración al poseer una estructura segmentada, permitiendo equipos numerosos afectar en distintas partes de la aplicación sin generar colisiones, y que personas externas al sistema pero con dominio en el framework, puedan integrarse de manera más rápida.

Muchos frameworks web implementan el patrón de diseño MVC. El patrón MVC, propone segmentar la aplicación en tres componentes: modelo, vista y controlador, encargadas de distintas responsabilidades.

---

[29] The PHP Group. What can PHP do? [en línea] <http://php.net/manual/en/intro-whatcando.php> [consulta: 26 octubre 2016].
[30] StackOverflow. What is a Web Framework? How does it compare with LAMP [en línea] <http://stackoverflow.com/questions/4507506/what-is-a-web-framework-how-does-it-compare-with-lamp> [consulta: 26 octubre 2016].

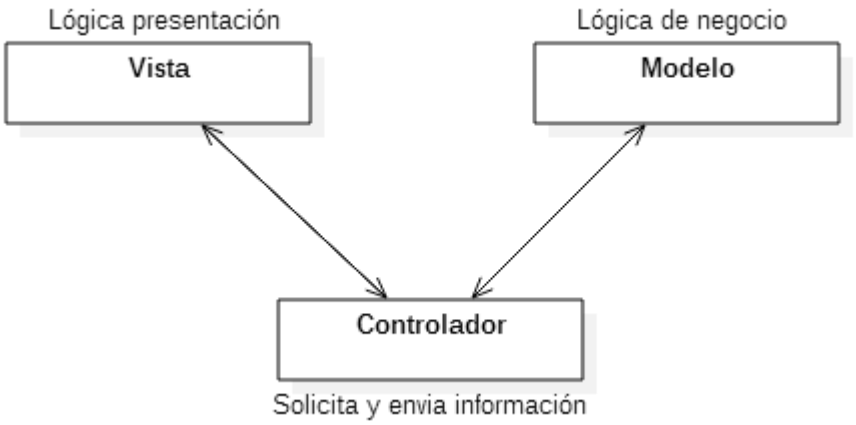

Figura 3.1: Esquema de interacción de componentes en patrón MVC.

El modelo, contiene las entidades del dominio, así como la lógica de negocio relacionada a estas. El controlador es el intermediario entre la vista y el modelo, responde a las llamadas de las vista, solicita información del modelo, la cual es procesada y enviada de vuelta a la vista. La vista es finalmente encargada de presentar la información en una interfaz gráfica al usuario, y de capturar eventos para ser enviados al controlador.

El patrón MVC fue originalmente propuesto para el desarrollo de interfaces gráficas en SmallTalk [12], pero se han logrado adaptar al paradigma web: Mediante solicitudes HTTP, los clientes llaman a los controladores (por medio del enrutador), los cuales procesan información desde el modelo y lo envían a las vistas. Las vistas definen la organización del contenido y responden al usuario con contenido HTML, el cual es visible desde el navegador. Los modelos corresponden a entidades del dominio, y describen las reglas de negocio sobre cómo estas entidades operan. Esta última capa generalmente está vinculada con un motor de base de datos para persistir la información de las entidades del modelo, mediante un ORM (Object Relational Mapping), el cual expone una interfaz para hacer uso de funciones de la base de datos sin exponer su funcionamiento interno, de modo que es “agnóstico” al motor de base de datos utilizado (los ORM robustos permiten ser vinculados a distintos motores funcionando de igual forma, sin cambio alguno en cómo la interfaz debe ser llamada).

Como se mencionó anteriormente, el modelo MVC en frameworks web generalmente integra un enrutador, el cual es el punto de entrada a la aplicación y se vincula a los controladores para procesar las solicitudes, el enrutador es el encargado de montar direcciones URL y asociarlas a acciones de los controladores, enviando los parámetros de la solicitud HTTP para que el controlador genera una respuesta. Un controlador puede contener múltiples acciones, y puede comunicarse con más de un modelo si fuese necesario.

Dadas estas diferencias, el MVC aplicado a frameworks web no es puro, ya que está apoyado estructuralmente con otros componentes para suplir otras funciones que no son responsabilidad habitual de MVC clásico.

En la actualidad existen numerosos frameworks web para cada lenguaje: Laravel, Codeigniter, CakePHP o Symphony para PHP, Django o Flask en el caso de Python, Sinatra o Ruby on Rails para Ruby. Algunos frameworks integran una gran serie de funcionalidades, mientras que otros intentan mantener el peso de su inclusión a un mínimo (micro frameworks), o difieren en los patrones de diseño utilizados (no todos se basan en un diseño MVC).

Para la implementación de este sistema, se hará uso del framework web para Ruby: Ruby on Rails (o comúnmente llamado solamente Rails), el cual es un framework web basado el patrón MVC para organizar la estructura de la aplicación[31].

Rails se integra como librería (gema) mediante el gestor de paquetes para Ruby, RubyGems. Rails es un framework para la creación de aplicaciones web, el cual asiste el proceso de implementación de "features" comunes, como generación de HTML dinámico, procesamiento de "forms" y acceso a base de datos.

Rails es un framework que propone una solución única para estos problemas, basada en la experiencia de sus autores ("opinionated framework", o framework "opinado"), de esta forma, establece convenciones a seguir y una forma de realizar tareas comunes, facilitando la toma de decisiones por desarrolladores y mejorando la mantenibilidad del código, ya que al seguir las convenciones el código se vuelve homogéneo y más fácil de comprender.

Rails se apoya fuertemente en las virtudes de Ruby: una sintaxis limpia y un alto nivel de abstracción, que permite la construcción de DSL (el cual es usado para la generación de vistas y configuración de Rails), y el gestor de paquetes RubyGems.

El beneficio de utilizar gemas (el nombre dado para los paquetes), es que las funcionalidades de Rails pueden ser extendidas, por gemas especialmente pensadas para ser integradas junto a Rails. Las gemas son subidas por la comunidad y dejadas a libre disposición para el uso libre de los desarrolladores, permitiendo disminuir los tiempos de desarrollo de aplicaciones web, ya que funcionalidades comunes están generalmente cubiertas por gemas populares.

Las gemas más populares son constantemente mantenidas por la comunidad, por lo que la calidad de estas es alta, y son mantenidas al día para funcionar con nuevas versiones de Ruby.

El concepto de gemas de Ruby concuerda con algunas de las características propias de arquitectura basada en componentes, diferenciándose a su vez del reuso normal, gracias a su estandarización en su distribución/uso y su definición clara de interfaces gracias a un desarrollo correctamente implementado.

---

[31] RailsApps Project. What is Ruby on Rails? [en línea] <http://railsapps.github.io/what-is-ruby-rails.html> [consulta: 26 octubre 2016].

En efecto, el concepto de gemas de Ruby se utilizará de las bases en la implementación del componente, ya que este será implementado como una gema, la cual se requerirá por el sistema ERP.

Por otra parte, el componente de facturación electrónica dependerá de otras gemas, las cuales asistirán procesos genéricos a cubrir: Generación de código de barras PDF417, manejo de XML y generación de firmas digitales son algunos puntos que serán cubiertos con estas.

Capítulo 4: Desarrollo de Componente

En base a la propuesta, este capitulo considera la especificación de las exigencias impuestas por el SII que fueron implementadas en el componente, además de mencionar requisitos externos al componente que pertenecen al sistema ERP en el cual se integrará. Esta sección también explicará cómo fue formada la integración del componente al sistema. La segunda parte de la propuesta será definida en detalle en el capitulo siguiente.

4.1 Exigencias impuestas por el SII

La normativa del SII, obliga a los contribuyentes a seguir con ciertas exigencias comunes para todos a la hora de emitir documentos tributarios electrónicos. A la hora de diseñar un componente de facturación electrónica se debe asegurar que, por lo mínimo, cada una de las exigencias impuestas esté cubierta.

En particular, estas exigencias pueden ser revisadas en detalle a partir de los documentos técnicos provistos por el SII. Para facilitar la comprensión, se pueden agrupar las exigencias en los siguientes grupos de roles, los cuales serán cubiertos por la implementación específica del componente.

4.1.1 Generación de Documentos Tributarios Electrónicos (DTE)

Se deberá ser capaz de procesar información de un DTE, mediante la creación de un archivo que cumpla con los estándares impuestos por el SII sobre estos. Estos estándares consideran la generación de documentos en formato XML, los cuales son validados mediante un schema provisto por el mismo SII.

Cada uno de los DTE generados deberá contener información del timbrado electrónico, el cual lo entrega el SII y permite la emisión de ese tipo de DTE. Adicionalmente, el formato de estos XML contendrá una o varias firmas, las cuales deben ser realizadas siguiendo el estándar XMLDSIG 1.1, recomendado por la W3C. Esta firma debe ser efectuada por la llave privada del certificado provisto por las empresas prestadoras acreditadas en la materia.

4.1.1.1 Sobre el formato de envíos

Cada DTE que se desee emitir, debe estar contenido dentro de un envío de DTE. Este contenedor se encarga de contener datos de información del emisor, así como la firma electrónica completa del documento. Cabe destacar que un envío de DTE usualmente contiene un DTE, pero el formato permite la inclusión de hasta 2000 DTE.

En particular, respecto al esquema XML exigido (“SiiDte EnvioDTE_v10.xsd”), la información contenida por cada DTE se define en un nodo <Documento>, el cual deberá ser contenido por un nodo <DTE>. Este nodo <DTE> contendrá a su vez la firma del nodo <Documento>, la cual es definida por XMLDSIG 1.1, y es contenida en un nodo <Signature>.

Luego, cada nodo <DTE> debe ser contenido en un nodo <SetDTE>, el cual puede contener uno o más <DTE>, <SetDTE> a su vez contiene información del emisor del envío, la cual es contenida en un nodo <Caratula>. Finalmente, <SetDTE> es contenido nuevamente por <EnvioDTE>, el cual es el nodo padre de todo DTE válido. Este nodo contiene la última firma, la cual se efectúa sobre los contenidos de <SetDTE>, y es contenida en un nuevo nodo <Signature>.

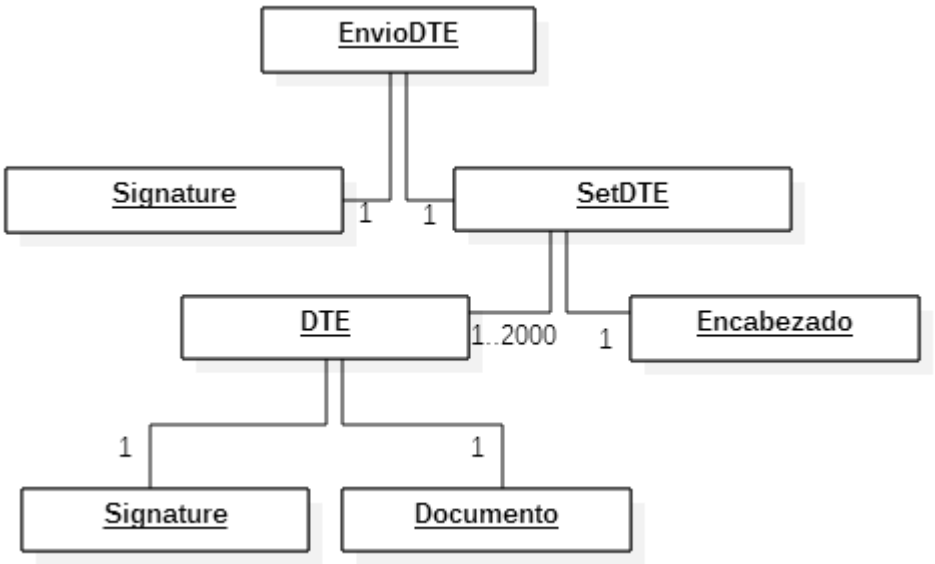

Figura 4.1: Esquema de formato XML de un EnvioDTE.

4.1.1.2 Sobre el formato de los DTE

La información de cada documento tributario perteneciente a un EnvioDTE, queda descrita en nodos tipo <Documento>, los cuales están regidos por el schema XML “DTE_v10.xsd”. Según el tipo de documento tributario electrónico, el SII define en su instructivo técnico,

cuáles campos son permitidos, señalando aquellos que son obligatorios, opcionales, o de carácter condicional[32].

A grandes rasgos, la estructura de un <Documento> es conformada por las siguientes zonas:

- Encabezado: Contiene datos del documento, como su tipo, folio utilizado, fecha de emisión, montos, e información de transporte, además de incluir datos del emisor y el receptor del DTE.

- Detalle de productos o servicios: Contiene el detalle de cada ítem perteneciente al documento: código, valor, nombre, descripción, unidad de referencia, descuentos y recargos aplicados a este, cantidad, precio unitario y monto total. El formato permite la ocurrencia de hasta 60 detalles, debiendo respetar además que estos puedan ser incluidos en una plana de la representación impresa.

- Subtotales informativos: Pueden desglosar detalle de las líneas o introducir nuevos conceptos a declarar en el documento. Los montos introducidos en estos no aumentan la base impositiva ni son acumulados en los totales, son solo de carácter informativo. El formato permite de 0 a 20 ocurrencias de estos. Cada subtotal debe incluir una glosa, la que especifica el concepto.

- Información de descuentos y recargos: Contiene los descuentos y recargos a aplicar de manera global en el documento. Los descuentos y recargos pueden ser de carácter porcentual (esto es, sobre el valor del total bruto de todas las líneas de detalle) o numérico (mediante un valor fijo). De esta forma, el valor neto del documento se calcula aplicando el valor IVA al monto bruto total, posterior a las modificaciones de los descuentos y recargos aplicados. El formato permite de 0 a 20 líneas.

- Información de referencia: Expresan referencias a otros documentos tributarios o de control interno. En el caso de una nota de crédito/débito electrónica, es aquí donde se declara el DTE al cual hacen efecto. Se permiten de 0 a 40 repeticiones.

- Comisiones y otros cargos: Se incorporan en caso de que apliquen otros cargos/comisiones, como en liquidaciones-factura electrónicas, en facturas de compra, o en notas de crédito/débito electrónicas que apliquen sobre facturas de compra. El formato permite de 0 a 20 repeticiones.

---

[32] Servicio de Impuestos Internos. Formato documentos tributarios electrónicos [en línea] <http://www.sii.cl/factura_electronica/factura_mercado/formato_dte.pdf> [consulta: 26 octubre 2016].

− Timbre electrónico del documento: Corresponde a una firma electrónica sobre algunos datos del DTE, la cual es creada a partir del CAF (código de autorización de folios) entregado. Su conformación y utilidad será explicada en “Sobre el uso de Timbraje Electrónico”.

4.1.1.3 Sobre la firma de los documentos

Cada documento EnvioDTE debe ser firmado, por lo menos dos veces. Por cada DTE, debe realizarse una firma electrónica sobre su respectivo nodo <Documento> y una sobre el <SetDTE>, el cual incluye el detalle de todos los DTE pertenecientes al envío y los datos del encabezado. Cada una de las firmas electrónicas es realizada utilizando la llave pública del certificado digital del contribuyente, el cual es una persona natural autorizada por la empresa para emitir los documentos.

El proceso de firmado debe considerar los estándares publicados en XMLDSIG 1.1 y otros detalles adicionales, los cuales han sido documentados de manera extraoficial en [33]. En detalle, el proceso de firmado supone los siguientes pasos:

− Canonizar el nodo bajo el cual se realizará la firma electrónica. Para efectos del SII, esto supone:

- Mantener todo salto de línea, espacio y tabulación entre los tags XML sin cambios.

- Codificación del contenido utilizando ISO-8859-1. El cual opera almacenando caracteres especiales, como diacríticos y letras especiales en dos bytes.

- Las 5 entidades predefinidas en XML (& < > ‘ y “) que ocurran como contenido dentro de un nodo, deberán ser escapadas utilizando su referencia como entidades (& < > " y ').

- Aquellos nodos que hayan sido declarados en su forma reducida (ej: <tag />), deberán ser expandidos a su forma normal (ej: <tag></tag>).

- Cualquier espacio en blanco adicional en la definición de atributos de un tag deberá ser remplazado por un único espacio. Los valores de los atributos deberán ser definidos encerrados entre comillas dobles exclusivamente.

---

[33] DI Management Services. XML-DSIG and the Chile SII [en línea] <http://www.cryptosys.net/pki/xmldsig-ChileSII.html> [consulta: 26 octubre 2016].

- El ordenamiento de los namespaces y atributos de los nodos deberá ser realizado de manera lexicográfica por su nombre local, con una preferencia a los namespaces de los nodos por sobre sus atributos (si no posee nombre local tiene prioridad por sobre los otros). Los atributos deberán ser ordenados luego de estos utilizando ordenamiento lexicográfico con dos claves, la URI del namespace del atributo como clave primaria y el nombre del atributo como clave secundaria. Aquellos atributos que no posean un namespace asignado poseen prioridad.

- Algunos namespaces deben ser propagados a los nodos a firmar. Para las firmas internas, el nodo <Documento> debe contener el namespace xmlns="http://www.sii.cl/SiiDte". En el caso de la firma al nodo <SetDTE>, se debe propagar los namespaces xmlns="http://www.sii.cl/SiiDte" y xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance", en ese mismo orden.

− Bajo el nodo XML canonizado, se debe utilizar la función de hashing criptográfico SHA1, para generar un message digest (el resultado de la operación de hash). Este digest debe ser insertado en un nodo <SignedInfo>, tal como es definido por el estándar XMLDSIG 1.1.

− Canonizar el nodo <SignedInfo> con el digest encontrado y aplicar nuevamente SHA1 para obtener un digest de este.

− Con la llave privada del contribuyente, se deberá firmar el digest SHA1 del nodo <SignedInfo> canonizado.

− El valor hexadecimal encontrado al firmar el digest deberá ser codificado en Base64 e insertado en el nodo <SignedValue>, el cual es un nodo hijo de <SignedInfo>. Este valor será el que permitirá verificar la firma electrónica del contribuyente, contra la información presentada en el nodo.

Para realizar el proceso de firmado, se utilizó Signer[34] como base para asistir con las especificaciones del estándar XMLDSIG 1.1, la cual trabaja utilizando la gema OpenSSL como base para la creación de los digest y para realizar el firmado, y la gema Nokogiri[35] para el manejo de documentos XML.

Para cubrir aquellos otros requisitos no incluidos en el estándar, se requirió hacer una copia local de la gema para adaptarla al funcionamiento del SII.

---

[34] ebeigarts. WS Security XML Certificate signing for Ruby [en línea] <https://github.com/ebeigarts/signer> [consulta: 26 octubre 2016].

[35] Sparkle Motion. Nokogiri is an HTML, XML, SAX, and Reader parser with XPath and CSS selector support [en línea] <https://github.com/sparklemotion/nokogiri> [consulta: 26 octubre 2016].

4.1.1.4 Sobre el uso de Timbraje Electrónico

El timbraje electrónico para cada tipo de DTE debe ser solicitado en el portal web del SII, y permite al contribuyente emitir un número válido de DTE en un rango de folios. Es responsabilidad del componente controlar que el rango de folios se respete (no permita la creación fuera de este), y que la asignación de folios sea única a cada DTE (SII rechazara los DTE si el folio para ese tipo de documento ya se encuentra recepcionado).

El rango de folios entregado por el SII consiste en un documento XML que cuenta con un CAF (código de autorización de folios) y un par de claves privado/pública que forman parte del proceso de timbrado. El procedimiento para timbrar un documento consiste en lo siguiente:

- Construcción de un nodo <TED>: El nodo <TED>, almacenado en la jerarquía del nodo <Documento>, contendrá información relevante del documento junto con el <CAF>, el cual deberá ser extraído desde el XML de los folios autorizados. Toda esta información deberá ser contenida bajo el nodo <DD>, el cual es nodo hijo de <TED>.

- Codificación de caracteres especiales: Similar a la canonización a cumplir para realizar las firmas XMLDSIG 1.1, se requiere la codificación de los caracteres especiales que formen parte de los valores presentes en el nodo <DD> o sus hijos: Escapar las 5 entidades especiales de XML y codificar utilizando ISO-8859-1. Como regla adicional, se requiere eliminar todo espacio en blanco que este entre los tags.

- Cálculo del digest: Bajo el nodo DD con los caracteres codificados, se calcula el digest SHA1.

- Obtención del valor de la firma: Utilizando la llave privada perteneciente al XML del rango de folios autorizado, se firma el digest del nodo <DD> encontrado previamente. Este valor deberá ser codificado en Base64 e insertado como valor del nodo <FRMT>, el cual es un hijo del nodo <TED>.

- Insertar fecha de la firma: Finalmente, se debe insertar el nodo <TmstFirma> a <TED>, el cual contiene el tiempo actual al momento de la firma.

Este proceso debe ser realizado para cada documento tributario dentro de un <EnvioDTE>.

Ya que el timbrado no es un proceso estandarizado ni rige a cabalidad el XMLDSIG, se realizó una solución propia, utilizando OpenSSL y Nokogiri como base para asistir el proceso.

4.1.2 Envío automático de DTE's al SII

La generación de cada documento tributario electrónico concreta su validez para efectos tributarios una vez que es recibido conforme sin rechazos ni reparos por el SII. Es necesaria la capacidad de que los DTE's generados sean enviados de forma automática. Para ello, el SII define servicios SOAP y API's HTTP las cuales permiten la autentificación y envío de DTE's por parte del contribuyente.

4.1.2.1 Autentificación al SII

Para hacer uso de la API de envío de DTE, es necesario obtener un token de autentificación por parte del SII. Para esto, se deben utilizar dos servicios SOAP: "CrSeed" y "GetTokenFromSeed".

El primero, expone un método para obtener una semilla, la cual es una cadena de texto generada aleatoriamente por el SII para facilitar la autenticación. Esta semilla debe ser compuesta en un documento XML, el cual debe ser firmado con la llave privada del contribuyente. Este documento XML luego se envía en el segundo servicio "GetTokenFromSeed", el cual valida la firma y entrega un token de autenticación, el cual sirve para hacer uso de otras API del SII en nombre del contribuyente. El token es válido por un corto periodo de tiempo, por lo que se recomienda que cada operación genere su propio token.

Para acceder a las API SOAP desde el componente, se utilizó Savon[36], cliente que permitió la obtención de los WSDL del SII y el envío de solicitudes.

