ESPECIFICACIONES
FACTURA CCI UBL

Versión 1.0

Junio 2007

HISTORIAL DEL DOCUMENTO

A. VERSIONES

Nº Revisión	Fecha Revisión	Resumen
WD 0.1	01/05/2007	Versión inicial
WD 0.2	15/05/2007	Primer borrador de trabajo
WD 0.3	01/06/2007	Segundo borrador
WD 0.4	15/06/2007	Tercer borrador
R 1.0	27/06/2007	Primera versión

B. EDITORES

Nombre	Iniciales	Organización	Correo Electrónico
Oriol Bausà Peris	OB	Invinet Sistemes	oriol@invinet.org
Pablo del Canto Rodrigo	PC	UPC	pcanto@ac.upc.edu
Sergi Cabré Longás	SC	UPC	scabre@ac.upc.edu
Juan Carlos Cruellas	JC	UPC	cruellas@ac.upc.edu
Brais Méndez Tellez	BM	Invinet Sistemes	brais@invinet.org
Jordi Bosom Valmaña	JB	Invinet Sistemes	jbosom@invinet.org

RESTRICCIONES DE PUBLICACIÓN

Tabla de Contenidos

Índice de Figuras

Ejemplos

1 INTRODUCCIÓN

Este proyecto consiste en la continuación de los trabajos realizados en el proyecto de convergencia entre la factura electrónica AEAT-CCI definida por el Centro de Cooperación Interbancaria y la Agencia Tributaria y el modelo internacional de factura UBL 2.0. Fruto de aquel primer proyecto resultaron las especificaciones para la factura CCI UBL [UBLCCI] que recoge los requisitos definidos en el modelo AEAT-CCI y los traslada al modelo UBL 2.0, obteniendo una personalización de la factura electrónica UBL 2.0.

El estándar UBL 2.0 consiste en una librería de componentes y en un conjunto de 22 documentos que cubren algunas de las áreas de eProcurement más relevantes:

- Etapa de selección de productos
  - Petición de Catálogo,
  - Catálogo,
  - Modificación de Especificación de Elemento de Catálogo,
  - Modificación de Precio de Catálogo,
  - Borrado de Catálogo,
  - Solicitud de Presupuesto,
  - Presupuesto
- Etapa de gestión del pedido
  - Pedido,
  - Respuesta a Pedido,
  - Respuesta a Pedido Sencilla,
  - Modificación de Pedido,
  - Cancelación de Pedido
- Etapa de entrega de bienes o provisión del servicio
  - Albarán de Entrega,
  - Notificación de Recepción
- Etapa de facturación
  - Factura,
  - Nota de Abono,
  - Nota de Débito,
  - Autofactura,

- Autonota de Abono,
- Recordatorio

- Etapa de pago
  - Notificación de pago,
  - Extracto

La definición de la factura CCI UBL permite utilizar la librería de componentes de UBL 2.0, o cuanto menos el conjunto de componentes personalizados de la librería de componentes de UBL 2.0 para la factura electrónica, para definir otros documentos que permitan la definición de procesos de negocio completos.

Este desarrollo tiene como finalidad facilitar la adopción de mecanismos de eProcurement a las empresas, ya sea a través de la adquisición de aplicaciones que incluyan estos mecanismos out-of-the-box, como a través del desarrollo de artefactos específicos para sus sistemas internos de gestión.

El trabajo realizado con la Factura CCI UBL constituye un primer paso que este proyecto pretende continuar y complementar con los siguientes elementos:

- Definición de los perfiles de intercambio que se pueden producir en situaciones de facturación.
- Definición de los posibles escenarios aplicables a cada uno de dichos perfiles de intercambio.
- Generación de modelos de ejemplo de distintos tipos de factura
- Análisis de requerimientos de los componentes y documentos para su posterior desarrollo
- Personalización de los documentos adicionales requeridos para la definición de los perfiles de intercambio
- Definición de los mecanismos necesarios para que se puedan desarrollar los sistemas de validación, entrada de datos y visualización de las instancias XML que participan en estos procesos de intercambio.

1.1 Organización del documento

El documento se organiza en dos partes, en la primera parte se definen los perfiles de intercambio para los procesos de facturación electrónica en España, los distintos escenarios para cada perfil y se proponen distintos modelos de documento factura. Se describen asimismo los documentos adicionales a utilizar en perfiles de facturación complejos. Aunque no es objeto de este documento realizar una descripción detallada de los documentos adicionales se hará referencia a ellos y se describirá someramente su uso.

En la segunda parte se especifica el análisis de las librerías a desarrollar para la factura CCI UBL. Como podremos comprobar, las librerías a desarrollar tienen cierto nivel de independencia con los perfiles de intercambio definidos en este documento, e incluso con los modelos o ejemplos de factura a utilizar, quedando a expensas del desarrollador el uso de los distintos componentes y clases definidos para la creación de facturas que resulten válidas para el perfil de intercambio al que deseen dar respuesta.

1.2 Audiencia

El público objetivo de este documento es personal técnico que desee añadir prestaciones de emisión y/o recepción de facturas electrónicas en sus procesos de negocio.

La primera parte tiene un público potencial más amplio, destacando consultores, responsables de áreas TIC e incluso gerentes o responsables de organizaciones que quieran introducir sistemas de intercambio electrónico de datos en sus procesos.

La segunda parte de análisis, sin embargo, tiene un alto contenido técnico, por lo que su lectura es recomendada para jefes de proyecto, analistas y programadores, que deseen incorporar facilidades de facturación electrónica en sus sistemas de gestión empresarial.

1.3 Asunciones

Se supone que el lector posee conocimientos en intercambio electrónico de datos y en tecnología XML y UBL y en circuitos de facturación. También se asumen conocimientos en desarrollo informático y en la utilización de patrones de diseño.

1.4 Convenciones tipográficas

En el documento se han seguido un conjunto de convenciones tipográficas y de formato a fin de facilitar la lectura al desarrollador de la librería.

Los fragmentos de código de ejemplo escritos en el documento son presentados usando el siguiente formato:

```
Código de ejemplo
```

El desarrollador debe tener en cuenta que tiene que realizar una serie de sustituciones en la nomenclatura de los métodos y/o clases:

- Cuando en el presente documento se cita *TipoElementoX*, el desarrollador deberá sustituir por el tipo o clase del elemento que corresponda.
- Cuando en el presente documento se cita *TipoDato*, el desarrollador deberá sustituir por el tipo de dato básico que corresponda.
- Cuando en el presente documento se cita *TipoBasicoLenguaje*,

el desarrollador deberá sustituir por el tipo de dato básico del lenguaje que corresponda.

- Cuando en el presente documento se cita *TipoEnumCodigo*, el desarrollador deberá sustituir por el tipo de dato o clase que represente la enumeración que corresponda.
- Cuando en el presente documento se cita *ElementoNombreX*, el desarrollador deberá sustituir por el nombre del elemento que corresponda.
- Cuando en el presente documento se cita *AtributoX*, el desarrollador deberá sustituir por el nombre del atributo que corresponda.

PARTE I
PERFILES

2 DEFINICIÓN DE PERFILES Y ESCENARIOS

La primera parte de este documento define los perfiles de intercambio para la facturación electrónica, desde los más sencillos que permitirán únicamente la remisión de documentos, sin ningún tipo de interacción entre las partes más allá de la transacción de factura, hasta los más complejos orientados a la resolución sistémica de errores en la facturación, como por ejemplo la resolución de diferencias en el precio, o errores en los conceptos u otros elementos de la factura.

El número de perfiles que se pueden gestionar con únicamente un solo documento son muy limitados, de hecho, en este documento se puede comprobar la necesidad de disponer de más documentos de manera que se puedan definir perfiles de intercambio más sistémicos. Aunque no se contemple en este proyecto, un caso muy claro de perfil complejo sería la incorporación del pedido en un circuito típico de aprovisionamiento de Pedido a Factura.

En el momento que se definan nuevos documentos del circuito de aprovisionamiento como por ejemplo el mismo pedido o las notificaciones de entrega, albaranes, etc, se podrá ampliar el número de perfiles y escenarios.

2.1 Definiciones

En este apartado se definirán conceptos necesarios para la mejor comprensión de esta parte del documento.

2.1.1 Conformidad

Para que una instancia de factura electrónica XML se considere que es conforme a la especificación CCI UBL, debe también ser conforme a la especificación UBL 2.0.

Una instancia de factura CCI UBL debe ser conforme a UBL 2.0, pero una instancia de factura UBL puede ser o no ser conforme a CCI UBL. Esto ocurre porque CCI UBL es un subconjunto de UBL..

Por tanto, para determinar que una instancia de factura electrónica XML es conforme a CCI UBL, se deberá comprobar que es conforme a UBL y que cumple con las restricciones impuestas por CCI UBL mediante los mecanismos de validación Schematron que se deberán desarrollar a tal efecto.

Tal como se define en UBL, una instancia XML se considera conforme a UBL si:

- no existen violaciones de las restricciones a nivel de documento cuando se valida la instancia contra los esquemas UBL XSD publicados.
- no hay elementos añadidos (excepto en el caso de utilizar el punto de Extensión presente en todos los documentos UBL)
- todos los elementos obligatorios están presentes.

- los valores siguen las reglas de los tipos de datos.

Existe efectivamente la posibilidad de incluir elementos externos a UBL en un punto de extensión establecido a este efecto, y en el ámbito de CCI UBL se utiliza esta facilidad para la inclusión de firmas electrónicas avanzadas en las facturas.

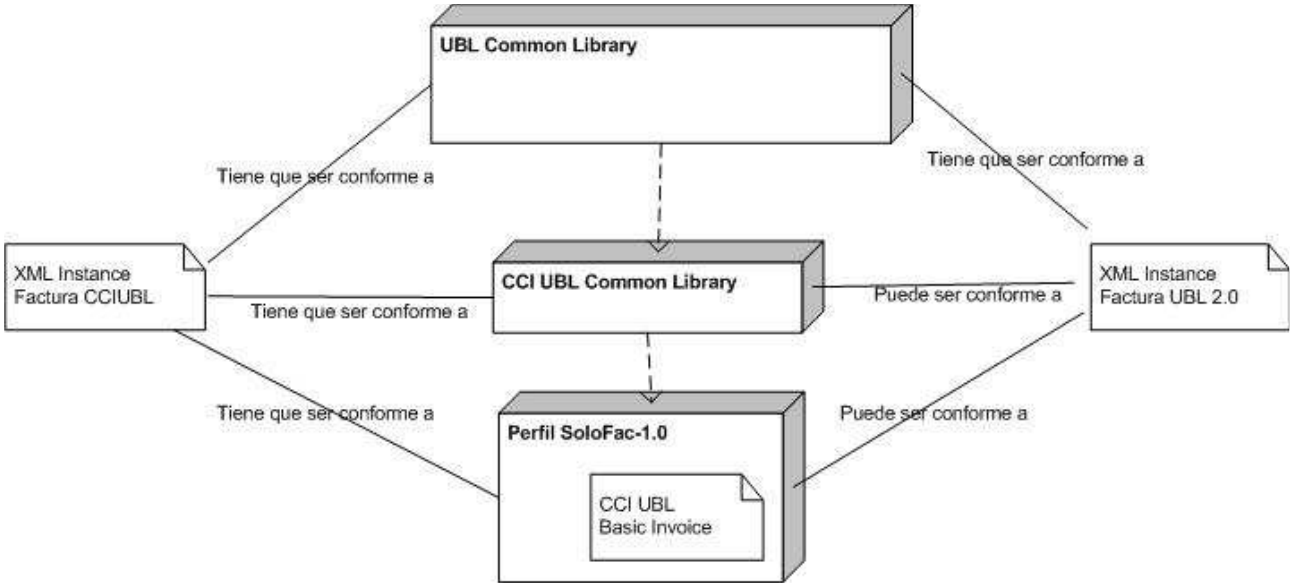


Figura 1: Conformidad de las instancias de factura XML

Las instancias de facturas CCI UBL tienen más restricciones que las instancias de facturas UBL, aunque estas últimas puedan cumplir las restricciones CCI UBL.

2.1.2 Personalización

Una definición de personalización especifica los siguientes aspectos:

- selección de los componentes de UBL predefinidos que son de interés para la personalización, entre los que deberán estar todos aquellos elementos obligatorios de UBL y los opcionales que sean de aplicación a la personalización,
- todos aquellos elementos que se definirán a partir del punto de extensión de UBL, y que no perteneciendo al estándar UBL son necesarios para la personalización. En el caso de CCI UBL se ha definido que en el punto de extensión debe aparecer la firma electrónica XAdES, en cualquiera de las formas que se requiera a nivel legal,
- especificación de las listas de códigos y las reglas de validación para cada uno de los elementos codificados,

- especificación de las reglas de negocio que obligan a las restricciones entre componentes (como por ejemplo, que el identificador del componente UBL Signature sea el mismo que el de la extensión XAdES)

Una personalización como la definida en el marco del proyecto Factura CCI UBL define el uso de instancias XML de factura electrónica en el contexto geográfico de España

.

2.1.3 Perfiles de intercambio y escenarios

Tal como se ha comentado, el documento Factura CCI UBL es una restricción de la factura Invoice UBL 2.0. La personalización se realizó de acuerdo con los requerimientos funcionales de la factura AEAT-CCI y básicamente consiste en la restricción del contenido del documento en base a la eliminación de elementos y de la cardinalidad de estos elementos del modelo original.

Desde un punto de vista de arquitectura de componentes, CCI UBL consiste pues en una vista restringida de la Librería UBL 2.0 y del documento factura UBL.

Con el fin de conseguir una adopción paulatina de los procesos de facturación electrónica, y por extensión, de futuros procesos de gestión de pedidos, entregas, etc. se definen los “perfiles” de intercambio.

Los perfiles de intercambio son conjuntos de escenarios o procesos de negocio que especifican las posibles transacciones sistémicas entre las partes. Cada usuario define el perfil que soportan sus sistemas de información, de manera que cada interlocutor conoce a priori los procesos de negocio que soporta cada uno de los demás participantes, y por tanto sabe qué documentos puede emitir y/o recibir facilitándose de este modo el establecimiento de acuerdos de intercambio espontáneos entre las partes.

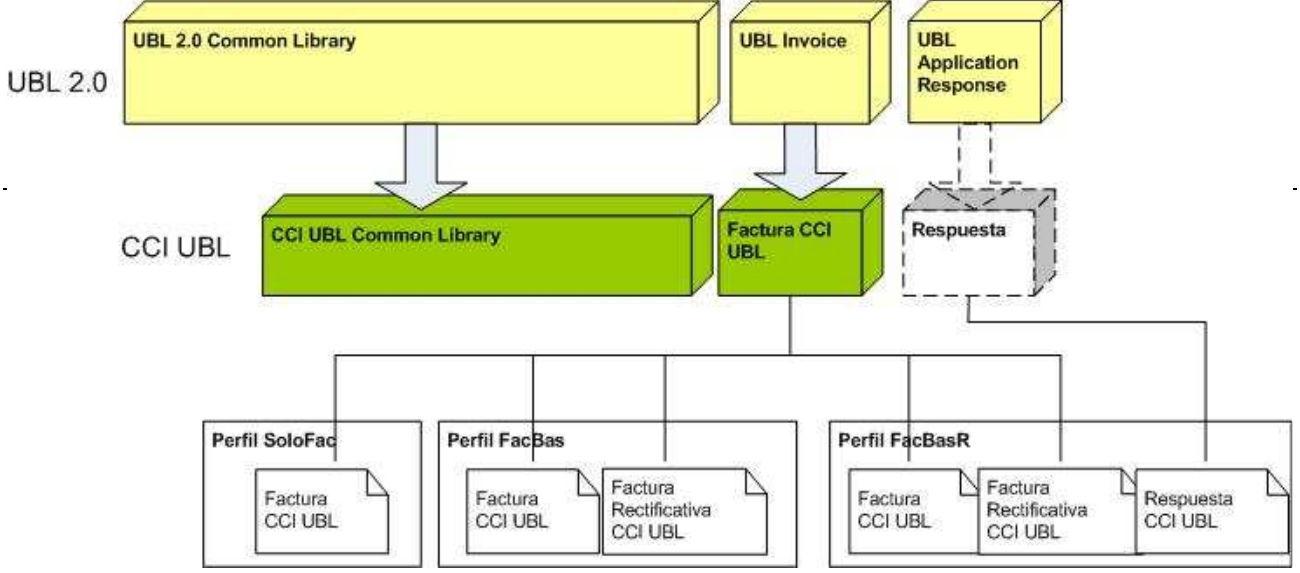


Figura 2: Personalización de UBL y perfiles CCI UBL

Un usuario puede pertenecer a un determinado perfil, si sus sistemas de información le permiten emitir y recibir todos los documentos que se definen en ese perfil.

Los perfiles se organizan como niveles de manera que a partir de un perfil básico se van incorporando nuevos escenarios o procesos de negocio incrementando la interacción sistémica entre los sistemas de información de los participantes en el intercambio electrónico de información.

Por ejemplo, el perfil más básico definido en este documento es el de emisión de la Factura, sin ningún otro tipo de interacción sistémica entre las partes. A partir de este perfil inicial, pensado para dar cabida a todo aquél que desee emitir o recibir facturas sin ningún otro tipo de interacción, se define el perfil de Facturación Básica, en el que además de la emisión/recepción de la factura, se puede emitir/recibir información de corrección de errores.

Cada perfil dispone de diversos escenarios, que describen las restricciones en tiempo de ejecución, por ejemplo, en el perfil de Facturación Básica que comentábamos en el párrafo anterior, los errores de la factura se pueden resolver mediante la emisión de facturas complementarias, o mediante la emisión de una factura rectificativa, dependiendo del tipo de error.

2.1.4 Modelos

Utilizaremos modelos para ilustrar los procesos de facturación desde una perspectiva de negocio, y mostrar como se pueden implementar estos procesos utilizando el documento factura CCI UBL.

Los modelos que se definirán en este documento facilitarán al desarrollador la utilización de las librerías ya que constituyen una capa superior con clases fachada y clases decorador que permiten el uso de las librerías sin tener que bajar al nivel de las clases de tipos de datos y de elementos básicos y agregados que forman el core de la librería.

Los modelos se relacionan con los perfiles en el sentido que en determinados perfiles existen restricciones que invalidan el uso de

determinados modelos. Por ejemplo, en un futuro perfil que corresponda al circuito Pedido-Factura, será obligatorio que se informe en la Factura el Número de Pedido al que hace referencia; la Factura Mínima, que no contiene esta información no se podría utilizar en este perfil, o cuanto menos se debería complementar con la información del Pedido asociado mediante el uso de un decorador.

2.1.5 Sistemas de Validación

Las instancias XML recibidas se pueden procesar con sistemas de validación que permiten la comprobación de la validez sintáctica, semántica y de reglas de negocio antes de ser cargadas por las librerías en la estructura de objetos. Para ello se pueden utilizar distintas herramientas entre las que destacamos:

- **Schema XSD**

  Dentro de los mecanismos de validación de instancias, la primera es la comprobación de la validez sintáctica y semántica del documento de acuerdo al esquema normativo.

  Los esquemas XSD son el primer mecanismo de validación de instancias Factura CCI UBL. Se utilizará el esquema normativo UBL 2.0 de la factura Invoice.xsd para determinar la validez de la factura CCI UBL.

  Para determinar la validez de las restricciones del documento que impone CCI UBL o bien las derivadas de los perfiles, se definirán ficheros Schematron adicionales.

- **Schematron**

  Schematron es un lenguaje XML basado en reglas que utiliza expresiones Xpath para describir reglas de validación de instancias.

  Cada documento Schematron debe contener las reglas de validación que deben cumplir las instancias de facturas XML para ser consideradas documentos válidos según la personalización CCI UBL, y en su caso, según cada uno de los perfiles definidos en este documento.

  Los perfiles incorporan reglas de negocio adicionales, como por ejemplo que una factura rectificativa debe contener el número de factura a rectificar.

- **Genericode**

  Las distintas listas de códigos requeridas dentro del esquema de la factura CCI UBL dispondrán de su correspondiente fichero Genericode que permite la validación en tiempo de ejecución de los códigos en las instancias XML recibidas.

  Las librerías incorporan un mecanismo que permite cargar listas

Genericode y facilita la definición de los metadatos de las listas de códigos, tanto en tipos de datos Code como en tipos de datos Identifier.

2.2 Actores

Existen diversos actores que pueden participar en un proceso de facturación, sin embargo, en el documento Factura CCI UBL existen únicamente dos actores obligatorios, el emisor de la factura y el receptor. Además de estos dos actores principales, la Factura CCI UBL puede contener otros actores que pueden ser importantes a la hora de procesar el contenido del documento (como por ejemplo el beneficiario en casos de facturas con factoring) pero no forman parte del núcleo del proceso. En alguno de los modelos veremos como intervienen estos actores adicionales.

Los actores tienen distintos nombres que expresan el rol que juegan cada uno de ellos en el proceso de facturación. Por claridad, utilizaremos el nombre Cliente y Proveedor para referirnos de forma genérica al AccountingCustomerParty y al AccountingSupplierParty que son los actores obligatorios en la Factura CCI UBL.

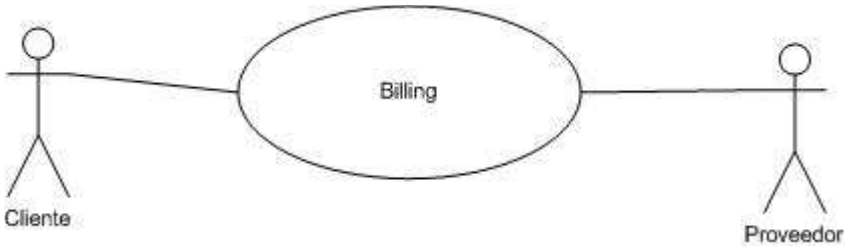


Figura 3: Caso de uso genérico Billing

2.3 Perfiles de intercambio

El objetivo de los perfiles de intercambio es que tanto el emisor como el receptor de un documento comercial puedan especificar los documentos que sus sistemas de gestión soportan y esperan recibir en un determinado contexto.

Un perfil es pues una descripción de uno o más procesos de negocio interconectados, consistiendo cada uno de ellos en el intercambio de uno o más documentos. En definitiva, un perfil expresa la existencia de un acuerdo para intercambiar documentos donde se establece:

- qué rol juega cada participante en el proceso de negocio
- qué documentos puede enviar y recibir cada participante en el

proceso de negocio electrónico.

- qué documentos debe poder enviar y recibir un determinado participante

El desarrollo de perfiles se basa en el modelo definido por UBL 2.0 y en el trabajo desarrollado por el Northern European Subset. En el caso de CCI UBL, y por lo tanto en este proyecto, únicamente se definirán los perfiles asociados al proceso de negocio de facturación (Billing).

Aunque sólo se definan los perfiles asociados a la facturación electrónica, se ha tenido que recurrir a la definición de nuevos documentos por compatibilidad con el estándar UBL 2.0, y para poder dar una idea de la riqueza que estos perfiles pueden suponer de cara a una difusión más amplia de la facturación electrónica y en general de los procesos de eProcurement en España. Más adelante se define el documento adicional que se ha requerido para poder definir los tres perfiles.

El primer objetivo de este documento consiste en obtener un perfil muy básico que permita una rápida y sencilla adopción por parte de un amplio grupo de usuarios, de manera que se pueda realizar un despliegue masivo de sistemas de facturación electrónica sin demasiadas restricciones, y sobretodo sin necesidad de realizar asunciones sobre los sistemas de los receptores. En este sentido, este tipo de perfil no requiere por ejemplo de la codificación o identificación de los elementos a facturar ya que su único propósito consiste en la digitalización de los procesos de facturación de las empresas.

Aparte de este primer perfil básico, cuyo único objetivo consiste en la emisión de facturas, se han definido otros dos perfiles adicionales donde se realiza interacción entre emisor y receptor, y las diferencias entre ellos, como veremos a continuación, dependen del nivel de automatización de procesos y de los documentos integrados en cada uno de los sistemas de gestión de los interlocutores.

Estos perfiles son:

- Sólo Factura
- Facturación Básica
- Facturación Básica con Control de Errores

Los perfiles de intercambio definidos en este documento se engloban en el espacio de nombres *urn:fundetec:profile-1.0* que cualificará el elemento ProfileID informándose en el atributo schemeDataURI.

Los valores de ProfileID serán:

- SoloFac-1.0 para el perfil de Solo Facturación
- FacBas-1.0 para el perfil de Facturación Básica
- FacBasR-1.0 para el perfil de Facturación Básica con Control de Errores.

2.3.1 Perfil Sólo Facturación

Especificación Perfil SoloFac-1.0

Identificador	SoloFac-1.0
Contexto	Este perfil aplica a • cualquier emisor o receptor de facturas de cualquier tamaño de empresa • de cualquier sector, privado o público • para cualquier tipo de bien, servicio o suministro
Descripción	Este perfil describe un proceso que únicamente comprende la emisión de un documento factura electrónica. Su objeto es que se aplique en situaciones donde la facturación sea electrónica pero no exista necesidad de relacionar el documento factura con otros documentos dentro de los circuitos de aprovisionamiento como pedidos o albaranes. El documento factura electrónica contiene todos los elementos necesarios para constituir un documento legal de acuerdo con la legislación establecida en España. La aceptación o rechazo del documento factura electrónica se debe realizar de forma manual y externa por lo tanto a este perfil de intercambio. Los objetivos principales de este perfil son permitir la sistematización de la gestión de facturas y eliminar la introducción de errores al reducir la gestión manual. No es objetivo de este perfil dar soporte a tareas de conciliación de facturas con albaranes o pedidos, o a discriminación automatizada en centros de coste, por lo que no será precisa información normalizada o identificadores alineados entre las partes. Este perfil se puede utilizar sin necesidad de realizar integración con los sistemas de gestión o ERPs. Si no se utilizan codificaciones, se puede utilizar con un mínimo acuerdo entre las partes, de hecho únicamente seria necesario el requerido a nivel legal de obtención del consentimiento por parte del receptor de la factura. En este perfil se pueden incorporar sistemas de pago avanzados como por ejemplo el factoring.
Documentos	Factura CCI UBL
Escenarios	• Factura aceptada • Factura rechazada
Requerimientos	1. El Proveedor envia una factura electrónica que puede ser recibida y procesada por el Cliente

	2. La factura electrónica debe cumplir con los requisitos legales para las facturas electrónicas en España, tanto a nivel de contenido como a nivel de firma electrónica. 3. El contenido de la factura electrónica debe permitir al sistema de información del Cliente su enrutado a la persona o departamento responsable.
Beneficios	Para el Proveedor • transferencia de facturas más rápida • acuse de recibo técnico de facturas emitidas • potencial para reducir el circuito factura-pago Para el Cliente • menor intervención manual y reducción de errores de tecleado • validación de facturas automático (importes, impuestos, sistemas de pago…) • incremento del control de las facturas recibidas
Actores	Proveedor • Quien exige el pago y se responsabiliza de la resolución de cualquier problema a nivel de facturación. Es el responsable de la emisión de la factura. Cliente • Quien recibe la factura y es responsable de su gestión.
Reglas de negocio	A nivel de documento • una factura representa la reclamación de un pago por la entrega de unos bienes o provisión de un servicio durante un período de tiempo • únicamente se puede hacer referencia a un contrato a nivel de documento • si aparecen, los medios y forma de pago, así como la información contable deben hacer referencia al conjunto de las líneas de la factura • la información de impuestos de la cabecera del documento debe aplicar a todas las líneas de la factura. • los anticipos deben hacer referencia a toda la factura. • la aceptación o rechazo de la factura no puede ser parcial • la factura debe contener una firma electrónica A nivel de línea de factura • cada línea de factura debe especificar un elemento ya sea codificado o descrito literalmente • la información de impuestos a nivel de línea se puede incluír a título informativo. Excepciones • Los problemas de rechazo de las facturas se deben gestionar fuera de este perfil.

	Precondiciones • El Cliente debe haber autorizado la recepción de facturas electrónicas • Se debe haber producido una relación contractual entre las partes, en forma de pedido o contrato.

Diagrama de actividad

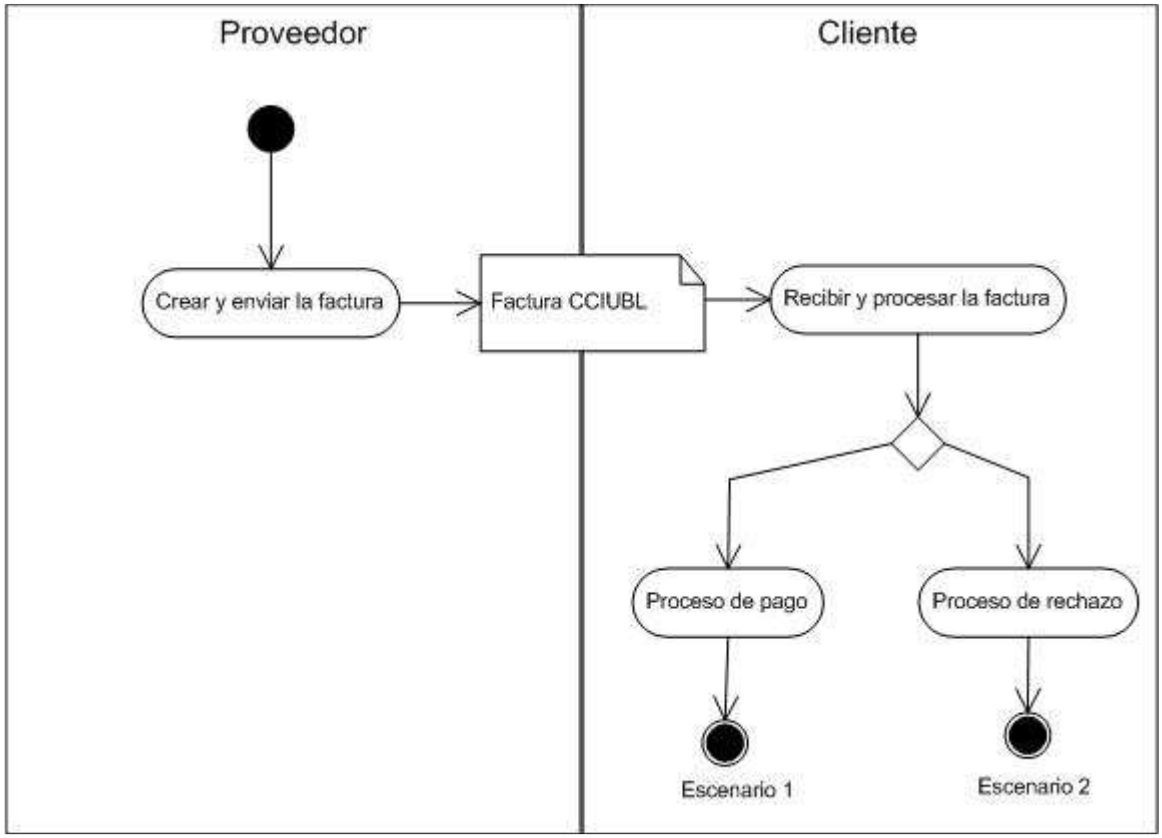


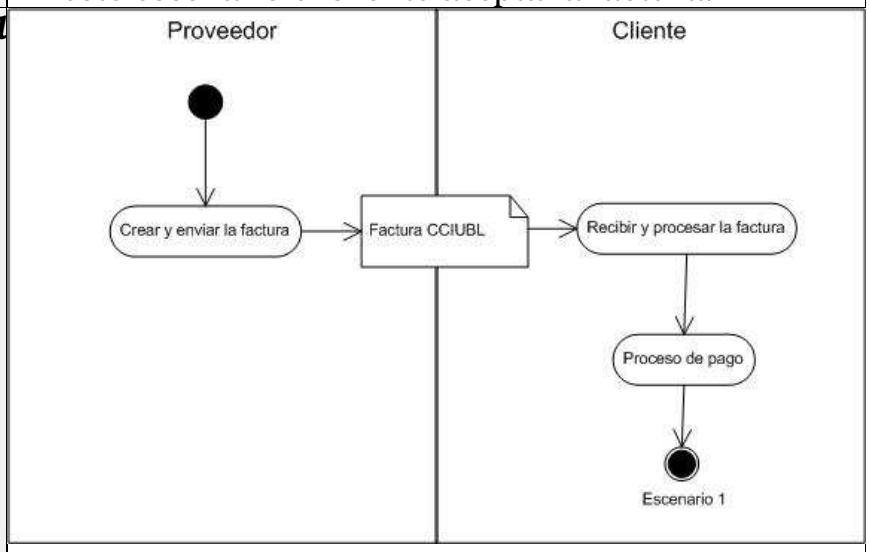
Figura 4: Diagrama de actividad Perfil SoloFac-1.0

Descripción del diagrama de actividad

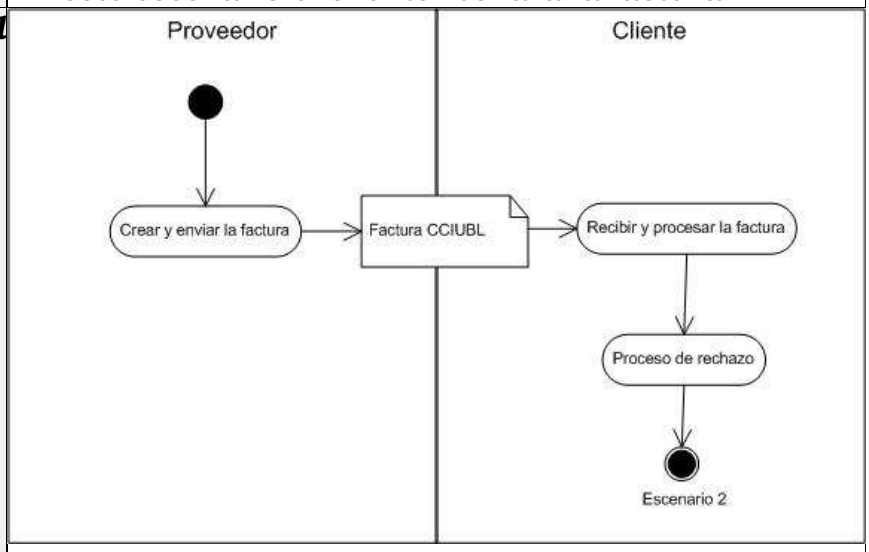
Rol	***Actividad***	***Descripción***
Proveedor	Crear y enviar la factura	El proveedor crea y envía la factura al cliente
Cliente	Recibir y procesar la factura	El cliente recibe y procesa la factura del proveedor
Cliente	Aceptar o rechazar la	El cliente acepta o rechaza toda la

	factura	factura. Si la acepta se inicia el proceso de pago. Si la rechaza debe iniciar un proceso externo de gestión del rechazo. La notificación de aceptación o rechazo de la factura es externa al perfil.

Escenario ***1***

Nombre	Factura aceptada
Descripción	En este escenario el cliente acepta la factura.
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El cliente acepta la factura electrónica
Postcondiciones	De acuerdo con los términos y condiciones acordadas en la factura, el Cliente procede a pago de la misma.

Escenario 2

Nombre	Factura rechazada
Descripción	En este escenario el cliente rechaza la factura
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El cliente rechaza toda la factura. La notificación y resolución se gestionan de forma externa a este perfil
Postcondiciones	El proveedor será informado externamente del rechazo de la factura.

2.3.2 Perfil Facturación Básica

Especificación Perfil FacBas-1.0

Identificador	FacBas-1.0
Contexto	Este perfil aplica a • cualquier emisor o receptor de facturas de cualquier tamaño • de cualquier sector, privado o público • para cualquier tipo de bien, servicio o suministro
Descripción	Este segundo perfil describe un proceso más complejo en el que se pueden corregir de forma sistémica facturas electrónicas erróneas. Como en el perfil anterior, no se prevé en este perfil la integración dentro de la cadena logística pero si que se pueden resolver los posibles errores en la factura original mediante la emisión de una factura rectificativa o una factura complementaria. Tanto el documento factura original como la factura rectificativa contienen todos los elementos necesarios para constituir documentos legales de acuerdo con la legislación establecida. La notificación de la aceptación o rechazo del documento factura electrónica se debe realizar de forma manual y externa por lo tanto a este perfil, sin embargo la resolución se realiza de forma sistémica. Como en el perfil anterior, tampoco es objetivo de este perfil facilitar tareas de conciliación de facturas con albaranes o pedidos, o de discriminación automatizada de facturas por centros de coste, por lo que no será precisa información normalizada o identificadores alineados entre las partes. Aplican las características del perfil anterior en cuanto a integración con ERPs y establecimiento de sistemas de pago avanzados.
Documentos	Factura CCI UBL (normal, complementaria) Factura Rectificativa CCI UBL (por definir)
Escenarios	• Factura aceptada • Factura rechazada por importe menor • Factura rechazada por error o por importe mayor
Requerimientos	1. El Proveedor envía una factura electrónica que puede ser recibida y procesada por el Cliente 2. La factura electrónica debe cumplir con los requisitos legales para las facturas electrónicas en España, tanto a nivel de contenido como a nivel de firma electrónica. 3. El contenido de la factura electrónica debe permitir al sistema de información del Cliente su enrutado a la persona o departamento responsable.

	4. Tanto los documentos Factura como los documentos Factura Rectificativa tienen la información estructurada de forma que se pueden validar impuestos, importes, descuentos y sistemas de pago. 5. Las Facturas Rectificativas deben referirse a la Factura original que corrigen.
Beneficios	Para el Proveedor • transferencia de facturas más rápida • acuse de recibo técnico de facturas emitidas • potencial para reducir el circuito factura-pago Para el Cliente • menor intervención manual y reducción de errores de tecleado • validación de facturas automático (importes, impuestos, sistemas de pago…) • potencial para ejercer un control sistémico de las facturas recibidas • menor necesidad de personal para la gestión de facturas
Actores	Proveedor • Quien exige el pago y se responsabiliza de la resolución de cualquier problema a nivel de facturación. Es el responsable de la emisión de la factura. • También es el responsable de enviar la Factura complementaria o Rectificativa. Cliente • Quien recibe la factura y/o factura rectificativa y es responsable de su gestión.
Reglas de negocio	A nivel de documento • una factura representa la reclamación de un pago por la entrega de unos bienes o provisión de un servicio durante un período de tiempo • una factura rectificativa debe hacer referencia a una factura original • una factura rectificativa siempre sustituye a una factura original • únicamente se puede hacer referencia a un contrato a nivel de cabecera del documento • si aparecen, los medios y forma de pago, así como la información contable deben hacer referencia al conjunto de las líneas de la factura o factura rectificativa • la información de impuestos de la cabecera del documento debe aplicar a todas las líneas de la factura o de la factura rectificativa. • los anticipos deben hacer referencia a toda la factura. • la aceptación o rechazo de una factura no puede ser parcial

	• la factura debe contener una firma electrónica A nivel de línea de factura • cada línea de factura debe especificar un elemento ya sea codificado o descrito literalmente • la información de impuestos a nivel de línea se puede incluír a título informativo. Excepciones • La notificación de rechazo de las facturas se deben gestionar fuera de este perfil, aunque su resolución si que está contemplada en el mismo. Precondiciones • El Cliente debe haber autorizado la recepción de facturas electrónicas con este perfil. • Se debe haber producido una relación contractual entre las partes, en forma de pedido o contrato.

Diagrama de actividad

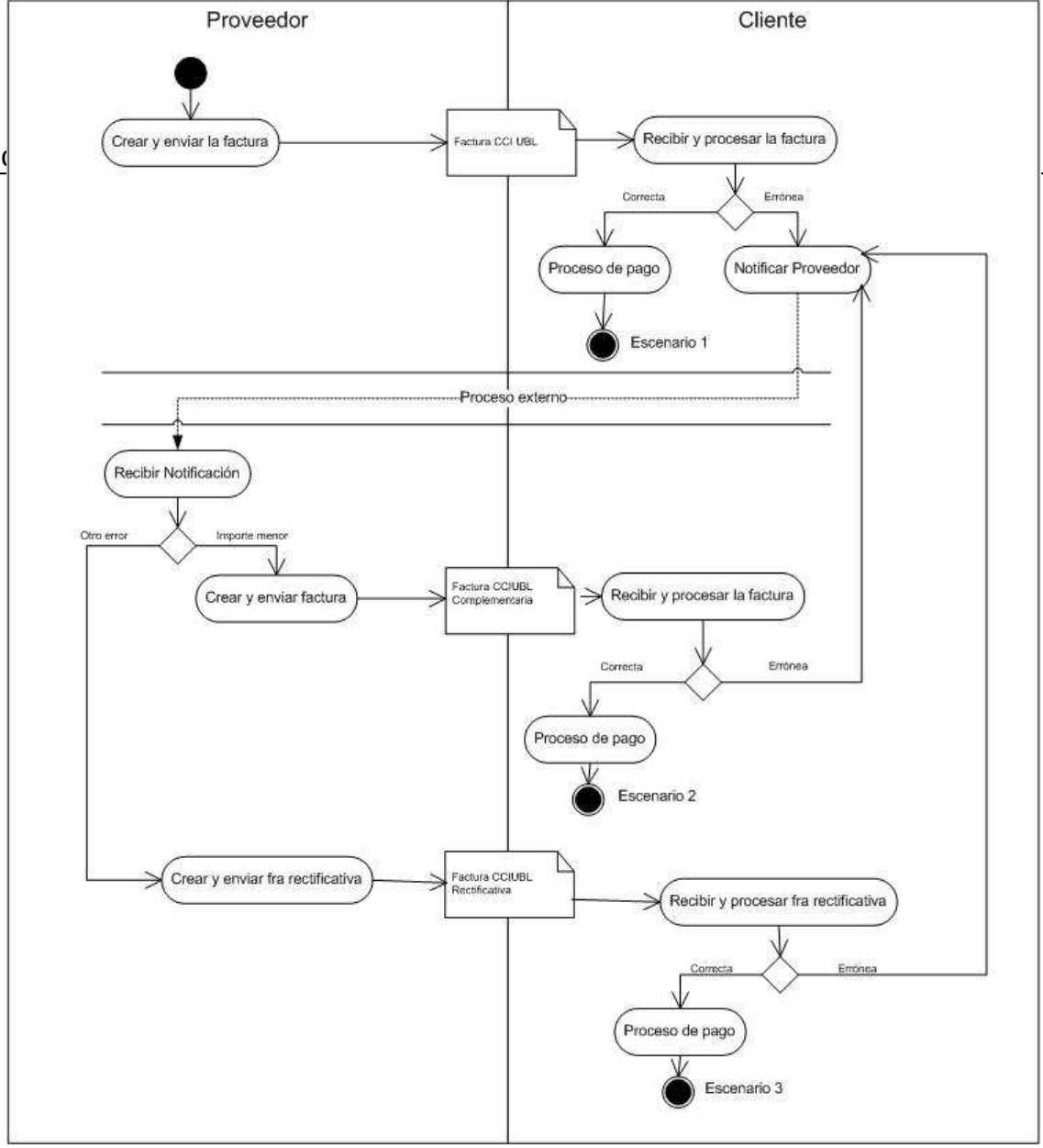


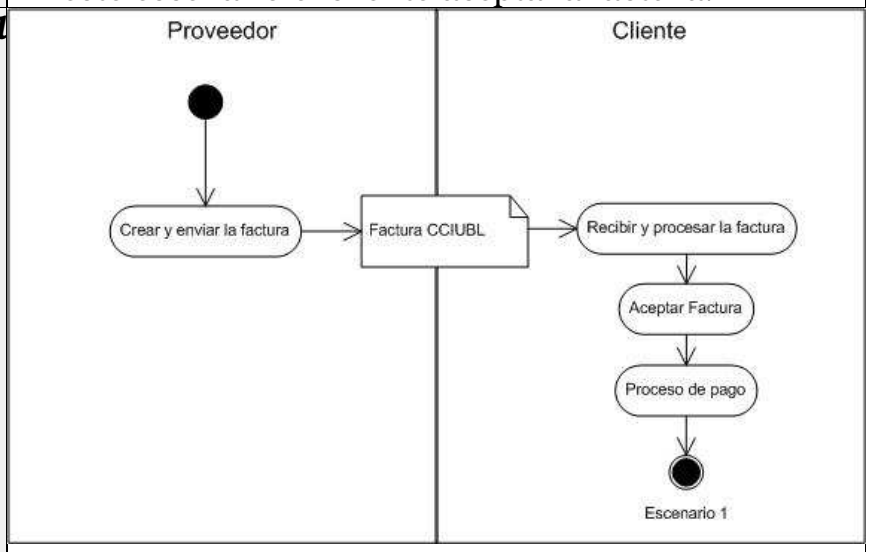
Figura 5: Diagrama de actividad Perfil FacBas-1.0

Descripción del diagrama de actividad

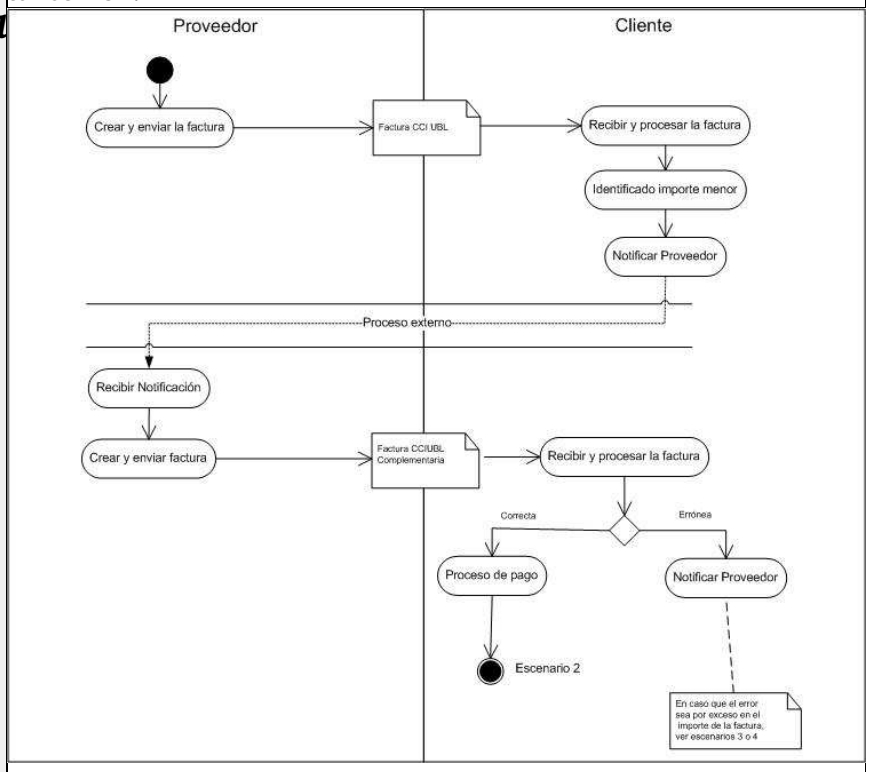
Rol	***Actividad***	***Descripción***
Proveedor	Crear y enviar la factura	El proveedor crea y envia la factura al cliente
Cliente	Recibir y procesar la factura	El cliente recibe y procesa la factura del proveedor
Cliente	Aceptar o rechazar la factura	El cliente acepta o rechaza toda la factura. Si la acepta se inicia el proceso de pago. Si la rechaza debe iniciar un proceso externo de gestión del rechazo. La notificación de aceptación o rechazo de la factura es externa al

		perfil.
Cliente	Acepta la factura y procede al pago	Al aceptar la factura, el Cliente procede al pago mediante un proceso externo a este perfil.
Cliente	Rechaza la factura y notifica al Proveedor	Al rechazar la factura el Cliente notifica al Proveedor mediante mecanismos externos a este perfil.
Proveedor	Recibe la Notificación externa	El Proveedor recibe la notificación de forma manual y procede a su resolución.
Proveedor	Crear y enviar Factura Complementaria	Si el error es debido a que el importe en la Factura Original era menor al esperado, el Proveedor emite una Factura Complementaria con el importe restante. La Factura Complementaria es de tipo Comercial, igual que la Factura Original.
Proveedor	Crear y enviar Factura Rectificativa	Si el error es debido a que el importe en la Factura Original era mayor que el pactado, o existen errores de otro tipo, se procede a la emisión de una Factura Rectificativa.
Cliente	Recibe y procesa la Factura Complementaria	El Cliente recibe y procesa la Factura Complementaria, y si el importe es correcto, se inicia el pago de forma externa.
Cliente	Recibe y procesa la Factura Rectificativa	El Cliente recibe y procesa la Factura Rectificativa, si el importe es ahora correcto, se inicia el pago de forma externa.