4.1.2.2 Envío de DTE al SII

Una vez que se obtiene el token de autenticación, se puede realizar el envío de DTE al SII mediante la API HTTP definida para tal propósito (/cgi_dte/UPL/DTEUpload), mediante un POST se envía la información del DTE junto con el token obtenido. En caso de éxito, el SII responde con un XML, el cual contiene el TrackID del envío, el cual es un número de seguimiento para hacer la posterior consulta del estado de este. En caso contrario, este XML también retornará el status de envío, mostrando el código y razón del fallo.

Para acceder a esta API HTTP, se utilizo Unirest[37], el cual es un cliente de solicitudes HTTP que permite el envío de archivos (mediante solicitudes "mutlipart").

---

[36] SavonRB. Heavy metal SOAP client [en línea] <https://github.com/savonrb/savon> [consulta: 26 octubre 2016].

4.1.3 Consulta de envío de los DTE

Una vez que el envío haya sido realizado al SII, y este haya procesado y probado su validez, el SII podrá determinar el estado de aceptación del documento. Es fundamental verificar que cada envío de DTE resulte aceptado, o sino se estarían utilizando documentos inválidos en los pasos posteriores.

La consulta al SII se puede realizar a través del sitio web, con el TrackID del envío a consultar, así como también por medio de las API designadas para tal propósito. Para efectos del componente, nos interesa la versión automatizada mediante las API.

La consulta de los envió se realiza mediante una API SOAP ("QueryEstUp"), la cual usa como parámetros el TrackID del envío a consultar, el RUT del contribuyente y un token de autenticación (el cual se obtiene de la misma manera de que se utiliza para envío de DTE). La respuesta contiene un XML el cual contiene el estado de aceptación de cada tipo de DTE que formaba parte del envío.

4.1.4 Intercambio de DTE entre contribuyentes

Se requiere que una vez que el DTE sea aprobado por el SII, se haga envío del DTE al contribuyente receptor mediante correo, el cual debe contener una copia del XML enviado al SII. En el caso de ser receptor de los DTE, se requiere generar mensajes de respuesta (acuses de recibo) en el formato apropiado para ellos por el SII. Los mensajes de respuesta son archivos XML que deben ser enviados al emisor del DTE al cual se responde, y contienen información de aceptación al DTE según validaciones que el componente deberá realizar. Dependiendo del estado de conformidad del contribuyente receptor del DTE, este también puede optar por rechazarlo mediante una justificación expresada en una glosa.

4.1.4.1 Tipos de Mensaje de Respuesta

Según el nivel de detalle de la respuesta, el SII define dos tipos de mensaje de respuesta: El primero permite responder a la totalidad del envío, y es considerado el acuse de recibo de este, el segundo tipo permite la revisión individual de cada DTE compuesto en un envío, permitiendo aceptar, aceptar con reparos, o rechazar por separado cada uno de estos. Ambos tipos siguen un esquema XML común, y comparten campos obligatorios (carátula con datos del contribuyente emisor de la respuesta, y firma electrónica del XML). El esquema XML que define las respuestas es "SiiDte RespuestaEnvioDTE_v10.xsd".

---

[37] Mashape. Unirest in Ruby: Simplified, lightweight HTTP client library [en línea] <https://github.com/Mashape/unirest-ruby> [consulta: 26 octubre 2016].

Adicionalmente, para efecto de Ley N°19983, el SII define un tercer tipo de respuesta, la cual da cuenta del acuse de recibo de mercaderías o servicios prestados. Esta respuesta solo deja en claro la recepción conforme de bienes por parte del emisor, por lo que debe ser realizado posteriormente efectuado la aceptación correcta de los DTE, y solo si los bienes son recibidos de manera conforme. El esquema XML que define este tipo de respuesta está definido en "SiiDte EnvioRecibos_v10.xsd".

4.1.4.2 Validaciones a realizar

Para el primer mensaje de respuesta, independiente del rechazo o conformidad que el contribuyente pueda determinar sobre el DTE, es necesario realizar comprobaciones de rigor, las cuales también pueden llevar al rechazo del envío. Estas consisten en validación contra el esquema XML del documento, validación de las firmas electrónicas utilizadas, y validación de que el RUT del receptor del documento (aquel que lo está revisando), sea el correcto.

Para la validación de las firmas electrónicas, se utilizó la gema xmldsig[38], la cual funciona con ayuda de Nokogiri y OpenSSL para la validación de firmas XMLDSIG 1.1.

4.1.5 Generación de copias impresas de los DTE

Para aquellos contribuyentes que continúen operando de forma manual, se requiere que el DTE sea enviado a estos bajo cierto formato, en un documento impreso. El documento debe ser un archivo PDF, el cual debe reflejar la información de su contraparte electrónica.

Detalles relevantes sobre la impresión, son que debe provenir de un documento PDF, y deben contar con la generación de un código de barras bidimensional PDF417, el cual contiene información codificada del DTE, a modo de permitir verificar su autenticidad.

Para generar las copias impresas en formato PDF, se utilizó Prawn[39].

4.1.5.1 Sobre el formato de las muestras impresas

El SII define un reglamento a seguir respecto al formato que deben contener las muestras impresas[40]. A continuación se expone un resumen del formato exigido, para aquellas muestras impresas en formato hoja papel:

---

[38] benoist. Implementation of the xmldsig specification [en línea] <https://github.com/benoist/xmldsig> [consulta: 26 octubre 2016].
[39] PrawnPDF. Fast, Nimble PDF Writer for Ruby [en línea] <https://github.com/prawnpdf/prawn> [consulta: 26 octubre 2016].

Dimensiones para hojas de papel:

- Dimensión mínima de muestra impresa: 21,5 x 11 centímetros. (largo y ancho o viceversa) (1/3 de papel oficio).
- Dimensión máxima de muestra impresa: 21,5 x 33 centímetros. (largo y ancho o viceversa) (tamaño papel oficio).

Borde sin letras: Mínimo de 0,5 centímetros.

Logotipo (opcional): Posicionado en lado superior izquierdo sin interferir información de muestra impresa. No debe ser mayor a 1/5 del documento.

Zona superior izquierda: Debe contener los siguientes datos del emisor del documento:

- Razón social del emisor
- Giro del emisor
- Dirección casa del emisor
- Dirección sucursales

La información del emisor debe ser idéntica a la dispuesta por el SII, la cual es consultable en su sitio web[41].

Recuadro de documento: Debe contener el nombre de documento al cual hace referencia la muestra impresa junto con el RUT del contribuyente y Folio utilizado. Bajo el recuadro se debe detallar la dirección regional o unidad SII correspondiente al emisor.

- Tamaño mínimo: de 1,5 x 5,5 centímetros. en el lado derecho color negro o rojo, enmarcado por una línea de entre 0,5 a un milímetro de grosor.
- Tamaño máximo del recuadro: 4 x 8 centímetros.
- Letras: tamaño igual o superior a 10 en alta y negritas.

Cuerpo del documento: Debe indicar los siguientes campos:

- Fecha de emisión
- Datos del receptor

---

[40] Servicio de Impuestos Internos. Manual de muestras impresas [en línea] <http://www.sii.cl/factura_electronica/factura_mercado/manual_muestras_impresas.pdf> [consulta: 26 octubre 2016].

[41] Servicio de Impuestos Internos. Datos para impresión de documentos tributarios [en línea] <https://maullin.sii.cl/cvc_cgi/dte/pe_construccion_dte> [consulta: 26 octubre 2016].

    - Razón social del receptor
    - RUT del receptor
    - Giro del receptor
    - Dirección del receptor
- Datos de referencia (si corresponde)
    - Tipo de documento (si corresponde)
    - N° Folio
    - Fecha de emisión de documento referencia
    - Motivo
    - Otros
- Zona de detalle (Ítems y descuentos y recargos)

Zona inferior del documento: Debe incluir los siguientes campos:

- Zona de totales
- Recuadro acuse de recibo (Ley 19.983)
- Timbre electrónico (a una distancia mayor o igual a 2 cm de borde izquierdo del documento
- Leyenda de destino (solo para copia cedible) en zona inferior derecha del documento.

4.1.5.2 Sobre el código de barras bidimensional PDF417

La información que debe contener el código de barras PDF417 corresponde a la información del nodo <TED> del XML respectivo al DTE. Este nodo trae información básica del documento (tipo de documento, monto total, datos del emisor e información del primer ítem detallado en el documento) y la información del timbre electrónico. Por lo cual se puede verificar a partir de este que:

- El tipo de documento emitido es el correcto
- El folio utilizado para este documento es válido
- El emisor es el correcto, y
- El monto declarado en el documento es el correcto

La generación del código PDF417 fue realizada con ayuda de la gema PDF417[42].

[42] American Society for Engineering Education. A Ruby wrapper for the PDF417 barcode library [en línea] <https://github.com/asee/pdf417> [consulta: 26 octubre 2016].

4.1.6 Generación y envío de información de compras y ventas

Los contribuyentes electrónicos están obligados a llevar la emisión de sus libros de compra y venta en formato electrónico, los cuales deben contener la información de compras y ventas de un periodo mensual. El plazo que los contribuyentes poseen para enviar esta información, es hasta el final del próximo mes (al cual hace referencia el periodo).

Cada libro de compra y venta consiste en un archivo XML, definiendo una estructura similar a la de los realizados para los DTE: Presencia de un código de autorización para la emisión de libros (CAL) y una firma electrónica sobre la información del libro contable, utilizando el estándar anteriormente mencionado XMLDSIG 1.1.

4.1.6.1 Estructura de los XML

Los XML de libros de compra y ventas están definidos por el esquema "XMLSiiDte LibroCV_v10.xsd". La información de cada libro de venta queda definida en un nodo padre llamado <LibroCompraVenta>, este nodo tiene como hijo <EnvioLibro> y <Signature>, el cual es una firma electrónica realizada sobre los contenidos del primer hijo.

<EnvioLibro> contiene <Caratula>, la cual posee información del envío, y las zonas de resúmenes y detalles respectivamente, las cuales se definen mediante la inclusión del nodo <ResumenPeriodo> con sus hijos <TotalesPeriodo>, y nodos tipo <Detalle>. Finalmente, se inserta el código de autorización de libros (CAL) dentro del <EnvioLibro> en el nodo <LceCal>.

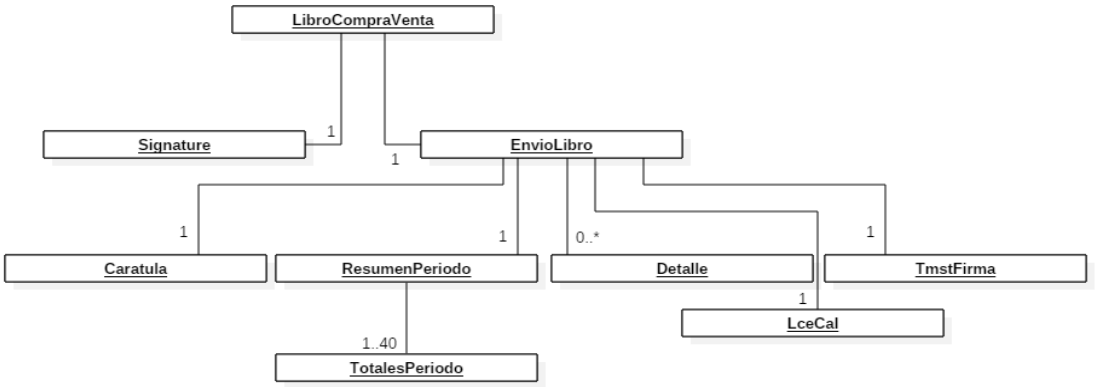

Figura 4.2: Esquema de formato XML de un LibroCompraVenta.

Para la realización de la firma electrónica bajo el nodo <Signature> se asistió el proceso utilizando la gema Signer[43], de manera similar como se realizaron las firmas XMLDSIG para los EnvioDTE.

4.1.6.2 Sobre la obligatoriedad de los detalles

Los libros de compra y venta deben registrar todo el detalle de los documentos tributarios emitidos en el periodo, detalles que deben ser resumidos en una sección que totaliza los movimientos agrupándolos por tipo de documento tributario.

En el caso de algunos documentos manuales, solo es necesario proveer los totales en la sección de resumen, sin hacer uso del detalle para cada uno de estos documentos, algunos de los documentos exceptuados son: Boleta de Ventas y Servicios (afecta) (Código 35), Boleta de Ventas y Servicios no afectos o exentos de IVA (Código 38), Boleta electrónica (afecta) (Código 39), Boleta no afecta o exenta electrónica (Código 41) y Comprobante de pago electrónico (Código 48)

4.1.6.3 Alimentación de información para los libros de compra y venta

La información presente en los libros de compra y venta puede ser completada parcialmente de aquella que está almacenada en el sistema. Lo que no puede ser completado corresponde a documentos tributarios de origen manual, los cuales deben ser digitados de forma manual. El sistema encargado debe facilitar que estos libros parciales sean generados, e integrar métodos para permitir completar la información faltante.

4.1.7 Herramientas de generación de pruebas para certificación

Cada contribuyente que desee operar con software de mercado o elaboración propia, debe pasar por las pruebas de certificación que el SII envía. Si bien la certificación da cuenta de que las otras responsabilidades estén cubiertas de forma efectiva, el proceso de certificación requiere que la preparación de estas cumpla con ciertas condiciones especiales. Por esta misma razón, aunque la existencia de herramientas para la certificación no es necesaria, otorga un valor agregado al componente ya que permite disminuir el esfuerzo a la hora de integrar a un sistema y certificar un contribuyente.

En particular, interesa la implementación de herramientas que apoyen en 3 de las 4 fases del proceso de certificación: generación de envíos únicos que contengan todos los DTE del set de pruebas (incluyendo también la generación de los libros de compra y venta), set de simulación

---

[43] ebeigarts. WS Security XML Certificate signing for Ruby [en línea] <https://github.com/ebeigarts/signer> [consulta: 26 octubre 2016].

a partir de la información requerida, y generación automatizada de muestras impresas a partir de los set de prueba y simulación. La fase de intercambio entre contribuyentes no requiere un tratamiento especial en relación a su uso fuera de certificación, ni tampoco supone un esfuerzo considerable, por lo que la inclusión de herramientas para este queda descartada.

4.2 Especificaciones de Sistema ERP

4.2.1 Descripción General del Sistema ERP

El sistema a construir consiste en una herramienta de gestión pensada para las necesidades de una empresa de rubro maderero y de construcción de muebles, el cual fue exigido por un cliente para su realización. El sistema debe ser capaz de apoyar el proceso de ventas de la empresa, llevando un seguimiento de las órdenes pedidas por los clientes, además de integrar funciones que permitan manejar el stock de productos (inventario).

Las órdenes deben ser emitidas a partir de un vendedor autorizado en el sistema, las cuales se hacen a un cliente existente en la base de datos. Los productos que puede comprar el cliente son los insumos propios del dominio del rubro: herramientas de construcción, tablas/planchas de madera y cubiertas postformadas, estas últimas, difieren en que son elaboradas aplicando una lámina de color en su cubierta. De esta forma, el sistema debe llevar un apropiado control de los artículos y láminas a utilizar en los pedidos.

Debido a que el precio de cada cubierta difiere por sus dimensiones, color de lámina utilizada y complejidad de la forma utilizada, el precio debe poder ser ajustado de forma manual. El sistema debe proveer precios referenciales de las láminas, los cuales deberán poder ser consultados a la hora de realizar las órdenes de pedido. El vendedor puede agregar recargos y descuentos a que afectan a sólo un ítem de la orden, o globales, que se aplican al final de la orden y afectan a todo lo pedido.

A partir de las órdenes de venta, se tendrá la opción de generar las facturas y guías de despacho correspondiente, por lo que la interacción con el componente se generará entre estos modelos. El vendedor tendrá la opción de facturar la totalidad de la orden de venta, o solo parte de esta, y el sistema se deberá encargar de llevar un control de lo facturado y lo que falta por facturar. Este comportamiento deberá ser idéntico para el caso de las guías de despacho, trabajando con una numeración independiente a la de las facturas.

Sobre las otras funcionalidades de facturación electrónica mencionadas en la sección 1.2.6, estas también son necesarias de integrar (generación de notas de crédito y notas de débito, generación de DTE impresos, generación de información de compras y ventas, e intercambio con otros contribuyentes).

Una descripción completa de los casos de uso requeridos a construir se encuentra presente en el anexo A, “Casos de uso de sistema ERP”.

4.2.2 Modelo de datos

A partir de la definición dada del sistema ERP, y siendo complementado por los casos de uso del sistema, se construye un modelo conceptual de datos, el cual se utilizara para la implementación del sistema.

Aunque el componente de facturación se encargará de emitir los documentos tributarios, la persistencia necesaria para mantener la información relevante necesaria para estos deberá estar contenida en el modelo. Esto permite al componente desligarse de esa tarea, y reducir las dependencias asociadas, permitiendo su reuso en un mayor número de casos y plataformas. De modo similar, los modelos asociados a la administración de correos, quedarán también situados en el sistema.

Debido a la complejidad del sistema el modelo fue separado en tres partes: Modelo de sistema ERP, modelo de facturación, y modelo de administración de correos.

El primero interactúa con el modelo de facturación en los modelos “Ítem pedido” y “Nota pedido”, por lo que aparecen replicados en ambos esquemas.

El tercer modelo, de administración de correos, no tiene interacciones con el resto de los modelos, el modelo “DteMailConstant” no presenta relaciones con el resto de los modelos, ya que su función consiste en solo almacenar opciones de configuración para las funciones de correo en la aplicación.

Los esquemas generados se encuentran en el anexo B, “Esquemas de modelos de datos”.

4.3 Implementación de componente y sistema ERP

4.3.1 Estructura de aplicación Rails

Para entender cómo funciona una aplicación Rails, y como se desarrollará la solución, es necesario entender la estructura de una aplicación tradicional Rails. La jerarquía de directorios que define una aplicación Rails tradicional es la siguiente:

```
|-- app
|   |-- assets
|   |-- controllers
|   |-- helpers
|   |-- mailers
|   |-- models
|   `-- views
|       `-- layouts
|           `-- application.html.erb
|-- bin
|-- config
|   |-- environments
|   |   |-- development.rb
|   |   |-- production.rb
|   |   `-- test.rb
|   |-- initializers
|   |-- locales
|   |-- routes.rb
|   `-- secrets.yml
|-- db
|-- Gemfile
|-- Gemfile.lock
|-- lib
|   |-- assets
|   `-- tasks
|-- log
|-- public
|-- test
|-- tmp
`-- vendor
```

Figura 4.3: Jerarquía de directorios de una aplicación Rails tradicional.

El directorio “app”, contiene la mayoría del código propio de la aplicación web a implementar, en esta sección recaen los tres componentes típicos del MVC: los modelos en la carpeta “models”, los controladores en la carpeta “controllers” y las vistas en la carpeta “views”. Este directorio también contiene la carpeta “assets”, la cual contiene recursos para ser embebidos en las vistas generadas (hojas de estilo CSS, archivos javascript para ser ejecutados en el lado del cliente, imágenes y otros), la carpeta “helpers”, en la cual se definen funciones de soporte, para asistir procesos de los componentes MVC, y la carpeta “mailers”, la cual contiene funciones específicas al envío de mails de la aplicación[44].

El directorio “bin”, contiene vínculos a ejecutables de las gemas del proyecto, son automáticamente generados y permiten ejecutar tareas de las gemas, desde la raíz del directorio de Rails, sin conocer la ubicación exacta de su instalación (siendo estos conocidos como “binstubs”).

El directorio “config”, contiene los archivos de configuración de la aplicación, en el subdirectorio “environments”, se puede especificar configuraciones que solo aplican a un

[44] SitePoint. A Quick Study of the Rails Directory Structure [en línea] <https://www.sitepoint.com/a-quick-study-of-the-rails-directory-structure> [consulta: 26 octubre 2016].

ambiente en que la aplicación es lanzada (por defecto, está la distinción entre ambiente de desarrollo, ambiente de producción y ambiente de pruebas), el subdirectorio "initializers" define archivos de configuración que son ejecutados al momento de lanzar la aplicación, y son generalmente usados para la configuración de gemas asociadas al proyecto. El subdirectorio "locales", en donde se puede definir distintas localizaciones de la app para cada lenguaje, el archivo "routes.rb", el cual cumple la función de enrutador, redirigiendo las llamadas de los clientes a las respectivas acciones de los controladores, y el archivo "secrets.yml", el cual contiene información sensible de la app como credenciales de autenticación, los cuales quedan a disposición de la aplicación. Para mantener las credenciales del ambiente de producción, este último archivo es generalmente excluido del sistema de versionamiento, y su configuración se mantiene protegida sólo en el ambiente de producción.

El directorio "db", contiene los archivos relacionados con la base de datos, aquí se almacenan las bases de datos SQLite (generalmente utilizadas para entornos de desarrollo), las migraciones (archivos que son utilizados de manera secuencial para definir y mantener al día el esquema de base de datos de la aplicación), y los seeds (archivos utilizados para alimentar la base de datos, generalmente con información de prueba, en ambientes de desarrollo).

El archivo "Gemfile", en el cual la aplicación Rails define las dependencias de gemas a usar, este archivo es generado por defecto en la instalación de Rails, ya que contiene algunas dependencias básicas que Rails requiere, y otras sugeridas. Al momento de instalar las gemas, se revisa este archivo y se resuelven las dependencias de cada gema, así como las versiones a instalar de estas. El archivo "Gemfile.lock" asiste el proceso de instalación, congelando las versiones de cada gema que fueron resueltas al momento de la instalación. Esto permite al momento de colaborar, que se mantengan las mismas versiones de cada gema a utilizar[45].

El directorio "lib", contiene librerías de soporte que la aplicación pueda utilizar, y que no clasifiquen dentro de una gema o un helper. El subdirectorio "assets" está pensado para mantener recursos externos a la aplicación, y que sean usados por "lib". El otro subdirectorio, "tasks", permite definir tareas ejecutables por la utilidad rake, la cual corre a través de línea de comandos (CLI). Normalmente se recomienda mantener un tamaño moderado de código en "lib", y si este crece mucho, desligarlo del proyecto y generar una gema con tal funcionalidad. En el caso de la implementación a realizar, las funcionalidades que forman parte de la facturación electrónica, son posicionadas en este directorio, para luego constituir una gema. Los detalles de este proceso son explicados con detención más adelante.