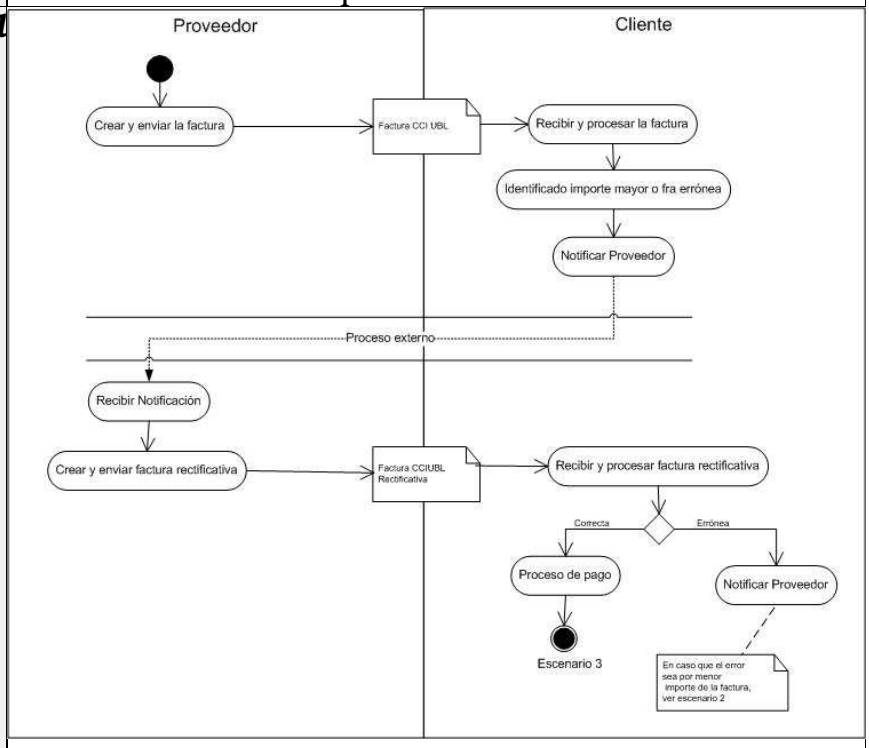
Escenario 1

Nombre	Factura aceptada
Descripción	En este escenario el cliente acepta la factura.
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El cliente acepta la factura electrónica
Postcondiciones	De acuerdo con los términos y condiciones acordadas en la factura, el Cliente procede a pago de la misma.

Escenario 2

Nombre	Factura rechazada por importe menor
Descripción	En este escenario el cliente rechaza la factura porque el importe de la misma es menor del que se estableció en el pedido o contrato, externos a este perfil. La notificación del rechazo se establece por medios externos a este perfil. El Proveedor recibe la notificación de rechazo y procede, en caso de acuerdo en el error, a emitir una factura que complemente la anterior.
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El Cliente rechaza toda la factura. El Cliente notifica de forma externa al Proveedor. El Proveedor si está de acuerdo envía una Factura Complementaria con el importe adicional al Cliente El Cliente recibe y procesa la Factura Complementaria, y si el importe es correcto inicia el proceso de pago. Si es incorrecto, el Cliente notifica al Proveedor de forma externa. El Proveedor puede volver a emitir otro documento según sea el error.
Postcondiciones	Los importes debidos se resuelven y el pago queda cerrado o no se resuelven los errores y se vuelve a notificar al Proveedor.

Escenario 3

Nombre	Factura rechazada por error y emisión de Factura Rectificativa
Descripción	En este escenario el Cliente rechaza la factura ya que su importe es mayor que el pactado en un proceso de pedido o contratación previo y externo a este perfil o por cualquier otro tipo de error en la información de la factura original. La notificación del rechazo se realiza por mecanismos manuales externos a este proceso y el Proveedor, al recibir la notificación de rechazo procede a emitir una Factura Rectificativa para subsanar el error.
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El cliente recibe y rechaza la factura electrónica por medios externos a este perfil El Proveedor recibe la notificación de rechazo debido a un sobrecargo en la factura. El Proveedor crea y envía una Factura Rectificativa donde establece toda la información corregida. El Cliente recibe la Factura Rectificativa, y si está de acuerdo con la misma procede al pago. Si no está de acuerdo volvería a un escenario de resolución de errores dependiendo del tipo de error.
Postcondiciones	Los importes debidos se resuelven y el pago queda cerrado o no se resuelven los errores y se vuelve a notificar al Proveedor.

2.3.3 Perfil Facturación Básica con Respuesta

Especificación Perfil FacBasR-1.0

Identificador	FacBasR-1.0
Contexto	Este perfil aplica a • cualquier emisor o receptor de facturas de cualquier tamaño • de cualquier sector, privado o público • para cualquier tipo de bien, servicio o suministro
Descripción	El tercer perfil describe un proceso de relación sistémica entre las partes, en el que se puede notificar y corregir de forma sistémica facturas electrónicas erróneas. Es un perfil muy similar al anterior pero añade el proceso de notificación para evitar transacciones manuales en todo el proceso de Facturación. Tanto el documento factura electrónica como la factura rectificativa contienen todos los elementos necesarios para constituir documentos legales de acuerdo con la legislación establecida. La notificación de la aceptación o rechazo del documento factura electrónica se realiza de forma sistémica mediante el documento Respuesta de Aplicación. Aplican las características del perfil anterior en cuanto a integración con ERPs y establecimiento de sistemas de pago avanzados.
Documentos	Factura CCI UBL (InvoiceTypeCode: comercial) Factura Rectificativa CCI UBL (InvoiceTypeCode: rectificativa) Respuesta de Aplicación (ver Apartado Nuevos Documentos)
Escenarios	• Factura aceptada • Factura rechazada por importe menor • Factura rechazada por error y emisión de factura rectificativa
Requerimientos	1. El Proveedor envía una factura electrónica que puede ser recibida y procesada por el Cliente 2. La factura electrónica debe cumplir con los requisitos legales para las facturas electrónicas en España, tanto a nivel de contenido como a nivel de firma electrónica. 3. El contenido de la factura electrónica debe permitir al sistema de información del Cliente su enrutado a la persona o departamento responsable. 4. Tanto los documentos Factura como los documentos Factura Rectificativa tienen la información estructurada de forma que se pueden validar impuestos, importes, descuentos y sistemas de pago. 5. Las Facturas Rectificativas deben referirse a la Factura original que corrigen.

	6. Las Facturas Rectificativas sustituyen a una y solo una Factura Original. 7. La Respuesta de Aplicación contiene información del error encontrado y referencia a la Factura que corrige.
Beneficios	Para el Proveedor • transferencia de facturas más rápida • acuse de recibo técnico de facturas emitidas • potencial para reducir el circuito factura-pago Para el Cliente • menor intervención manual y reducción de errores de tecleado • validación de facturas automática (importes, impuestos, sistemas de pago…) • potencial para ejercer un control sistémico de las facturas recibidas • menor necesidad de personal para la gestión de facturas
Actores	Proveedor • Quien exige el pago y se responsabiliza de la resolución de cualquier problema a nivel de facturación. Es el responsable de la emisión de la factura. • También es el responsable de enviar la Factura Complementaria o Rectificativa. Cliente • Quien recibe la factura y/o factura rectificativa y es responsable de su gestión. • Es responsable de emitir la notificación de rechazo en caso de error en la Factura Original.
Reglas de negocio	A nivel de documento • una factura representa la reclamación de un pago por la entrega de unos bienes o provisión de un servicio durante un período de tiempo • una factura rectificativa debe hacer referencia a una factura original • una factura rectificativa sustituye a una factura original. • únicamente se puede hacer referencia a un contrato a nivel de cabecera del documento • los medios y forma de pago, así como la información contable deben hacer referencia al conjunto de las líneas de la factura • la información de impuestos de la cabecera del documento debe aplicar a todas las líneas de la factura o de la factura rectificativa. • los anticipos deben hacer referencia a toda la factura. • la aceptación o rechazo de una factura no puede ser parcial

	• la factura debe contener una firma electrónica A nivel de línea de factura • cada línea de factura debe especificar un elemento ya sea codificado o descrito literalmente • la información de impuestos a nivel de línea se puede incluír a título informativo. Precondiciones • El Cliente debe haber autorizado la recepción de facturas electrónicas con este perfil. • Se debe haber producido una relación contractual entre las partes, en forma de pedido o contrato. • El Proveedor acepta la recepción de las Respuestas de Aplicación que se utilizan en este perfil

Diagrama de actividad

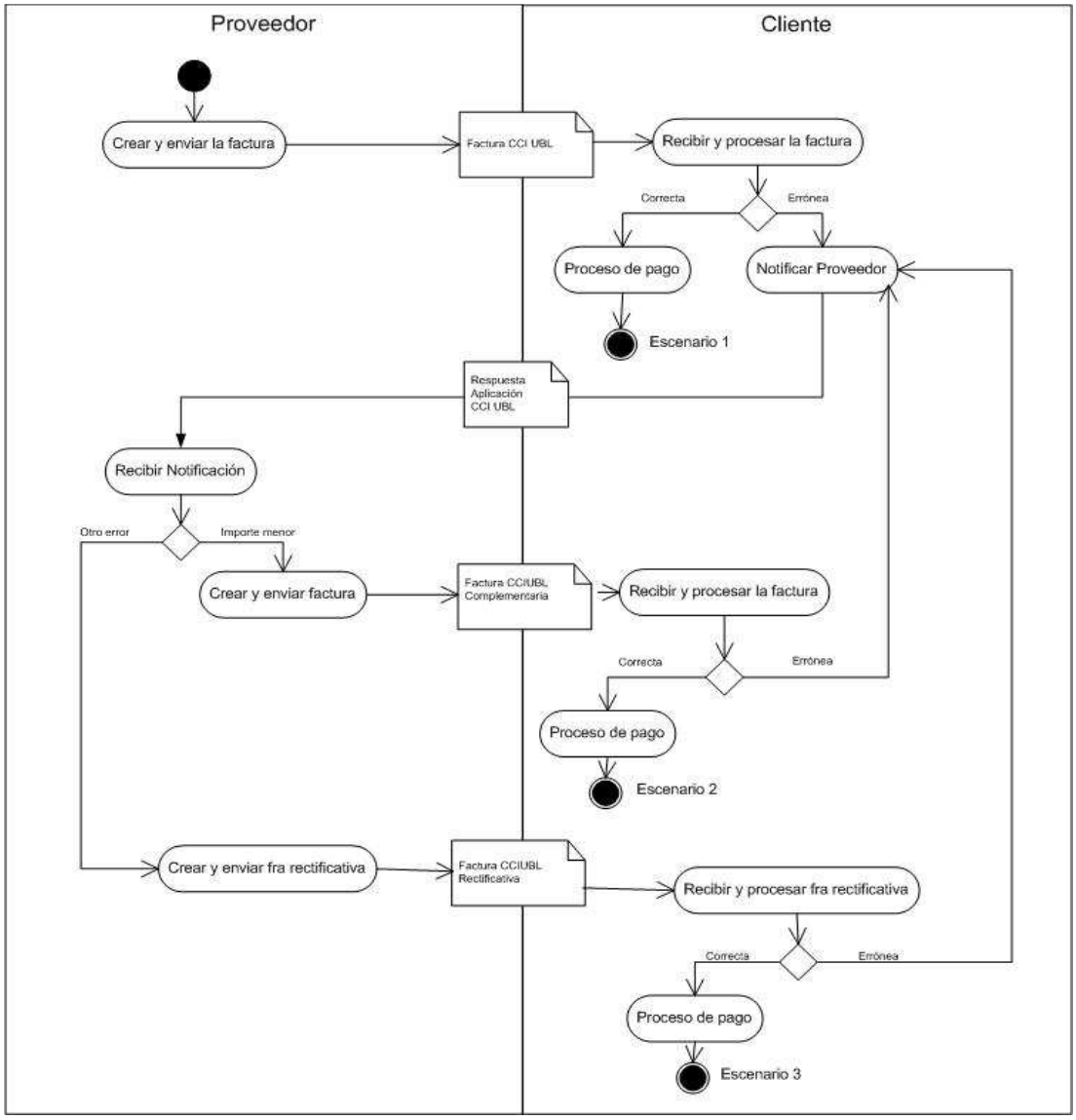


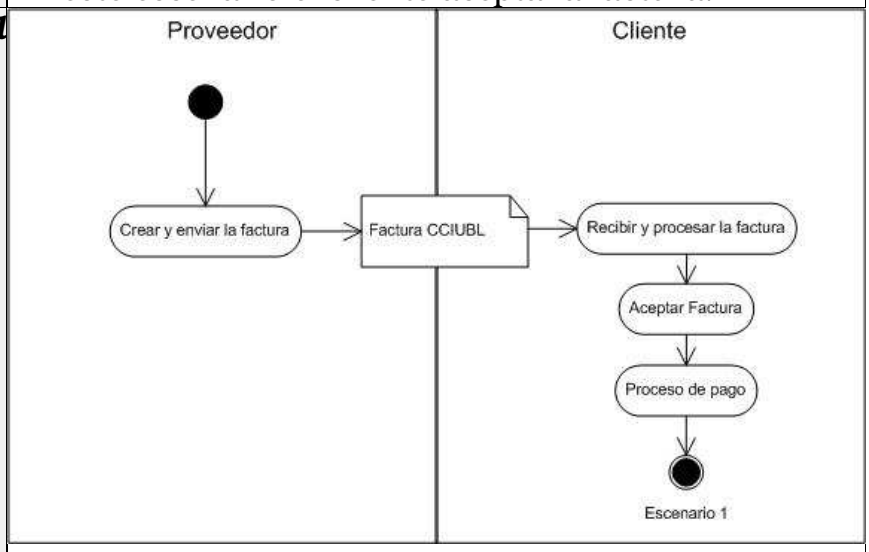
Figura 6: Diagrama de actividad Perfil FacBas-R1.0

Descripción del diagrama de actividad

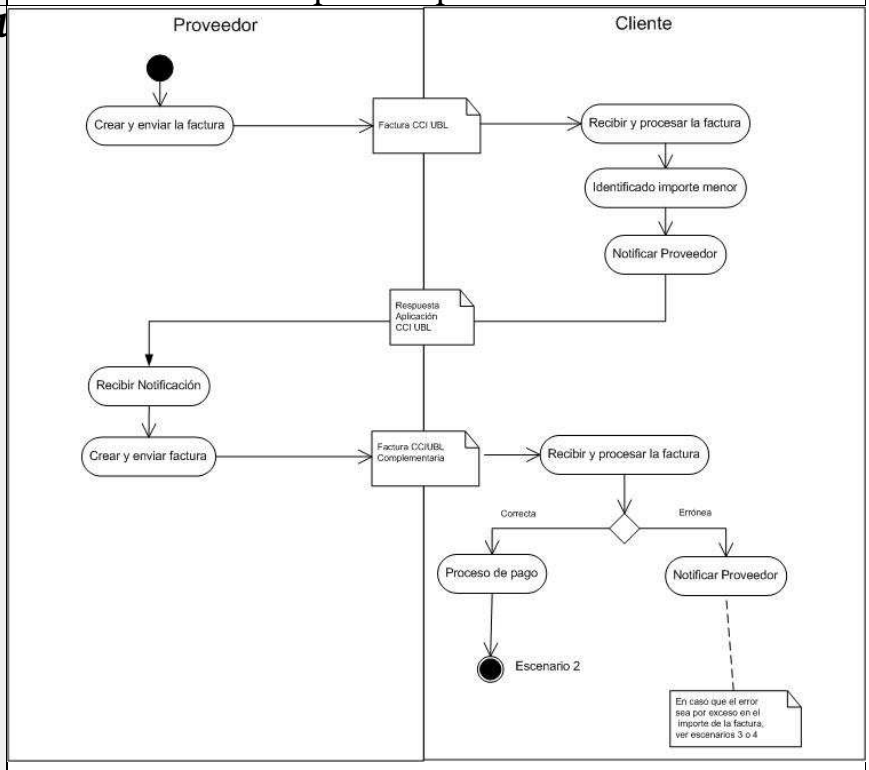
Rol	***Actividad***	***Descripción***
Proveedor	Crear y enviar la factura	El proveedor crea y envia la factura al cliente
Cliente	Recibir y procesar la factura	El cliente recibe y procesa la factura del proveedor
Cliente	Aceptar o rechazar la factura	El cliente acepta o rechaza toda la factura. Si la acepta se inicia el proceso de pago.

		Si la rechaza debe emitir una notificación de rechazo mediante una Respuesta de Aplicación.
Cliente	Acepta la factura y procede al pago	Al aceptar la factura, el Cliente procede al pago mediante un proceso externo a este perfil.
Cliente	Rechaza la factura y notifica al Proveedor	Al rechazar la factura el Cliente notifica al Proveedor mediante la creación y envio de una Respuesta de Aplicación que contiene la referencia a la Factura Original y el motivo del rechazo.
Proveedor	Recibe y procesa la Respuesta de Aplicación	El Proveedor recibe y procesa la Respuesta de Aplicación.
Proveedor	Crear y enviar Factura Complementaria	Si el error es debido a que el importe en la Factura Original era menor, el Proveedor emite una Factura Complementaria con el importe restante.
Proveedor	Crear y enviar Factura Rectificativa	Si el error es debido a que el importe en la Factura Original era mayor que el pactado, o existe algún otro tipo de error, se procede a la emisión de una Factura Rectificativa.
Cliente	Recibe y procesa la Factura Complementaria	El Cliente recibe y procesa la Factura Complementaria, y si el importe es correcto, se inicia el pago de forma externa.
Cliente	Recibe y procesa la Factura Rectificativa	El Cliente recibe y procesa la Factura Rectificativa, si el importe es ahora correcto, se inicia el pago de forma externa.

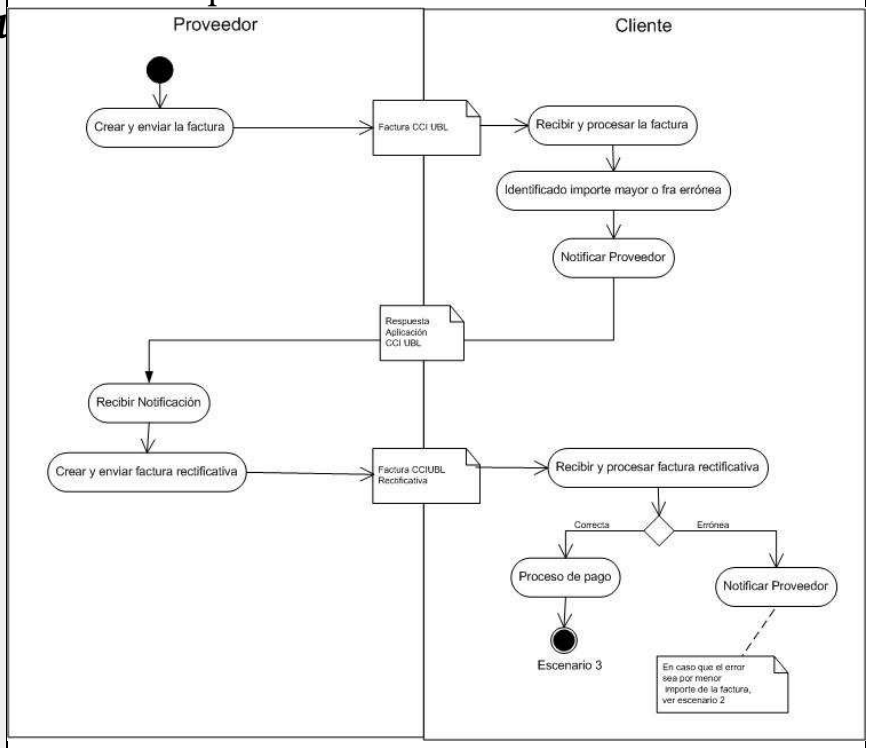
Escenario 1

Nombre	Factura aceptada
Descripción	En este escenario el cliente acepta la factura.
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El cliente acepta la factura electrónica
Postcondiciones	De acuerdo con los términos y condiciones acordadas en la factura, el Cliente procede a pago de la misma.

Escenario 2

Nombre	Factura rechazada por importe menor
Descripción	En este escenario el cliente rechaza la factura porque el importe de la misma es menor del que se estableció en el pedido o contrato, externos a este perfil. La notificación del rechazo se establece mediante el envío de una Respuesta de Aplicación. El Proveedor recibe y procesa la Resupuesta de Aplicación y procede, en caso de acuerdo en el error, a emitir una factura que complemente la anterior.
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El Cliente rechaza toda la factura. El Cliente crea y envía una Respuesta de Aplicación. El Proveedor si está de acuerdo envía una Factura Complementaria con el importe adicional al Cliente El Cliente recibe y procesa la Factura Complementaria, y si el importe es correcto inicia el proceso de pago. Si es incorrecto, el Cliente notifica al Proveedor de forma externa. El Proveedor puede volver a emitir otro documento según sea el error.
Postcondiciones	Los importes debidos se resuelven y el pago queda cerrado o no se resuelven los errores y se vuelve a notificar al Proveedor.

Escenario ***3***

Nombre	Factura rechazada por error y emisión de Factura Rectificativa
Descripción	En este escenario el Cliente rechaza la factura ya que su importe es mayor que el pactado en un proceso de pedido o contratación previo y externo a este perfil o bien porque existe algún otro error en la Factura Original. La notificación del rechazo se realiza mediante la creación y envío de una Respuesta de Aplicación y el Proveedor, al recibirla procede a emitir una Factura Rectificativa para subsanar el error.
Diagrama de actividad del escenario	
Reglas de negocio en tiempo de ejecución	El cliente recibe y rechaza la factura electrónica mediante la creación y envío de una Respuesta de Aplicación. El Proveedor recibe y procesa la Respuesta de Aplicación debido a un sobrecargo en la factura u otro error. El Proveedor crea y envía una Factura Rectificativa. El Cliente recibe la Factura Rectificativa, y si está de acuerdo con la misma procede al pago. Si no está de acuerdo volvería a un escenario de resolución de errores dependiendo del tipo de error.
Postcondiciones	Los importes debidos se resuelven y el pago queda cerrado o no se resuelven los errores y se vuelve a notificar al Proveedor.

2.4 Modelos

Tal como se ha indicado anteriormente, los modelos consisten en particularizaciones o ejemplos de uso de la factura electrónica. A nivel de programación, se podrá utilizar la librería completa para generar facturas completas de cualquier tipo, siempre de acuerdo con las restricciones de la Especificación de la Factura CCI UBL, pero existirán mecanismos de ayuda para la generación de facturas específicas con el objeto de facilitar la labor de desarrollo.

Los modelos se han utilizado a nivel de librerías para definir una capa superior que permita generar facturas sin tener que bajar a nivel de los componentes.

Se han definido los siguientes modelos de factura:

- Factura Mínima
- Factura Recapitulativa
- Factura Rectificativa
- Información de pago

2.4.1 Factura Mínima

El modelo de Factura Mínima es el que contiene los datos mínimos obligatorios legalmente para que el documento se pueda considerar una factura, pero que no se tiene que integrar en ningún circuito de aprovisionamiento, y por tanto carece de sistemas de códigos o identificadores alineados entre cliente y proveedor. Ni tan siquiera se incluyen en la factura mínima los mecanismos de pago o información extensa de producto que se podrán incorporar mediante decoradores.

Estos datos son:

- Núm. Factura
  - cbc:ID
- Fecha expedición
  - cbc:IssueDate
- Razón Social emisor y receptor
  - cac:AccountingSupplierParty
  - cac:AccountingCustomerParty
- NIF emisor y receptor
  - cac:AccountingCustomerParty/cac:Party/cac:PartyIdentification/cbc:ID
  - cac:AccountingCustomerParty/cac:Party/cac:PartyIdentification/cbc:ID@schemeDataURI

- cac:AccountingCustomerParty/cac:Party/cac:PartyIdentification/cbc:ID@schemeName
- cac:AccountingSupplierParty/cac:Party/cac:PartyIdentification/cbc:ID
- cac:AccountingSupplierParty/cac:Party/cac:PartyIdentification/cbc:ID@schemeDataURI
- cac:AccountingSupplierParty/cac:Party/cac:PartyIdentification/cbc:ID@schemeName

- Domicilio emisor y receptor
  - ….cac:Party/cac:PostalAddress/cbc:CityName
  - ….cac:Party/cac:PostalAddress/cbc:PostalZone
  - ….cac:Party/cac:PostalAddress/cac:AddressLine/cbc:Line
  - ….cac:Party/cac:PostalAddress/cac:Country/cbc:IdentificationCode
  - ….cac:Party/cac:PostalAddress/cac:Country/cbc:IdentificationCode@listSchemeURI
  - ….cac:Party/cac:PostalAddress/cac:Country/cbc:Name
- Descripción de las operaciones
  - cac:InvoiceLine/cbc:ID
  - cac:InvoiceLine/cac:Item/cbc:Name
- Tipo impositivo
  - cac:TaxTotal/cac:TaxSubtotal/cac:TaxCategory/cbc:Percent
  - cac:TaxTotal/cac:TaxSubtotal/cac:TaxCategory/cac:TaxScheme/cbc:TaxTypeCode
  - cac:TaxTotal/cac:TaxSubtotal/cac:TaxCategory/cac:TaxScheme/cbc:TaxTypeCode@listSchemeURI
- Cuota tributaria
  - cac:TaxTotal/cbc:TaxAmount
  - cac:LegalMonetaryTotal/cbc:TaxExclusiveAmount
- Fecha prestación del servicio (si distinta a expedición)
  - cbc:TaxPointDate

Además de estos datos obligatorios, se informan también en la Factura Mínima otros datos requeridos según la personalización CCI UBL como por ejemplo

- Identificador de versión
  - cbc:UBLVersionID

- Identificador de personalización
  - cbc:CustomizationID
- Identificador de perfil
  - cbc:ProfileID
  - cbc:ProfileID@schemeDataURI
- Indicador de copia
  - cbc:CopyIndicator
- Tipo de factura (Comercial, Proforma,Factoring, Rectificativa)
  - cbc:InvoiceTypeCode
  - cbc:InvoiceTypeCode@listURI
  - cbc:InvoiceTypeCode@listSchemeURI
- Moneda del documento
  - cbc:DocumentCurrencyCode
  - cbc:DocumentCurrencyCode@listSchemeURI
- Cantidades y precios para las facturas de bienes o productos
  - cac:InvoiceLine/cbc:InvoicedQuantity
  - cac:InvoiceLine/cac:Price/cbc:PriceAmount

Veamos a continuación una instancia XML con todos los datos necesarios para una Factura Mínima:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited by Invinet Sistemes 2003, S.L. -->
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:qdt="urn:oasis:names:specification:ubl:schema:xsd:QualifiedDatatypes-2"
xmlns:udt="urn:un:unece:uncefact:data:specification:UnqualifiedDataTypesSchemaModule:2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2
http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/maindoc/UBL-Invoice-2.0.xsd">
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>CCIUBL</cbc:CustomizationID>
    <cbc:ProfileID schemeDataURI="urn:fundetec:profile-1.0">SoloFac-1.0</cbc:ProfileID>
    <cbc:ID>001-2007</cbc:ID>
    <cbc:CopyIndicator>false</cbc:CopyIndicator>
    <cbc:IssueDate>2007-05-01</cbc:IssueDate>
    <cbc:InvoiceTypeCode listSchemeURI="urn:fundetec:codelist:InvoiceTypeCode-
2.0">Comercial</cbc:InvoiceTypeCode>
    <cbc:DocumentCurrencyCode
listSchemeURI="urn:un:unece:uncefact:codelist:specification:54217:2001">EUR</cbc:DocumentCurrencyCode>
```

```
<cac:Signature>
    <cbc:ID>uuid-invinet-6378-20070501</cbc:ID>
    <cac:SignatoryParty>
        <cac:PartyIdentification>
            <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0" schemeName="NIF">46228159T</cbc:ID>
        </cac:PartyIdentification>
        <cac:PartyName>
            <cbc:Name>Oriol </cbc:Name>
        </cac:PartyName>
    </cac:SignatoryParty>
    <cac:DigitalSignatureAttachment>
        <cac:ExternalReference>
            <cbc:URI>#123454</cbc:URI>
        </cac:ExternalReference>
    </cac:DigitalSignatureAttachment>
</cac:Signature>
<cac:AccountingSupplierParty>
    <cac:Party>
        <cac:PartyIdentification>
            <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0" schemeName="CIF">B63776174</cbc:ID>
        </cac:PartyIdentification>
        <cac:PartyName>
            <cbc:Name>Invinet Sistemes 2003, S.L.</cbc:Name>
        </cac:PartyName>
        <cac:PostalAddress>
            <cbc:CityName>Barcelona</cbc:CityName>
            <cbc:PostalZone>08013</cbc:PostalZone>
            <cac:AddressLine>
                <cbc:Line>Ribes, 31</cbc:Line>
            </cac:AddressLine>
            <cac:Country>
                <cbc:IdentificationCode listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-2.0">ES</cbc:IdentificationCode>
                <cbc:Name>España</cbc:Name>
            </cac:Country>
        </cac:PostalAddress>
    </cac:Party>
</cac:AccountingSupplierParty>
<cac:AccountingCustomerParty>
    <cac:Party>
        <cac:PartyIdentification>
            <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
```

```
schemeName="NIF">B87227388</cbc:ID>
                    </cac:PartyIdentification>
                    <cac:PartyName>
                        <cbc:Name>Ingent Grup Systems, S.L.</cbc:Name>
                    </cac:PartyName>
                    <cac:PostalAddress>
                        <cbc:CityName>Vilafranca del Penedes</cbc:CityName>
                        <cbc:PostalZone>08720</cbc:PostalZone>
                        <cac:AddressLine>
                            <cbc:Line>Sant Pere, 17</cbc:Line>
                        </cac:AddressLine>
                        <cac:Country>
                            <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
                            <cbc:Name>España</cbc:Name>
                        </cac:Country>
                    </cac:PostalAddress>
                </cac:Party>
            </cac:AccountingCustomerParty>
            <cac:TaxTotal>
                <cbc:TaxAmount currencyID="EUR">160</cbc:TaxAmount>
                <cac:TaxSubtotal>
                    <cbc:TaxableAmount currencyID="EUR">1000</cbc:TaxableAmount>
                    <cbc:TaxAmount currencyID="EUR">160</cbc:TaxAmount>
                    <cac:TaxCategory>
                        <cbc:Percent>16</cbc:Percent>
                        <cac:TaxScheme>
                            <cbc:TaxTypeCode      listSchemeURI="urn:fundetec:codelist:TaxTypeCode-
2.0">IVA</cbc:TaxTypeCode>
                        </cac:TaxScheme>
                    </cac:TaxCategory>
                </cac:TaxSubtotal>
            </cac:TaxTotal>
            <cac:LegalMonetaryTotal>
                <cbc:LineExtensionAmount currencyID="EUR">1000</cbc:LineExtensionAmount>
                <cbc:TaxExclusiveAmount currencyID="EUR">160</cbc:TaxExclusiveAmount>
                <cbc:PayableAmount currencyID="EUR">1160</cbc:PayableAmount>
            </cac:LegalMonetaryTotal>
            <cac:InvoiceLine>
                <cbc:ID>1</cbc:ID>
                <cbc:InvoicedQuantity unitCode="ZZ">2</cbc:InvoicedQuantity>
                <cbc:LineExtensionAmount currencyID="EUR">1000</cbc:LineExtensionAmount>
                <cac:Item>
```

```
                    <cbc:Name>Ordenador</cbc:Name>
                </cac:Item>
                <cac:Price>
                    <cbc:PriceAmount currencyID="EUR">500</cbc:PriceAmount>
                </cac:Price>
            </cac:InvoiceLine>
</Invoice>
```

Ejemplo 1: Factura Mínima[1]

1 No se ha incorporado al ejemplo la firma electrónica correspondiente que podría aparecer como una extensión de UBL o bien como un envoltorio al documento factura CCI UBL.

2.4.2 Factura Recapitulativa

Las facturas recapitulativas incluyen en un solo documento distintas operaciones realizadas en diferentes fechas para un mismo destinatario, siempre durante el mismo mes natural. Se trata por tanto de Facturas comerciales convencionales en las que se debe especificar

- el período de facturación
  - cac:InvoicePeriod/cbc:StartDate
  - cac:InvoicePeriod/cbc:EndDate
- las operaciones realizadas durante el período.

A nivel de código de factura InvoiceTypeCode se utiliza el mismo tipo "Comercial" ya que no existe un requerimiento expreso de diferenciar las facturas recapitulativas de las facturas convencionales.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited by Invinet Sistemes 2003, S.L. -->
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:qdt="urn:oasis:names:specification:ubl:schema:xsd:QualifiedDatatypes-2"
xmlns:udt="urn:un:unece:uncefact:data:specification:UnqualifiedDataTypesSchemaModule:2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2
http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/maindoc/UBL-Invoice-2.0.xsd">
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>CCIUBL</cbc:CustomizationID>
    <cbc:ProfileID schemeDataURI="urn:fundetec:profile-1.0">SoloFac-1.0</cbc:ProfileID>
    <cbc:ID>002-2007</cbc:ID>
    <cbc:CopyIndicator>false</cbc:CopyIndicator>
    <cbc:IssueDate>2007-04-30</cbc:IssueDate>
    <cbc:InvoiceTypeCode listSchemeURI="urn:fundetec:codelist:InvoiceTypeCode-
2.0">Comercial</cbc:InvoiceTypeCode>
    <cbc:DocumentCurrencyCode
listSchemeURI="urn:un:unece:uncefact:codelist:specification:54217:2001">EUR</cbc:DocumentCurrencyCode>
    <cac:InvoicePeriod>
        <cbc:StartDate>2007-04-01</cbc:StartDate>
        <cbc:EndDate>2007-04-30</cbc:EndDate>
    </cac:InvoicePeriod>
    <cac:Signature>
        <cbc:ID>uuid-invinet-98348-20070430</cbc:ID>
        <cac:SignatoryParty>
            <cac:PartyIdentification>
                <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="NIF">46228159T</cbc:ID>
```

```
                    </cac:PartyIdentification>
                    <cac:PartyName>
                        <cbc:Name>Oriol </cbc:Name>
                    </cac:PartyName>
                </cac:SignatoryParty>
                <cac:DigitalSignatureAttachment>
                    <cac:ExternalReference>
                        <cbc:URI>#123454</cbc:URI>
                    </cac:ExternalReference>
                </cac:DigitalSignatureAttachment>
            </cac:Signature>
            <cac:AccountingSupplierParty>
                <cac:Party>
                    <cac:PartyIdentification>
                        <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="CIF">B63776174</cbc:ID>
                    </cac:PartyIdentification>
                    <cac:PartyName>
                        <cbc:Name>Invinet Sistemes 2003, S.L.</cbc:Name>
                    </cac:PartyName>
                    <cac:PostalAddress>
                        <cbc:CityName>Barcelona</cbc:CityName>
                        <cbc:PostalZone>08013</cbc:PostalZone>
                        <cac:AddressLine>
                            <cbc:Line>Ribes, 31</cbc:Line>
                        </cac:AddressLine>
                        <cac:Country>
                            <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
                            <cbc:Name>España</cbc:Name>
                        </cac:Country>
                    </cac:PostalAddress>
                </cac:Party>
            </cac:AccountingSupplierParty>
            <cac:AccountingCustomerParty>
                <cac:Party>
                    <cac:PartyIdentification>
                        <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="NIF">B87227388</cbc:ID>
                    </cac:PartyIdentification>
                    <cac:PartyName>
                        <cbc:Name>Ingent Grup Systems, S.L.</cbc:Name>
                    </cac:PartyName>
                    <cac:PostalAddress>
```

```
                    <cbc:CityName>Vilafranca del Penedes</cbc:CityName>
                    <cbc:PostalZone>08720</cbc:PostalZone>
                    <cac:AddressLine>
                        <cbc:Line>Sant Pere, 17</cbc:Line>
                    </cac:AddressLine>
                    <cac:Country>
                        <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
                        <cbc:Name>España</cbc:Name>
                    </cac:Country>
                </cac:PostalAddress>
            </cac:Party>
        </cac:AccountingCustomerParty>
        <cac:TaxTotal>
            <cbc:TaxAmount currencyID="EUR">160</cbc:TaxAmount>
            <cac:TaxSubtotal>
                <cbc:TaxableAmount currencyID="EUR">1000</cbc:TaxableAmount>
                <cbc:TaxAmount currencyID="EUR">160</cbc:TaxAmount>
                <cac:TaxCategory>
                    <cbc:Percent>16</cbc:Percent>
                    <cac:TaxScheme>
                        <cbc:TaxTypeCode      listSchemeURI="urn:fundetec:codelist:TaxTypeCode-
2.0">IVA</cbc:TaxTypeCode>
                    </cac:TaxScheme>
                </cac:TaxCategory>
            </cac:TaxSubtotal>
        </cac:TaxTotal>
        <cac:LegalMonetaryTotal>
            <cbc:LineExtensionAmount currencyID="EUR">1000</cbc:LineExtensionAmount>
            <cbc:TaxExclusiveAmount currencyID="EUR">160</cbc:TaxExclusiveAmount>
            <cbc:PayableAmount currencyID="EUR">1160</cbc:PayableAmount>
        </cac:LegalMonetaryTotal>
        <cac:InvoiceLine>
            <cbc:ID>1</cbc:ID>
            <cbc:LineExtensionAmount currencyID="EUR">500</cbc:LineExtensionAmount>
            <cac:Item>
                <cbc:Name>Intervención 14 - 4</cbc:Name>
            </cac:Item>
        </cac:InvoiceLine>
        <cac:InvoiceLine>
            <cbc:ID>2</cbc:ID>
            <cbc:LineExtensionAmount currencyID="EUR">500</cbc:LineExtensionAmount>
            <cac:Item>
```

```
                    <cbc:Name>Intervención 22 - 4</cbc:Name>
                </cac:Item>
            </cac:InvoiceLine>
</Invoice>
```

Ejemplo 2: Factura Recapitulativa

En una factura de tipo recapitulativo se pueden facturar diversas operaciones efectuadas durante el mismo mes

Podrán incluirse en una sola factura, todas las operaciones realizadas dentro de un mismo mes natural para un mismo destinatario

2.4.3 Factura Rectificativa

Una Factura Rectificativa se debe expedir de forma obligatoria para corregir una factura anterior que no contenga todas las menciones obligatorias que se exigen en el Reglamento de facturación o cuando las cuotas impositivas del IVA se hubiesen determinado de forma incorrecta, o se hayan producido alguna de las circunstancias de modificación de la base imponible previstas en la Ley. La norma prevé una excepción a esta regla, cuando la rectificación sea debida a la devolución de mercancías o retorno de envases y se produzca un suministro posterior con el mismo tipo impositivo, pudiéndose informar de la rectificación en la nueva factura. En este caso se utilizaría una factura comercial convencional informando en el elemento AllowanceCharge del ajuste.

La Factura Rectificativa debe contener los datos del documento rectificado y la rectificación efectuada, además de la causa de la rectificación. Los datos descriptivos de las operaciones, tipos impositivos y cuotas tributarias serán los que resulten de la rectificación efectuada.

A nivel general, adicionalmente a los datos de la factura, una Factura Rectificativa debe informar de:

- el número de la factura o facturas originales que rectifica o en su caso del período de rectificación
    - cac:BillingReference/cac:InvoiceDocumentReference/cbc:ID
    - cac:BillingReference/cac:InvoiceDocumentReference/cbc:Issue Date
- el motivo por el cual se procede a la emisión de una Factura Rectificativa.
    - cbc:Note

Una Factura Rectificativa puede rectificar diversas Facturas Originales (o

Comerciales, tal como las definimos en el InvoiceTypeCode).

Se debe utilizar un nuevo tipo InvoiceTypeCode para identificar la Factura Rectificativa, concretamente se utiliza el código “Rectificativa”.

Veamos a continuación un ejemplo de factura rectificativa que corrige la factura mínima del primer apartado.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited by Invinet Sistemes 2003, S.L. -->
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:qdt="urn:oasis:names:specification:ubl:schema:xsd:QualifiedDatatypes-2"
xmlns:udt="urn:un:unece:uncefact:data:specification:UnqualifiedDataTypesSchemaModule:2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2
http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/maindoc/UBL-Invoice-2.0.xsd">
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>CCIUBL</cbc:CustomizationID>
    <cbc:ProfileID schemeDataURI="urn:fundetec:profile-1.0">SoloFac-1.0</cbc:ProfileID>
    <cbc:ID>001-2007-R</cbc:ID>
    <cbc:CopyIndicator>false</cbc:CopyIndicator>
    <cbc:IssueDate>2007-06-01</cbc:IssueDate>
    <cbc:InvoiceTypeCode listSchemeURI="urn:fundetec:codelist:InvoiceTypeCode-
2.0">Rectificativa</cbc:InvoiceTypeCode>
    <cbc:Note>Devolución de una silla</cbc:Note>
    <cac:BillingReference>
        <cac:InvoiceDocumentReference>
            <cbc:ID>001-2007</cbc:ID>
            <cbc:IssueDate>2007-05-01</cbc:IssueDate>
        </cac:InvoiceDocumentReference>
    </cac:BillingReference>
    <cac:Signature>
        <cbc:ID>uuid-invinet-73423-20070601</cbc:ID>
        <cac:SignatoryParty>
            <cac:PartyIdentification>
                <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="NIF">46228159T</cbc:ID>
            </cac:PartyIdentification>
            <cac:PartyName>
                <cbc:Name>Oriol </cbc:Name>
            </cac:PartyName>
        </cac:SignatoryParty>
        <cac:DigitalSignatureAttachment>
            <cac:ExternalReference>
                <cbc:URI>#123454</cbc:URI>
```

```
                    </cac:ExternalReference>
                </cac:DigitalSignatureAttachment>
            </cac:Signature>
            <cac:AccountingSupplierParty>
                <cac:Party>
                    <cac:PartyIdentification>
                        <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="CIF">B63776174</cbc:ID>
                    </cac:PartyIdentification>
                    <cac:PartyName>
                        <cbc:Name>Invinet Sistemes 2003, S.L.</cbc:Name>
                    </cac:PartyName>
                    <cac:PostalAddress>
                        <cbc:CityName>Barcelona</cbc:CityName>
                        <cbc:PostalZone>08013</cbc:PostalZone>
                        <cac:AddressLine>
                            <cbc:Line>Ribes, 31</cbc:Line>
                        </cac:AddressLine>
                        <cac:Country>
                            <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
                            <cbc:Name>España</cbc:Name>
                        </cac:Country>
                    </cac:PostalAddress>
                </cac:Party>
            </cac:AccountingSupplierParty>
            <cac:AccountingCustomerParty>
                <cac:Party>
                    <cac:PartyIdentification>
                        <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="NIF">B87227388</cbc:ID>
                    </cac:PartyIdentification>
                    <cac:PartyName>
                        <cbc:Name>Ingent Grup Systems, S.L.</cbc:Name>
                    </cac:PartyName>
                    <cac:PostalAddress>
                        <cbc:CityName>Vilafranca del Penedes</cbc:CityName>
                        <cbc:PostalZone>08720</cbc:PostalZone>
                        <cac:AddressLine>
                            <cbc:Line>Sant Pere, 17</cbc:Line>
                        </cac:AddressLine>
                        <cac:Country>
                            <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
```