El directorio "log" mantiene los archivos con los registros de actividad reportados por el sistema. El directorio "public" contiene archivos que son accesibles públicamente por los

---

[45] StackOverflow. Gemfile.lock Use in Rails? [en línea] <http://stackoverflow.com/questions/9208240/gemfile-lock-use-in-rails> [consulta: 26 octubre 2016].

clientes, y algunas vistas relacionadas con mensajes de error típicos (HTTP 404, 422 y 500). En esta carpeta residen los recursos de assets en ambiente de producción, una vez que estos son compilados.

Luego está el directorio “tests”, en la que residen los archivos de pruebas utilizados para hacer testing funcional de la aplicación, el directorio “tmp” donde residen archivos generados temporalmente por Rails, y el directorio “vendor”, en donde se contienen los recursos externos a la aplicación.

4.3.2 Estructura de aplicación a desarrollar

Para la implementación del sistema a desarrollar, gran parte de la estructura tradicional de una aplicación Rails se mantendrá sin cambios. Los modelos contienen las entidades del modelo con sus respectivas validaciones y reglas, los controladores establecen el vínculo entre la lógica transaccional de los modelos y el despliegue de información de las vistas, definiendo por lo bajo acciones suficientes para realizar un CRUD de un modelo (creación, lectura, actualización y eliminación de entidades). En los casos de uso más complejos, los controladores son provistos de acciones adicionales para responder a los clientes, y llaman a más de un tipo de entidad del modelo para presentar o manejar la información.

La interfaz de usuario es generada por las vistas, las cuales son asistidas por javascript para definir el comportamiento del frontend. Se utilizan componentes de Bower[46] para integrar paquetes “front-end” en la aplicación, consistentes primordialmente en librerías javascript, junto con hojas de estilo CSS e imágenes.

En el caso de la implementación de código relacionado con facturación electrónica, se utilizó una estructura distinta a la usual. Las diferencias son detalladas en la sección 4.1.3.4.

4.3.3 Gemas a utilizar

Para cubrir algunas de las funciones necesarias a cumplir por el sistema, se utilizaron gemas disponibles en el repositorio RubyGems, las más relevantes son mencionadas a continuación.

- devise para la autenticación y registro de usuarios en el sistema (vendedores)
- cancancan para manejar la autorización de el sistema (que acciones pueden concretar cada tipo de usuario).

[46] Bower. Bower: A package manager for the web [en línea] <https://bower.io> [consulta: 26 octubre 2016].

- ajax-datatables-rails para asistir la creación de datatables (componente javascript para generación de tablas dinámicas con opciones de filtrado y búsqueda), kaminari para la paginación en estas.
- rut_chileno, para la validación de RUT en el backend
- pg, que es requerido por el ORM de Rails (ActiveRecord) para manejar el motor de base de datos Postgresql, el cual fue utilizado en ambientes de desarrollo y producción.
- sass-rails, para el uso de hojas de estilo SCSS en la aplicación.
- prawn, para la generación de PDF lo cual se usa generar historial de clientes
- axlsx, para la generación de hojas de cálculo XLS, y roo para la lectura de estas, lo cual se usa para exportar/importar artículos entre el sistema y archivos XLS.
- bower-rails para asistir el proceso de instalación e integración de componentes de "front-end" desde el repositorio de Bower.
- paperclip para adjuntar archivos a modelos, usado para guardar los DTE, recibos de DTE y el certificado a utilizar. Los archivos se almacenan en el servicio de almacenamiento en la nube Amazon S3, con ayuda de la gema aws-sdk, la cual contiene las credenciales y autoriza el uso del almacenamiento.

En el ambiente de pruebas, se utilizaron adicionalmente las siguientes gemas:

- rspec como framework de testing en la aplicación, utilizado para realizar pruebas de integración e unitarios.
- capybara para realizar pruebas frontend en la aplicación.
- poltergeist como driver javascript, el cual se complementa a capybara para hacer pruebas frontend que dependen de scripts javascript en la aplicación.
- sqlite3, utilizado para manejar el motor de base de datos SQLite en ambiente de pruebas.
- factory_girl para crear instancias de modelos en la aplicación para el ambiente de pruebas.
- database_cleaner para purgar la base de datos entre las pruebas realizadas.
- faker para alimentar las instancias de los modelos con información ficticia.

4.3.4 Uso de engines en la aplicación

Con ayuda de las gemas y del framework Rails, se pretenden implementar los casos de uso que el sistema requiere. Para aquellos que tengan un vínculo con las funcionalidades de facturación electrónica, deberán ser resueltos con ayuda del componente.

Las llamadas al componente deberán ser realizadas a través de la interfaz definida para realizar estas acciones. Es necesario que exista un código "puente", encargado de establecer el vínculo entre entidades propias del dominio de la aplicación (y del framework rails) y las llamadas al componente, dando el formato que el componente requiere para que estas

funcionen de forma apropiada. De forma similar, las respuestas necesitan ser interpretadas y procesadas por la aplicación, si esto fuera necesario.

Dado que el componente de facturación electrónica se desarrolla como gema, no puede estar ligado con entidades internas de la aplicación, y por lo tanto no debe conocer de las facilidades de framework Rails, ni de los modelos propios de la aplicación. Esto implica que todas las funciones de persistencia de información, deben ser realizadas fuera del componente. De manera similar con las vistas relacionadas, ya que estas deben ser generadas por la aplicación con ayuda del modelo MVC propuesto por Rails,

Este desentendimiento del componente, puede ser desventajoso para esta implementación en particular, pero de esa forma el componente se vuelve mucho más portable. Podrá ser integrado a otras aplicaciones Ruby, u otros lenguajes, utilizando solicitudes HTTP por ejemplo. Ya que el componente no se encarga de la lógica de almacenamiento, ni de la presentación, podrá ser utilizado en un número mucho más amplio de contextos.

La interacción con el componente quedará resuelta por una parte de la aplicación. Para reducir la complejidad agregada que esto puede provocar, es necesario que la interacción entre la interfaz del componente y la aplicación sea lo mínima posible, y que esta parte de la aplicación (el delegado de comunicarse con el componente), esté separado del resto de la aplicación. Esta parte también debe ser la encargada de mantener los modelos relacionados de facturación electrónica, junto con las vistas que sean requeridas por los casos de uso relacionados con esta.

Otro beneficio de realizar esta separación, es que para efectos de desarrollo, el código relacionado del componente se puede implementar en este módulo de la aplicación. Dado que el componente y la aplicación fueron desarrollados en paralelo, resulta una manera más natural de trabajar en la creación de un componente reusable. Una vez que el componente se encuentre funcional y haya sido probado apropiadamente, el código se puede extraer de la aplicación, siendo empaquetado en una gema y publicándola.

Para realizar esta división del módulo de la aplicación relacionado con funciones de facturación electrónica, se utilizaron engines de Rails, como es propuesto en [13]. Los engines de Rails pueden ser considerados pequeñas aplicaciones, las cuales se pueden montar en una aplicación padre, otorgándole sus funcionalidades. A nivel de clases, una aplicación Rails hereda del engine Rails (Rails::Application y Rails::Engine respectivamente), por lo que sus comportamientos son similares[47].

---

[47] RailsGuides. Getting Started with Engines [en línea] <http://guides.rubyonrails.org/engines.html> [consulta: 26 octubre 2016].

Es necesario explicar la diferencia de utilizar un plugin, ya que estos son generados de manera similar por la CLI de Rails, utilizando “rails plugin new”. Un plugin consiste en una librería embebida dentro de la aplicación (utilizando una carpeta lib), de comportamiento similar al de una gema corriente, permitiendo su posterior reuso a través del Gemfile entre aplicaciones, o su distribución por RubyGems. Un engine a diferencia, además de contener el directorio lib, trae consigo la estructura típica de una aplicación Rails.

Un engine igualmente puede ser distribuido como una gema, aunque esto es menos frecuente, dado la dificultad de que una parte de la aplicación, conteniendo vistas y modelos, sea ampliamente reusable por otros usuarios.

Existen dos tipos de engines los cuales varían ligeramente en su estructura, utilizando distintas opciones en la línea de comandos. El engine tipo --full (completo) genera una estructura de directorios similar al de una aplicación Rails normal: app está presente, permitiendo la construcción de vistas, modelos y controladores en el modelo. El directorio bin también es incluido, lo que permite la ejecución de comandos Rails dentro del contexto del engine, lo que permite que un engine pueda generar modelos, vistas, o scaffolds, de manera idéntica al de una aplicación Rails corriente. Incluso se pueden generar migraciones, las cuales serán propiedad del engine, residiendo en su propio directorio db/migrations.

Los contenidos del engine son directamente añadidos a la aplicación padre, lo cual puede provocar colisiones por ejemplo, a la hora de definir modelos. Si la aplicación padre comparte el nombre de clase de un modelo en particular, se provocará un comportamiento inesperado, y existirá una incerteza en la aplicación al momento de integrar, dependiendo del orden en que se carguen las dependencias. Este problema se acentúa sobre todo cuando el número de engines en la aplicación crece, ya que las colisiones son mucho más probables.

El otro tipo de engine, --mountable (montable), resuelve exactamente el problema anteriormente mencionado. Cada parte del engine es contenido en un namespace único, idéntico al nombre del engine, lo cual también se propaga para la generación por línea de comandos. Estos engines dejan de ser montados directamente a la aplicación, por lo que para agregarlos, es necesario montar las rutas de estos en la aplicación padre. Este es uno de los pocos puntos en donde los engines montables interactúan con la aplicación padre, ya que se evita cualquier interacción innecesaria, maximizando el grado de aislación que el engine tiene respecto a su aplicación padre.

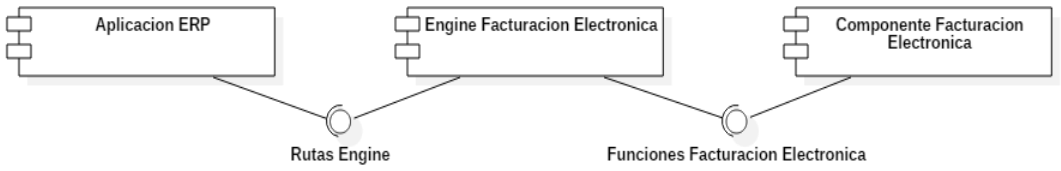

Figura 4.4: Diagrama de dependencias entre partes de la aplicación ERP con el componente de facturación electrónica.

La estructura de la aplicación, junto con la utilización del engine y del componente de facturación electrónica, queda demostrada en la figura 4.4. La aplicación ERP se conecta con el engine de facturación electrónica montando sus rutas en routes.rb y explicitando su dependencia en el Gemfile. Por otra parte, el engine se conecta con el componente de facturación electrónica explicitando su dependencia en el gemspec del engine, el cual es utilizado para definir las dependencias de una gema, similar al Gemspec para una aplicación rails. Debido a que la gema de facturación electrónica no está publicada, el engine debe además explicitar su dependencia como gema local en su Gemfile correspondiente.

```
# ERP/Gemfile
source 'https://rubygems.org'
# ...
# ...
# Modules
path "modules" do
 gem "facturacion_electronica"
end
```

Figura 4.5: Gemfile de aplicación ERP definiendo requerimiento de engine de facturación electrónica.

```
# ERP/components/facturacion_electronica/Gemfile

source "https://rubygems.org"

path "gem" do
 gem "facturacion_electronica_gem"
end

gemspec
```

Figura 4.6: Gemfile de engine de facturación electrónica definiendo que la gema de facturación electrónica se encuentra disponible localmente.

```
# ERP/modules/facturacion_electronica/facturacion_electronica.gemspec
Gem::Specification.new do |s|
 # ...
 s.add_dependency "facturacion_electronica_gem"
 # ...
end
```

Figura 4.7: Gemspec de engine de facturación electrónica definiendo requerimiento de gema (componente) de facturación electrónica.

La aplicación a desarrollar es suficientemente compleja como para que más engines montables puedan ser generados, repartiendo la aplicación en varias sub-aplicaciones, cada una de ellas exponiendo sus rutas para ser montadas en la aplicación principal. Dado que está fuera del enfoque de este trabajo, esta división no fue realizada, aunque es deseable.

4.3.5 Implementación del componente

Como se sugirió en la sección anterior, la implementación del componente fue creada en paralelo con la del engine, y fue desarrollado como una gema la cual hace uso el engine para funcionar. Esta gema contiene las funcionalidades base, que están relacionadas con facturación electrónica, sin manejar la lógica de despliegue de información, ni persistencia de datos.

El componente a crear no depende del framework Rails para su funcionamiento, permitiendo ser usada en otros contextos. Aun así, presenta dependencias en algunas gemas del ecosistema de Ruby para asistir la implementación de sus funciones:

- Nokogiri para la lectura y creación de archivos XML, utilizado en los documentos tributarios generados, en la lectura de los recibidos, generación de los acuse de recibo y para realizar el proceso de autentificación con el SII.
- Signer para efectuar las firmas electrónicas necesarias en los DTE, acuse de recibo y para el proceso de autentificación con el SII.
- Xmldsig para verificar la validez de las firmas electrónicas contenidas en los XML recibidos.
- Savon para acceder a los WSDL del SII, y comunicación con las API SOAP.
- Unirest para realizar solicitudes HTTP a las API del SII que no utilizan SOAP.
- mail para recepción y envío de correos (utilizando los protocolos POP3 y SMTP respectivamente).
- prawn, para la generación de las copias impresas de los DTE en formato PDF.
- pdf417 para la generación del código de barras bidimensional en representación PDF417 en las copias impresas.
- rut_chileno para la validación de RUT dentro del componente.

Para la comunicación con el componente, se definen funciones de acceso público a las clases del componente. Los parámetros de cada llamada toman tipos primitivos, o estructuras de información corrientes en Ruby (OpenStruct). El engine toma información de los modelos, y adapta para que las llamadas se efectúen correctamente.

Capitulo 5: Evaluación y análisis comparativo de componentes

Este capitulo expone el estudio comparativo de las alternativas presentes en el mercado, respecto del componente construido en la primera parte de la propuesta. Con la información recaudada en la sección de estado del arte, se formará una selección de componentes para conformar el estudio.

La rúbrica será sustentada con la ayuda de modelos de calidad para componentes, permitiendo perfilar el desempeño de distintas dimensiones de calidad de cada uno de los componentes participantes en el análisis.

Los resultados obtenidos en esta fase, servirán para más adelante para caracterizar los escenarios de uso favorables de cada uno de los componentes.

5.1 Selección de Componentes

La selección de componentes para hacer parte de la evaluación se desprende de la búsqueda realizada en el estado del arte. Dado que el mercado de componentes relacionados con la facturación electrónica no es tan basto (siendo la mayoría de estas, soluciones de facturación electrónica completos, provistos con un ERP), el proceso de filtrado no fue exhaustivo en relación al cumplimiento de los requerimientos descritos anteriormente. De este modo, los principales puntos de importancia a la hora de definir la selección fueron el costo de evaluar el producto, y la factibilidad de que estos sean probados en fase de certificación (dado que algunos proveedores de soluciones son imperantes en gestionar el proceso de certificación a la hora de entregar la solución, permitiendo solamente utilizarlo bajo el ambiente de producción).

Los componentes seleccionados para formar parte del análisis son los siguientes:

5.1.1 LibFacturista - FacturaElectronicaChile.com

LibFacturista es una solución propuesta por FacturaElectronicaChile.com, la cual consiste en una librería que provee funciones de firma y envío de DTE. La librería es integrada mediante ligado dinámico (en tiempo de ejecución) y es distribuida como una DLL o SO (para funcionamiento en sistemas Windows y Unix). De esta forma, la librería puede ser integrada para ser utilizada en una gran variedad de lenguajes.

Para realizar las pruebas y evaluación del componente, se obtuvo una demo del componente. La cual fue utilizada usando su integración bajo Python 3.4 y la librería ctypes para realizar el vínculo dinámico con LibFacturista.

5.1.2 LibreDTE-lib - LibreDTE

LibreDTE-lib es una librería PHP para la construcción de aplicaciones con funcionalidades de facturación electrónica. La librería es utilizada la aplicación web de uso gratuito LibreDTE, la cual es una aplicación PHP que utiliza el framework SowerPHP. LibreDTE permite a los contribuyentes llevar un control de sus documentos tributarios electrónicos, sin incluir manejo de inventario, órdenes de pedidos, ni otras funcionalidades propias de un ERP (es similar a las funcionalidades otorgadas por el portal gratuito del SII).

Para realizar las pruebas y evaluación de esta solución, se utilizó PHP 5.5.17 para ejecutar LibreDTE-lib y composer, como manejador de dependencias (y de esa forma importar LibreDTE-lib).

5.1.3 Facturación_Electronica

Este es el último candidato a evaluar, y corresponde al componente implementado en el presente trabajo. El componente de facturación electrónica corresponde a una gema de Ruby, la cual fue integrada dentro del engine de facturación electrónica para la implementación del ERP.

Para evaluar el componente, se utilizó la versión de Ruby del ambiente de desarrollo (2.1.5). Las pruebas fueron realizadas con una versión standalone del componente, sin ser integrado al sistema ERP desarrollado.

5.2 Modelos de calidad

Esta sección pretende discutir los modelos de calidad presentados en la literatura, dando un vistazo en los modelos históricos de uso general, y los modelos de calidad especialmente diseñados para componentes de software. Este estudio posteriormente permitirá definir el modelo de calidad en el cual se basará la evaluación de componentes.

5.2.1 Primeros modelos de calidad

La calidad de software puede ser definida como “el grado en que un sistema, componente de sistema o procesos cumplen requerimientos específicos”, o como “el grado en que un sistema,

componente de sistema o proceso cumple las expectativas de un cliente o usuario" [14]. Un modelo de calidad de software es definido como "la serie de características y las relaciones entre estas que proveen la base para especificar y evaluar requerimientos de calidad" [16].

Los primeros modelos de calidad de software, propuestos para uso general, fueron el de McCall (1977) [15], Boehm(1978) [17] y FURPS(1992) [18]. El modelo de McCall define la calidad del producto de software en tres aspectos: Operación del producto, Revisión del producto y Transición del producto, estos tres aspectos se dividen en 11 factores de calidad externos.

El modelo de calidad propuesto por Boehm identifica características que determinan la calidad del código fuente de un software, y construye un árbol de dependencia entre las características. Dicho árbol lo denomina "Árbol de características de calidad de software" y define un nodo base como Utilidad general el cual se separa hasta en 3 niveles. A partir de cada característica hoja, Boehm logra definir métricas, que permiten medir la utilidad general de la calidad. Boehm además propone la correlación del nodo hoja con la calidad total, el beneficio potencial que tendría un asesoramiento en la materia, su facilidad de cuantificar, la facilidad de automatizar la medición, y el grado de completitud que una medición automatizada puede asegurar en la métrica.

FURPS fue originalmente presentado por Robert Grandy, su nombre se descompone en las características propuestas para definir la calidad: Funcionalidad, Usabilidad, Confiabilidad (Reliability), Desempeño (Performance) y Soportabilidad (Supportability), a su vez cada una de las características se separa en sub-características. FURPS ha sido actualizado en FURPS+, el cual añade características al modelo: Requerimientos de diseño, Requerimientos de implementación, Requerimientos de interfaz y Requerimientos físicos[48].

Existen estándares ISO que tratan la calidad de los productos de software: La norma ISO 9126-1 [19] la cual está basada en los trabajos de Boehm y McCall. Esta norma establece dos dimensiones de la calidad del software: Calidad en el uso, los cuales son aspectos de la calidad perceptibles por el usuario final, y Calidad del producto, las cuales son características intrínsecas al producto, y están contenidas en su estructura interna y externa. En el proceso de desarrollo de software, las características internas son influenciadas por la calidad del proceso de desarrollo, siendo a su vez las características internas influenciadoras de la calidad externa del producto. De la misma forma, las características de calidad en uso son influenciadas por las características externas del producto de software.

---

[48] IBM developerWorks. Capturing Architectural Requirements [en línea] <http://www.ibm.com/developerworks/rational/library/4706.html#N100A7> [consulta: 26 octubre 2016].

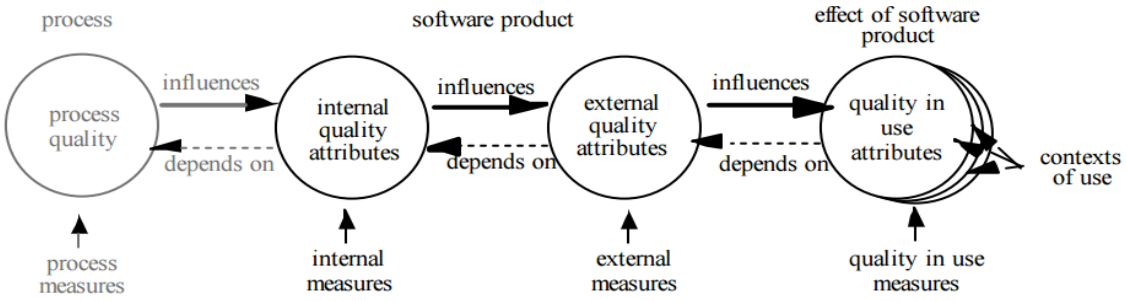

Figura 4.7: Calidad en el ciclo de vida de un producto de software [19].

En base a estas dos dimensiones anteriormente mencionadas, ISO 9126-1 define características y subccaracterísticas las cuales definen la calidad de manera jerárquica. La norma ISO 9126-1 define una jerarquía de calidad de software del cual muchos modelos de calidad posteriores se basan para formular su propuesta.

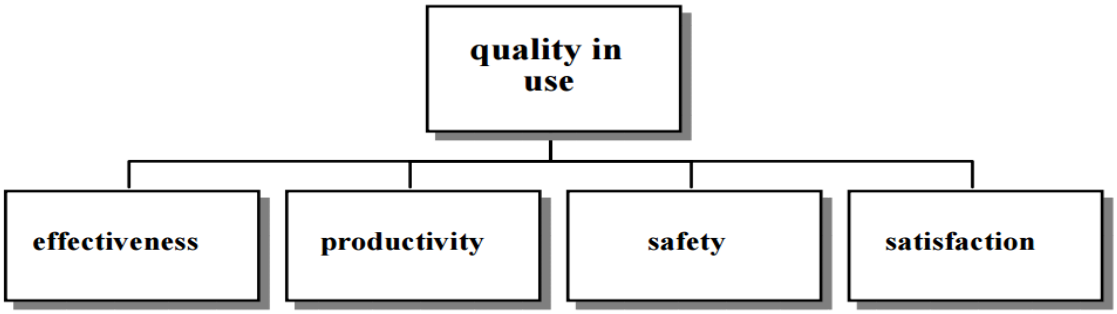

Figura 4.8: Árbol de características y subccaracterísticas para calidad en uso según norma ISO/IEC 9126-1 [19].