```
2.0">ES</cbc:IdentificationCode>
                                        <cbc:Name>España</cbc:Name>
                                </cac:Country>
                        </cac:PostalAddress>
                </cac:Party>
        </cac:AccountingCustomerParty>
        <cac:TaxTotal>
                <cbc:TaxAmount currencyID="EUR">80</cbc:TaxAmount>
                <cac:TaxSubtotal>
                        <cbc:TaxableAmount currencyID="EUR">500</cbc:TaxableAmount>
                        <cbc:TaxAmount currencyID="EUR">80</cbc:TaxAmount>
                        <cac:TaxCategory>
                                <cbc:Percent>16</cbc:Percent>
                                <cac:TaxScheme>
                                        <cbc:TaxTypeCode      listSchemeURI="urn:fundetec:codelist:TaxTypeCode-
2.0">IVA</cbc:TaxTypeCode>
                                </cac:TaxScheme>
                        </cac:TaxCategory>
                </cac:TaxSubtotal>
        </cac:TaxTotal>
        <cac:LegalMonetaryTotal>
                <cbc:LineExtensionAmount currencyID="EUR">500</cbc:LineExtensionAmount>
                <cbc:TaxExclusiveAmount currencyID="EUR">80</cbc:TaxExclusiveAmount>
                <cbc:PayableAmount currencyID="EUR">580</cbc:PayableAmount>
        </cac:LegalMonetaryTotal>
        <cac:InvoiceLine>
                <cbc:ID>1</cbc:ID>
                <cbc:InvoicedQuantity unitCode="ZZ">1</cbc:InvoicedQuantity>
                <cbc:LineExtensionAmount currencyID="EUR">500</cbc:LineExtensionAmount>
                <cac:Item>
                        <cbc:Name>Ordenador</cbc:Name>
                </cac:Item>
                <cac:Price>
                        <cbc:PriceAmount currencyID="EUR">500</cbc:PriceAmount>
                </cac:Price>
        </cac:InvoiceLine>
</Invoice>
```

Ejemplo 3: Factura Rectificativa[2]

2 No se ha incorporado al ejemplo la firma electrónica correspondiente que podría aparecer como una extensión de UBL o bien como un envoltorio al documento factura CCI UBL.

Las particularidades de las facturas rectificativas se recogen en los siguientes elementos de datos:

Nombre UBL	Nombre CCI	Uso	Observaciones
ID	Número de factura	1	El identificador único de la factura se forma mediante el número de factura y el número de serie. Para las facturas rectificativas se debe informar una serie diferenciada de la serie para las facturas originales.
InvoicePeriod	Período de facturación	0..1	Se debe utilizar en caso de rectificarse un período.
Note	Descripción	1..n	Descripción textual de la discrepancia.
BillingReference	Referencia a factura	1..n	Se debe informar mediante este ABIE el número o números de factura que se rectifican.
BillingReference/InvoiceDocumentReference/ID	Número de factura	1	Número de factura que se rectifica.
BillingReference/InvoiceDocumentReference/IssueDate	Fecha de factura	1	Fecha de factura que se rectifica.

El resto de elementos de la factura corresponderán a los datos rectificados y deben contener toda la información requerida a las facturas originales.

2.4.4 Información de Pago

Las facturas electrónicas pueden incluir información acerca de la forma de pago de la misma por parte del cliente. En las facturas electrónicas CCI UBL se puede informar de diversos mecanismos de pago, incluyendo la posibilidad de definir diversos vencimientos y especificar anticipos sobre la factura.

En este apartado veremos qué elementos se requieren para incorporar información de pago sencilla en las facturas, para un único vencimiento y sin anticipos. En caso de requerir mayor detalle para la información de pago, se deberá proceder a través del modelo de clases completo de la factura electrónica CCIUBL.

Veremos cuatro formas de informar los datos para el pago de la factura electrónica, al contado, pago por transferencia, domiciliación bancaria y pago a través de una entidad financiera en modalidad de factoring.

Pago al contado

Para especificar un pago por transferencia, el documento de factura electrónica deberá informar del:

- Identificador de la forma de pago.
  - cac:PaymentMeans/cbc:ID
- El código de forma de pago deberá ser "10"
  - cac:PaymentMeans/cbc:PaymentMeansCode
  - cac:PaymentMeans/cbc:PaymentMeansCode@listSchemeURI
- Fecha de vencimiento.
  - cac:PaymentMeans/cbc:PaymentDueDate

Pago por Transferencia

Para especificar un pago por transferencia, el documento de factura electrónica deberá informar del:

- Identificador de la forma de pago.
  - cac:PaymentMeans/cbc:ID
- El código de forma de pago deberá ser "30"
  - cac:PaymentMeans/cbc:PaymentMeansCode
  - cac:PaymentMeans/cbc:PaymentMeansCode@listSchemeURI
- Fecha de vencimiento.
  - cac:PaymentMeans/cbc:PaymentDueDate
- El número de cuenta del proveedor

- cac:PaymentMeans/cac:PayeeFinancialAccount/cbc:ID

Pago por Domiciliación bancaria

En caso de domiciliación bancaria habrá que informar de los siguientes datos en la factura:

- Identificador de la forma de pago.
  - cac:PaymentMeans/cbc:ID
- El código de forma de pago deberá ser "42"
  - cac:PaymentMeans/cbc:PaymentMeansCode
  - cac:PaymentMeans/cbc:PaymentMeansCode@listSchemeURI
- Fecha de vencimiento.
  - cac:PaymentMeans/cbc:PaymentDueDate
- El número de cuenta del Cliente
  - cac:PaymentMeans/cac:PayerFinancialAccount/cbc:ID

Pago mediante Factoring

En caso de cesión del cobro a un tercero, se debe especificar los datos del Beneficiario del pago así como la información legal correspondiente. En este caso, además, el InvoiceTypeCode especificará Factoring como tipo de factura para permitir validar la presencia de los elementos requeridos en facturas de este tipo. Concretamente, se incluirán a una factura convencional los siguientes elementos:

- Tipo de factura
  - cbc:InvoiceTypeCode
  - cbc:InvoiceTypeCode@listURI
  - cbc:InvoiceTypeCode@listSchemeURI
- Datos del Beneficiario
  - cac:Payee
- Cláusula para la cesión de cobro
  - cbc:Note
- Identificador de la forma de pago.
  - cac:PaymentMeans/cbc:ID
- Fecha de pago de la cesión
  - cac:PaymentMeans/cbc:PaymentDueDate
- Forma de pago de la cesión
  - cac:PaymentMeans/cbc:PaymentMeansCode
  - cac:PaymentMeans/cbc:PaymentMeansCode@listSchemeURI

- Cuenta del beneficiario
  - cac:PaymentMeans/cac:PayeeFinancialAccount/cbc:ID
- Referencia del pago
  - cac:PaymentMeans/cbc:PaymentID

Veamos a continuación un ejemplo de factura con información de domiciliación bancaria.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited by Invinet Sistemes 2003, S.L. -->
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:qdt="urn:oasis:names:specification:ubl:schema:xsd:QualifiedDatatypes-2"
xmlns:udt="urn:un:unece:uncefact:data:specification:UnqualifiedDataTypesSchemaModule:2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2
http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/maindoc/UBL-Invoice-2.0.xsd">
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>CCIUBL</cbc:CustomizationID>
    <cbc:ProfileID schemeDataURI="urn:fundetec:profile-1.0">SoloFac-1.0</cbc:ProfileID>
    <cbc:ID>001-2007</cbc:ID>
    <cbc:CopyIndicator>false</cbc:CopyIndicator>
    <cbc:IssueDate>2007-05-01</cbc:IssueDate>
    <cbc:InvoiceTypeCode listSchemeURI="urn:fundetec:codelist:InvoiceTypeCode-
2.0">Comercial</cbc:InvoiceTypeCode>
    <cbc:DocumentCurrencyCode
listSchemeURI="urn:un:unece:uncefact:codelist:specification:54217:2001">EUR</cbc:DocumentCurrencyCode>
    <cac:Signature>
        <cbc:ID>uuid-invinet-6378-20070501</cbc:ID>
        <cac:SignatoryParty>
            <cac:PartyIdentification>
                <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="NIF">46228159T</cbc:ID>
            </cac:PartyIdentification>
            <cac:PartyName>
                <cbc:Name>Oriol </cbc:Name>
            </cac:PartyName>
        </cac:SignatoryParty>
        <cac:DigitalSignatureAttachment>
            <cac:ExternalReference>
                <cbc:URI>#123454</cbc:URI>
            </cac:ExternalReference>
        </cac:DigitalSignatureAttachment>
```

```
        </cac:Signature>
        <cac:AccountingSupplierParty>
                <cac:Party>
                        <cac:PartyIdentification>
                                <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="CIF">B63776174</cbc:ID>
                        </cac:PartyIdentification>
                        <cac:PartyName>
                                <cbc:Name>Invinet Sistemes 2003, S.L.</cbc:Name>
                        </cac:PartyName>
                        <cac:PostalAddress>
                                <cbc:CityName>Barcelona</cbc:CityName>
                                <cbc:PostalZone>08013</cbc:PostalZone>
                                <cac:AddressLine>
                                        <cbc:Line>Ribes, 31</cbc:Line>
                                </cac:AddressLine>
                                <cac:Country>
                                        <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
                                        <cbc:Name>España</cbc:Name>
                                </cac:Country>
                        </cac:PostalAddress>
                </cac:Party>
        </cac:AccountingSupplierParty>
        <cac:AccountingCustomerParty>
                <cac:Party>
                        <cac:PartyIdentification>
                                <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="NIF">B87227388</cbc:ID>
                        </cac:PartyIdentification>
                        <cac:PartyName>
                                <cbc:Name>Ingent Grup Systems, S.L.</cbc:Name>
                        </cac:PartyName>
                        <cac:PostalAddress>
                                <cbc:CityName>Vilafranca del Penedes</cbc:CityName>
                                <cbc:PostalZone>08720</cbc:PostalZone>
                                <cac:AddressLine>
                                        <cbc:Line>Sant Pere, 17</cbc:Line>
                                </cac:AddressLine>
                                <cac:Country>
                                        <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
```

```
                                        <cbc:Name>España</cbc:Name>
                                    </cac:Country>
                                </cac:PostalAddress>
                            </cac:Party>
                        </cac:AccountingCustomerParty>
                        <cac:PaymentMeans>
                            <cbc:PaymentMeansCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:PaymentMeansCode-
2.0">42</cbc:PaymentMeansCode>
                            <cbc:PaymentDueDate>2007-06-30</cbc:PaymentDueDate>
                            <cac:PayerFinancialAccount>
                                <cbc:ID>2100-0829-00-1234567890</cbc:ID>
                            </cac:PayerFinancialAccount>
                        </cac:PaymentMeans>
                        <cac:TaxTotal>
                            <cbc:TaxAmount currencyID="EUR">160</cbc:TaxAmount>
                            <cac:TaxSubtotal>
                                <cbc:TaxableAmount currencyID="EUR">1000</cbc:TaxableAmount>
                                <cbc:TaxAmount currencyID="EUR">160</cbc:TaxAmount>
                                <cac:TaxCategory>
                                    <cbc:Percent>16</cbc:Percent>
                                    <cac:TaxScheme>
                                        <cbc:TaxTypeCode listSchemeURI="urn:fundetec:codelist:TaxTypeCode-
2.0">IVA</cbc:TaxTypeCode>
                                    </cac:TaxScheme>
                                </cac:TaxCategory>
                            </cac:TaxSubtotal>
                        </cac:TaxTotal>
                        <cac:LegalMonetaryTotal>
                            <cbc:LineExtensionAmount currencyID="EUR">1000</cbc:LineExtensionAmount>
                            <cbc:TaxExclusiveAmount currencyID="EUR">160</cbc:TaxExclusiveAmount>
                            <cbc:PayableAmount currencyID="EUR">1160</cbc:PayableAmount>
                        </cac:LegalMonetaryTotal>
                        <cac:InvoiceLine>
                            <cbc:ID>1</cbc:ID>
                            <cbc:InvoicedQuantity unitCode="ZZ">2</cbc:InvoicedQuantity>
                            <cbc:LineExtensionAmount currencyID="EUR">1000</cbc:LineExtensionAmount>
                            <cac:Item>
                                <cbc:Name>Ordenador</cbc:Name>
                            </cac:Item>
                            <cac:Price>
                                <cbc:PriceAmount currencyID="EUR">500</cbc:PriceAmount>
```

```
                </cac:Price>
            </cac:InvoiceLine>
</Invoice>
```

Ejemplo 4: Factura electrónica con información de domiciliación

2.5 Procesos

La intención de sistematizar o automatizar el envío y recepción de facturas que se deriva de la creación de perfiles y modelos de documentos, requiere asimismo de la definición de determinados procesos que participan en el intercambio. En este apartado se pretenden especificar estos procesos adicionales para permitir que los desarrolladores conozcan las herramientas y facilidades que existen y que se pueden implementar en sus sistemas de gestión con un bajo coste de implementación y por tanto con cierta rapidez.

Analizaremos los procesas de validación de instancias recibidas y los sistemas de generación de respuesta. Este último es básico para la adopción del perfil de facturación básico con respuesta, y constituye un hito en las aplicaciones que lo implementen ya que conlleva la integración B2B bidireccional, pudiendo los sistemas dar respuesta a transacciones recibidas erróneas.

2.5.1 Proceso Sistémico de Validación

Los sistemas de validación se utilizan siempre que se recibe un documento. Se trata de realizar una validación técnica del mismo para verificar que la instancia cumple con las restricciones especificadas tanto a nivel de esquema como de reglas de restricción en Schematron. Con este conjunto de validaciones, el receptor del documento se asegura de la validez sintáctica y semántica del contenido del documento así como de otras reglas de negocio como por ejemplo validación de los cálculos mediante reglas algorítmicas.

El proceso sistémico de validación también debe asegurar que el receptor del documento soporta el perfil del documento recibido. La finalidad de definir perfiles consiste precisamente en la estandarización de los intercambios, de forma que cada pareja emisor/receptor conoce los requisitos de las transacciones. Si un usuario envía un documento con un perfil no soportado por el receptor, éste debería responder con un documento de respuesta ApplicationResponse donde se informara de la incapacidad de procesar este tipo de perfil.

Por tanto, los errores que se pueden desprender del sistema de validación son:

- “Aceptación” del documento, por ser conforme tanto al esquema como a las restricciones y por ser de un perfil soportado
- “Rechazo técnico” por ser un documento inválido sintácticamente o que no cumpla las restricciones semánticas.
- “Perfil inválido” en caso de recibirse un documento de un perfil no soportado por el sistema de gestión del receptor.
- “Error en regla de negocio” si la instancia recibida no cumple

alguna de las reglas de negocio establecidas.

Los sistemas de validación se pueden dividir en tres:

1. Validación sintáctica y semántica contra los esquemas normativos. En el caso de CCIUBL se utilizan siempre los esquemas normativos definidos por UBL 2.0. Esto permite la recepción por parte de sistemas de gestión desarrollados en el ámbito de CCIUBL de instancias generadas en otras áreas geográficas o mediante otras personalizaciones como por ejemplo la del NES. Es decir que una instancia de factura emitida por un país como Dinamarca, pasaria este primer nivel de validación al ser las facturas generadas por el NES un subconjunto de las definidas por UBL 2.0.

2. En segundo lugar, se realizaría una validación contra las restricciones de las facturas CCIUBL mediante la aplicación de una transformación XSLT donde estuvieran contenidas las reglas Schematron requeridas a nivel de la personalización CCIUBL. En este punto se identificarán posibles disfunciones entre instancias de distintas personalizaciones, y se deberán generar respuestas sistémicas especificando los requerimientos no cumplidos por las instancias recibidas.

3. Adicionalmente, se pueden incluir nuevas reglas de negocio tal como se establece en el punto 2, es decir mediante Schematron, donde especificar reglas de negocio particulares de cada organización. La definición y publicación de estas reglas de negocio es competencia de cada organización. Un ejemplo de este tipo de restricciones podria ser el requerimiento de informar del código contable (AccountCostCode) de forma obligatoria en la factura.

2.5.2 Proceso Sistémico de Generación de Respuesta

Este proceso se debe implementar para generar documentos ApplicationResponse de respuesta al documento recibido.

Aunque el documento ApplicationResponse sólo se obliga en el Perfil de Facturación Básico con Respuesta, se podría utilizar en cualquiera de ellos para informar de aceptación o rechazo técnico del documento recibido. Es por tanto recomendable que las aplicaciones de gestión implementen la generación de respuestas de forma genérica.

El documento ApplicationResponse sirve como documento de acuse técnico, que permite al emisor del documento original conocer la aceptación o rechazo técnico del documento, y permitirá también dar información de negocio, conformándose en un documento de Respuesta de Negocio a cualquier documento emitido.

En un apartado más adelante se describe sucintamente el documento

ApplicationResponse. Sería conveniente definir una guía de implementación del documento ApplicationResponse más amplia para evitar malinterpretaciones.

Cuando un sistema recibe un documento de ApplicationResponse se debe chequear el valor del elemento ResponseCode.

Si es un código de Aceptación significa que el emisor del ApplicationResponse ha aceptado el documento que se referencia en la clase DocumentReference. Este tipo de documentos ApplicationResponse positivos se producen en aquellos perfiles donde se requiere la emisión de una ApplicationResponse, como por ejemplo en el perfil de facturación básica con respuesta.

Por el contrario, se puede recibir un ApplicationResponse cuyo ResponseCode sea de rechazo. Se diferencian tres tipos de rechazo del documento original:

- "Rechazo Técnico" para especificar que el documento original no cumple con las restricciones de esquema o las reglas de la personalización.
- "Rechazo de Perfil" para especificar que el emisor no soporta el perfil especificado en el documento original.
- "Rechazo de Negocio" cuando el documento original no cumple con las reglas de negocio del emisor.

Tal como se verá en el apartado de Nuevo Documento, existen diversos elementos dentro del documento ApplicationResponse para especificar los motivos del rechazo.

3 NUEVO DOCUMENTO

La definición de los perfiles y escenarios establecidos en los apartados anteriores obligan a la definición de un nuevo documento que permita que el receptor de una factura pueda responder al emisor para posibilitar la rectificación sistémica de las facturas.

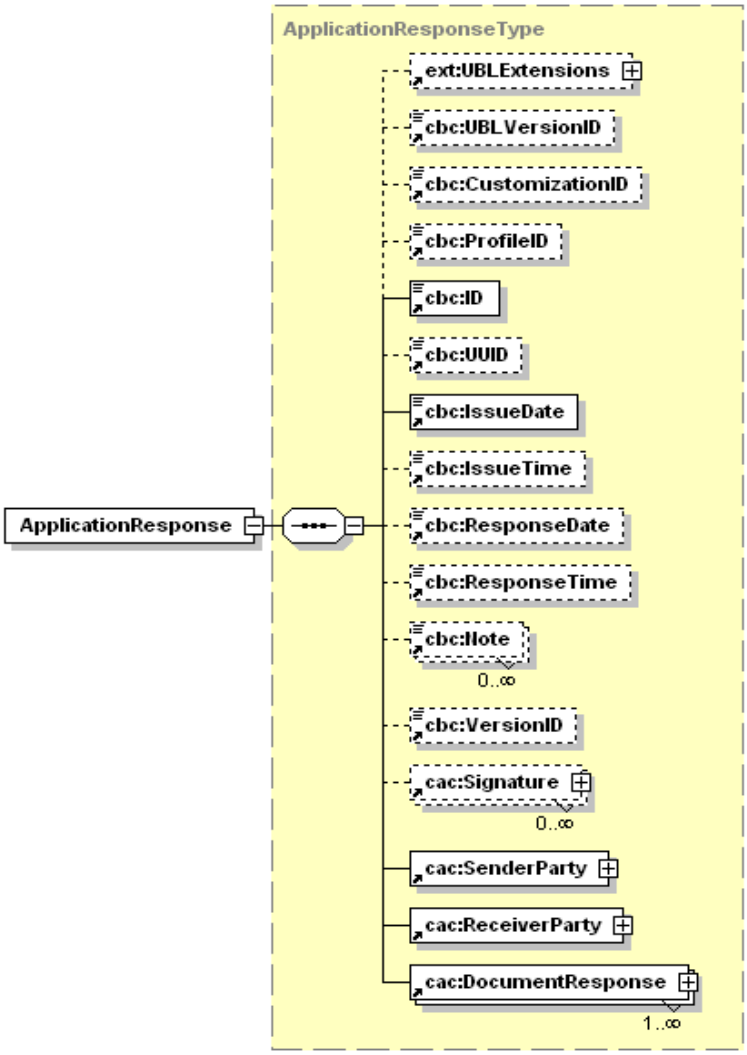
El sistema de información del receptor de facturas electrónicas debe estar preparado para la generación de respuestas de este tipo si quiere acceder al tercer perfil de intercambio que se establece en este documento.

No se definen en este documento otros documentos como los Recordatorios de facturas o las Autofacturas que constituyen nuevos modelos de documento UBL y que por tanto seria necesario modelar o personalizar tal como se ha realizado con el documento factura (Invoice).

3.1 Respuesta de Aplicación (UBL-ApplicationResponse-2.0)

El documento Respuesta de Aplicación de UBL 2.0 es un documento que se utiliza para intercambiar justificantes de recepción y recibos electrónicos para procesar documentos UBL. Se utiliza este documento tanto para la emisión de recibos a nivel de aplicación o regla de negocio como para la remisión de recibos técnicos.

Se intercambian Respuestas de Aplicación entre un Emisor y un Receptor. Aparte de la información sobre el emisor y el receptor y sus respectivos puntos de entrega de los documentos, una Respuesta de Aplicación contiene una referencia al documento original y un código de respuesta.

Diagrama			
Espacio de nombres	urn:oasis:names:specification:ubl:schema:xsd:ApplicationResponse-2		
Nombre	ApplicationResponse	Tipo	cac:ApplicationResponseType
Término CCI	Respuesta de Aplicación	Cardinalidad	

3.1.1 Atributos Simples

En esta sección veremos la lista de atributos de tipo simple más relevantes de la Respuesta de Aplicación UBL. Se trata de un subconjunto de los atributos simples del documento ApplicationResponse UBL 2.0.

Nombre UBL	Nombre CCI	Uso	Observaciones
UBLVersionID	Versión UBL	1	Versión de UBL
CustomizationID	Personalización	1	Perfil de personalización
ProfileID	Perfil	1	Identificador de perfil
ID	ID	1	El identificador único de la Respuesta de Aplicación
IssueDate	Fecha de emisión	1	Fecha de emisión en formato AAAA-MM-DD
ResponseDate	Fecha de respuesta	0..1	Fecha de respuesta en formato AAAA-MM-DD
Note	Nota	0..n	Nota

Atributos simples de la ApplicationResponse UBL

Especificación

Nombre	UBLVersionID	Tipo	Identifier
Término CCI	Versión UBL	Cardinalidad	1
Descripción	Versión de la librería UBL utilizada		
Valor por defecto	2.0		

Nombre	CustomizationID	Tipo	Identifier
Término CCI	Personalización de UBL	Cardinalidad	1
Descripción	Para este documento de Respuesta de Aplicación se establece la personalización CCIUBL		
Valor por defecto	CCIUBL		

Nombre	ProfileID	Tipo	Identifier
Término CCI	Perfil	Cardinalidad	1
Descripción	Se establecerá el perfil de intercambio apropiado.		

Nombre	ID	Tipo	Identifier
Término CCI	Identificador de documento	Cardinalidad	1
Descripción	El identificador de la respuesta de aplicación asignada por el emisor.		

Ejemplo	982349

Nombre	IssueDate	Tipo	Date
Término CCI	Fecha de emisión	Cardinalidad	1
Descripción	Fecha de emisión del documento. El formato es AAAA-MM-DD.		
Ejemplo	2006-11-23		

Nombre	ResponseDate	Tipo	Date
Término CCI	Fecha de respuesta	Cardinalidad	0..1
Descripción	Fecha en la que se responde al documento original en formato AAAA-MM-DD.		

3.1.2 Clases agregadas o componentes

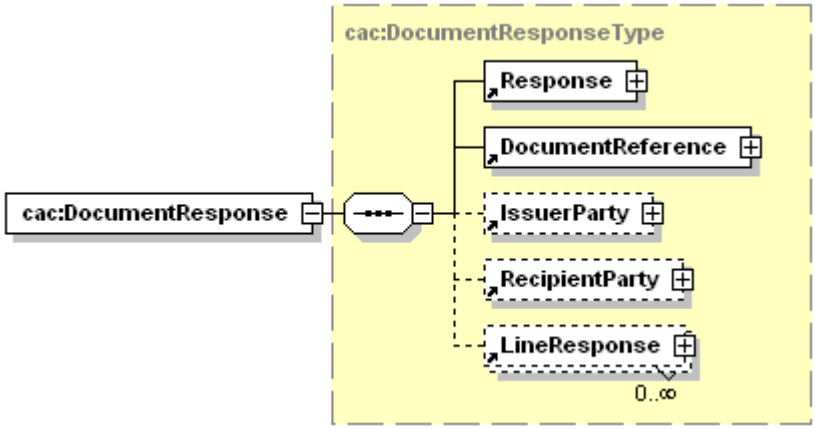
El resto de información del documento ApplicationResponse UBL se estructura en las siguientes clases.

Nombre UBL	Nombre CCI	Uso	Observaciones
UBLExtensions	Extensiones	0..1	Extensiones a la factura electrónica. Ver guía de implementación de la factura CCI UBL.
Signature	Firma	0..n	Datos de firma para no repudio
SenderParty	Emisor	1	Emisor del documento
ReceiverParty	Receptor	1	Receptor del documento
DocumentResponse	Documento de respuesta	1	Información de respuesta

Clases agregadas del documento ApplicationResponse UBL

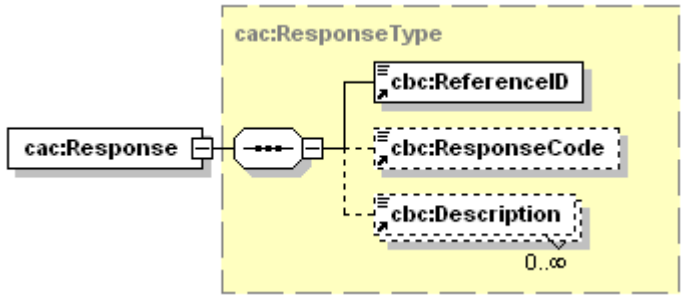
No entraremos en el detalle de las clases UBLExtensions ni Signature que ya están descritas en la guía de implementación de la Factura Electrónica CCI UBL [UBLCCI]. Tampoco describiremos los agregados SenderParty ni ReceiverParty que corresponden a la descripción del Emisor y del Receptor del documento, y que son de tipo PartyType.

Veamos pues el agregado específico del documento DocumentResponse:

Diagrama			
Espacio de nombres	urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2		
Nombre	DocumentResponse	Tipo	cac:DocumentResponseType
Término CCI	Documento Respuesta	Cardinalidad	1..n
Descripción	Conjunto de respuestas al documento original, se puede establecer una respuesta para todo el documento original o realizar respuestas independientes por cada línea del documento original.		

El agregado de DocumentResponse contiene a su vez dos componentes obligatorios, Response y DocumentReference, y tres opcionales. Veamos a continuación el detalle del componente Response. El componente DocumentReference no se detalla en este documento aún siendo obligatorio ya que se definió en la Guía de Implementación de la Factura Electrónica CCI UBL [UBLCCI]. . Únicamente destacar que en este contexto, el componente DocumentReference se utiliza para identificar el documento que se responde.

Tampoco se verán los componentes opcionales entre los que destaca el LineResponse ya que en el perfil FacBasR-1.0 se utiliza la Respuesta de Aplicación para responder a documentos Factura completos, y no se realizan referencias a líneas de Factura.

Diagrama			
Espacio de nombres	urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2		
Nombre	Response	Tipo	cac:ResponseType
Término CCI	Respuesta	Cardinalidad	1
Descripción	Descripción de la respuesta.		

La clase Response tiene los siguientes elementos:

Nombre UBL	Nombre CCI	Uso	Observaciones
ReferenceID	ID Referencia	1	Identificador de la respuesta
ResponseCode	Código de Respuesta	0..1	Código descriptivo de la respuesta
Description	Descripción	0..n	Descripción textual de la respuesta

El elemento ResponseCode representa un código de respuesta definida por la correspondiente lista de códigos Genericode. Se puede detallar la respuesta en diversas repeticiones del elemento Description.