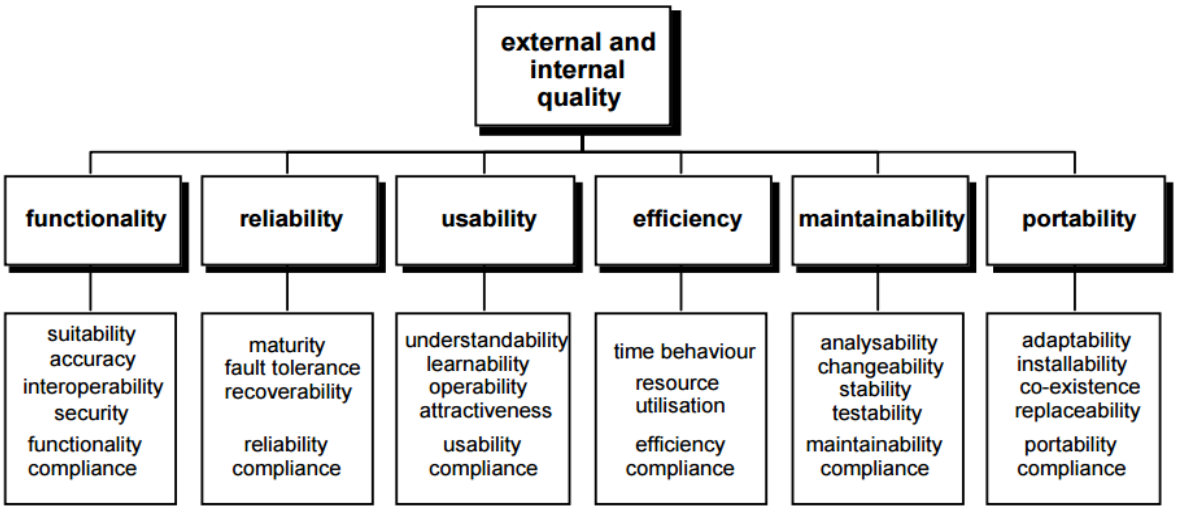

Figura 4.9: Árbol de características y subccaracterísticas para calidad de producto de software según norma ISO/IEC 9126-1 [19].

La norma ISO/IEC 9126-1 es posteriormente actualizada en la ISO/IEC 25010, agregando más características al modelo de calidad del producto (Seguridad y Compatibilidad) reagrupando las subccaracterísticas, además de expandir el modelo de calidad en el uso (redefiniendo características, agregando subccaracterísticas y añadiendo una nueva característica "Cobertura contextual") [20].

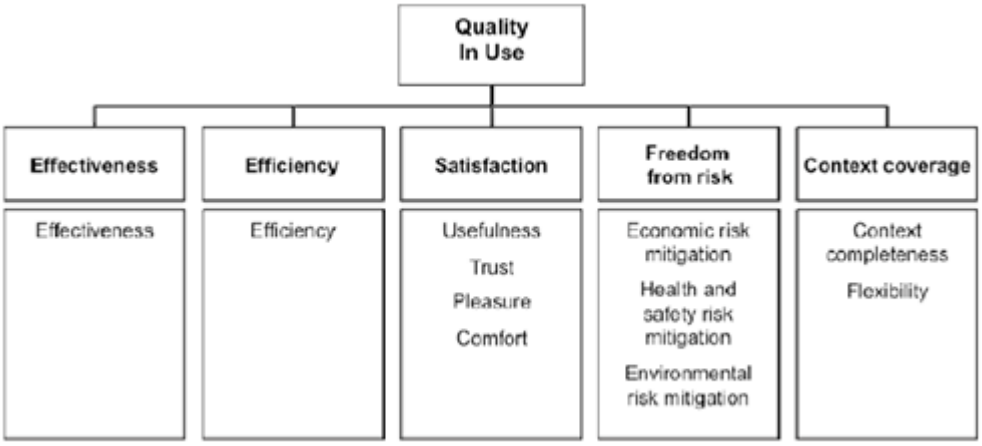

Figura 4.10: Árbol de características y subccaracterísticas para calidad en uso según norma ISO/IEC 25010 [20].

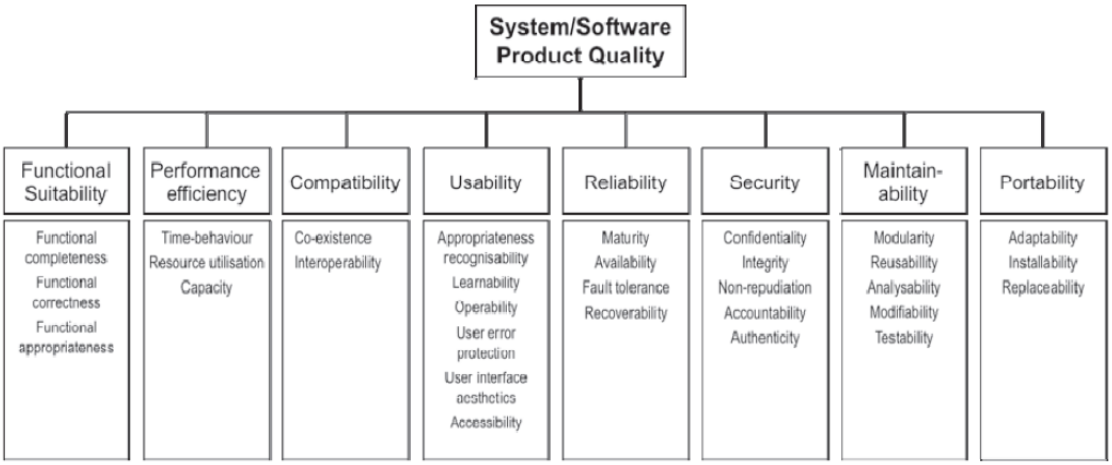

Figura 4.11: Árbol de características y subccaracterísticas para calidad de producto de software según norma ISO/IEC 25010 [20].

5.2.2 Modelos de calidad para COTS

Los componentes COTS ("Commercial off the shelf") son componentes distribuidos comercialmente por un proveedor para su reuso en productos de propiedad de un cliente. Una definición más comprensiva de un componente COTS es aquel que cumple alguna de las

siguientes condiciones: (i) es vendido, prestado o licenciado al público general, (ii) ofrecido por un proveedor tratando de obtener beneficio comercial por el, (iii) es soportado y modificado por el proveedor, (iv) disponible en múltiples copias idénticas o (v) utilizado sin modificación alguna de su estructura interna [5].

La utilización de modelos de calidad de software (en particular, aquellos de uso estandarizado como la ISO o IEEE), han demostrado ser demasiado generales para tratar las características especiales de los componentes de software. Aun cuando algunas características de estos son apropiadas para la evaluación de componentes de software, otras no son aplicables al contexto de componentes [21].

Bertoa et al. proponen un modelo de calidad para apoyar a la evaluación de componentes COTS presentes en el mercado [22]. Para ello, se realiza un análisis en que separa las características en tres agrupaciones: locales o globales (siendo las locales aquellas que pueden estar auto contenidas en un componente), características en tiempo de ejecución (runtime) y propias del producto, y características internas y características externas (siendo las internas la que respecta al funcionamiento interno del componente, y que en casos de componentes tipo "black-box", estas no son de fácil acceso).

Finalmente, el modelo propuesto consiste en una adaptación del estándar ISO 9126 para facilitar su uso en componentes basado en el análisis anteriormente mencionado. El modelo define 5 características y 16 subccaracterísticas, de las cuales proponen métricas para realizar la evaluación.

Tabla 4.12: Características y subccaracterísticas del modelo propuesto por Bertoa et al. para la evaluación de componentes.

Characteristics	Sub-characteristics (Product)	Sub-characteristics (Runtime)
Functionality	Accuracy	Suitability
		Interoperability
		Compliance
	Security	Compatibility
Reliability	Recoverability	Maturity
Usability		Learnability
		Understandability
		Operability
		Complexity
Efficiency	Time Behaviour	
	Resource Behaviour	

Maintainability		Changeability
		Testability

En el trabajo de Álvaro et al. se propone un modelo de calidad para componentes, con el fin de ser utilizado para facilitar los procesos de selección de COTS, con énfasis en realizar una proposición aplicable para uso real en la industria, y para facilitar la certificación de los componentes existentes en el mercado (siendo parte de un proceso más grande, dedicado a la certificación de componentes de software).

El modelo propuesto nuevamente es una modificación del modelo ISO 9126, ajustado a aquellas características accesibles en los componentes, y que son relevantes [21]. Siendo uno de los cambios más notorios la inclusión de la característica negocio (business) en el modelo, la cual refiere a las características de marketing del componente de software, complementando a las otras características de calidad.

Tabla 4.13: Características y subcaracterísticas de modelo de calidad propuesto por Alvaro et al. para la selección de componentes COTS.

Characteristics	Sub-characteristics
Functionality	Suitability, Accuracy, Interoperability, Security, Compliance, Self-contained
Reliability	Maturity, Recoverability, Fault Tolerance
Usability	Understandability, Configurability, Learnability, Operability
Efficiency	Time Behaviour,Resource Behaviour, Scalability
Maintainability	Stability, Changeability, Testability
Portability	Deployability, Replaceability, Adaptability, Reusability
Business	Development time, Cost, Time to market, Targeted market, Affordability

Rawashdeh y Matalkah, proponen un framework para la construcción de un modelo de calidad para la evaluación de COTS, basado en los modelos de calidad ISO 9126, Dromey y ISO/IEC TR 15504-2 [23].

En base a los requerimientos solicitados por el sistema, y las necesidades de cada stakeholder presente en la construcción del sistema, propone la adaptación de estos tres modelos de calidad para responder a la calidad interna y externa del producto, señalando a ISO 9126 y Dromey como los encargados de estas dos dimensiones respectivamente (ISO/IEC TR 15504-

2 es utilizado para complementar la “process-efficiency” y “product-efficiency”, similarmente a como son vistas en el Quality Cube Model).

Utilizando esta técnica, logran definir un modelo de calidad con características similares a las de ISO 9126, más la adición de la característica Gestionabilidad (“Manageability”), la cual responde a la necesidad de poder estimar tiempos y definir plazos en el proyecto que se desea integrar el componente.

Tabla 4.14: Características y subccaracterísticas de producto y proceso para modelo de calidad propuesto por Rawashdeh y Matalkah.

Characteristics	Sub-characteristics (Product)	Sub-characteristics (Process)
Functionality	Accuracy, Security	Suitability, Interoperability, Compliance, Compatibility
Reliability	Recoverability	Maturity
Usability		Learnability, Understandability, Operability, Complexity
Efficiency	Time behavior, Resource behavior	
Maintainability		Changeability, Testability
Manageability	Quality management	

Se propone el modelo de calidad Q’Facto 10, el cual es un modelo de calidad para la evaluación y selección de componentes COTS tipo “black-box”, basado fuertemente en el modelo de calidad ISO 9126. Posteriormente se propone una versión actualizada, Q’Facto 12. La cual es una versión simplificada de Q’Facto 10, con implementación de nuevos estándares basados en la ISO 25010, y poniendo más énfasis en la calidad percibida por el usuario final. Estos dos modelos son de naturaleza jerárquica, y evalúan los COTS en subfactores los cuales son medidos en una escala de 0 a 1 [24].

Singh et al. proponen un framework para la construcción de un modelo de calidad para COTS con énfasis en la reusabilidad. Define un proceso para la construcción de un modelo de calidad basado en las métricas que se descomponen de las metas de calidad requeridas por los stakeholders y propone un proceso para la selección de componentes, utilizando el modelo de calidad anteriormente generado, otorgando pesos a las métricas propias del modelo basado en la especificación de requerimientos provista a cumplir por el componente. Adicionalmente propone un modelo de calidad generado utilizando el framework, inspirado en los atributos más frecuentemente propuestos en la literatura (Funcionalidad, usabilidad, confiabilidad, eficiencia, mantenibilidad y portabilidad) [25].

5.3 Criterios de Evaluación

Para realizar la evaluación de los componentes seleccionados, se construyó una rúbrica de criterios a medir. Una vez obtenidas las medidas de los componentes en relación a las métricas, se podrá perfilar aspectos positivos y negativos de las soluciones.

La rúbrica de criterios de evaluación es confeccionada pensando en cubrir dos aspectos: atributos de calidad, los cuales fueron tomados a partir de los más frecuentemente utilizados en la literatura, para conducir este tipo de análisis, y la cobertura de requerimientos funcionales, los cuales fueron extraídos del modelo de funcionamiento del SII y de las funcionalidades provistas por las alternativas a evaluar, permitiendo medir el grado de cobertura que cada componente posee para realizar el trabajo de componente de facturación electrónica.

Es importante considerar que el enfoque de esta rúbrica es medir la calidad del componente con el fin de ser integrado en un software, no para ser percibido por un usuario final (contribuyente que utiliza software ERP con facturación electrónica). Por esto, la noción de usabilidad para el producto de software cambia, alterando el significado de esta característica.

5.3.1 Atributos de calidad

Para la selección de atributos de calidad, se utilizó como base las características más frecuentemente nombradas en los modelos de calidad para componentes existentes en la literatura [25]. Los atributos son los siguientes:

- Funcionalidad (Functionality): Expresa la habilidad del componente para proveer los servicios y funciones, cuando es utilizado bajo condiciones específicas.
- Confiabilidad (Reliability): Expresa la habilidad del componente para mantener un cierto nivel de desempeño, cuando es utilizado bajo condiciones específicas.
- Usabilidad (Usability): Expresa la habilidad de un componente para ser entendido, aprendido, usado, configurado y ejecutado, cuando es utilizado bajo condiciones específicas.
- Eficiencia (Efficiency): Expresa la habilidad de un componente proveer un apropiado desempeño, en relación al número de recursos utilizado.
- Mantenibilidad (Maintainability): Expresa la habilidad de un componente para adaptarse a lo largo del tiempo.
- Portabilidad (Portability): Expresa la habilidad de un componente para ser transferido de un ambiente operacional a otro.

Para definir las subccaracterísticas, se utilizó el trabajo de Álvaro et al. [21] el cual define un modelo de calidad con las características mencionadas (basadas en ISO 9126), adecuándose al contexto de componentes de software: redefine subccaracterísticas, elimina algunas que son

irrelevantes o no aplican, y agrega cuatro nuevas subcaracterísticas (Autosuficiencia, Configurabilidad, Escalabilidad y Reusabilidad).

Dada la naturaleza y los datos que se poseen para realizar la evaluación de los componentes de facturación electrónica, algunas subcaracterísticas fueron adicionalmente omitidas del modelo. La subcaracteristica pertinente a la cobertura funcional del software, fue extraída para ser revisada en detalle, descomponiéndose en una nueva jerarquía.

Para efectos de este trabajo, la característica de negocio (Business) propuesta en dicho trabajo es dejada fuera de análisis. Aunque es un aspecto de interés a considerar, los distintos tipos de comercialización de los componentes candidatos no lo permiten.

Tabla 4.15: Características y subcaracterísticas de calidad, propuestas para evaluar componentes de facturación electrónica.

Funcionalidad	Adecuación, Precisión, Interoperabilidad, Seguridad, Conformidad, Autosuficiencia
Usabilidad	Comprensibilidad, Esfuerzo de aprendizaje, Configurabilidad
Confiabilidad	Recuperabilidad, Tolerancia a fallos
Eficiencia	Eficiencia temporal, Eficiencia material
Mantenibilidad	Posibilidad de cambiar, Comprobabilidad
Portabilidad	Adaptabilidad, Reusabilidad

Las subcaracterísticas son nuevamente desprendidas en atributos de calidad, los que entregan una forma puntual de medir la calidad de ese aspecto específico a asesorar. Los tipos de métricas a utilizar para realizar las mediciones de los atributos, son los siguientes:

- Presencia: Este tipo de métrica indica si un componente posee el atributo de calidad indicado o no. Por lo que su valor es de tipo booleano y en caso de que lo posea. Puede ser acompañado por un campo de texto que señale las condiciones de como lo posee.
- Valores numéricos: Este tipo de métrica se utiliza para describir valores exactos de la información de un componente. Consiste en un valor numérico acompañado de una unidad (MB, veces, segundos, milisegundos).
- Porcentual: Este tipo de métricas es utilizada para describir porcentajes. Tomando un valor numérico entre 0 y 100.
- Calificación: Este tipo de métricas describe una escala cualitativa que muestra el grado de completitud del componente respecto a un atributo de calidad. Su valor es de un número entero entre 1 a 5, siendo 5 el valor de completitud total, y 1 la ausencia total del atributo de calidad.

A continuación se definen las subcaracterísticas y los atributos a utilizar para medir cada una de estas. La rúbrica completa de atributos de calidad queda descrita en la tabla 4.16.

5.3.1.1 - Funcionalidad - Subcaracterísticas

Adecuación (Suitability): Esta característica expresa que tan bien el componente se adecua a los requerimientos especificados.

Métricas utilizadas para "Adecuación":

- Pre-condicionado y Post-condicionado: Señala si el componente define sus pre- y post- condiciones, determinando exactamente qué es lo que necesita y que es lo que provee (Presencia).

Precisión (Accuracy): Evalúa el porcentaje de resultados que poseen el nivel de correctitud demandado:

Métricas utilizadas para "Precisión":

- Correctitud (Correctness): Este atributo evalúa el porcentaje de los resultados obtenidos con la precisión especificada por los requerimientos del usuario (Porcentual).

Interoperabilidad (Interoperability): Encapsula la habilidad del componente para comunicarse con otros componentes.

Métricas utilizadas para "Interoperabilidad":

- Compatibilidad de información: Este atributo indica si el formato de la información manejado por el componente (aquello que recibe o genera) obedece a algún estándar internacional o convención, como XML (Presencia).

Seguridad (Security): Esta sub-característica indica la capacidad que tiene el componente para controlar el acceso a los servicios provistos.

Métricas utilizadas para "Seguridad":

- Encriptación de la información: Este atributo indica si el componente posee la habilidad de encriptar la información de modo de proteger la información que maneja (Presencia).
- Controlabilidad: Este atributo indica la capacidad que tiene el componente para controlar el acceso a las interfaces provistas (Calificación).

- Auditabilidad: Este atributo indica si el componente cuenta con un sistema de auditoría, con la capacidad de registrar el acceso de los usuarios al componente y su información (Presencia).

Conformidad: Esta sub-característica indica si el componente tiene conformancia con algún estándar (estándares internacionales, certificaciones, etc.).

Métricas utilizadas para "Conformidad":

- Estandarización: Este atributo indica si el componente hace conformancia a estándares internacionales (Presencia).
- Certificación: Este atributo indica si el componente está certificado por alguna organización interna o externa (Presencia).

Autosuficiencia (Self-Contained): Expresa la capacidad del componente para realizar sus funciones de manera autónoma.

Métricas utilizadas para "Autosuficiencia":

- Grado de independencia: Este atributo indica si el componente es autosuficiente, para realizar sus tareas, no dependiendo de otros componentes para proveer los servicios especificados (Calificación).

5.3.1.2 - Usabilidad - Subcaracterísticas

Comprensibilidad (Understandability): Esta característica mide el grado de facilidad de comprender el componente (a través de documentación, descripciones, demos, API's, o tutoriales del componente).

Métricas utilizadas para "Comprensibilidad":

- Documentación disponible: Este atributo indica la cantidad de documentación disponible, descripciones, demos o tutoriales disponibles, que tienen un directo impacto en la comprensibilidad del componente (Presencia).
- Calidad de documentación: Este atributo indica la calidad de la documentación presente en el componente (Calificación).

Esfuerzo de aprendizaje (Learnability): Esta característica estima el tiempo y esfuerzo requerido para dominar tareas específicas del componente (uso, configuración, administración del componente, etc.).

Métricas utilizadas para "Esfuerzo de aprendizaje":

- Facilidad de aprendizaje: Este atributo intenta estimar la cantidad de esfuerzo y tiempo necesario para dominar las tareas específicas (usar, configurar, administrar y dominar el componente) (Calificación).

Configurabilidad (Configurability): Demuestra la habilidad de un componente para ser configurado (a través de un archivo XML, el número de parámetros, entre otros).

Métricas utilizadas para "Configurabilidad":

- Esfuerzo para configurar: Este atributo mide la capacidad del componente para ser configurado (Calificación).

5.3.1.3 - Confiabilidad - Subcaracterísticas

Recuperabilidad (Recoverability): Indica si el componente puede soportar situaciones de error, y el mecanismo implementado en dicho caso para cubrirla.

Métricas utilizadas para "Recuperabilidad":

- Manejo de errores: Este atributo indica si el componente puede manejar situaciones de error (Presencia).

Tolerancia a fallos (Fault Tolerance): Indica si el componente puede mantener un específico nivel de rendimiento en caso de fallos.

Métricas utilizadas para "Tolerancia a fallos":

- Mecanismo disponible: Este atributo indica si el componente presenta un mecanismo de tolerancia a fallos (Presencia).

5.3.1.4 - Eficiencia - Subcaracterísticas

Eficiencia temporal (Temporary efficiency): Esta característica indica la habilidad del componente para realizar tareas específicas en el tiempo dado, bajo condiciones.

Métricas utilizadas para "Eficiencia temporal":

- Tiempo de respuesta: Este atributo mide el tiempo desde que se envía una petición hasta que se el componente envía una respuesta (Valor numérico).

Eficiencia material (Resource efficiency): Esta característica indica la cantidad de recursos utilizados para realizar tareas específicas, bajo condiciones.

Métricas utilizadas para "Eficiencia material":

- Utilización de memoria: Mide la cantidad de memoria necesitada por el componente para operar (Valor numérico).
- Utilización de disco: Mide la cantidad de espacio en disco utilizado por el componente (Valor numérico).

5.3.1.5 - Mantenibilidad - Subcaracterísticas

Posibilidad de cambiar (Changeability): Esta característica indica si cambios especificados pueden ser concretados y si el componente puede fácilmente ser extendido con nuevas funcionalidades.

Métricas utilizadas para "Posibilidad de cambiar":

- Extensibilidad: Este atributo indica la capacidad de extender la funcionalidad de un componente (Calificación).
- Personalización: Este atributo mide el número de parámetros que son configurables en el funcionamiento del componente (Valor numérico).

Comprobabilidad (Testability): Esta característica mide el esfuerzo requerido para probar un componente en orden de comprobar si realiza la función a la cual fue destinado.

Métricas utilizadas para "Comprobabilidad":

- Set de pruebas provisto: Este atributo indica si existen set de tests provistos para verificar la funcionalidad del componente y/o medir alguna de sus propiedades (Presencia).

5.3.1.6 - Portabilidad - Subcaracterísticas

Adaptabilidad (Adaptability): Esta característica indica si el componente puede ser adaptado a distintos ambientes.

Métricas utilizadas para “Adaptabilidad”:

- Esfuerzo para adaptar: Este atributo indica la cantidad de cambios requeridos para transferir un componente a otro ambiente (Calificación).

Reusabilidad (Reusability): Esta característica evalúa la habilidad del componente para ser reutilizado.

Métricas utilizadas para “Reusabilidad”:

- Abstracción de dominio: Este atributo indica el nivel de abstracción del componente respecto a su específico dominio de negocio (Calificación).
- Dependencia arquitectural: Este atributo indica el nivel de dependencia del componente respecto a una arquitectura en específico (Calificación).
- Modularidad: Este atributo indica el nivel de modularidad del componente en su estructura: La presencia de módulos, paquetes, o si todo el código fuente se encuentra junto (Calificación).

Tabla 4.16: Definición de atributos de calidad a evaluar en rubrica.

Característica	Subcaracteristica	Atributo	Tipo Métrica	Valores
Funcionalidad	Adecuación	Pre-condicionado y Post-condicionado	Presencia	No/Si
	Precisión	Correctitud	Porcentual	0-100 (%)
	Interoperabilidad	Compatibilidad de información	Presencia	No/Si
	Seguridad	Encriptación de la información	Presencia	No/Si
		Controlabilidad	Calificación	1-5
		Auditabilidad	Presencia	No/Si
	Conformidad	Estandarización	Presencia	No/Si
		Certificación	Presencia	No/Si
	Autosuficiencia	Grado de independencia	Calificación	1-5
Usabilidad	Comprensibilidad	Documentación disponible	Presencia	No/Si
		Calidad de documentación	Calificación	1-5
	Esfuerzo de aprendizaje	Facilidad de aprendizaje	Calificación	1-5
	Configurabilidad	Esfuerzo para configurar	Calificación	1-5
Confiabilidad	Recuperabilidad	Manejo de errores	Presencia	No/Si
	Tolerancia a fallos	Mecanismo disponible	Presencia	No/Si
Eficiencia	Eficiencia temporal	Tiempo de respuesta	Numérico	Valor numérico [ms]
	Eficiencia material	Utilización de memoria	Numérico	Valor numérico [MB]
		Utilización de disco	Numérico	Valor numérico [MB]
Mantenibilidad	Posibilidad de cambiar	Extensibilidad	Calificación	1-5
		Personalización	Numérico	Valor numérico [cant.]
	Comprobabilidad	Set de pruebas provisto	Presencia	No/Si
Portabilidad	Adaptabilidad	Esfuerzo para adaptar	Calificación	1-5
	Reusabilidad	Abstracción de dominio	Calificación	1-5
		Dependencia arquitectural	Calificación	1-5
		Modularidad	Calificación	1-5

5.3.2 Cobertura funcional

Ya que la oferta de funciones de los componentes es amplia y las necesidades de los stakeholders son variables, la cobertura funcional fue extraída de la definición de atributos de calidad para ser evaluada en detalle.

A continuación se define una rúbrica de requerimientos funcionales a cumplir a partir de las responsabilidades a cumplir del componente, definidas en la sección "Exigencias impuestas por el SII". Los requerimientos funcionales son expresados de manera jerárquica en funcionalidades de alto nivel, las que son divididas en sub-funcionalidades.

Para evaluar la cobertura funcional. Se hará uso de una escala de 3 valores posibles: sin cobertura (0), cobertura parcial (1) y cobertura total (2).

La rúbrica generada será combinada con la entregada por los atributos de calidad, formando la rúbrica final.

Tabla 4.17: Definición de funcionalidades y subfuncionalidades a evaluar en rubrica.

Funcionalidad	Sub-funcionalidad
Generación de DTE	Generación de XML para Factura Electrónica
	Generación de XML para Guía de Despacho
	Generación de XML para Nota de Crédito
	Generación de XML para Nota de Débito
	Soporte Campos adicionales en generación de DTE
	Firmado electrónico de XML
	Manejo de timbraje Electrónico
	Autentificación al SII
Envío Automático al SII	Envío de DTE al SII
	Envío de LibroCompraVenta al SII
Consulta Envío de DTE	
Intercambio entre Contribuyentes	Recepción de DTE desde Mail
	Revisión de DTE
	Generación de respuesta de DTE

	Envío de respuesta de DTE
Generación de Copias Impresas	Generación Muestra Impresa Factura Electrónica
	Generación Muestra Impresa Guía de Despacho
	Generación Muestra Impresa Nota de Crédito
	Generación Muestra Impresa Nota de Débito
Generación información Compras y Ventas	Generación de Libro de Compra
	Generación de Libro de Venta

5.4 Evaluación de componentes

Con la rúbrica formada, se evaluó los componentes candidatos en sus respectivos entornos de prueba. Dependiendo del tipo de métrica, el resultado originado fue una medición exacta, o una calificación cualitativa, la cual será fundamentada. Esta sección se encargará de mostrar los resultados obtenidos para cada métrica de la rúbrica, en los 3 candidatos seleccionados.

5.4.1 Resultados de "Atributos de calidad"

5.4.1.1 Característica "Funcionalidad"

La característica "Funcionalidad" se separa en las subccaracterísticas "Adecuación", "Precisión", "Interoperabilidad", "Seguridad", "Conformidad" y "Autosuficiencia", las cuales se desglosan en 9 atributos de calidad.

Resultados para "Pre-condicionado y Post-condicionado"

Los tres candidatos cuentan con una documentación que define las interfaces propias de cada uno, con sus precondiciones (argumentos) y postcondiciones (valor de regreso y efectos colaterales). Por lo que los tres candidatos obtienen una calificación de "Si".

Resultados para "Correctitud"

Para medir la correctitud de las funciones otorgadas por los candidatos, se realizan 2 series de pruebas relacionadas con la funcionalidad del componente:

- Firma de un XML utilizando un certificado digital: Se evalúa que el componente sea capaz de realizar firmas válidas para efectos del SII (que cumplan el estándar XMLDSIG 1.1).

− Generación de un EnvioDTE y Envío al SII: Se evalúa que el componente sea capaz de generar DTE's empaquetados en un EnvioDTE mediante los parámetros de entrada, y que estos puedan ser enviado al SII sin errores, ni rechazos.

Se construyen 8 casos para los dos sets de prueba, variando la estructura del XML entregado, y los DTE's requeridos a generar. Cada una de las pruebas se ejecuta 2 veces. Los resultados para cada componente pueden ser observados en la tabla 4.18:

Tabla 4.18: Resultados obtenidos para atributo de calidad "Correctitud".

	LibFacturista				LibreDTE-lib									Facturación_Electronica								
	1	2	3-8	%	1	2	3	4	5	6	7	8	%	1	2	3	4	5	6	7	8	%
Firma de XML	2	2	--	100	2	2	2	2	2	2	2	2	100	2	2	2	2	2	2	2	2	100
Generación Envío DTE	--	--	--	--	2	2	2	2	2	2	2	2	100	2	2	2	2	2	2	2	2	100

En el caso de LibFacturista, solo fueron posibles realizar las pruebas que eran soportadas por las funcionalidades del componente. Debido a que este no puede generar el XML de los DTE, ni tampoco soporta el firmado de XML con más de un DTE en este.

Resultados para "Compatibilidad de información"

El SII en el modelo de operación define los formatos a utilizar para los archivos (principalmente XML y PDF), los métodos de transmisión de información al SII y a sus contribuyentes (API's HTTP puro y SOAP para comunicación con el SII, correo electrónico para intercambio entre contribuyentes), y otros aspectos de formato (estándar XMLDSIG 1.1 para las firmas, las guías de estilo para las muestras impresas) que deben ser obedecidos de manera irrenunciable por los componentes para operar. Como los tres candidatos poseen las facultades para ser utilizados, todos obtienen una calificación de "Si".

Resultados para "Encriptación de la información"

Los componentes de facturación electrónica trabajan con información sensible, la cual permite emitir DTE a nombre de un contribuyente. En particular manejan los certificados digitales de los contribuyentes, los folios de timbraje y almacenan los DTE enviados (e incluso los recibidos) en la máquina que actualmente están operando. Procedimientos que limiten el acceso a esta información o la limiten quedan fuera de la responsabilidad de los componentes, por lo que los tres candidatos son valorizados con un "No" en este atributo.

Resultados para "Controlabilidad"

"LibFacturista" define el comportamiento y acceso de cada una de sus interfaces en el .DLL (o .SO) provisto, el cual se liga dinámicamente a la aplicación que lo desee integrar para

acceder a sus funciones. Este archivo se encuentra pre-compilado, por lo que para modificar y acceder a funciones de carácter privado se necesitan aplicar procedimientos de ingeniería inversa.

En el caso de “LibreDTE-lib” y “FacturaciónElectronica”, estos componentes son requeridos mediante manejadores de dependencias (Composer y RubyGems respectivamente) e instalados como código fuente en la máquina del cliente, el cual es cargado al momento de iniciar el servidor. Ambos componentes definen sus métodos públicos y privados posibles de acceder, definiendo un control de sus interfaces, pero al disponer el código fuente en el servidor, el control que pueden poseer es menor al provisto por “LibFacturista”, ya que alguien con acceso a este podría realizar modificaciones y cambiando el control/funcionamiento del componente.

Realizando aquel contraste, “LibFacturista” obtiene una calificación de 5, y los dos componentes restantes obtienen una calificación de 4.

Resultados para “Auditabilidad”

En el caso de “LibFacturista”, las llamadas retornan códigos de estado, que permiten saber el éxito de estas. El control de las interfaces llamadas no está provisto por estos. Para “LibreDTE-lib” y “FacturaciónElectronica”, estos cuentan con un log, en el cual registran el acceso a ciertas interfaces y los errores en ejecución.

Resultados para “Estandarización”

Para este atributo se utiliza el mismo razonamiento que en “Compatibilidad de información”. Los tres componentes son sometidos a estándares por el SII para poder operar. De forma adicional, la integración de los tres componentes es realizada utilizando canales estandarizados (Librerías dinámicas, paquetes de composer, y gemas de Ruby respectivamente).

La calificación obtenida para los tres componentes es de “Si”.

Resultados para “Certificación”

De la información provista por los proveedores de los dos componentes candidatos, ninguno posee alguna certificación (diferente a la del SII) acreditado por una organización. Por lo que la calificación obtenida para los tres componentes es de “No”.

Resultados para “Grado de independencia”

Para medir el grado de independencia de los componentes, se consideró observar que tan dependientes son estos en el caso de uso de “Generar y enviar un DTE al SII”, los resultados obtenidos son los siguientes:

− LibFacturista: Este componente que cuenta con funcionalidades limitadas, solo es capaz de firmar y enviar los DTE al SII. La generación del XML debe ser implementada fuera del componente, generando un alto grado de dependencia en otros componentes. La calificación puesta a este componente es de 2.

− LibreDTE-lib: La información de los DTE a enviar es ingresada como parámetro para la construcción de un objeto sasco\LibreDTE\Sii\Dte, el cual es posteriormente firmado y timbrado (con las funciones timbrar y firmar respectivamente). El DTE es posteriormente añadido a un objeto sasco\LibreDTE\Sii\EnvioDte, el cual es exportado a XML mediante la función generar. El envío del XML se realiza utilizando la función sasco\LibreDTE\Sii::enviar, el cual toma el XML anteriormente generado y un token de autenticación como parámetros (siendo el token generado con la función sasco\LibreDTE\Sii\Autenticación::getToken). Para este caso de uso el componente es autosuficiente, por qué se califica a este componente con un valor de 5 en este aspecto.

− FacturaciónElectronica: Los DTE son ingresados como parámetro para la construcción de un objeto SiiEnvioDte, el cual permite el envío de los DTE al SII mediante la función "send_dte". En esta función el objeto generar los XML, firma, firma y envía el documento al SII, con la configuración previamente ingresada utilizando "FactGem.configure". Para este caso de uso el componente es autosuficiente, por qué se califica a este componente con un valor de 5 en este aspecto.

5.4.1.2 Característica "Usabilidad"

La característica "Usabilidad" se separa en las subccaracterísticas "Comprensibilidad", "Esfuerzo de aprendizaje" y "Configurabilidad", las cuales se desglosan en 4 atributos de calidad.

Resultados para "Documentación disponible"

Los dos componentes comerciales cuentan con una documentación. "LibFacturista" cuenta con una documentación en texto plano que define la firma de los métodos (parámetros y valores de retorno), junto con ejemplos de integración para distintos lenguajes. "LibreDTE-lib" mantiene una documentación creada utilizando doxygen, la cual es una herramienta que genera la documentación a partir del código fuente del proyecto. En el caso de "FacturaciónElectronica", el código se encuentra comentado para facilitar la comprensión, pero no existe una documentación oficial disponible.

Resultados para "Calidad de documentación"

Para "LibFacturista", la documentación disponible entrega una descripción de las funciones expuestas por la interfaz, con sus valores y tipos de retorno/entrada. Aunque algunos parámetros son ingresados mediante tuplas clave-valor separadas por punto y coma, se

desconoce los posibles atributos que el componente permite. En el caso de “LibreDTE-lib”, la documentación generada es exhaustiva y estandarizada. Cómo se origina del código fuente, es un reflejo más idóneo del funcionamiento de este, y se mantiene al día con la modificación de éste de manera semi-automatizada.

Las calificaciones resultantes para los tres componentes son de 3, 5 y 1 respectivamente (“FacturaciónElectronica” no posee documentación disponible).

Resultados para “Facilidad de aprendizaje”

Para calificar este atributo, se utilizaron dos razonamientos: La calidad y presencia de la documentación, y los aspectos que el componente delega al implementador para que realice la integración. En el caso de “LibFacturista”, dado que la documentación disponible no es de alta calidad, y el componente delega tareas esenciales al implementación (como es la generación de los XML, lo cual supone una cantidad de información bastante extensa a entender para la construcción de una implementación), el componente es calificado con una facilidad de aprendizaje baja (2).

Para “LibreDTE-lib”, la documentación es bastante completa, y el componente presenta un alto grado de independencia (obteniendo una calificación de 5 en este atributo), por lo tanto es calificado con una facilidad de aprendizaje alto (5).

En el caso de “FacturaciónElectronica”, no existe una documentación disponible, por lo que la forma de aprender a utilizar este componente es mediante la comprensión del código fuente (el cual se encuentra comentado), o leyendo las pruebas unitarias que son realizadas a las clases del componente (las cuales denotan algunos ejemplos de uso). Aun así, posee un alto grado de independencia para cumplir sus labores, reduciendo bastante el esfuerzo para integrar a un sistema. De este modo, el componente es calificado con una facilidad de aprendizaje media (3).

Resultados para “Esfuerzo para configurar”

Para evaluar este atributo, se hicieron observaciones en dos aspectos de los componentes: localización y formato en el que son guardadas las configuraciones, y la magnitud de parámetros que son configurables mediante el método provisto.

Para “LibFacturista”, se provee un archivo de configuración que sigue una estructura clave-valor similar a los ficheros .ini, la cual permite definir aspectos regionales del componente y del agente encargado de realizar la integración. El resto de las configuraciones es entregado mediante un string que separa las tuplas clave-valor utilizando punto y coma. Ya que la documentación para estos parámetros no es exhaustiva, y su magnitud es limitada, se evalúa al componente en este aspecto con una calificación de 2 (Esfuerzo alto).

En el caso de “LibreDTE-lib”, toda la configuración es provista exclusivamente al componente por medio de los parámetros permitidos para cada una de sus interfaces. Los parámetros otorgan una gran flexibilidad a la hora de configurar el componente (con una alta magnitud de parámetros por configurar), pero al no estar centralizados, suponen un esfuerzo para el integrador, teniendo que poseer nociones más exactas de cómo el componente opera, y en cuál de sus llamadas debe ser configurado. Por aquellas razones, el componente obtiene una calificación de 3 (Esfuerzo medio).

“FacturaciónElectronica” provee una interfaz definida para configurar los parámetros generales del componente. Esta interfaz toma un bloque, en el cual se asignan las configuraciones que se deseen personalizar. Las configuraciones son almacenadas en el módulo del componente, y son leídas para realizar las distintas funcionalidades que el componente otorga.

Los parámetros posibles por configurar consideran: el certificado electrónico a utilizar, credenciales de correo electrónico (mediante protocolos de recepción y envío de correos), datos de carátula del contribuyente (RUT, razón social, giro, entre otros), y algunas rutas donde el componente almacenará los DTE generados. De esta forma, gran parte de la configuración queda centralizada, ahorrando trabajo al integrador y reduciendo la repetición de información que debe ser otorgada al componente por medio de los parámetros de entrada de cada una de sus interfaces. El componente obtiene una calificación de 5 en este aspecto (Esfuerzo mínimo).

5.4.1.3 Característica “Confiabilidad”

La característica “Confiabilidad” se separa en las subccaracterísticas “Recuperabilidad” y “Tolerancia a fallos”, las cuales se desglosan en 2 atributos de calidad.

Resultados para "Manejo de errores”

Este atributo observa la capacidad del componente de manejar los posibles errores de uso, que el integrador experimente al momento de llamar sus funciones.

En el caso de “LibFacturista”, cada llamada regresa un código numérico, que entrega información del resultado de la operación. Los mensajes asociados a cada uno de los códigos pueden ser revisados llamando a la función “lf_error”.

Para “LibreDTE-lib”, los errores son almacenados en un “log” (registro) de mensajes, común en el componente. Este “log” puede ser revisado al momento posterior de realizar una operación para revisar si este ocurrió sin errores.

En “FacturaciónElectronica”, aunque algunas interfaces retornan un valor de verdad que indica el éxito de la llamada, no existe un comportamiento estandarizado para manejar los posibles errores, y no están todos debidamente catalogados.

Resultados para "Mecanismo disponible"

Este atributo observa si existe un mecanismo disponible para tolerar los fallos inesperados en la ejecución del componente. Algunos lenguajes de programación integran de por sí mecanismos de Excepciones, como es en el caso de Ruby o PHP (en el cual está construido “FacturaciónElectronica” y “LibreDTE-lib”), facilitando el trabajo al momento de la integración.

Para “LibFacturista”, el ligado dinámico imposibilita tener un control de comportamientos inesperados del componente, por lo que no existe un mecanismo disponible para este.

5.4.1.4 Característica “Eficiencia”

La característica “Eficiencia” se separa en las subccaracterísticas “Eficiencia temporal” y “Eficiencia material”, las cuales se desglosan en 3 atributos de calidad.

Resultados para "Tiempo de respuesta"

Para medir el tiempo de respuesta de los componentes, se generaron 4 pruebas de tareas básicas relacionadas a los componentes, estas fueron:

- Firma de un XML: Determina cuánto tarda el componente en realizar el proceso de firma de un EnvioDTE, contenedor de un DTE (por lo tanto, se aplican 2 firmas).
- Envío de un DTE al SII: Determina cuánto tarda el componente desde la realización del envío de un DTE al SII, hasta recibir la respuesta de este.
- Generación de 1 EnvioDTE, Set Básico: determina cuánto tarda la generación de un set de prueba de 8 DTE propuesto por el SII en el proceso de certificación, junto con el envío al SII de este.
- 100 EnvioDTE de 10 facturas, sin envío: determina cuánto tarda la generación de 100 EnvioDTE de 10 facturas cada uno, sin enviarlos al SII.

En el caso de LibFacturista, debido a las limitaciones funcionales de este componente, solo se pudieron ejercer las primeras dos pruebas. Esto es debido a la carencia del componente para generar el XML de los DTE y EnvioDTE.

Se realizaron 5 muestras las cuales fueron promediadas para cada prueba. Los resultados están expuestos en la tabla 4.19:

Tabla 4.19: Resultados encontrados para atributo de calidad “Tiempo de respuesta”.

	LibFacturista	LibreDTE-lib	Facturación_Electronica
Tiempo de respuesta: Firma de un XML	373 [ms]	13 [ms]	26 [ms]
Tiempo de respuesta: Envío de DTE al SII	3571 [ms]	719 [ms]	1642 [ms]
Tiempo de respuesta: 1 EnvioDTE Set Básico	--	1794 [ms]	15510 [ms]
Tiempo de respuesta: 100 EnvioDTE de 10 facturas, sin envío	--	21334 [ms]	141181 [ms]

Resultados para "Utilización de memoria"

Para medir la utilización de memoria, se consideró la asignación de memoria de los tres componentes realizando dos tareas: firma de un EnvioDTE, y generación y envío de un DTE al SII. La memoria fue observada mediante de la utilización de 3 librerías, especificas a los lenguajes utilizados para integrar los 3 candidatos ([49], [50], [51]). Los resultados se observan en la tabla 4.20:

Tabla 4.20: Resultados encontrados para atributo de calidad “Utilización de memoria”.

	LibFacturista	LibreDTE-lib	Facturación_Electronica
Memoria: Firma de un XML	15,524 [MB]	0,011 [MB]	0,694 [MB]
Memoria: Generación y envío de un DTE	--	2,868 [MB]	26,601 [MB]

Resultados para "Utilización de disco"

La utilización de disco de los componentes fue calculada a partir del peso del componente, tal como es distribuido (sin uso de compresión). Los resultados encontrados quedan informados en la tabla 4.21:

---

[49] Arnaud Le Blanc. PHP memory profiler extension [en línea] <https://github.com/arnaud-lb/php-memory-profiler> [consulta: 26 octubre 2016].

[50] Python. memory_profiler 0.41: A module for monitoring memory usage of a python program [en línea] <https://pypi.python.org/pypi/memory_profiler> [consulta: 26 octubre 2016].

[51] Sam Saffron. memory_profiler for ruby [en línea] <https://github.com/SamSaffron/memory_profiler> [consulta: 26 octubre 2016].

Tabla 4.21: Resultados encontrados para atributo de calidad “Utilización de disco”.

LibFacturista	LibreDTE-lib	Facturación_Electronica
7,61 [MB]	2,37 [MB]	2,12 [MB]

5.4.1.5 Característica “Mantenibilidad”

La característica “Mantenibilidad” se separa en las subccaracterísticas “Posibilidad de cambiar”, y “Comprobabilidad”, las cuales se desglosan en 3 atributos de calidad.

Resultados para "Extensibilidad"

Ya que LibFacturista es distribuido como librería dinámica precompilada, su extensibilidad es nula/mínima (1), toda funcionalidad extra debe ser añadida fuera del componente.

En el caso de LibreDTE-lib y FacturaciónElectronica, estos son distribuidos con su código fuente expuesto, por lo que pueden ser adaptados en una copia local si fuese necesario. Sin embargo, estos no proveen un mecanismo documentado para extender su comportamiento. De esta forma, son calificados con extensibilidad moderada (3).

Resultados para "Personalización"

Este atributo contabiliza la cantidad de parámetros que son configurables en el componente, por los medios que estos designan para ser configurados (explicados en detalle en la sección de resultados de ”Esfuerzo para configurar”).

En el caso de LibFacturista, estos son 3 parámetros (Información regional, e información del integrador); para LibreDTE-lib son 0 (no existe un método de configuración, toda la personalización se realiza a través de los parámetros que cada una de las interfaces provee); y 34 para el caso de FacturaciónElectronica.

Resultados para "Set de pruebas provisto"

El set de pruebas está presente en dos de los tres componentes: LibreDTE-lib, que contiene un set de pruebas que utiliza PHPUnit como framework de testing; y FacturaciónElectronica, que provee su set de pruebas compatible con RSpec. LibFacturista no lo provee: su distribución incluye únicamente la librería dinámica precompilada, adjunta a la documentación y ejemplos de uso.

5.4.1.6 Característica “Portabilidad”

La característica “Portabilidad” se separa en las subccaracterísticas “Adaptabilidad” y “Reusabilidad”, las cuales se desglosan en 4 atributos de calidad.

Resultados para "Esfuerzo para adaptar"´

Este atributo intenta capturar el esfuerzo que el componente requiere para ser trasladado a otro ambiente.

Para analizar este aspecto, se realiza el supuesto de querer adaptar el formato de los DTE para contener otro formato debido a un cambio regional, teniendo que hacer variaciones en la estructura de los DTE, campos de carátula y nodos de detalle; y en la ruta HTTP a la cual se debe enviar los DTE.

- LibFacturista: Este componente recibe un XML ya conformado para ser firmado y enviado al SII. Un cambio en el formato en aquellos campos del DTE no requiere alteraciones en el componente. Un cambio en la ruta HTTP no está permitido por el componente, por lo que se debería exigir una nueva versión de este. Es calificado con esfuerzo medio (3).
- LibreDTE-lib: La creación de los XML pertenecientes a cada DTE es entregada como parámetro a en la construcción de un objeto sasco\LibreDTE\Sii\Dte. Si se entregan parámetros adicionales estos son utilizados al momento de la generación sin problemas. Sobre la ruta HTTP, esta está codificada en el componente, por lo que requerirá adaptar el componente a la nueva ruta. Es calificado con esfuerzo medio (3).
- FacturaciónElectronica: La generación de los XML pertenecientes a cada DTE es entregada como parámetros a un objeto SiiFactura, el cual solo acepta una lista restringida de estos. Las rutas de envío para ambientes de certificación y producción de los envío de DTE están codificadas. Es calificado con esfuerzo alto (1).

Ya que los componentes candidatos responden a un fin muy específico en un formato exacto, y las modificaciones que pueden ser requeridas para trasladarlos a otro ambiente pueden ser mucho mayores a las propuestas, esta aproximación es minimalista al problema de estimar este atributo. Aun así, nos entrega un dato de las magnitudes relativas de los esfuerzos necesarios para adaptar estos componentes.

Resultados para "Abstracción de dominio"

Para evaluar la abstracción de dominio de los componentes, se analizaron las funcionalidades de los componentes que podrían ser deseables de adaptar para ser utilizados en otros contextos, pero que por limitaciones técnicas estos componentes no lo permiten. Para foco de este análisis, se tomó en cuenta la actividad específica de la emisión de un DTE.

- LibFacturista: La generación de XML viene dada a partir de implementación propia al momento de integrar, por lo que los contenidos del DTE, considerando que respeten el

esquema definido por el SII, son completamente maleables (exceptuando las firmas electrónicas y el timbrado de los documentos, procesos que son realizados por el componente). Dado esto, el componente fue calificado con una alta abstracción de dominio (5).

- LibreDTE-lib: Este componente se encarga de generar los DTE a partir de la información entregada como parámetro (mediante la utilización de un objeto sasco\LibreDTE\Sii\Dte). Los parámetros posibles son aquellos que el SII permite, incluyendo tipos opcionales y de carácter condicional para contribuyentes especiales (impuestos adicionales según actividad, retenciones, valores en otra moneda y otros). Como el componente se puede adaptar a estos casos, fue calificado con una alta abstracción de dominio (5).
- FacturaciónElectronica: La generación de los DTE es realizada dentro del componente al momento de realizar el envío, mediante un objeto SiiFactura. Este toma una serie de parámetros disponibles en los cuales se inserta la información que el XML final contendrá. La generación de estos campos solo considera aquellos de carácter obligatorio para 4 tipos de DTE (facturas, guías de despacho, notas de crédito y notas de débito electrónicas). Dado que el componente sólo puede generar DTE bajo esas condiciones, se calificó con una baja abstracción de dominio (2).

Resultados para "Dependencia arquitectural"

Los componentes candidatos permiten ser integrados a sistemas de software para realizar funcionalidades de facturación electrónica bajo el modelo del SII. Estos componentes no se encargan de la presentación ni la persistencia de la información, ni tampoco imponen un estilo arquitectural de manera excluyente. La única dependencia que estos generan, son el tipo de lenguajes de programación en los cuales el componente deberá ser ejecutado, y por lo tanto, limitan el tipo de plataformas en los que estos pueden ser ejecutados.

De esta forma, “LibFacturista” es el más independiente de los candidatos (5), ya que su modo de uso mediante ligado dinámico, puede ser integrado a una alta variedad de lenguajes y sistemas operativos. Para “LibreDTE-lib”, y “FacturaciónElectronica”, estos componentes son locales a un lenguaje de programación, por lo que generan un nivel medio de dependencia (3).

Resultados para "Modularidad"

Para LibreDTE-lib y FacturaciónElectronica, ambos presentan una estructura modular que separa el código, por ejemplo: LibreDTE-lib define los módulos “Base”, “Certificación”, “Dte” y “PDF” para estructurar las distintas responsabilidades del componente. Ambos componentes son calificados con alta modularidad (5).

En el caso de LibFacturista, se desconoce su estructura interna, por lo que este atributo no pudo ser calificado.

5.4.2 Resultados de “Cobertura funcional”

A continuación se presentan los resultados de la evaluación de cobertura funcional para los candidatos. La calificación de la cobertura es realizada a partir de las funcionalidades presentes en los componentes distribuidos, y complementada con la información disponible en la documentación de estos:

Tabla 4.22: Resultados encontrados para cobertura funcional en componentes evaluados.

Funcionalidad	Subfuncionalidad	LibFacturista	LibreDTE-lib	Facturación_Electronica
Generación de DTE	Generación de XML para Factura Electrónica	0	2	2
	Generación de XML para Guía de Despacho	0	2	2
	Generación de XML para Nota de Crédito	0	2	2
	Generación de XML para Nota de Débito	0	2	2
	Soporte Campos adicionales en generación de DTE	0	2	1
	Firmado electrónico de XML	2	2	2
	Manejo de timbraje Electrónico	0	2	2
Envío Automático al SII	Autentificación al SII	2	2	2
	Envío de DTE al SII	2	2	2
	Envío de LibroCompraVenta al SII	2	2	2
Consulta Envío de DTE		0	2	2
Intercambio entre Contribuyentes	Recepción de DTE desde Mail	0	0	2
	Revisión de DTE	0	2	2
	Generación de respuesta de DTE	0	2	2
	Envío de respuesta de DTE	0	0	2
Generación de Copias Impresas	Generación Muestra Impresa Factura Electrónica	2	2	2
	Generación Muestra Impresa Guía de Despacho	2	2	2
	Generación Muestra Impresa Nota de Crédito	2	2	2

	Generación Muestra Impresa Nota de Débito	2	2	2
Generación información Compras y Ventas	Generación de Libro de Compra	0	2	2
	Generación de Libro de Venta	0	2	2

De los 42 puntos totales de cobertura de funcionalidad presentes en la rúbrica, LibFacturista obtiene 16, LibreDTE-lib 38 y Facturación_Electronica 41:

- Para LibFacturista, muchas de las funciones requeridas, son delegadas a la implementación propia. El componente otorga las bases mínimas para el firmado de documentos, comunicación con el SII y generación de copias impresas.

- LibreDTE-lib obtiene un puntaje de 38 sobre 41. Los aspectos que este no cubre están relacionados con el intercambio entre contribuyentes, ya que este delega la recepción y envío de correos a una implementación externa al componente.

- Facturación_Electronica posee una cobertura prácticamente total en los aspectos requeridos. Las deficiencias que posee son en la generación de campos adicionales en los DTE, los cuales son de carácter obligatorio para algunos contribuyentes en condiciones especiales.

Capítulo 6: Validación

La propuesta de componente fue descrita ampliamente junto a sus detalles de implementación en la sección de “Desarrollo”, la cual fue acompañada a la creación de un sistema ERP que era beneficiado por sus funcionalidades. El sistema ERP construido a medida, fue necesario puesto que los requerimientos provenían de un rubro con necesidades particulares, las cuales no eran fácilmente cubiertas por las ofertas existentes en el mercado.

Posteriormente, se elaboró una rúbrica para la evaluación de algunos componentes del mercado, junto a la solución propia de este trabajo. La rúbrica considero atributos de calidad, los cuales fueron extraídos de modelos de calidad para componentes, y enfatizó la cobertura funcional de estos, siendo detallados en una sección independiente. La rúbrica elaborada fue ajustada especialmente a la evaluación de componentes de facturación electrónica en Chile, por lo puede ser reutilizada en el futuro, para realizar una evaluación con nuevos candidatos en la materia.

Los resultados de la evaluación encontrados permiten perfilar las soluciones en distintas dimensiones. Respecto a la cobertura funcional, soluciones como LibreDTE-lib y Facturación_Electronica demostraron ser más completas en torno a lo exigido por el SII, en relación a LibFacturista, componente que entrega bases mínimas para operar.

En torno a aspectos relacionados con la reusabilidad. Facturación_Electronica obtuvo una alta modularidad, dependencia arquitectural media (está diseñado para ser utilizado exclusivamente en ambiente Ruby), y una abstracción de dominio baja. Esto último se debe a que el mecanismo de generación de DTE del componente, es exclusivo en la designación de parámetros, prohibiendo cualquier inserción en el XML que no se encuentre permitida.

Dado esto, se afirma que el componente diseñado (Facturación_Electronica) es suficientemente diseñado en conformancia para operar bajo el modelo propuesto por el SII, y además posee un carácter reusable. Como este además fue posible de integrar a un sistema ERP a medida, se constata que es posible de integrar, cumpliendo el objetivo principal de este trabajo.

A continuación, se presenta el plan de transición y despliegue utilizado para el sistema ERP, el cual fue ejecutado para poner en producción el sistema. Posteriormente, a partir de los resultados de la evaluación, se identificarán escenarios que favorezcan la elección de un componente sobre el otro.

6.1 Plan de transición y despliegue

6.1.1 Transición del sistema ERP

La construcción del sistema ERP se realizó de forma iterativa, poniendo énfasis en cubrir las necesidades de facturación electrónica en las últimas iteraciones, y delegando la construcción de funciones del sistema ERP a iteraciones tempranas. Se designaron 3 iteraciones, las cuales dividen los casos de uso asociados al sistema ERP considerando la anterior premisa.

Cada iteración a desarrollar supone un entregable a otorgar al cliente, el cual podrá evaluar mediante el uso y validar su correctitud. La validación temprana de los casos de uso más simples, permite disminuir los riesgos asociados a dedicar esfuerzos de manera equivoca, los cuales son más costosos de corregir en fases tardías del desarrollo. El hecho que el cliente pueda evaluar cada entregable, permite ajustar la aplicación rápidamente para las siguientes iteraciones.

La capacitación fue realizada acompañando la entrega de cada gran hito, de manera presencial, en las dependencias del cliente. El servicio fue puesto al alcance de la red del cliente, por lo que podían evaluar su uso de manera prolongada.

Para realizar la transición de la información, fue necesario construir pequeñas utilidades que permitieron el traspaso de información entre el sistema viejo y el actual. Estas aplicaciones utilizan algunas de las gemas anteriormente mencionadas en la “Implementación del Sistema ERP”, como roo o axlsx.

Una vez que las tres iteraciones fueron construidas, entregadas y evaluadas por el cliente, se inició una fase correctiva, en la cual el sistema fue adaptando sus funciones para concordar con las expectativas del cliente. Luego de esta fase correctiva y realizando las capacitaciones guiadas a todo usuario que hará uso del sistema en las dependencias del cliente, el sistema fue puesto en marcha.

6.1.2 Despliegue de sistema ERP

El despliegue del sistema ERP fue desarrollado de forma paralela al plan de transición, dado que cada iteración una vez finalizada, fue desplegada en infraestructura de producción. El plan de despliegue necesita permitir la incorporación rápida de nuevas versiones del sistema a producción, teniendo en cuenta la prevención de bajas en el funcionamiento del sistema.

Un factor a considerar que se desprende de la toma de requerimientos es la disponibilidad del sistema. Es necesario que la infraestructura selecta asegure un alto grado de disponibilidad. Infraestructura propietaria depende de la disponibilidad de red del cliente, y no garantiza que

la aplicación sea accesible desde otras localizaciones geográficas. Por lo que se optó por utilizar infraestructura en la nube.

Por ello, se pensó en una solución PaaS (Platform as a Service), que proveen hardware y herramientas de software, necesarias para levantar aplicaciones en la web a sus usuarios como un servicio[52]. El usuario no maneja ni controla directamente la infraestructura Cloud (la cual incluye redes, servidores, sistemas operativos o almacenamiento), pero tiene control acerca de las aplicaciones desplegadas y posiblemente opciones de configuración para el ambiente en el cual corre la aplicación[53].

Ejemplos de soluciones PaaS existentes son: EngineYard, Google App Engine, Heroku, Microsoft Azure, Amazon AWS, Nitrous.io, OpenShift, entre otros[54].

Para el despliegue del sistema ERP, se eligió Amazon AWS, los cuales ofrecen una gran cantidad de soluciones IaaS (Amazon EC2, Amazon RDS, Amazon S3), las cuales son accesibles desde su servicio PaaS, conocido como Amazon Elastic Beanstalk.

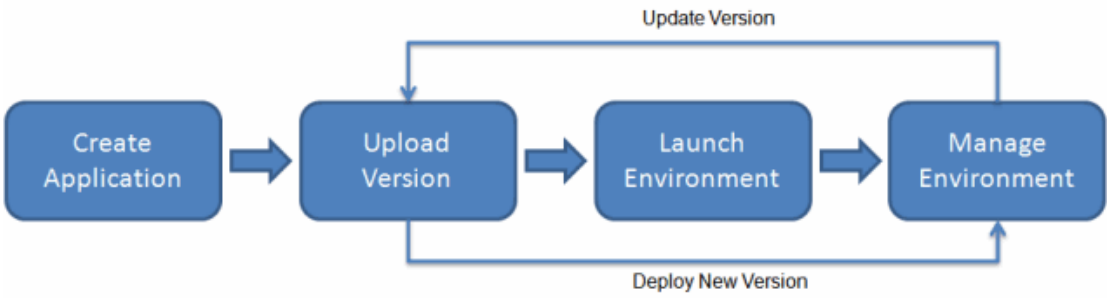

Figura 5.1: Ciclo de vida de una aplicación en Amazon Elastic Beanstalk.

Una vez que se sube una versión de la aplicación, la plataforma se encarga de los pasos posteriores de despliegue[55]. Cada aplicación vive en un ambiente, el cual es el encargado de preocuparse del despliegue, balanceo de carga, escalamiento y monitoreo de la aplicación.

La infraestructura a utilizar es configurable dependiendo de las necesidades de cada aplicación, servicios como Amazon EC2 (dedica a prestar hosting de virtual server) otorgan la

---

[52] TechTarget. Platform as a Service (PaaS) [en línea] <http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS> [consulta: 26 octubre 2016].

[53] IBM Cloud. What is platform as a service (PaaS)? [en línea] <http://www.thoughtsoncloud.com/2014/02/what-is-platform-as-a-service-paas> [consulta: 26 octubre 2016].

[54] PaaSfinder. Find your PaaS [en línea] <https://www.paasify.it/filter> [consulta: 26 octubre 2016].

[55] Amazon Web Services. AWS Elastic Beanstalk [en línea] <https://aws.amazon.com/elasticbeanstalk> [consulta: 26 octubre 2016].

posibilidad de pagar precios fijos por una reserva de tiempo, u “on-demand” (solo por la cantidad de horas utilizadas). Este último modelo se puede combinar con la utilidad de escalamiento automatizado provista por Amazon, cubriendo las necesidades de los usuarios que acceden a la aplicación, y pagando solo los recursos necesarios por ello.

Para el levantamiento del sistema ERP, se utilizaron tres servicios de Amazon, en conjunto con Elastic Beanstalk para el despliegue:

- Amazon EC2 para el despliegue del servidor que ejecuta el sistema ERP. Se utilizó una instancia t2.micro, la cual fue configurada con el sistema operativo Amazon Linux 2016.03 v2.1.0 de 64 bits, y con la versión 2.3 de Ruby.

- Amazon RDS para contener la base de datos relacional utilizada por las instancias. Se utilizó una instancia db.t2.micro, con PostgreSQL 9.5.1 como motor de base de datos.

- Amazon S3 para el almacenamiento persistente de archivos de las instancias, utilizado para almacenar los DTE emitidos y recibidos, y los certificados del contribuyente necesarios para firmar los DTE. Se creó un bucket (nombre que da Amazon S3 a unidad de almacenamiento persistente de tamaño variable) para tal propósito. El coste del bucket es dinámico y depende de la cantidad de datos almacenados, las transferencias y el tipo de bucket. Se utilizó un bucket de tipo almacenamiento estándar, el cual promete un alto nivel de disponibilidad (99,99% en un año), baja latencia, alta velocidad y durabilidad alta (99,999999999% de los casos)[56].

El despliegue de cada nueva versión fue asistido con la utilidad CLI de Elastic Beanstalk, eb. La utilidad permite integrarse al sistema de versionamiento git, apuntando el despliegue de una branch a una aplicación en Elastic Beanstalk. Para la aplicación, se apuntó que el despliegue de las versiones se realizará desde la branch master. Se utilizó git-flow como estrategia para el desarrollo de cada iteración/casos de uso, y para la fase correctiva del sistema. La utilidad eb se asegura de realizar el deployment de la aplicación, en el estado del último commit de la branch apuntada.

Una vez que la aplicación es subida a Elastic Beanstalk, esta es distribuida a la(s) instancias, las cuales realizan el proceso de actualización correspondiente, reportando su estado a Elastic Beanstalk. Los reportes permiten tener control de la información de despliegue, si este ha finalizado correctamente, o si se presentaron errores.

---

[56] Amazon Web Services. Amazon S3 Storage Classes [en línea] <https://aws.amazon.com/s3/storage-classes> [consulta: 26 octubre 2016].

6.2 Identificación de escenarios

Para identificar los escenarios, es necesario discutir los resultados encontrados en cada atributo de la rúbrica.

6.2.1 Discusión de resultados

Como se comentó previamente, la cobertura funcional está bien resuelta en LibreDTE-lib y Facturación_Electronica, componentes que obtuvieron puntuaciones de 38 y 41 puntos sobre 42 respectivamente. LibFacturista se queda bastante atrás (16 puntos), poseyendo solo funcionalidades mínimas para operar.

Aspectos que requieran la generación avanzada de XML (en el caso de contribuyentes que poseen una situación tributaria que se escapa del modelo más usual. LibreDTE-lib posee el soporte adecuado para estos casos. En los aspectos que Facturación_Electronica sobresale es en la recepción y envío de DTE por parte de otros contribuyentes, ya que puede ser configurado para el cumplimiento de dichos propósitos.

Si no se desea realizar un esfuerzo adicional para la cobertura de este aspecto, Facturación_Electronica es el candidato a favorecer. Un beneficio del componente es que logra cubrir autónomamente gran parte del modelo de operación del SII, minimizando el esfuerzo adicional por parte del usuario integrador.

En relación a la característica “Funcionalidad” propia de los atributos de calidad, los resultados son parciales en relación a los tres candidatos: la precisión de los tres es del 100% de correctitud; la compatibilidad de la información está obligada por el SII, y por lo tanto todos los candidatos cumplen este aspecto; la encriptación de la información no está presente en los componentes, ya que no es el propósito de los componentes.

Todos los componentes están estandarizados por las obligaciones impuestas por el SII para operar; ninguno de los candidatos cuenta con una certificación adicional a la otorgada por el SII, al cumplir las pruebas del ambiente de certificación.

Los aspectos a resaltar de esta característica son: controlabilidad, en la que LibFacturista obtiene una mejor calificación debido a la forma en la que es empaquetado, la que previene la edición de esta; y el grado de independencia, atributo que recalca la capacidad de los componentes para operar de forma autónoma. En este, LibFacturista obtuvo una pobre calificación, debido a que su cobertura funcional es demasiado baja y requiere de otros componentes u implementación propia para que efectivamente sea de utilidad como componente de facturación electrónica.

Respecto a la característica Usabilidad: los candidatos LibFacturista y LibreDTE-lib poseen una documentación disponible, siendo la de LibreDTE-lib de mejor calidad (alta). Facturación_Electronica no fue documentado apropiadamente, siendo este uno de los aspectos futuros a mejorar.

Aun así, para la facilidad de aprendizaje, Facturación_Electronica obtuvo una puntuación media, ya que los ejemplos de uso pueden ser encontrados en las pruebas unitarias presentes en el componente. LibFacturista es la solución de mayor dificultad de aprender, ya que su alto grado de dependencia requiere al integrador manejar aspectos detallados del modelo propuesto por el SII, a diferencia de los otros dos componentes, los cuales abstraen los detalles finos del procedimiento. LibreDTE-lib gracias a su autonomía y alta calidad de documentación, hacen que sea la más sencilla de aprender.

Sobre la configurabilidad, el esfuerzo para configurar es menor en Facturación_Electronica, ya que este centraliza todos sus parámetros en un solo lugar a modificar. LibreDTE-lib posee un alto nivel de personalización, pero al estar desacoplado supone un esfuerzo mayor. LibFacturista es la más compleja en este aspecto, ya que la configuración es acotada y no se posee documentación de su uso.

Bajo la característica confiabilidad, los resultados favorecen a LibreDTE-lib, el cual posee un manejo de errores y un control de excepciones apropiado gracias al lenguaje en el que está basado. LibFacturista posee un manejo de errores, pero no de excepciones; y Facturación_Electronica solo posee control de excepciones.

En la característica de eficiencia, LibreDTE-lib demostró ser altamente más eficiente que los otros candidatos, ganando todas las pruebas exceptuando el de la utilización en disco (la cual para la mayoría de los efectos prácticos, no posee un valor relevante), la cual fue menor en Facturación_Electronica por un 11,8% (0,25 [MB]). El segundo más eficiente es Facturación_Electronica, el cual tiene tiempos de respuesta cercanos al de LibreDTE-lib en algunos casos. LibFacturista fue el más ineficiente en los tres aspectos (utilización de memoria, disco y tiempos de respuesta).

Para la característica de mantenibilidad, Facturación_Electronica destaca al poseer una serie de atributos de personalización expuestos (LibreDTE-lib también es personalizable, pero esta configuración se realiza a través de parámetros al momento de realizar llamadas a sus interfaces). LibFacturista es el que posee más baja mantenibilidad, ya que su extensibilidad es nula.

En relación a la característica de portabilidad, los resultados indican que Facturación_Electronica posee poca facilidad para adaptar, esto es debido parcialmente a que está demasiado ajustado al modelo de operación del SII (misma razón por la que posee una

poca abstracción de dominio). Los otros componentes poseen un nivel medio de esfuerzo, por lo que son más adecuados para una implementación diferente a la propuesta por el SII.

Para la subcaracteristica de reusabilidad, LibFacturista demostró ser el más independiente de la arquitectura a usar, dado que la librería de ligado dinámico es pensada para ser integrada a una gran cantidad de ambientes, y los otros candidatos son locales a un lenguaje de programación. La modularidad de los componentes solo pudo ser analizada en LibreDTE-lib y Facturación_Electronica, los cuales obtuvieron una alta modularidad.

Finalmente, la abstracción de dominio es alta en LibFacturista y LibreDTE-lib; y baja en Facturación_Electronica. Esto se debe parcialmente a las características de diseño de este último, que hacen que la generación DTE sea de carácter fijo, sin tener la libertad de insertar información adicional al documento que los otros dos candidatos proveen.

6.2.2 Caracterización de componentes

En base a lo discutido, se construye una tabla resumen de los componentes en los aspectos macro evaluados en la rúbrica:

Tabla 5.2: Resumen de resultados obtenidos tras evaluación de rúbrica en componentes candidatos, para cada una de las características evaluadas y la cobertura funcional de estos.

	LibFacturista	LibreDTE-lib	Facturación_Electronica
Cobertura Funcional	Baja	Alta	Alta
Funcionalidad	Regular	Regular	Regular
Usabilidad	Regular	Alta	Regular
Confiabilidad	Regular	Alta	Regular
Eficiencia	Baja	Alta	Regular
Mantenibilidad	Baja	Regular	Alta
Portabilidad	Alta	Alta	Regular

El componente LibFacturista posee baja cobertura funcional, una usabilidad (para integrar) regular y confiabilidad regular. Gracias a su independencia arquitectural está pensado para ser utilizado en soluciones en las que se necesita una gran cobertura de plataformas, o que se requiere trabajar con una plataforma en específico. Requiere ser apoyado con otros COTS o una implementación propia para cubrir el modelo de operación del SII.

LibreDTE-lib destaca en la mayoría de los atributos, es una solución robusta para ambientes de desarrollo PHP. Siendo el único de los tres que provee opciones de generación avanzada de DTE, es el candidato a utilizar en casos que se escapen del contribuyente tradicional, o en

aquellos que la eficiencia es requerida (como es el caso en contribuyentes que emitan o reciban una gran cantidad de documentos tributarios). Requiere un esfuerzo de configuración medio y debe ser complementado con una estrategia de intercambio entre otros contribuyentes, ya que carece de esta.

Facturación_Electronica destaca por su alta cobertura funcional, librando de casi todas las tareas pertinentes al integrador. Aunque su eficiencia, facilidad de uso, confiabilidad son menores a las de LibreDTE-lib. Se identifica que es útil en casos que requieren la menor intervención posible en el modo de operación del SII, y donde la magnitud de información con la que trabaja el componente, no sea muy grande (por temas de eficiencia). Es un componente que destaca por su facilidad de integración (gracias a su configurabilidad y facilidad de aprendizaje). Su uso queda recomendado para la construcción de sistemas ERP a medida, que asistan a empresas de pequeña o mediana escala.

Capitulo 7: Conclusiones

En este trabajo se observó el desarrollo de una solución de facturación electrónica, en conformancia con el modelo de operación definido por el SII. Dado que todos los contribuyentes electrónicos requerían una solución de este carácter, independiente de su rubro o modelo de operación, se construyó una solución capaz de abstraerse de la implementación particular, permitiendo ser reutilizada. Se encontró que la arquitectura basada en componentes y el uso de COTS respondía a este propósito en particular, por lo que se procedió a la construcción de un componente de facturación electrónica.

Para verificar que el componente era capaz de resolver la problemática, este fue integrado a un sistema ERP a medida, el cual fue construido en paralelo a la realización del proyecto. El ecosistema de gemas de Ruby demostró ser particularmente beneficioso a la hora de encapsular el componente, y definir sus dependencias a utilizar.

Por otra parte, la utilización de engines propuesta en [13] ayudo a separar las funcionalidades del sistema ERP en módulos (componentes). Se cree que en proyectos grandes de Rails, este beneficio es significante, disminuyendo la complejidad de las dependencias y facilitando el trabajo entre equipos.

Otro aspecto que se cubrió en este trabajo fue el otorgar una herramienta para asesorar la evaluación de componentes de facturación electrónica. Se descubrió que la evaluación de la calidad de componentes de software no calzaba idealmente con los modelos de calidad para software existentes, puesto que la visión de otros stakeholders, y los posibles atributos a medir, necesitaban cambiar la perspectiva de la evaluación. Se desarrolló una rúbrica ajustada únicamente a este propósito, bajo la cual se evaluaron tres componentes (incluyendo el desarrollado).

Los resultados de la evaluación son parcialmente satisfactorios: dan cuenta de que el componente de facturación electrónica desarrollado posee una cobertura funcional bastante aceptable en relación a su competencia, aunque aspectos de eficiencia, usabilidad, confiabilidad y portabilidad, son importantes de mejorar.

A modo de detalle, el objetivo general de este trabajo fue cubierto en la sección 4.1 del desarrollo. Sección que otorga gran detalle técnico de: las obligaciones impuestas por el SII que fueron necesarias de implementar; especificaciones del componente y del sistema ERP a desarrollar; y detalles de la integración.

Puesto que muchos de los detalles técnicos de bajo nivel son difíciles de hallar en la documentación u son omitidos, esta sección resulta particularmente valiosa. La información rescatada para lograr este propósito se originó de varios sitios. Se destaca la importancia de

los documentos del SII; la mesa de ayuda del SII; el sitio de CyptoSys que profundiza en las firmas electrónicas XMLDSIG[57]; y el sitio de Factura Electrónica Desarrolladores[58], donde se congrega una gran cantidad de inquietudes técnicas.

El estudio cauteloso de esta sección, complementado con una lectura del instructivo técnico del SII, pueden ser de gran utilidad a la hora de implementar una solución de facturación electrónica.

En relación a cada uno de los objetivos específicos de este trabajo, se resumen los resultados obtenidos:

- "Realizar una evaluación comparativa de componentes de facturación electrónica existentes en el mercado con el componente a implementar, utilizando modelos de calidad para componentes": Se logró este objetivo en la sección 4.2, en la que se definió una rúbrica basada en el modelo de calidad para componentes propuesto por Álvaro et al. [21], ajustada para propósitos de esta evaluación en particular. La rúbrica incluye además un énfasis en la cobertura funcional, la cual se desprende del estudio de la sección 4.1, complementada con la información investigada en la sección 1.2. La evaluación y sus resultados se pueden encontrar en la sección 4.2.4.

- **"Integrar componente de facturación electrónica a sistema web ERP"**: Los esfuerzos situados en la integración de componente al sistema ERP fueron detallados en la sección 4.1.3. Se utilizaron engines de Rails para aislar el desarrollo de facturación electrónica dentro del sistema, del resto de este. La utilización demostró ser útil para organizar el sistema en partes independientes, similar a lo formulado por la arquitectura basada en componentes. Este engine fue integrado a la gema de facturación electrónica, comunicándose exclusivamente con esta y definiendo las vistas y persistencia requeridas para este, utilizando el patrón MVC al que Rails está suscrito.

- **"Diseñar un plan de transición y despliegue, que permita poner el sistema web ERP en producción"**: Este objetivo fue cumplido en la sección 5.1, la cual detalla cómo fue la transición a producción mediante un proceso iterativo, dividiendo el desarrollo del sistema en 3 fases. El despliegue fue realizado utilizando la plataforma PaaS Elastic Beanstalk de Amazon, la cual fue utilizada pensando en maximizar la disponibilidad del sistema, ya que permite el despliegue de nuevas versiones sin alterar la disponibilidad del servicio. La plataforma utiliza otros servicios de infraestructura Amazon para desplegar la aplicación, de los cuales en particular se utilizaron Amazon EC2, Amazon S3 y Amazon RDS. El uso de esta

---

[57] DI Management Services. XML-DSIG and the Chile SII [en línea] <http://www.cryptosys.net/pki/xmldsig-ChileSII.html> [consulta: 26 octubre 2016].

[58] Marcelo Rojas. Factura Electrónica Desarrolladores .Net [en línea] <http://lenguajedemaquinas.blogspot.cl> [consulta: 26 octubre 2016].

plataforma demostró ser útil para abstraerse de los detalles finos del despliegue, además de proveer un sistema de monitoreo y opciones de escalabilidad, permitiendo ajustar dinámicamente la cantidad de infraestructura para responder a las demandas de usuarios finales.

- "Identificar escenarios que favorezcan la elección de un componente sobre el otro, basados en los resultados obtenidos por la evaluación comparativa de componentes": La discusión e identificación de escenarios fue realizada en la sección 5.2. A grandes rasgos: el componente LibFacturista demostró ser útil en su independencia arquitectural, para soluciones que requieran ser utilizados en una gran cantidad de plataformas; LibreDTE-lib es beneficioso gracias a su amplia cobertura funcional y eficiencia, por lo que es adecuado para soluciones que escapan de los requerimientos de un contribuyente común, o en soluciones que requieran procesar un número amplio de documentos tributarios; Facturación_Electronica (el componente creado) posee una gran cobertura funcional, pero presenta defectos en su usabilidad, confiabilidad, eficiencia y portabilidad. Es recomendado en situaciones en la que el esfuerzo a utilizar por parte del integrador sea limitado, y en las cuales se esté construyendo un sistema para apoyar a contribuyentes de pequeña a mediana escala. Los resultados a grandes rasgos fueron resumidos en la tabla 5.2.

En relación a trabajos futuros relacionados a la propuesta, se ha pensado la posibilidad de:

- Evaluar un mayor número de candidatos utilizando la rúbrica, los cuales no fueron posibles de evaluar debido a limitantes económicas, y ver si esta logra caracterizar los atributos de calidad y diferentes necesidades de los stakeholders.

- Realizar una revisión de la rúbrica en base al modelo de calidad ISO 25010, puesto que la rúbrica originada está basada en modelos de calidad ajustados para componentes, que estaban fuertemente ligados a ISO 9126. De las propuestas encontradas, Q'Facto 12 fue la única en hacer un nexo con ISO 25010.

- Agregar a la rúbrica las características de calidad en uso: Evaluar atributos de calidad en uso pueden otorgar nuevas perspectivas de los beneficios o contras de la utilización de cada uno de los componentes. Estas características fueron omitidas en este trabajo dado que la calidad de uso es relativa los contextos, los cuales son propios de un usuario y sus condiciones (equipamiento, tareas a realizar, conocimientos), por lo que son difíciles de comparar dado a lo condicionados que están. Se requiere además, tener una base de usuarios que hayan sido enfrentados directamente al producto final, hecho que para el componente creado no se cumple. Como trabajo futuro, se propone establecer un método cuantitativo que permita comparar atributos propios de calidad en uso como satisfacción, productividad o eficiencia. O en su defecto, validar la calidad en uso del componente desarrollado.

Referencias

[1] CENTRO de estudios de la economía digital. Perspectivas de la factura electrónica en Chile. Santiago, Cámara de comercio de Santiago, 2003. 2 p.

[2] Pour G. (1998), Component-based software development approach: new opportunities and challenges

[3] LEAL B., Tatiana. y NAVEA B., Paola. Facturación electrónica. Revista de Estudios Tributarios (10): 51-67, Julio, 2014.

[4] Ley N° 825. Diario Oficial de la República de Chile, Santiago, 30 de noviembre de 1976.

[5] A PROCESS for COTS Software Product Evaluation por S. COMELLA-DORDA [et al]. Estados Unidos, Software Engineering Institute, Carnegie Mellon University, 2004.

[6] SONI, Nikita, JHA, S. K. Component Based Software Development : A new Paradigm. En: International Journal Of Scientific Research And Education (2°, 2014, India). Amity Institute of Information and Technology, 2014 pp. 969-974.

[7] TAHVILDARI, Ladan. Component-Based Software Systems: Testing and Management of Component-Based Systems [diapositivas]. Canadá. University of Waterloo, 2011. 77 diapositivas, col..

[8] TAHVILDARI, Ladan. Component-Based Software Systems: Quality Attributes and CBSE: Concerning Predictability [diapositivas]. Canadá. University of Waterloo, 2011. 47 diapositivas, col..

[9] TAHVILDARI, Ladan. Component-Based Software Systems: Component-Based Development Process [diapositivas]. Canadá. University of Waterloo, 2011. 72 diapositivas, col..

[10] AOYAMA, Mikio. New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development?. Japon, Department of Information and Electronics Engineering, Niigata Institute of Technology, 1998.

[11] PRADEEP, Tomar, NASIB, Singh G. New Algorithm for Component Selection to Develop Component-Based Software with X Model. En: Lecture Notes on Software Engineering (3°, 2013, Estados Unidos). IACSIT, 2013 pp. 298-302.

[12] KRASNER, Glen E., POPE, Stephen T. A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80. J. Object Oriented Program. (1): 26-49, Agosto-Septiembre, 1988.

[13] COMPONENT-BASED Rails Applications por HAGEMANN Stephan [et al]. Leanpub, 2016.

[14] IEEE. IEEE Standard Glossary of Software Engineering Terminology en: IEEE Std 610.12 (1990). 1990 pp 1-84.

[15] McCall, RICHARDS, Jim A., WALTERS, Gene F., RICHARDS, Paul K. Factors in Software Quality. Estados Unidos, General Electric Company, 1977. 1 v.

[16] BAWANE, Neelam, SRIKRISHNA, C. V. A Novel Method for Quantitative Assessment of Software Quality. En: International Journal of Computer Science and Security (3°, 2009, Malasia). PES Institute Technology, 2009 pp 508-517.

[17] BOEHM, Barry W., BROWN, John R., LIPOW, M. Quantitative evaluation of software quality. En: Proceedings of the 2nd international conference on Software engineering. IEEE Computer Society Press (2°, 1976, Estados Unidos), IEEE Computer Society, 1976 pp. 592-605.

[18] GRADY, Robert B. Practical software metrics for project management and process improvement, Prentice Hall, 1992.

[19] ISO. IEC 9126-1: Software Engineering-Product Quality-Part 1: Quality Model. Geneva, International Organization for Standardization, 2001. 27 p.

[20] ISO. IEC25010 (2011) Systems and software engineering–Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. International Organization for Standardization. Geneva, International Organization for Standardization, 2011. 34 p.

[21] TOWARDS a Software Component Quality Model por Álvaro Alexandre [et al]. Brasil, Federal University of Pernambuco and C.E.S.A.R – Recife Center for Advanced Studies and Systems, 2005.

[22] BERTOA, Manuel F., VALLECILLO, Antonio. Quality attributes for COTS components. España, Universidad de Málaga, Departamento de Lenguajes y Ciencias de la Computación, 2002.

[23] RAWASHDEH, Adnan, MATALKAH, Bassem. A new software quality model for evaluating COTS components. Journal of Computer Science (2°, 2006, Estados Unidos). Science Publications, 2006 pp 373-381.

[24] KALAIMAGAL, Sivamuni, SRINIVASAN, Rengaramanujam. Q'Facto 12: an improved quality model for COTS components en ACM SIGSOFT Software Engineering Notes (35, 2):1-4. Marzo. 2010.

[25] THAPAR, Simrandeep Singh, SINGH, Paramjeet, RANI, Shaveta. Reusability-based quality framework for software components en ACM SIGSOFT Software Engineering Notes (39, 2):1-5. Marzo. 2014.

Anexo

A - Casos de uso de sistema ERP

Iniciar Sesión

Nombre: Iniciar sesión
Actores: Vendedor
Precondiciones: Vendedor debe existir en el sistema
Escenario principal:

1. Vendedor pone su nombre de usuario y contraseña
2. Sistema valida el vendedor y lo dirige a menú principal

Extensiones
2a Validación de usuario o contraseña inválida

1. Sistema responde con error de validación
2. Vendedor ingresa sus datos correctamente
3. Sistema valida el vendedor y lo dirige a menú principal

Administrar Cuenta

Nombre: Administrar cuenta
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

1. Vendedor elige la opción de Administrar cuenta
2. Vendedor modifica su nombre y/o cambia su contraseña
3. Sistema modifica cuenta de vendedor exitosamente

Administrar Vendedores

Nombre: Administrar vendedores
Actores: Administrador
Precondiciones: Administrador existente y autentificado en el sistema
Escenario principal:

Este escenario describe la creación de un nuevo vendedor en el sistema.

1. Administrador elige la opción de crear nuevo vendedor en el sistema
2. Administrador elige nombre, contraseña y rol a nuevo vendedor.
3. Sistema crea exitosamente nuevo vendedor en el sistema

Extensiones
1a Administrador desea modificar vendedor existente en el sistema

1. Administrador busca vendedor por nombre
2. Sistema despliega resultados en una lista

3. Administrador elige el indicado
4. Administrador modifica nombre, contraseña o rol de vendedor existente
5. Sistema modifica exitosamente vendedor del sistema

1b Administrador desea eliminar vendedor existente en el sistema

1. Administrador busca vendedor por nombre
2. Sistema despliega resultados en una lista
3. Administrador elige el indicado
4. Administrador elige "Eliminar vendedor" y confirma
5. Sistema elimina exitosamente vendedor del sistema
    a. Vendedor no podrá ser eliminado si tiene guías de venta asociadas

Administrar Clientes

Nombre: Administrar Clientes
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

Este escenario se da cuando el vendedor quiere crear un nuevo cliente en el sistema

1. Vendedor elige la opción de crear nuevo cliente
2. Vendedor ingresa datos personales del clientes: Nombre, RUT, giro, teléfono fijo, y móviles (si tuviera), dirección, ciudad, comuna y fija una condición de pago para el cliente (de las ya existentes).
3. Sistema ingresa exitosamente cliente al sistema

Extensiones:

1a El vendedor quiere modificar un cliente ya existente

1. Vendedor elige la opción de modificar cliente
2. Vendedor busca al cliente ya existente por alguna de las siguientes propiedades: Nombre, RUT, email
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige el cliente indicado
5. Vendedor modifica alguno de los datos personales del cliente: Nombre, RUT, giro, teléfono fijo, y móviles (si tuviera), dirección, ciudad, comuna y fija una condición de pago para el cliente (de las ya existentes).
6. Sistema modifica exitosamente al cliente del sistema

1b El vendedor quiere eliminar un cliente ya existente

1. Vendedor elige la opción de modificar cliente
2. Vendedor busca al cliente ya existente por alguna de las siguientes propiedades: Nombre, RUT, email
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige el cliente indicado
5. Vendedor selecciona "Eliminar cliente" y confirma

6. Sistema eliminar exitosamente al cliente junto con sus guías de venta y otras entidades débiles del modelo asociadas.

Administrar Láminas

Nombre: Administrar Láminas
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

Este escenario se da cuando el vendedor quiere crear una nueva lámina en el sistema

1. Vendedor elige la opción de crear nueva lámina
2. Vendedor ingresa datos de la lámina: Nombre, código, marca, acabado, tipo de lámina, precios de referencia en el mercado (precio general neto y precio comprado neto) y el stock actual de la lámina en bodega (si fuera aplicable).
3. Sistema ingresa exitosamente la lámina al sistema

Extensiones:

1a El vendedor quiere modificar una lámina ya existente

1. Vendedor elige la opción de modificar lámina
2. Vendedor busca a la lámina ya existente por alguna de las siguientes propiedades: Nombre, Código, Marca
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige la lámina indicada
5. Vendedor modifica alguno de los datos de la lámina: Nombre, código, marca, acabado, tipo de lámina, precios de referencia en el mercado (precio general neto y precio comprado neto) y el stock actual de la lámina en bodega (si fuera aplicable).
6. Sistema modifica exitosamente la lámina del sistema.

1b El vendedor quiere eliminar una lámina ya existente

1. Vendedor elige la opción de modificar lámina
2. Vendedor busca a la lámina ya existente por alguna de las siguientes propiedades: Nombre, Código, Marca
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige la lámina indicada
5. Vendedor selecciona “Eliminar lámina” y confirma
6. Sistema elimina exitosamente a la lámina junto con sus entidades débiles asociadas.

Administrar Artículos

Nombre: Administrar Artículos
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

Este escenario se da cuando el vendedor quiere crear un nuevo artículo en el sistema

1. Vendedor elige la opción de crear nuevo artículo
2. Vendedor ingresa datos de la lámina: Nombre, código, precio neto, tipo y unidad de medida principal (largo en caso de muebles, unidad caso de clavos, etc.).
3. Sistema ingresa exitosamente el artículo al sistema

Extensiones:

1a El vendedor quiere modificar un artículo ya existente

1. Vendedor elige la opción de modificar artículo
2. Vendedor busca el artículo ya existente por alguna de las siguientes propiedades: Nombre, Código
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige el artículo indicado
5. Vendedor modifica alguno de los datos de la lámina: Nombre, código o precio neto.
6. Sistema modifica exitosamente el artículo del sistema.

1b El vendedor quiere eliminar un artículo ya existente.

1. Vendedor elige la opción de modificar artículo
2. Vendedor busca el artículo ya existente por alguna de las siguientes propiedades: Nombre, Código
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige el artículo indicado
5. Vendedor selecciona “Eliminar artículo” y confirma
6. Sistema elimina exitosamente el artículo junto con sus entidades débiles asociadas.

Emitir o actualizar una guía de venta

Nombre: Emitir o actualizar una guía de venta

Actores: Vendedor, Cliente

Precondiciones: Vendedor existente y autentificado en el sistema

Escenario principal:

Este escenario describe la emisión de una nueva guía de venta en el sistema.

1. Vendedor elige la opción de emitir guía de venta
2. Vendedor elige el RUT del cliente a emitir la guía de venta
3. Vendedor ingresa hora y fecha de entrega del pedido
4. Vendedor ingresa cada artículo de la guía de venta
    a. Vendedor busca el artículo por su código o nombre
    b. Sistema despliega lista de resultados que coincidan con la búsqueda
    c. Vendedor selecciona el artículo indicado
    d. Vendedor determina el faldón, fondo y largo (En el caso de un artículo tipo mueble), en otro caso, vendedor determina el tamaño (en unidad a definir por el artículo) del artículo que se lleva.
    e. Vendedor determina la lámina que irá con el mueble

i. Vendedor busca la lámina por su código, nombre o marca

ii. Sistema despliega lista de resultados que coincidan con la búsqueda

iii. Vendedor selecciona el artículo indicado

f. Vendedor determina cantidad a llevar del artículo

5. Vendedor agrega recargos y/o descuentos al pedido
6. Vendedor ingresa abono por parte del cliente.
7. Sistema ingresa correctamente guía de venta al sistema.

Extensiones:

2a El RUT ingresado no existe en el sistema

1. Vendedor ingresa datos personales del clientes: Nombre, RUT, giro, teléfono fijo, y móviles (si tuviera), dirección, ciudad, comuna y fija una condición de pago para el cliente (de las ya existentes).
2. Sistema ingresa exitosamente cliente al sistema

1a Vendedor quiere modificar una guía de venta ya existente en el sistema

1. Vendedor elige la opción de modificar una guía de venta
2. Vendedor busca la guía de venta por el RUT del cliente, o por el identificador de la guía de venta
3. Vendedor puede modificar/eliminar/agregar cualquier artículo de la guía de venta

a. Siempre que este artículo no se encuentre asociado a ninguna factura

4. Vendedor puede modificar el abono, recargos y/o descuentos asociados al pedido
5. Sistema modifica correctamente la guía de venta en el sistema

Emitir una factura a partir de una guía de venta

Nombre: Emitir una factura a partir de una guía de venta

Actores: Vendedor

Precondiciones: Vendedor existente y autentificado en el sistema, guía de venta existente en el sistema

Escenario principal:

1. Vendedor elige la opción de emitir factura
2. Vendedor busca una guía de venta por su identificador o RUT de cliente asociado
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige la guía de venta indicada
5. Vendedor elige el monto de cada artículo a facturar (sistema muestra saldo remanente a facturar)
6. Sistema emite factura correctamente en el sistema

Emitir una factura electrónica a partir de una factura

Nombre: Emitir una factura electrónica a partir de una factura
Actores: Vendedor, Facturación SII.cl
Precondiciones: Vendedor existente y autentificado en el sistema, guía de venta existente en el sistema, Factura existente en el sistema
Escenario principal:

1. Vendedor elige la opción de emitir factura electrónica
2. Vendedor busca una factura por su número de documento, RUT a facturar o guía de venta asociada
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige la factura indicada
5. Sistema genera XML de factura electrónica acuerdo a los estándares del SII.cl
6. Sistema envía bajo conexión segura a Facturación SII.cl el XML de la factura
7. Facturación SII.cl verifica la factura y manda información de éxito a Sistema
8. Sistema confirma la transacción y marca la factura como firmada electrónicamente.

Obtener historial de un cliente

Nombre: Obtener historial de un cliente
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

Este escenario describe la consulta de historial de un cliente existente en el sistema

1. Vendedor elige la opción de modificar cliente
2. Vendedor busca al cliente ya existente por alguna de las siguientes propiedades: Nombre, RUT, email
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige el cliente indicado
5. Sistema despliega datos personales del cliente, junto con detalle de guías de venta históricas de cliente, facturas y cheques asociados.
    a. Cada venta histórica tendrá su detalle de artículos, abono y estado de guía de venta
    b. Facturas mostrarán los ítems facturados, con sus cargos, RUT, a facturar, giro, fecha de emisión, número documento, y pedido al cual se asocia
    c. Cheques mostraran pedido al cual se asocia, banco, plaza, RUT girador, cuenta corriente, valor, fecha vencimiento, si fue cancelado o no (y en qué fecha), y el estado actual del mismo

Ingresar cheque abonado por el cliente

Nombre: Ingresar cheque abonado por el cliente

Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

1. Vendedor elige la opción ingresar compromiso
2. Vendedor busca una guía de venta por su identificador o RUT de cliente asociado
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige la guía de venta indicada
5. Vendedor ingresa datos del nuevo compromiso: Fecha del cheque, rut girador, glosa (si aplica), número documento (numero cheque), cuenta corriente, valor, banco y plaza, estado actual, y si esta cancelado o no (si está cancelado se incluye fecha).
6. Sistema ingresa exitosamente el compromiso al sistema.

Administrar estados de guía de venta

Nombre: Administrar estados de guía de venta
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

Este escenario se da cuando el vendedor quiere crear un estado de guía de venta en el sistema

1. Vendedor elige la opción ingresar estado de guía de venta
2. Vendedor ingresa descripción del nuevo estado de guía de venta.
3. Sistema ingresa exitosamente el estado de guía de venta al sistema.

Extensiones:

1a El vendedor quiere modificar un estado de guía de venta ya existente en el sistema

1. Vendedor elige la opción modificar estado de guía de venta
2. Vendedor elige el estado de guía de venta que desea modificar de una lista
3. Vendedor cambia la descripción del estado de guía de venta
4. Sistema modifica exitosamente el estado de guía de venta del sistema.

1b El vendedor quiere eliminar un estado de guía de venta ya existente en el sistema

1. Vendedor elige la opción modificar estado de guía de venta
2. Vendedor elige el estado de guía de venta que desea modificar de una lista
3. Vendedor elige “Eliminar estado de guía de venta” y confirma
4. Sistema elimina exitosamente el estado de guía de venta

Administrar recargos y descuentos

Nombre: Administrar recargos y descuentos
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

Este escenario se da cuando el vendedor quiere crear un nuevo recargo o descuento en el sistema

1. Vendedor elige la opción ingresar recargo/descuento
2. Vendedor ingresa descripción, tipo (VF,VV,PF,PV) y cantidad (si aplica) del recargo/descuento.
3. Sistema ingresa exitosamente el recargo/descuento al sistema.

Extensiones:

1a El vendedor quiere modificar un recargo/descuento ya existente en el sistema

1. Vendedor elige la opción modificar recargo/descuento
2. Vendedor elige el recargo/descuento que desea modificar de una lista
3. Vendedor cambia la descripción, tipo o cantidad del recargo/descuento.
4. Sistema modifica exitosamente el recargo/descuento del sistema.

1b El vendedor quiere eliminar un recargo/descuento ya existente en el sistema

1. Vendedor elige la opción modificar recargo/descuento
2. Vendedor elige el recargo/descuento que desea modificar de una lista
3. Vendedor elige "Eliminar recargo/descuento" y confirma
4. Sistema elimina exitosamente el recargo/descuento.
    a. Guías que ya se encuentren con el recargo/descuento asociado, permanecerán con una copia estática de dicho recargo/descuento, por lo que su eliminación no produce efecto cascada

Administrar bancos y plazas

Nombre: Administrar bancos y plazas

Actores: Vendedor

Precondiciones: Vendedor existente y autentificado en el sistema

Escenario principal:

Este escenario se da cuando el vendedor quiere crear un nuevo banco/plaza en el sistema

1. Vendedor elige la opción crear banco/plaza
2. Vendedor ingresa el nombre en el caso del banco, descripción y que banco se asocia de una lista desplegable en el caso de una plaza
3. Sistema crea exitosamente banco/plaza en el sistema.

Extensiones:

1a Vendedor quiere modificar un banco/plaza ya existente en el sistema

1. Vendedor elige la opción de modificar banco/plaza
2. Vendedor busca el banco/plaza a modificar de una lista
3. Vendedor selecciona el indicado
4. Vendedor modifica el nombre en el caso del banco, descripción en el caso de una plaza
5. Sistema modifica exitosamente banco/plaza en el sistema.

1b Vendedor quiere eliminar un banco/plaza ya existente en el sistema

1. Vendedor elige la opción de modificar banco/plaza

2. Vendedor busca el banco/plaza a modificar de una lista
3. Vendedor selecciona el indicado
4. Vendedor elige “Eliminar banco/plaza” y confirma
5. Sistema elimina exitosamente banco y plaza del sistema
    a. En el caso de existir un compromiso existente con esa plaza, no permitirá la eliminación
    b. La eliminación de un banco eliminará todas sus plazas asociadas, siempre que una de sus plazas no se encuentre asociada a un compromiso existente, en dicho caso, no permitirá la eliminación.

Imprimir o enviar por correo guía de venta

Nombre: Imprimir o enviar por correo guía de venta
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema, guía de venta ya existente en sistema
Escenario principal:

1. Vendedor elige imprimir o enviar por correo guía de venta
2. Vendedor elige guía de venta
3. Vendedor buscar guía de venta por su identificador o RUT de cliente
4. Sistema despliega una lista de resultados
5. Vendedor elige el indicado
6. Vendedor selecciona si se desea imprimir o enviar por correo, y si es para cliente o uso interno
7. Sistema genera documento para imprimir o mandar por correo.

Imprimir lista de láminas

Nombre: Imprimir listado de láminas
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema
Escenario principal:

1. Vendedor elige la opción de listado de láminas
2. Vendedor selecciona si lista es para uso interno o vendedores
3. Sistema genera documento para imprimir de listado de láminas

Imprimir factura electrónica

Nombre: Imprimir factura electrónica
Actores: Vendedor
Precondiciones: Vendedor existente y autentificado en el sistema, factura existente y marcada como ya facturada electrónicamente
Escenario principal:

1. Vendedor elige la opción de emitir factura electrónica
2. Vendedor busca una factura por su número de documento, RUT a facturar o guía de venta asociada
3. Sistema despliega lista de resultados que coincidan con la búsqueda
4. Vendedor elige la factura indicada
5. Sistema genera código de barra bidimensional PDF417 con información del timbre electrónica e información necesaria para validarlo (acorde a sii.cl)
6. Sistema genera detalle de la factura acorde a formato de sii.cl, listo para su impresión.

B - Esquemas de modelos de datos

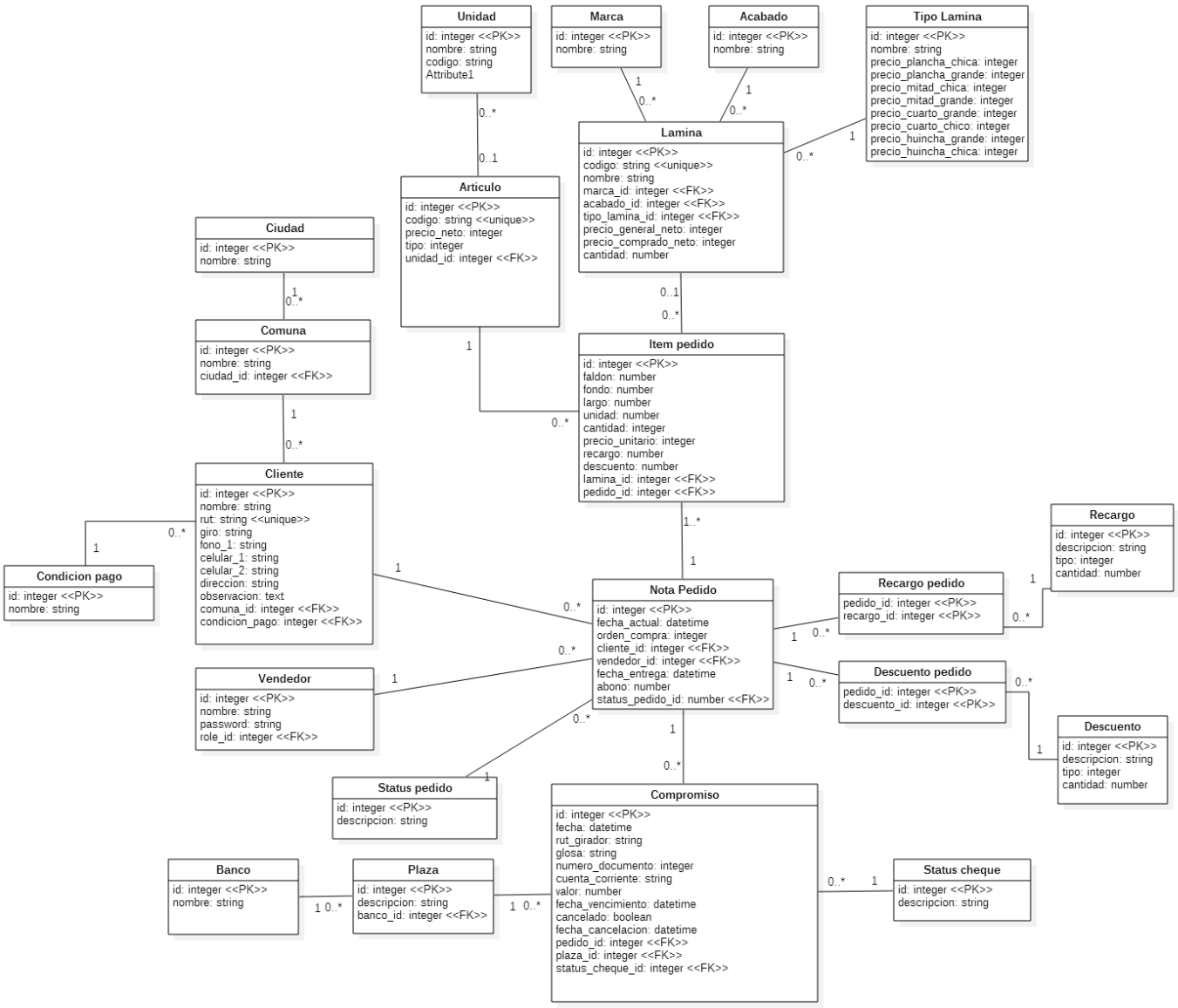

Figura B.1: Esquema de modelo de datos para sistema ERP.

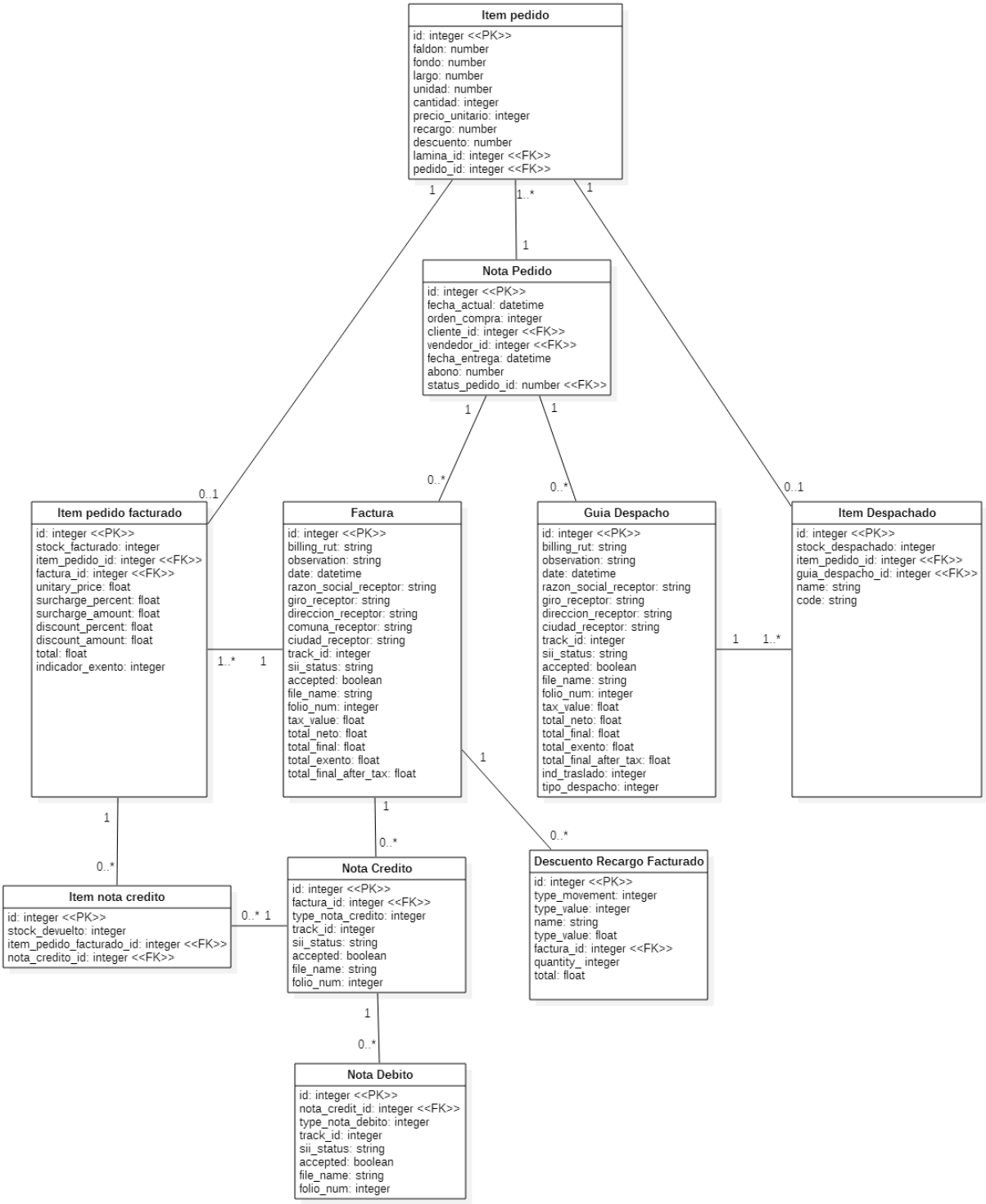

Figura B.2: Esquema de modelo de datos para sistema ERP, módulo de facturación electrónica.

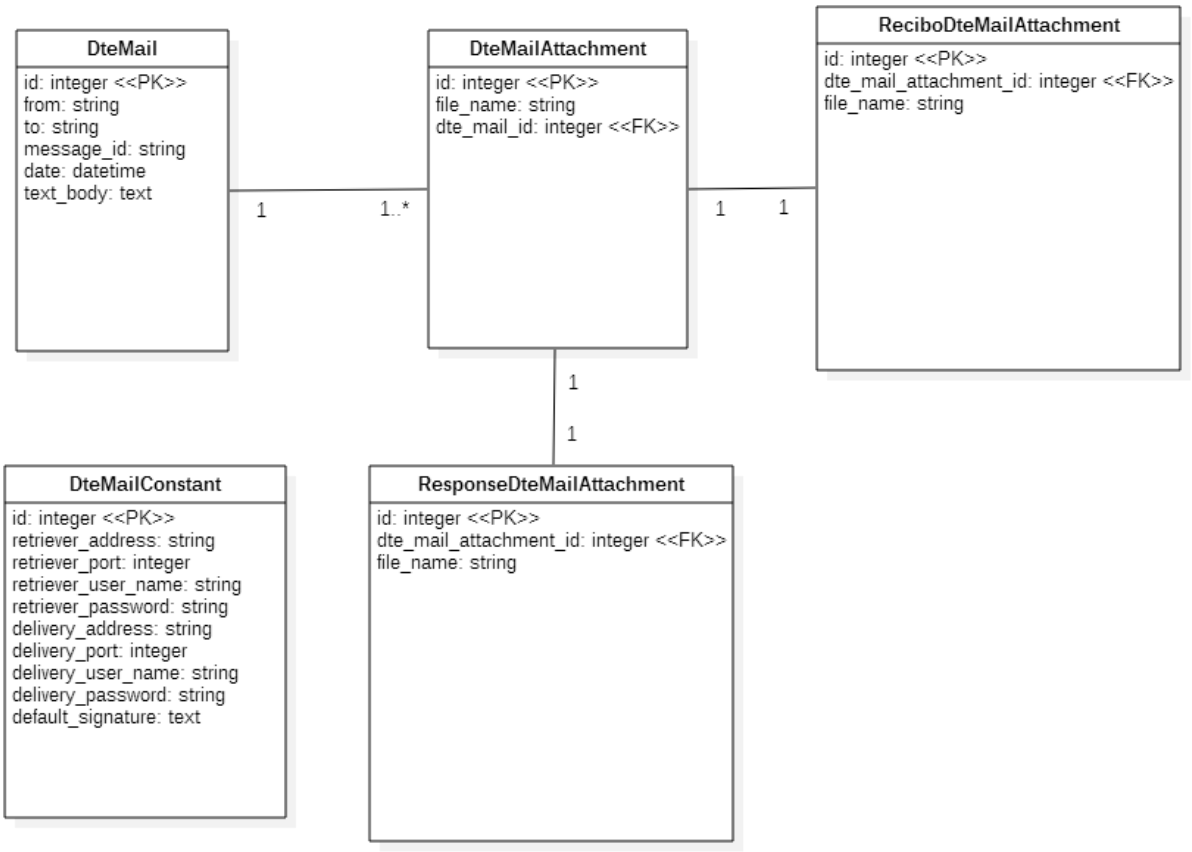

Figura B.3: Esquema de modelo de datos para sistema ERP, administración de correos.