Veamos a continuación un ejemplo de una ApplicationResponse.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited by Invinet Sistemes 2003, S.L.-->
<ApplicationResponse xmlns="urn:oasis:names:specification:ubl:schema:xsd:ApplicationResponse-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:qdt="urn:oasis:names:specification:ubl:schema:xsd:QualifiedDatatypes-2"
xmlns:udt="urn:un:unece:uncefact:data:specification:UnqualifiedDataTypesSchemaModule:2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:ApplicationResponse-2
http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/maindoc/UBL-ApplicationResponse-2.0.xsd">
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>CCIUBL</cbc:CustomizationID>
    <cbc:ProfileID schemeDataURI="urn:fundetec:profile-1.0">FacBasR-1.0</cbc:ProfileID>
    <cbc:ID>874389</cbc:ID>
```

```
        <cbc:IssueDate>2007-05-14</cbc:IssueDate>
        <cac:SenderParty>
                <cac:PartyIdentification>
                        <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="NIF">B87227388</cbc:ID>
                </cac:PartyIdentification>
                <cac:PartyName>
                        <cbc:Name>Ingent Grup Systems, S.L.</cbc:Name>
                </cac:PartyName>
                <cac:PostalAddress>
                        <cbc:CityName>Vilafranca del Penedes</cbc:CityName>
                        <cbc:PostalZone>08720</cbc:PostalZone>
                        <cac:AddressLine>
                                <cbc:Line>Sant Pere, 17</cbc:Line>
                        </cac:AddressLine>
                        <cac:Country>
                                <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
                                <cbc:Name>España</cbc:Name>
                        </cac:Country>
                </cac:PostalAddress>
        </cac:SenderParty>
        <cac:ReceiverParty>
                <cac:PartyIdentification>
                        <cbc:ID schemeDataURI="urn:fundetec:scheme:partyidentification-1.0"
schemeName="CIF">B63776174</cbc:ID>
                </cac:PartyIdentification>
                <cac:PartyName>
                        <cbc:Name>Invinet Sistemes 2003, S.L.</cbc:Name>
                </cac:PartyName>
                <cac:PostalAddress>
                        <cbc:CityName>Barcelona</cbc:CityName>
                        <cbc:PostalZone>08013</cbc:PostalZone>
                        <cac:AddressLine>
                                <cbc:Line>Ribes, 31</cbc:Line>
                        </cac:AddressLine>
                        <cac:Country>
                                <cbc:IdentificationCode
listSchemeURI="urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-
2.0">ES</cbc:IdentificationCode>
                                <cbc:Name>España</cbc:Name>
                        </cac:Country>
                </cac:PostalAddress>
```

```
        </cac:ReceiverParty>
        <cac:DocumentResponse>
                <cac:Response>
                        <cbc:ReferenceID>001-20070514</cbc:ReferenceID>
                        <cbc:ResponseCode                listSchemeURI="urn:fundetec:codelist:responsecode-
1.0">Negocio</cbc:ResponseCode>
                        <cbc:Description>Importe de factura incorrecto</cbc:Description>
                </cac:Response>
                <cac:DocumentReference>
                        <cbc:ID>001-2007</cbc:ID>
                        <cbc:IssueDate>2007-05-01</cbc:IssueDate>
                </cac:DocumentReference>
        </cac:DocumentResponse>
</ApplicationResponse>
```

Ejemplo 5: Respuesta de aplicación

PARTE II
ESPECIFICACIONES

4 ARQUITECTURA DE COMPONENTES DE LIBRERÍAS Y PAQUETES

4.1 Plataforma y lenguajes de programación

Las librerías especificadas en el presente documento deben ser desarrolladas para plataformas Java 1.4 o superior y Microsoft .NET 1.1 o superior. Los lenguajes de programación serán respectivamente Java y C#.

4.2 Organización y paquetes

Los distintos componentes de la librería están distribuidos en distintas capas según su funcionalidad. Las capas inferiores son las que tratan con los datos de más bajo nivel mientras que las superiores ofrecen servicios más a alto nivel. Estas últimas serán las típicamente usadas por un cliente de la librería.

Las capas ordenadas de menor a mayor nivel son las siguientes:

- Tipos básicos. Componentes destinados a tratar elementos básicos XML. Incluyen la gestión de tipos básicos XSD, listas de códigos, tipos de datos *qualified* y tipos *unqualified*.
- Tipos agregados. Componentes destinados a gestionar los tipos CBC, CEC y CAC definidos en CCI UBL. Estos componentes harán uso de los componentes de la capa inferior.
- Servicios. En esta capa se agrupan los componentes que proporcionan servicios relacionados con la factura electrónica. Se distinguen componentes para proporcionar servicios de serialización, validación, seguridad y transformación. Los componentes para la gestión de la factura CCI UBL están a caballo entre esta capa y la capa de tipos agregados.
- Perfiles y modelos. Capa de nivel superior que permite el perfilado de la factura electrónica y la gestión de los diferentes modelos especificados en el apartado 2.4 . El perfilado y los modelos harán uso de los distintos servicios y de los tipos xml para la gestión de los documentos de factura. Gracias a la organización por capas, mediante la librería un cliente podrá definir su propio perfil y modelos asociados.

A continuación se muestra un gráfico que ilustra la organización de los componentes de la librería.

Figura 7: Diagrama de componentes de las librerías

A continuación se detallan las relaciones existentes entre los componentes de las dos capas inferiores. Entre ellos existen relaciones de herencia y de agregación. La figura 9 no representa relaciones de herencia y agregación entre clases sino entre paquetes de clases del mismo tipo.

Así, por ejemplo, la forma correcta de leer la figura es que los tipos CBC derivan de su correspondiente *UnqualifiedDataType* y que los tipos de CAC se componen de CBCs de diferentes tipos y de otros CACs según el caso.

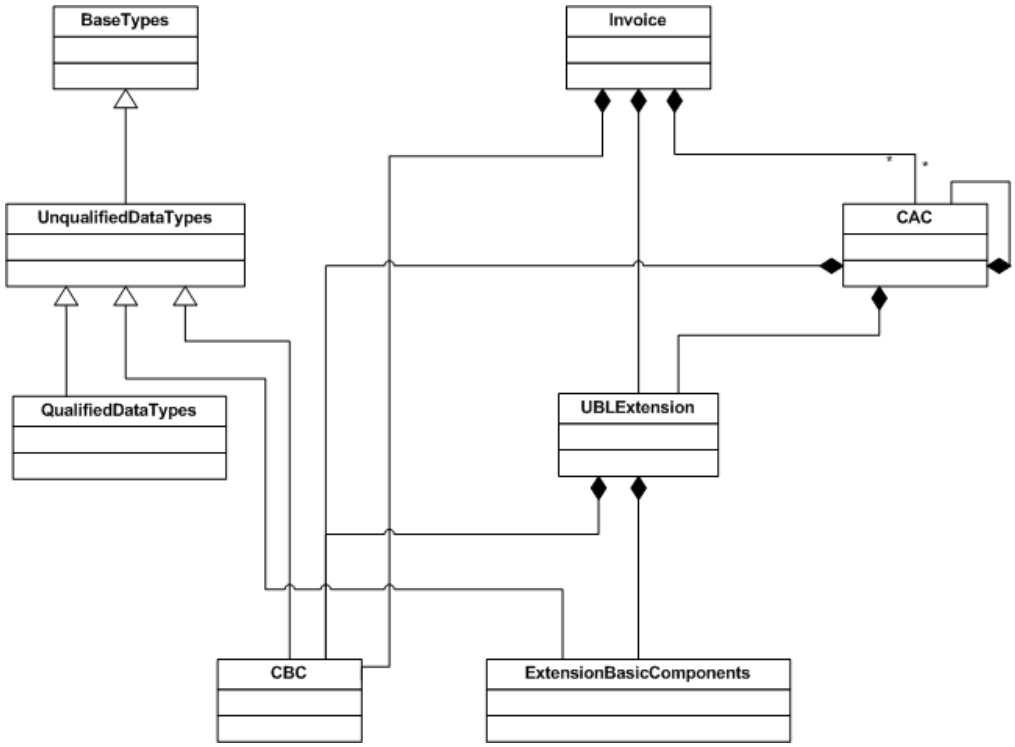


Figura 8: Relaciones de herencia y composición entre los componentes de la factura CCI UBL

4.3 Consideraciones comunes

4.3.1 Nomenclatura de clases

Los nombres de las clases de la librería deben seguir una serie de reglas:

- Las clases para la gestión de elementos XML deben nombrarse de forma igual a la definición del tipo XML. Por ejemplo, la clase para gestionar al elemento *AccountingCostType* debe tener por nombre *AccountingCostType*.
- El package de las clases de gestión de los elementos XML debe seguir las reglas siguientes:
    - La primera parte del package debe ser la determinada por el implementador (común a la librería), quién es responsable de evitar la colisión con otras implementaciones.
    - El último segmento del package debe ser igual al namespace del elemento XML al que corresponde la clase.

4.4 Clases Básicas

Las clases dedicadas a la gestión de elementos XML derivan de una misma

clase llamada *XMLObject*. Esta clase proporcionará funcionalidades comunes así como un conjunto de métodos que deberán ser implementados por cada una de las clases derivadas. Asimismo, la clase *XMLObject* almacenará los datos del elemento XML necesarios para la posterior serialización (nombre del elemento, namespace, etc…).

Las clases que deriven de *XMLObject* deberán implementar los métodos declarados como abstractos que son descritos a continuación:

Método serialize(*)

Descripción	Serializa el elemento, generando la representación textual del mismo. Debe serializar también sus elementos hijos (mediante una invocación a su método para serializar).
Parámetros	Destino de la serialización (flujo, cadena de caracteres, etc…)
Valor de retorno	Ninguno.
Excepciones	En el caso de producirse algún error en la escritura (por ejemplo, error de I/O).

*Dados los múltiplos destinos posibles de serialización, el método deberá ser sobrecargado para cada tipo distinto.

Método validate()

Descripción	Valida el contenido del elemento XML. Debe comprobar que el valor del elemento y de sus atributos cumpla con las restricciones impuestas. Asimismo, el método debe comprobar que los elementos hijos sean también correctos (mediante una llamada a sus métodos validate).
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	Lanzará una *XMLException* en caso de que el contenido de elemento al que representa no se ajuste al esquema.

4.5 Tipos básicos XSD y enumeraciones

Para cada tipo de datos definido en el *xml-schema* se debe crear una clase de *wrapping*, que mapea el tipo a su correspondiente en el lenguaje de programación y proporciona métodos para asignar y obtener su valor de acuerdo con las restricciones de formato especificadas por el *schema*. El interfaz proporcionado por estas clases de *wrapping* será el usado por las clases de nivel superior para gestionar sus valores.

Por otro lado, para la gestión de los elementos y atributos que solo puedan contener un valor perteneciente a una lista se creará un conjunto de clases.

En el caso de listas de valores estáticas, las clases contendrán dichos valores. Por otro lado, para listas de valores dinámicas, las clases deberán leer los datos de un fichero. En ambos casos, las clases permitirán el acceso a los valores a las clases que gestionen los elementos XML.

4.6 Tipos de datos UBL

A partir de los tipos de datos básicos y de las enumeraciones, se definen en UBL un conjunto de tipos de datos básicos de dos tipos: *QualifiedDataType* y *UnqualifiedDataType*. Cada tipo *UnqualifiedDataType* encapsulará un tipo básico *xml-schema* que constituye el valor del elemento. En algunos casos el *UnqualifiedDataType* también incluye atributos que pueden ser opcionales u obligatorios. Por otro lado, los tipos de datos *QualifiedDataTypes* se forman a partir de los *UnqualifiedDataType*. La definición de los tipos *Qualified* añade restricciones a los valores de los tipos *Unqualified*.

4.7 Componentes CBC, CAC y Extensiones

Los componentes CBC, CAC y Extensiones (CEC) se definen a partir de los componentes básicos UBL. Para cada componente CBC, CAC o CEC se definirá una clase que contendrá como datos miembro aquellos tipos de datos, componentes básicos o agregados que le correspondan.

Las clases que implementen componentes CBC derivarán de los correspondientes *UnqualifiedDataType*, mientras que las clases que implementen componentes CAC y CEC se formarán a partir de agregaciones de componentes CBC.

4.8 Documento CCI UBL Invoice

Por encima de los distintos componentes e integrándolos todos se presenta la clase para la gestión de la factura CCI UBL. Esta clase es la que gestiona una factura y permite acceder a su contenido, siendo usada por los servicios, modelos y perfiles.

4.9 Clases de gestión de modelos de CCI UBL Invoice

Por encima de los servicios asociados a la factura se encuentran los distintos modelos. Estos pueden hacer uso de los servicios o comunicarse directamente con la clase Invoice.

En el documento se especifica un conjunto de clases que facilitan la generación de distintos modelos de factura. El diseño pretende facilitar la ampliación a nuevos modelos de factura o a una ampliación de los existentes.

4.10 Servicios asociados a CCI UBL Invoice

Los componentes presentados permiten una gestión básica a nivel XML de las facturas. Mediante ellos se permite el uso, la generación y la modificación de una factura electrónica con su perfil asociado. A partir de ellos se define un conjunto de servicios útiles para complementar la gestión básica de los tipos de datos que representan el contenido del XML.

4.10.1 *Servicios* de Serialización/Deserialización

En primer lugar, se definen los servicios de serialización y deserialización. El objetivo de los mismos es permitir la exportación de los datos de una factura a un formato XML y el proceso inverso, la lectura de una factura en formato XML y generación de los objetos correspondientes para su gestión.

El servicio de serialización es el encargado de realizar el paso de los datos contenidos en los diferentes objetos de la factura a un documento XML. El resultado es un documento XML conforme con la especificación UBL.

El servicio de deserialización realiza el proceso inverso al de serialización. La entrada de dicho servicio es el documento XML que contiene la factura UBL y a partir de él genera los objetos necesarios para su gestión.

4.10.2 Servicios de validación

Los servicios descritos en este apartado tienen por objetivo la comprobación de la validez de una factura. Dicha comprobación se realizará tanto a nivel de formato como a nivel de contenido.

Los servicios de validación son los siguientes:

- Servicio de validación contra *schema*. Realiza una validación de la factura contra el *schema*. Se comprueba que los campos del XML de la factura estén acorde con lo especificado en el *schema*.
- Servicio de validación de reglas de negocio. Realiza las verificaciones siguientes:
    - Validación *schematron*. Se valida el contenido de la factura mediante el procesado de documentos *schematron* para comprobar que sea correcto.
    - Validación de *genericodes*. Comprueba que los valores de los elementos o atributos que deban formar parte de una lista de valores especificada en un archivo de *genericodes* sean correctos.

4.10.3 Servicios de Seguridad

Los servicios de seguridad garantizan la integridad, la autenticidad del origen y la confidencialidad de la factura electrónica.

El servicio de firma electrónica permite generar y verificar firmas electrónicas sobre una factura. Mediante la firma se puede garantizar la integridad del documento, haciendo detectable su modificación. Asimismo, será posible por el receptor conocer la identidad del firmante.

Por otro lado, el servicio de cifrado tiene como objetivo proporcionar confidencialidad a la factura. Mediante la aplicación del cifrado, el emisor puede tener la certeza de que la factura solo podrá ser vista por quién tenga la clave de descifrado.

4.10.4 Servicios de Transformación

Por último, la librería debe ofrecer servicios de transformación de formato de la factura. El objetivo de las transformaciones puede ser la obtención de la misma factura representada en un formato de visualización, otro formato de facturación electrónica e incluso la conversión a un formato no XML que facilite su visualización para el usuario.

Las transformaciones consistirán principalmente en transformaciones XSLT. El servicio de transformación aceptará un archivo XSLT que especifique los pasos a seguir para ser realizada. Al aceptar transformaciones XSLT la herramienta aceptará por lo tanto nuevas transformaciones definidas por el cliente de la librería.

4.11 Jerarquía de excepciones

Los componentes de la librería deben lanzar excepciones en el caso de darse una situación anómala o incorrecta. Para una mayor comodidad en su gestión, se han organizado las excepciones siguiendo la siguiente jerarquía (Figura 10 y Figura 11):

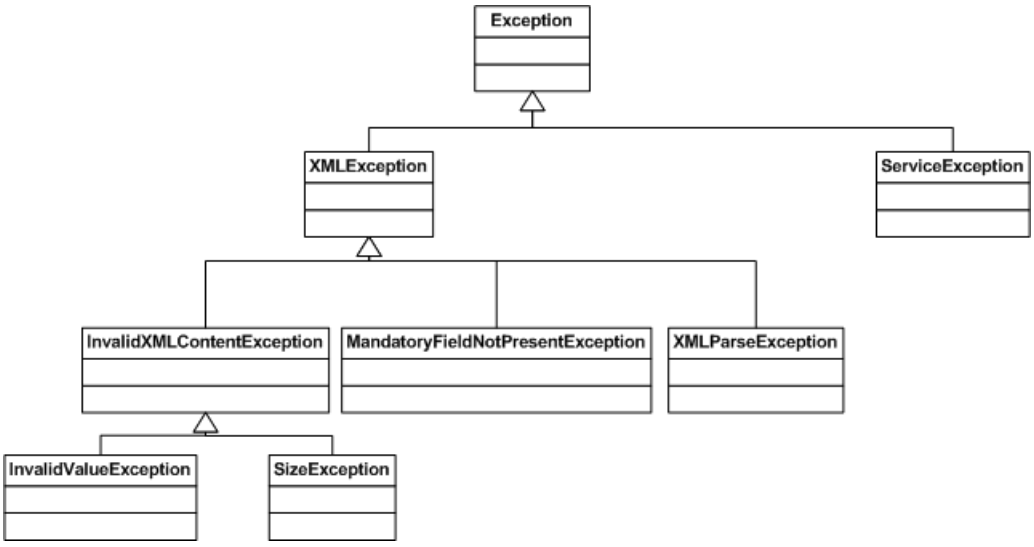


Figura 9: Diagrama de clases de excepción

En el diagrama se puede ver el conjunto de excepciones definidas. La primera gran división se realiza según si la causa de la excepción producida está relacionada con la generación, gestión o parseado del documento XML o si se ha producido al ejecutar un servicio. A continuación se describe cuando debe utilizarse cada uno de los tipos de excepciones.

Excepciones relacionadas con los elementos XML:

Nombre	***Significado***
InvalidXMLContentException	Hay un error en el contenido del XML o se está intentando asignar un contenido incorrecto a un elemento o un atributo. Su propósito principal es de actuar como superclase de InvalidValueException y de SizeException.
InvalidValueException	Error en el valor del elemento XML.
SizeException	Se ha intentado exceder el límite de elementos fijado por el schema.
MandatoryFieldNotPresentException	Un campo obligatorio no está presente.
XMLParseException	Error parseando el XML.

Excepciones relacionadas con los servicios:

Nombre	***Significado***
MarshallingException	Excepción que agrupa los errores del servicio de serialización/deserialización a xml.
SerializationException	Error durante la ejecución del servicio de serialización.
DeserializationException	Error durante la ejecución del servicio de deserialización.
TransformException	Error durante la ejecución de una transformación.
ValidationException	Excepción que agrupa los errores del servicio de validación de las facturas contra el schema y las reglas de negocio.
SchemaException	Error durante la validación de una factura contra el *xml schema*.
SchematronException	Error durante la validación de una factura contra un schematron.
BusinessRulesException	Error durante la validación de una factura contra las reglas de negocio.
GenericodeException	Error durante la validación de una factura contra una lista de códigos.
SecurityException	Excepción que agrupa los errores producidos por los servicios de seguridad.
CipherException	Error durante la ejecución del servicio de cifrado.
SignatureException	Error durante la ejecución del servicio de generación/validación de firmas.

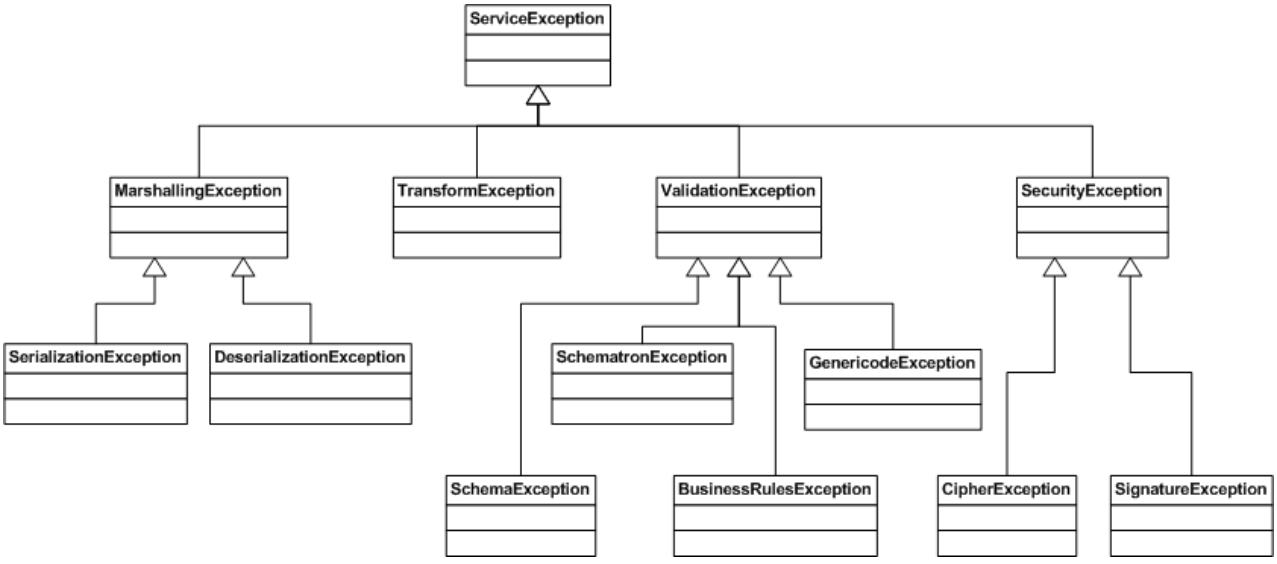


Figura 10: Diagrama de clases de excepción de los servicios

5 ESPECIFICACIÓN DE LOS COMPONENTES BÁSICOS

5.1 Especificación de las clases de tipos básicos xml-xchema (xsd)

Los tipos básicos *xml-schema* utilizados en la factura CCI UBL se corresponden según la siguiente tabla a los tipos básicos y clases de los lenguajes de programación Java y C#:

xsd	*Tipo de datos Java*	*Tipo de datos C#*
xsd:string	String	string
xsd:nomalizedStrin g	String	string
xsd:boolean	boolean	bolean
xsd:base64Binary	byte[]	byte[]
xsd:anyURI	java.net.URI	string
xsd:dateTime	java.util.Calendar	System.DateTime
xsd:date	java.util.Calendar	System.DateTime
xsd:time	java.util.Calendar	System.DateTime
xsd:decimal	java.math.BigDecimal *	Decimal
xsd:language	String	String
xsd:token	String	String

* para optimizar, el tipo xsd:decimal se puede convertir a:
- BigInteger si el atributo fractionDigit es igual a 0.
- long si el atributo totalDigits <=18 o si el rango entre min y max está dentro del representable con 64-bits complemento a 2.

Para cada uno de los tipos de datos de la columna *xsd* se debe crear una clase de *wrapping*, que mapea el tipo a su correspondiente en el lenguaje de programación y proporciona métodos para asignar y obtener su valor de acuerdo con las restricciones de formato especificadas por el *schema*.

Todas las clases de *wrapping* de los tipos *xml-schema* deben derivar de una misma clase XmlObject.

Cada una de las clases de *wrapping* debe contener:

Método set*DataType*Value(*DataType* x)

Descripción	Asigna un valor al tipo de dato xsd que encapsula la clase de *wrapping*.
Parámetros	Nuevo valor del tipo de dato como tipo de dato / clase en el lenguaje de programación.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setStringValue(String x)

Descripción	Asigna un valor al tipo de dato xsd que encapsula la clase de *wrapping*.
Parámetros	Nuevo valor del tipo de dato pasado como su representación textual en el xml.
Valor de retorno	Ninguno.

Excepciones	En el caso que la cadena de caracteres recibida como parámetro no tenga el formato correcto para ser un valor válido del tipo básico *xml-schema* correspondiente lanzará una excepción *InvalidValueException*.

Método get*DataType*Value()

Descripción	Método para acceder al tipo de dato encapsulado por la clase de *wrapping*.
Parámetros	Ninguno.
Valor de retorno	El dato
Excepciones	Ninguna.

Método getStringValue()

Descripción	Método para acceder al tipo de dato encapsulado por la clase de *wrapping*.
Parámetros	Ninguno.
Valor de retorno	Una cadena de caracteres con la representación textual del tipo de dato en el documento xml. Si el dato está vacío devuelve null.
Excepciones	Ninguna.

A modo de ejemplo, en la implementación en Java, para el tipo *xsd:dateTime* se debe implementar una clase de *wrapping* con un dato miembro privado del tipo *Calendar* y los siguientes métodos:

```
void setCalendarValue(Calendar d)
void setStringValue(String value) throws InvalidValueException
Calendar getCalendarValue()
String getStringValue()
```

Ejemplo 6: Implementación de un tipo básico xml-schema en Java

Se debe prestar especial atención a la representación textual de los tipos básicos *xml-schema time*, *date* y *dateTime* en el fichero xml. Toda la información de los tipos definidos en la especificación *xml-schema* y su formato se puede consultar en la referencia [XSD].

5.2 Especificación de las clases de las listas de códigos

Los elementos/atributos dentro del esquema de factura electrónica CCI UBL que se definen como un elemento dentro de listas de códigos o que deben tomar un valor de ellas se deben implementar según lo especificado en el presente apartado.

Estos tipos se han de implementar mediante una clase que contenga todos

los valores de la lista en una enumeración o una estructura análoga. La clase debe permitir seleccionar un único valor para el elemento y solo entre los múltiples posibles códigos de la lista. La clase debe contener también un método que retorne el valor textual del elemento. También es aconsejable que cada clase que da suporte a una lista de códigos disponga de otro método que retorne un valor entero asociado a la posición del valor seleccionado dentro de la lista para poderlo utilizar en sentencias de tipo *switch*.

En C# estas clases se pueden implementar mediante un *enum*.

En Java las posibilidades de los tipos enumerados (*enum*) varían según la versión del lenguaje que se utilice, por lo que se debe prestar atención si se está generando código para las versiones de JRE 1.5 o superiores, o para versiones anteriores. Las librerías objeto de esta especificación deben funcionar sobre JRE 1.4, por lo que se proponen las siguientes directivas para la programación de las clases de enumeración de códigos:

- Deben contener un conjunto de valores estáticos y públicos con todos los valores posibles de la enumeración.
- Deben contener un diccionario (se recomienda una tabla de hash) estático donde se deben añadir el conjunto de valores del punto anterior. La tabla debe relacionar la posición del valor dentro de la enumeración con su valor textual correspondiente.

5.2.1 Listas de códigos de atributos definidas en xml-schemas

Las clases de enumeración de códigos que se corresponden con los siguientes tipos *xml-schema* definidos por UN/CEFACT se utilizan para fijar el valor de los atributos de los siguientes *UnqualifiedDataType* [Apartado 5.1]: *AmountType*, *BinaryObjectType*, *MeasureType* y *QuantityType*.

CurrencyCodeContentType	Códigos de moneda internacionales
LanguageCodeContentType	Códigos de idioma internacionales
UnitCodeContentType	Códigos de unidades
BinaryObjectMimeType	Códigos de tipos MIME para archivos

Las 4 listas de códigos anteriores no necesitan añadir ninguna propiedad o método a lo especificado en el punto anterior de forma estándar para todas las listas de códigos.

5.2.2 Clases para la gestión de listas genericode

Las objetos que gestionan listas de códigos / identificadores definidos en archivos genericode deben permitir su carga dinámica. Se deben implementar dos clases para la creación de estos objetos, una para gestionar listas de códigos (*GenericodeCodeList*) y otra para la gestión de identificadores (*GenericodeIdentifierList*). Albergaran en su interior una tabla de códigos según lo especificado en el apartado 5.1 y los dos métodos siguientes para la carga dinámica de los mismos:

Método loadGenericodeFile(String fileName)

Descripción	Carga una lista de códigos genericode en la tabla de la clase para la gestión de códigos.
Parámetros	Nombre del fichero.
Valor de retorno	Ninguno.
Excepciones	Lanzará una excepción de I/O en caso de no encontrar el fichero o problemas de lectura. En caso de encontrar un problema al tratar con la estructura xml del fichero de códigos lanzará una *XmlParseException*.

Método loadGenericodeFile(URL value)

Descripción	Carga una lista de códigos genericode en la tabla de la clase para la gestión de códigos.
Parámetros	URL con la ubicación del fichero de genericodes.
Valor de retorno	Ninguno.
Excepciones	Lanzará una excepción de I/O en caso de no poder descargar el fichero o de problemas de lectura. En caso de encontrar un problema al tratar con la estructura xml del fichero de códigos lanzará una *XmlParseException*.

5.2.3 Listas de Códigos definidas en archivos genericode

Las listas de códigos de este apartado fijan el valor del elemento derivado a partir de un *UnqualifiedDataType CodeType* entre los contenidos en una lista especificada en forma de archivo genericode. Las listas de la siguiente tabla se deberán cargar dinámicamente sobre un objeto de la clase *GenericodeCodeList* [Apartado 5.2.2]:

Lista de códigos	***listSchemeURI***
InvoiceTypeCode	urn:fundetec:codelist:InvoiceTypeCode-2.0
CurrencyCode	urn:un:unece:uncefact:codelist:specification:54217:2001
CountryIndentificationCode	urn:oasis:names:specification:ubl:codelist:gc:CountryIdentificationCode-2.0
INE_CountrySubEntityCode	urn:fundetec:codelist:INE_CountrySubentityCode-2.0
OperatorCode	urn:oasis:names:specification:ubl:codelist:gc:OperatorCode-2.0
AllowanceChargeReasonCode	urn:oasis:names:specification:ubl:codelist:gc:AllowanceChargeReasonCode-2.0
TaxExcemptionReasonCode	urn:fundetec:codelist:TaxExemptionReasonCode-2.0
TaxTypeCode	urn:fundetec:codelist:TaxTypeCode-2.0
PaymentMeansCode	urn:oasis:names:specification:ubl:codelist:gc:PaymentMeansCode-2.0
UnitOfMeasureCode	urn:un:unece:uncefact:codelist:specification:66411:2001
DocumentTypeCode	urn:fundetec:codelist:DocumentTypeCode-2.0
BinaryObjectMimeCode LanguageCode	urn:un:unece:uncefact:codelist:specification:IANA_7_04:MIMEMediaType:2003 urn:un:unece:uncefact:codelist:specification:63453:2005

Además la clase *GenericodeCodeList* deberá contener a parte de la enumeración de códigos dos datos miembro para guardar el valor de los atributos *listURI* y *listschemeURI* del elemento *CodeType* con sus correspondientes métodos **set** y **get**, que se utilizan respectivamente para informar de la dirección donde se puede descargar el archivo genericode usado y de su URI canónica asociada.

5.2.4 Listas de identificadores definidas en archivos genericode

Las listas de códigos de este apartado fijan un valor del atributo *schemeName* de un elemento derivado a partir del *UnqualifiedDataType IdentifierType* entre los contenidos en una lista especificada en forma de archivo genericode. Las listas de la siguiente tabla se deberán cargar dinámicamente sobre un objeto de la clase *GenericodeIdentifierList* [Apartado 5.2.2]:

Lista de códigos	*schemeDataURI*
PartyIdentificationID	urn:fundetec:scheme:partyidentification-1.0
TaxSchemeID	urn:fundetec:scheme:taxscheme-1.0
TaxCategoryID	urn:fundetec:scheme:taxcategory-1.0

Estas clases deberán contener además de la tabla de códigos un dato miembro para guardar el valor del atributo *schemeDataURI* del elemento *IdentifierType*, que se utiliza para informar de la URI canónica asociada a la lista de genericodes en uso para ese atributo.

5.3 Especificación de las clases del paquete UN/CEFACT Unqualified Data Types

Todos los tipos básicos, agregados y las extensiones de UBL se construyen a partir de la siguiente lista de tipos:

Tipo UN/CEFACT Unqualified Data Type	***Tipo básico xsd***	***Atributo/s***
AmountType	xsd:decimal	Sí (obligatorio)
BinaryObjectType	xsd:base64Binary	Sí (obligatorio y opcionales)
GraphicType	xsd:base64Binary	Sí (obligatorio y opcionales)
PictureType	xsd:base64Binary	Sí (obligatorio y opcionales)
SoundType	xsd:base64Binary	Sí (obligatorio y opcionales)
VideoType	xsd:base64Binary	Sí (obligatorio y opcionales)
CodeType	xsd:normalizedString	Sí (obligatorio y opcionales)
DateTimeType	xsd:dateTime	No
DateType	xsd:date	No
TimeType	xsd:time	No
IdentifierType	xsd:normalizedString	Sí (opcionales)
IndicatorType	xsd:normalizedString	No
MeasureType	xsd:decimal	Sí (obligatorio)
NumericType	xsd:decimal	No
ValueType	xsd:decimal	No
PercentType	xsd:decimal	No
RateType	xsd:decimal	No
QuantityType	xsd:decimal	Sí (opcional)
TextType	xsd:string	Sí (opcional)
NameType	xsd:string	Sí (opcional)

Cada tipo de la lista (primera columna, UN/CEFACT Unqualified Data Type) encapsula un tipo básico *xml-schema* que constituye el valor del elemento. En algunos casos el *UnqualifiedDataType* también incluye atributos que pueden ser opcionales u obligatorios. Estos atributos se detallan en las tablas siguientes:

AmountType	
currencyID	Obligatorio

BinaryObjectType	
format	Opcional
mimeCode	Obligatorio
encodingCode	Opcional
characterSetCode	Opcional
uri	Opcional
filename	Opcional

GraphicType	
format	Opcional
mimeCode	Obligatorio
encodingCode	Opcional
uri	Opcional
filename	Opcional

PictureType	
format	Opcional
mimeCode	Obligatorio
encodingCode	Opcional
uri	Opcional
filename	Opcional

SoundType	
format	Opcional
mimeCode	Obligatorio
encodingCode	Opcional
uri	Opcional
filename	Opcional

VideoType	
format	Opcional
mimeCode	Obligatorio
encodingCode	Opcional
uri	Opcional
filename	Opcional

CodeType	
listID	Opcional
listAgencyID	Opcional
listAgencyName	Opcional
listName	Opcional
listVersionID	Opcional
name	Opcional
languageID	Opcional
listURI	Opcional
listSchemeURI	Opcional

IdentifierType	
schemeID	Opcional
schemeName	Opcional
schemeAgencyID	Opcional
schemeAgencyName	Opcional
schemeVersionID	Opcional
schemeDataURI	Opcional
schemeURI	Opcional

MeasureType	
unitCode	Obligatorio

QuantityType	
unitCode	Opcional

TextType	
languageID	Opcional

NameType	
languageID	Opcional

Cada *UnqualifiedDataType* constituirá una clase y recibirá / devolverá sus tipos básicos equivalentes en el lenguaje de programación. Todas las clases que implementan *UnqualifiedDataType* derivarán de la clase *wrapper* [Apartado 5.1] del tipo básico *xml-schema* que les corresponda. El constructor de estas clases no recibirá parámetros y creará un objeto vacío, los datos de la clase se deben rellenar a posteriori mediante las operaciones destinadas a añadir/modificar su contenido. El acceso a los valores del elemento y sus atributos se debe realizar conforme a la especificación de los próximos dos apartados.

Todos los atributos subrayados de las clases *UnqualifiedDataType* toman su valor de una secuencia de valores posibles especificada mediante *xml-schema* y corresponden a códigos estandarizados por UN/CEFACT.

5.3.1 Métodos para la gestión del valor de un elemento

Los métodos para acceder al valor del elemento serán los que se hereden de la clase *wrapper* del tipo básico *xml-schema* correspondiente [Apartado 5.1 **¡Error! No se encuentra el origen de la referencia.**].

Para la clase *CodeType* se implementará también el siguiente método:

Método *setElementX(EnumCodeType* x)

Descripción	Método para asignar un valor de los posibles dentro de una lista de códigos a un elemento del tipo *CodeType*. x es un objeto de la clase *GenericodeCodeList* [Apartado **¡Error! No se encuentra el origen de la referencia.**] y el método fijará automáticamente el valor de los atributos *listURI* y *listschemeURI* a partir del contenido en x.
Parámetros	Objeto de la clase *EnumCodeType* [Apartado 5.1] correspondiente a la enumeración de códigos.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

5.3.2 Métodos para la gestión de atributos

Los elementos que puedan contener atributos deberán disponer de los métodos necesarios para su gestión. Concretamente, se puede dar el caso de atributos que sean obligatorios y otros que sean opcionales. Asimismo, algún atributo puede tener un valor por defecto.

En cualquiera de los casos citados, todas las clases que contengan atributos deben proporcionar los siguientes métodos para cada uno de ellos:

Método get*AttributeX*()

Descripción	Método para acceder al atributo de nombre *AttributeX*.
Parámetros	Ninguno.
Valor de retorno	Objeto de la clase correspondiente al atributo solicitado. Si el atributo no tiene ningún valor asignado, devuelve null o el valor por defecto en el caso de haber uno especificado.
Excepciones	Ninguna.

Método set*AttributeX*(*LanguageBaseType* x)

Descripción	Método para asignar un valor al atributo de nombre *AttributeX*.
Parámetros	Objeto de la clase *LanguageBaseType* (cadena de caracteres, decimal, etc…) que contendrá el valor a asignar al atributo.
Valor de retorno	Ninguno.
Excepciones	En el caso de incumplir las restricciones impuestas por el *schema* lanzará una excepción *InvalidValueException*.

Método set*AttributeX*(*EnumCodeType* x)

Descripción	Método para asignar un valor al atributo de nombre *AttributeX*, cuyo valor se encuentra en una lista de códigos. Para los objetos del tipo *IdentifierType*, si x es un objeto de la clase *GenericodeIdentifierList* [Apartado **¡Error! No se encuentra el origen de la referencia.**] *y AttributeX* es *schemeName*, fijará automáticamente el valor del atributo *schemeDataURI* a partir del contenido en x.
Parámetros	Objeto de la clase *EnumCodeType* [Apartado 5.1] correspondiente a la enumeración de códigos.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Para los atributos opcionales se deben añadir los siguientes métodos:

Método has*AttributeX*()

Descripción	Permite determinar si el atributo opcional de nombre *AttributeX* está presente en el elemento.
Parámetros	Ninguno.
Valor de retorno	Valor booleano cierto si el atributo existe en el elemento o falso en el caso contrario.
Excepciones	Ninguna.

Método remove*AttributeX*()

Descripción	Elimina el atributo opcional de nombre *AttributeX* del elemento si está presente. En caso contrario no hace nada. Una vez ejecutado el método, si se invoca al método has*AttributeX*(), este debe devolver el valor falso.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

A modo de ejemplo, en Java, para implementar la gestión de atributos del *UnqualifiedDataType* TextType:

```
String getLanguageID()
boolean hasLanguageID()
void setLanguageID(String value) throws InvalidValueException
void removeLanguageID()
```

Ejemplo 7: Implementación de métodos para la gesitión de atributos opcionales de un Unqualifed Data Type en Java

5.4 Especificación de las clases del paquete UBL Qualified Data Types

Los tipos de datos *QualifiedDataTypes* se forman a partir de los *UnqualifiedDataType* descritos en el apartado anterior. La definición de los tipos *Qualified* añade restricciones a los valores de los tipos *Unqualified*. Por este motivo la clase que implemente a un tipo *Qualified* derivará de la clase correspondiente al tipo *Unqualified* al cual añade restricciones. Los métodos serán los mismos de la clase base, pero se deberán volver a implementar a fin de introducir la variación de comportamiento que suponen las restricciones.

En la siguiente tabla se muestra de qué clase usada para implementar un tipo *Unqualified* debe derivar cada una de las clases correspondientes a los tipos *Qualified*.

Clase Qualified	***Clase Unqualified***
AllowanceChargeReasonCodeType	CodeType
ChannelCodeType	CodeType
ChipCodeType	CodeType
ContainerSizeTypeCodeType	CodeType
CountryIdentificationCodeType	CodeType
CurrencyCodeType	CodeType
DocumentStatusCodeType	CodeType
LatitudeDirectionCodeType	CodeType
LineStatusCodeType	CodeType
LongitudeDirectionCodeType	CodeType
OperatorCodeType	CodeType
PackagingTypeCodeType	CodeType
PaymentMeansCodeType	CodeType
PortCodeType	CodeType
SubstitutionStatusCodeType	CodeType
TransportationStatusCodeType	CodeType
TransportEquipmentTypeCodeType	CodeType
TransportModeCodeType	CodeType
UnitOfMeasureCodeType	CodeType

Las restricciones se basan en el uso de listas de códigos [Apartado 5.2.2] para limitar el conjunto de valores posibles.

6 ESPECIFICACIÓN DE CLASES CBC, CAC Y EXTENSIONES

En este apartado se detallan los métodos que deben ser implementados por las clases destinadas a la gestión de los componentes básicos (CBC), agregados (CAC) y extensiones (CEC) definidos en la personalización CCI UBL. Dichos métodos deben permitir la gestión del contenido de los elementos XML a los que representan.

Cada componente CBC, CAC o CEC constituye una clase, que contendrá como datos miembro aquellos tipos de datos, componentes básicos o agregados que le correspondan. Para acceder a estos datos se deben implementar los métodos según proceda en cada caso siguiendo las especificaciones de los siguientes apartados.

Las clases que implementan componentes CBC deben derivar de su correspondiente *UnqualifiedDataType* [Apartado 5.1]. Algunas de las extensiones que denominaremos *BasicExtensionComponents* también deben derivar de su correspondiente *UnqualifiedDataType*.

La tabla que sigue muestra las clases *UnqualifiedDataType* de las que derivan las clases que implementan los correspondientes componentes CBC:

CBC	***UnqualifedDataType***
AccountingCostCodeType	CodeType
AccountingCostType	TextType
ActualDeliveryDateType	DateType
AllowanceChargeReasonCodeType	CodeType
AllowanceChargeReasonType	TextType
AllowanceTotalAmountType	AmountType
AmountType	AmountType
BaseAmountType	AmountType
BrandNameType	NameType
CalculationRateType	RateType
CalculationSequenceNumericType	NumericType
CanonicalizationMethodType	TextType
ChargeIndicatorType	IndicatorType
ChargeTotalAmountType	AmountType
CityNameType	NameType
CompanyIDType	IdentifierType
CopyIndicatorType	IndicatorType
CountrySubentityCodeType	CodeType
CountrySubentityType	TextType
CurrencyCodeType	CodeType
CustomerAssignedAccountIDType	IdentifierType
CustomizationIDType	IdentifierType
DateType	DateType
DescriptionType	TextType
DocumentCurrencyCodeType	CodeType
DocumentHashType	TextType
DocumentType	TextType
DocumentTypeCodeType	CodeType
ElectronicMailType	TextType
EmbeddedDocumentBinaryObjectType	BinaryObjectType
EndDateType	DateType
EndpointIDType	IdentifierType
FamilyNameType	NameType
FirstNameType	NameType
FreeOfChargeIndicatorType	IndicatorType
ID	IdentifierType
IdentificationCodeType	CodeType
InstructionNoteType	TextType
InvoiceTypeCodeType	CodeType
InvoicedQuantityType	QuantityType
IssueDateType	DateType
LineCountNumericType	NumericType

CBC	***UnqualifedDataType***
LineExtensionAmountType	AmountType
LineType	TextType
MathematicOperatorCodeType	CodeType
ModelNameType	NameType
MultiplierFactorNumericType	NumericType
NameType	NameType
NoteType	TextType
OrderableUnitFactorRateType	RateType
OrganizationDepartmentType	TextType
PaidAmountType	AmountType
PaidDateType	DateType
PayableAmountType	AmountType
PaymentDueDateType	DateType
PaymentIDType	IdentifierType
PaymentMeansCodeType	CodeType
PaymentMeansIDType	IdentifierType
PercentType	PercentType
PostalZoneType	TextType
PrepaidAmountType	AmountType
PrepaidIndicatorType	IndicatorType
PrepaidPaymentReferenceIDTyp e	IdentifierType
PriceAmountType	AmountType
ProfileIDType	IdentifierType
QuantityType	QuantityType
ReceivedDateType	DateType
RegistrationNameType	NameType
SequenceNumericType	NumericType
SignatureMethodType	TextType
SourceCurrencyBaseRateType	RateType
SourceCurrencyCodeType	CodeType
StartDateType	DateType
SupplierAssignedAccountIDTyp e	IdentifierType
TargetCurrencyBaseRateType	RateType
TargetCurrencyCodeType	CodeType
TaxAmountType	AmountType
TaxCurrencyCodeType	CodeType
TaxExclusiveAmountType	AmountType
TaxExemptionReasonCodeType	CodeType
TaxExemptionReasonType	TextType
TaxInclusiveAmountType	AmountType
TaxTypeCodeType	CodeType
TaxableAmountType	AmountType
TelefaxType	TextType
TelephoneType	TextType

CBC	***UnqualifedDataType***
TitleType	TextType
TransactionCurrencyTaxAmountType	AmountType
UBLVersionIDType	IdentifierType
URI	IdentifierType
WebsiteURIType	IdentifierType

La tabla que sigue muestra las clases *UnqualifiedDataType* de las que derivan las clases que implementan las correspondientes extensiones encuadradas en lo que se ha dado en llamar *BasicExtensionsComponents*:

BasicExtensionComponents	***UnqualifedDataType***
ExtensionAgencyIDType	IdentifierType
ExtensionAgencyNameType	TextType
ExtensionAgencyURIType	IdentifierType
ExtensionContentType	TextType
ExtensionReasonType	TextType
ExtensionReasonCodeType	CodeType
ExtensionURIType	IdentifierType
ExtensiónVersionIDType	IdentifierType

Por otro lado, para la gestión del documento *ApplicationResponse*, se deben implementar los componentes citados en la siguiente lista:

CBC	***UnqualifedDataType***
UUID	IdentifierType
IssueTime	TimeType
ResponseTime ResponseDate	TimeType DateType

CAC	*UnqualifedDataType*
SenderParty	PartyType
ReceiverParty	PartyType

6.1 Métodos para la gestión del valor y los atributos de componentes CBC y BasicExtensionComponents

Solo los componentes CBC y las clases del paquete de extensiones básicas anteriormente listadas pueden tener un valor de elemento y atributos asociados.

Como las clases de los componentes que contendrán atributos y un valor se crean por derivación de su correspondiente *UnqualifiedDataType* los métodos para acceder a sus valores serán los mismos que los de sus tipos base [Apartado 5.1].

6.2 Métodos para la gestión de elementos CAC y del elemento UBLExtension

Los elementos CAC y el elemento UBLExtension están formados por agregaciones de los componentes anteriores y solo tienen métodos para acceder a los elementos que los constituyen. Estas clases tendrán un constructor por defecto sin parámetros y el valor de sus elementos hijos se fijará a posteriori mediante los métodos disponibles a tal efecto. La implementación de todos los métodos de acceso se deberá realizar según lo detallado en los apartados 6.2 .x.

6.2.1 Métodos para la gestión de elementos obligatorios

Cada elemento que contenga otros elementos debe proporcionar un interfaz adecuado para acceder a ellos. Para la gestión de elementos cuya presencia dentro del elemento padre sea obligatoria, la clase correspondiente debe proporcionar los siguientes métodos:

Método get*ElementNameX*()

Descripción	Método para acceder al elemento hijo de nombre *ElementNameX*.
Parámetros	Ninguno.
Valor de retorno	Objeto de la clase correspondiente al elemento solicitado. Si el elemento no existe, devuelve null.
Excepciones	Ninguna.

Método set*ElementNameX* (*ElementTypeX* x)

Descripción	Método para asignar un objeto al elemento del tipo *ElementTypeX*.
Parámetros	Objeto de la clase correspondiente al elemento al que se quiere asignar el valor (CBC o *BasicExtensionComponent*).
Valor de retorno	Ninguno.
Excepciones	En el caso de incumplir las restricciones impuestas por el *schema* lanzará una excepción *InvalidValueException*.

A modo de ejemplo, en Java, para implementar la gestión del elemento obligatorio *Line* hijo del elemento *AddressLine*:

```
LineType getLine()
void setLine(LineType value) throws InvalidValueException
```

Ejemplo 8: Implementación de métodos para la gestión de elementos obligatorios de un CAC en Java

6.2.2 Métodos para la gestión de elementos opcionales

Para cada elemento hijo contenido cuya presencia sea opcional, se debe añadir un conjunto de métodos para completar su gestión. El interfaz se debe componer de los métodos descritos para la gestión de elementos obligatorios junto con los siguientes métodos:

Método has*ElementNameX* ()

Descripción	Permite determinar si el elemento opcional *ElementNameX* está presente en el componente.
Parámetros	Ninguno.
Valor de retorno	Valor booleano cierto si el elemento existe o falso en caso contrario.
Excepciones	Ninguna.

Método remove*ElementNameX*()

Descripción	Elimina el elemento opcional de nombre *ElementNameX* del conjunto de hijos del componente si está presente. En caso contrario no hace nada. Una vez ejecutado el método, si se invoca al método has*ElementNameX*(), éste debe devolver el valor falso.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

A modo de ejemplo, en Java, para implementar la gestión del elemento opcional *Country* hijo del elemento *AddressType*:

```
/* Métodos descritos para elementos obligatorios */
CountryType getCountry()
void setCountry(CountryType value) throws InvalidValueException
/* Métodos añadidos para elementos opcionales */
boolean hasCountry()
void removeCountry()
```

Ejemplo 9: Implementación de métodos para la gestión de elementos opcionales de un CAC en Java

6.2.3 Métodos para la gestión de elementos con multiplicidad mayor a uno

Cuando la multiplicidad de un elemento es mayor a uno se debe implementar el siguiente conjunto de métodos destinados a su correcta gestión:

Método get*ElementNameX*Array()

Descripción	Método para acceder al listado de elementos *ElementNameX*.
Parámetros	Ninguno.
Valor de retorno	Array con todos los elementos del tipo existentes.
Excepciones	Ninguna.

Método get*ElementNameX*ArrayAsList()

Descripción	Método para acceder al listado de elementos *ElementNameX*.
Parámetros	Ninguno.
Valor de retorno	Lista con todos los elementos del tipo existentes.
Excepciones	Ninguna.

Método get*ElementNameX*Array(int i)

Descripción	Método para acceder a uno de los elementos *ElementNameX* de la lista.
Parámetros	Entero indicando la posición que ocupa el elemento solicitado en la lista.
Valor de retorno	Objeto de la clase correspondiente al elemento solicitado. Si el elemento no existe, devuelve null.
Excepciones	En caso de que la posición esté fuera del rango de los elementos existentes lanzará una excepción *IndexOutOfBoundsException*.

Método set*ElementNameX*Array(*ElementTypeX[]* xx)

Descripción	Método para asignar una nueva lista de elementos *ElementNameX*.
Parámetros	Array con los nuevos elementos.
Valor de retorno	Ninguno.
Excepciones	En el caso de incumplir las restricciones impuestas por el *schema* se lanzará una excepción del tipo *InvalidValueException*. Si el número de elementos del Array está fuera del rango fijado por el *schema* se lanzará una *SizeException*.

Método set*ElementNameX*Array(List xx)

Descripción	Método para asignar una nueva lista de elementos *ElementNameX*.
Parámetros	Lista de elementos de tipo *ElementTypeX* con los nuevos elementos.
Valor de retorno	Ninguno.
Excepciones	En el caso de incumplir las restricciones impuestas por el *schema* se lanzará una excepción del tipo *InvalidValueException*. Si el número de elementos de la lista está fuera del rango fijado por el *schema* se lanzará una *SizeException*.

Método set*ElementNameX*Array(int i, *ElementTypeX* x)

Descripción	Método para asignar un objeto a una posición dada de la lista de elementos *ElementNameX*. Sustituye el elemento que hubiera anteriormente en dicha posición de la lista por el nuevo.
Parámetros	Entero indicando la posición que se desea sustituir por el nuevo elemento. Objeto de la clase correspondiente al elemento al que se quiere asignar el valor.
Valor de retorno	Ninguno.
Excepciones	En caso de incumplir las restricciones impuestas por el *schema* lanzará una excepción *InvalidValueException*. En caso de que la posición esté fuera del rango de los elementos existentes lanzará una excepción *IndexOutOfBoundsException*.

Método add*ElementNameX*Array(*ElementTypeX* x)

Descripción	Método para añadir un objeto al final de la lista de elementos *ElementNameX*.
Parámetros	Objeto de la clase correspondiente al tipo de elementos de la lista que se quiere añadir.
Valor de retorno	Ninguno.
Excepciones	En el caso de incumplir las restricciones impuestas por el *schema* se lanzará una excepción del tipo *InvalidValueException*. Si el nuevo número de elementos del Array está fuera del rango fijado por el *schema* se lanzará una *SizeException*.

Método insert*ElementNameX*Array(int i, *ElementTypeX* x)

Descripción	Método para insertar un objeto en una posición dada de la lista de elementos *ElementNameX*.
Parámetros	Entero indicando la posición donde se desea insertar el nuevo elemento. Objeto de la clase correspondiente a los elementos de la lista.
Valor de retorno	Ninguno.
Excepciones	En el caso de incumplir las restricciones impuestas por el *schema* se lanzará una excepción del tipo InvalidValueException. Si el nuevo número de elementos del Array está fuera del rango fijado por el *schema* se lanzará una SizeException. En caso de que la posición esté fuera del rango de los elementos existentes lanzará una excepción IndexOutOfBoundsException.

Inserta el elemento dado en la posición i, desplazando los elementos ya existentes para dejar espacio al nuevo elemento. Al finalizar la ejecución, el elemento que estuviera en la posición *i* estará en la posición *i+1*, el elemento x estará en la posición *i* y el elemento que estuviera en la posición *i-1* seguirá estando en la *i-1*. Si el hijo tiene algún tipo de restricción, el método deberá realizar la comprobación antes de asignar el nuevo valor. El método debe comprobar que al añadir un nuevo elemento no se exceda el límite de *ElementoTipoX* posibles si está fijado en el *schema*.

Método remove*ElementNameX*Array(int i)

Descripción	Elimina un elemento concreto del listado de elementos *ElementNameX*.
Parámetros	Entero indicando la posición que ocupa el elemento a eliminar.
Valor de retorno	Ninguno.
Excepciones	En el caso que la posición esté fuera del rango de los elementos existentes lanzará una excepción IndexOutOfBoundsException.

Método sizeOf*ElementNameX*Array()

Descripción	Permite consultar el número de elementos de nombre *ElementNameX* hijos del componente.
Parámetros	Ninguno.
Valor de retorno	Número de elementos del tipo *ElementoTipoX*.
Excepciones	Ninguna.

A modo de ejemplo, en Java, para implementar la gestión del listado de elementos *AddressLine* hijos del elemento *AddressType*:

```
AddressLineType[] getAddressLineArray()
AddressLineType getAddressLineArray(int i) throws IndexOutOfBoundsException
void setAddressLineArray (AddressLineType[] value) throws InvalidXMLContentException
void setAddressLineArray (List value) throws InvalidXMLContentException
void setAddressLineArray (int i, AddressLineType value) throws InvalidValueException, IndexOutOfBoundsException
void addAddressLineArray(AddressLineType value) throws InvalidXMLContentException
void insertAddressLineArray(int i, AddressLineType value) throws InvalidXMLContentException, IndexOutOfBoundsException
void removeAddressLineArray(int i) throws IndexOutOfBoundsException
int sizeOfAddressLineArray()
```

Ejemplo 10: Implementación de métodos para la gestión de elementos con multiplicidad mayor a uno de un CAC en Java

7 ESPECIFICACIÓN DE LA CLASE INVOICE

La clase *Invoice* contiene toda la información de una factura electrónica representada en un documento xml CCI UBL. Se corresponde con el elemento *Invoice* de UBL y está compuesta por una colección de CBCs, CACs y extensiones de diferentes tipos. Cada uno de estos componentes aporta un tipo de información a la factura tal y como se puede observar en la figura 12.

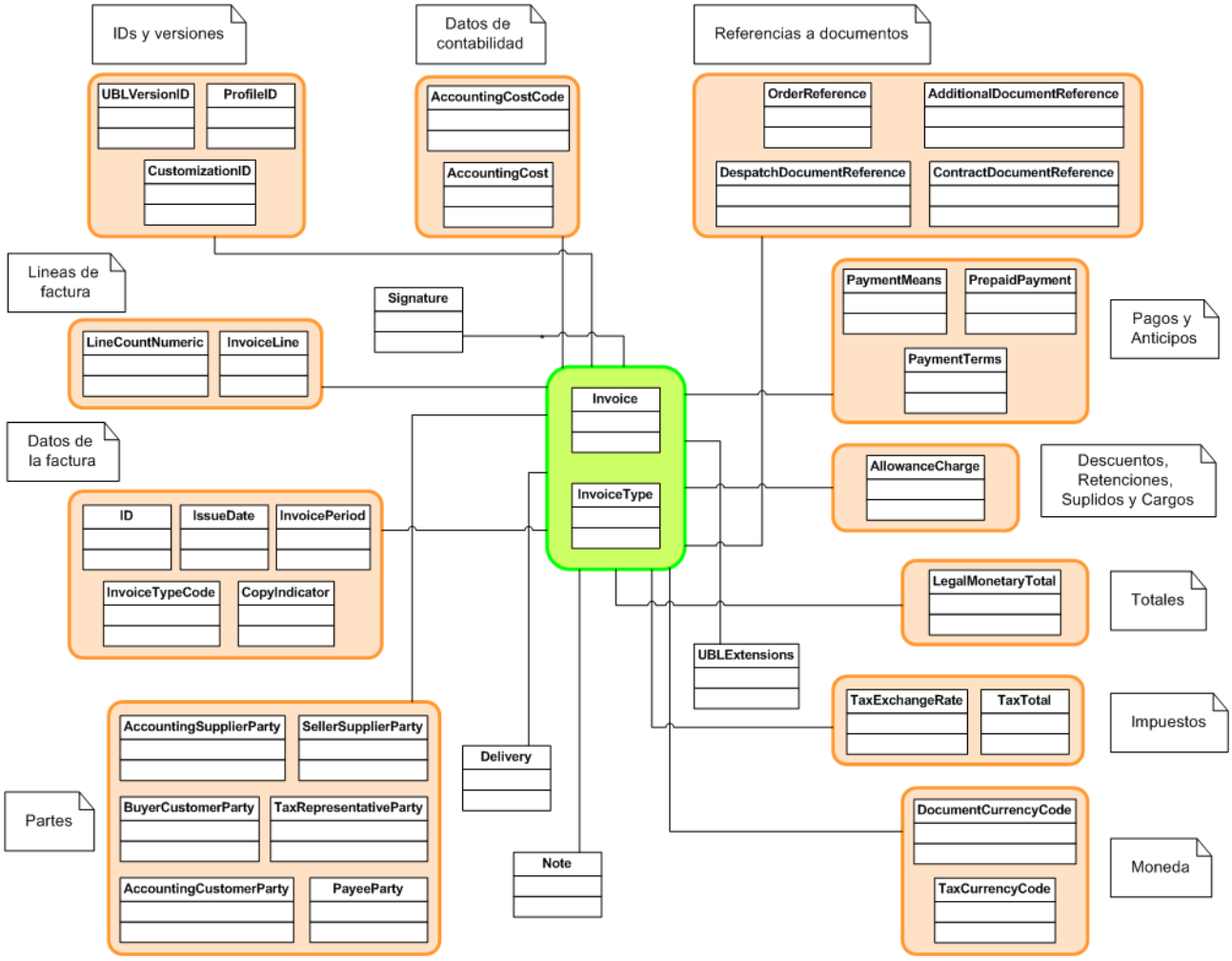


Figura 11: Componentes de la factura CCI UBL

De todos los componentes de la factura existen unos que son obligatorios y otros opcionales según lo definido por el *xml schema*. De la misma manera para cada componente se establece una multiplicidad que puede variar entre una única instancia o múltiples apariciones (constituyendo una lista de elementos del mismo nombre).

La clase *Invoice* como todas las de la librería que representan un

componente xml de UBL derivará del objeto *XmlObject* y extenderá sus métodos *validate* y *serialize.* Implementará métodos para acceder y modificar cualquier componente CBC, CAC o extensión que la constituye siguiendo estas reglas:

Para elementos obligatorios (CBCs o CACs) de multiplicidad uno:
Método get*ElementNameX*()
Método set*ElementNameX*(*ElementType* x)

Para elementos obligatorios (CAC) de multiplicidad mayor a uno:
Método get*ElementNameX*Array()
Método get*ElementNameX*ArrayAsList()
Método get*ElementNameX*Array(int i)
Método set*ElementNameX*Array(*ElementType[]* xx)
Método set*ElementNameX*Array(int i, *ElementType* x)
Método add*ElementNameX*Array(*ElementTypeX* x)
Método inset*ElementNameX*Array(int i, *ElementType* x)
Método remove*ElementNameX*Array(int i)
Método sizeOf*ElementNameX*Array(int i)

Para elementos opcionales (CBCs o CACs) de multiplicidad uno:
Método get*ElementNameX*()
Método set*ElementNameX*(*ElementType* x)
Método has*ElementNameX*()
Método remove*ElementNameX*()

Para elementos opcionales (CBCs, CACs o extensiones) de multiplicidad mayor a uno:
Método has*ElementNameX*Array()
Método remove*ElementNameX*Array()

Todos los métodos anteriores tienen los mismos nombres y comportamiento que los especificados en el apartado 6.2 .

8 ESPECIFICACIÓN DE LAS CLASES PARA GESTIONAR PERFILES

8.1 Modelos

Los modelos presentados en el apartado 2.4 se gestionarán de la forma presentada en este apartado. Las clases tiene como principal objetivo facilitar la generación de facturas con las características del modelo deseado.

El diseño de la jerarquía de clases sigue el patrón Facade [GOF] junto con el patrón decorador [GOF]. Mediante el patrón Facade se consigue un interfaz de generación de facturas simplificado para los casos descritos por los modelos. Por otro lado, mediante el patrón decorador se consigue añadir nuevas propiedades a la factura que permiten obtener los distintos tipos de factura o combinaciones de ellos.

Uno de los objetivos es permitir ampliar la funcionalidad a posteriori. El diseño proporciona la posibilidad de construir nuevos modelos de factura a partir de la creación de clases para la gestión de nuevas propiedades o la programación de nuevas facturas.

La jerarquía es la presentada en el diagrama y se compone de las siguientes clases:

- InvoiceModel
  - MinimalInvoice
  - AmendmentInvoice
- Decorador
  - SummarizeInvoiceData
  - PaymentData
  - PaymentInCashData
  - FactoringData
  - PaymentByTransferData
  - PaymentByDirectDebitData

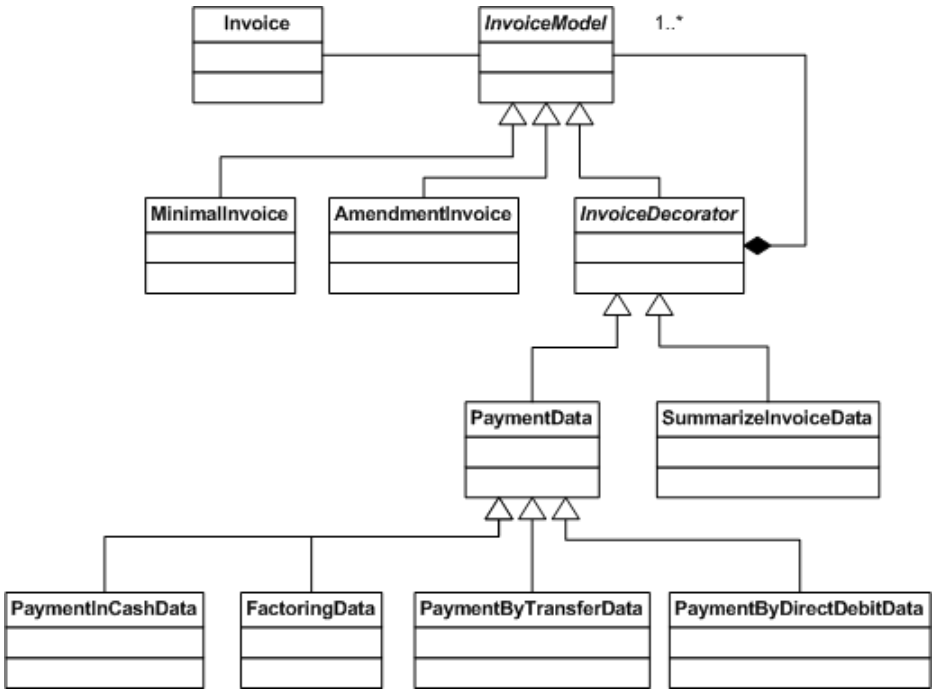


Figura 12: Diagrama de clases para la gestión de modelos de factura

8.1.1 Clase InvoiceModel

La clase *InvoiceModel* es una clase abstracta que actúa como base de todos los modelos de factura que se concreten. Para cada modelo de factura que se desee se debe derivar de esta clase, como se hace para implementar la factura mínima.

La clase debe ser abstracta ya que no se debe poder instanciar al no tratarse de una factura completa. Tiene como dato miembro un objeto de la clase *Invoice* correspondiente a la factura que se generará. Asimismo ofrece un conjunto de métodos que permiten fijar los datos básicos comunes y obligatorios para cualquier tipo de factura. Los métodos deben estar implementados en esta clase, ninguno de ellos debe ser abstracto. Estos métodos son:

En primer lugar, el valor del elemento *UBLVersionID* queda fijado de forma automática en la API y, por lo tanto, no existe ningún método para modificarlo.

Método setCustomizationID(String customizationID)

Descripción	Permite fijar el contenido del elemento *CustomizatonID*.
Parámetros	String con el contenido del elemento *CustomizatonID*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El valor del elemento *CustomizationID* debe ser fijado por la clase del modelo correspondiente.

Método setProfileID(String profileID)

Descripción	Permite fijar el contenido del elemento *ProfileID*.
Parámetros	String con el contenido del elemento *ProfileID*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setProfileID(*EnumTypeCode* profileID)

Descripción	Permite fijar el contenido del elemento *ProfileID*.
Parámetros	Tipo de dato con el valor deseado para *ProfileID* de una lista de valores especificado, por ejemplo, mediante una lista de genericodes.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El valor del elemento *ProfileID* debe ser fijado por la clase del perfil correspondiente.

Método setID(String id)

Descripción	Permite fijar el contenido del elemento *ID*.
Parámetros	String con el contenido del elemento *ID*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El valor del elemento *ID* corresponde a la concatenación del número y la serie que identifican la factura. Por lo tanto el cliente deberá fijar su valor al construir la factura.

Método setCopyIndicator(boolean copyIndicator)

Descripción	Permite fijar el contenido del elemento *CopyIndicator*.
Parámetros	Booleano con el contenido del elemento *CopyIndicator*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El valor del elemento *CopyIndicator* depende de la factura generada. Por lo tanto el cliente deberá fijar su valor al construir la factura.

Método setIssueDate(Date issueDate)

Descripción	Permite fijar el contenido del elemento *IssueDate*.
Parámetros	Fecha con el contenido del elemento *IssueDate*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El valor del elemento *IssueDate* depende de la factura generada. Por lo tanto el cliente deberá fijar su valor al construir la factura.

Método setInvoiceTypeCode(String invoiceTypeCode)

Descripción	Permite fijar el contenido del elemento *InvoiceTypeCode*.
Parámetros	String con el contenido del elemento *InvoiceTypeCode*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setInvoiceTypeCode(*EnumCodeType* invoiceTypeCode)

Descripción	Permite fijar el contenido del elemento *InvoiceTypeCode* a partir de la elección del valor de una lista.
Parámetros	Tipo de dato con el valor deseado de una lista de valores especificado, por ejemplo, mediante una lista de genericodes.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El valor del elemento *InvoiceTypeCode* depende de la factura generada. Por lo tanto el cliente deberá fijar su valor al construir la factura.

Método setDocumentCurrencyCode(String documentCurrencyCode)

Descripción	Permite fijar el contenido del elemento *DocumentCurrencyCode*.
Parámetros	String con el contenido del elemento *DocumentCurrencyCode*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setDocumentCurrencyCode(*EnumCodeType* documentCurrencyCode)

Descripción	Permite fijar el contenido del elemento *DocumentCurrencyCode* a partir de la elección del valor de una lista.
Parámetros	Tipo de dato con el valor deseado de una lista de valores especificado, por ejemplo, mediante una lista de genericodes.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El valor del elemento *DocumentCurrencyCode* depende de la factura generada. Por lo tanto el cliente deberá fijar su valor al construir la factura. Por defecto este valor será el correspondiente a la moneda Euro.

Método getInvoice()

Descripción	Método para acceder a la factura que se está generando.
Parámetros	Ninguno.
Valor de retorno	Objeto de la clase *Invoice* que contiene la factura que se está generando.
Excepciones	Ninguna.

Método validate()

Descripción	Comprueba que se hayan fijado todos los datos necesarios para el modelo deseado.
Parámetros	Ninguno.
Valor de retorno	Booleano con valor igual a cierto si la factura contiene todos los datos necesarios para el modelo deseado. Falso en caso contrario.
Excepciones	Ninguna.

8.1.2 Clase MinimalInvoice

La clase MinimalInvoice tiene como objetivo facilitar la generación de una factura mínima. Por ello, ofrece un interfaz que permite añadir y manipular únicamente las informaciones de una factura que se han enumerado en el apartado 2.4.1 . La clase deriva de la clase *InvoiceModel* y por ello contiene ya todos los métodos presentados anteriormente.

Los métodos añadidos en esta clase son:

Método setAccountingSupplierPartyIdentification(String id, String name)

Descripción	Permite fijar los datos de identificación del proveedor presentes en una factura mínima.
Parámetros	String con el identificador CIF del proveedor. String con la denominación del proveedor.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingSupplierPartyIdentification(String id, *EnumCodeType* idType, String name)

Descripción	Permite fijar los datos de identificación del proveedor presentes en una factura mínima.
Parámetros	String con el identificador del proveedor. Tipo de documento identificador del proveedor escogido de una lista de códigos (CIF, NIF,...) String con la denominación del proveedor.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingSupplierPartyAddress(String addressLine, String cityName, String postalCode, String country, *EnumCodeType* countryCode)

Descripción	Permite fijar los datos relativos a la dirección del proveedor presentes en una factura mínima.
Parámetros	String con una línea de dirección del proveedor. String con el nombre de la ciudad del proveedor. String con el código postal del proveedor. String con el nombre del país del proveedor. Código del país seleccionado de la lista de códigos correspondiente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingCustomerPartyIdentification(String id, String name)

Descripción	Permite fijar los datos de identificación del cliente presentes en una factura mínima.
Parámetros	String con el identificador del cliente. String con la denominación del cliente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingCustomerPartyIdentification(String id, *EnumCodeType* idType, String name)

Descripción	Permite fijar los datos de identificación del cliente presentes en una factura mínima.
Parámetros	String con el identificador del proveedor. Tipo de documento identificador del cliente escogido de una lista de códigos (CIF, NIF, ...) String con la denominación del proveedor.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingCustomerPartyAddress(String addressLine, String cityName, String postalCode, String country, *EnumCodeType* countryCode)

Descripción	Permite fijar los datos relativos a la dirección del cliente presentes en una factura mínima.
Parámetros	String con una línea de dirección del cliente. String con el nombre de la ciudad del cliente. String con el código postal del cliente. String con el nombre del país del cliente. Código del país seleccionado de la lista de códigos correspondiente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método addTaxes(double base, double percent, *EnumCodeType* typeCode)

Descripción	Añade un elemento *TaxSubtotal* correspondiente a los datos de un impuesto. El cálculo de la cantidad resultante de impuestos es realizada por esta función a partir de la base y el porcentaje. Este resultado debe ser añadido al valor del elemento *TaxAmount*, de forma que el valor de este elemento al finalizar la ejecución de la función sea igual a la suma del valor antes de su ejecución más el resultado del producto de la base por el porcentaje. El valor del atributo *currencyID* toma el mismo valor fijado en el elemento *DocumentCurrencyCode*.
Parámetros	Cantidad base donde aplicar el impuesto. La cantidad debe estar expresada en la moneda indicada en el elemento *DocumentCurrencyCode*. Porcentaje a aplicar. Código del impuesto escogido de la lista de códigos correspondiente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método addIVA(double base, double percent)

Descripción	Añade un elemento *TaxSubtotal* correspondiente a los datos de un impuesto cuyo código es IVA. El cálculo de la cantidad resultante de impuestos es realizada por esta función a partir de la base y el porcentaje. El valor del atributo *currencyID* toma el mismo valor fijado en el elemento *DocumentCurrencyCode*.
Parámetros	Cantidad base donde aplicar el impuesto. La cantidad debe estar expresada en la moneda indicada en el elemento *DocumentCurrencyCode*. Porcentaje a aplicar.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método addInvoiceLine(*EnumCodeType* unitCode, double quantity, double unitaryAmount, String itemDescription)

Descripción	Añade una nueva línea a la factura. La función realizará el cálculo del precio total como producto de la cantidad por el precio unitario. El valor del atributo *currencyID* toma el mismo valor fijado en el elemento *DocumentCurrencyCode*.
Parámetros	Unidades en las que se expresa la cantidad escogidas de la lista de códigos correspondiente. Cantidad de items. Precio unitario. Descripción que aparecerá en la línea.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Por otro lado, esta clase es la responsable de realizar los cálculos necesarios para llenar el elemento *LegalMonetaryTotal*. El valor del atributo *currencyID* de los hijos de *LegalMonetaryTotal* es, al igual que con los otros elementos, igual al fijado en el elemento *DocumentCurrencyCode*.

8.1.3 InvoiceDecorator

La clase abstracta *InvoiceDecorator* no implementa ninguna funcionalidad, simplemente pretende actuar como superclase de los distintos decoradores concretos, quienes serán los encargados de añadir sus prestaciones. Como dato miembro debe tener un objeto de la clase *InvoiceModel*, al que decora añadiéndole nueva funcionalidad. Si se desea añadir unas nuevas propiedades a una factura se debe derivar de esta clase, como se hace con la información de factoring.

La clase *InvoiceDecorator* debe tener un único constructor. Éste debe recibir como parámetro una referencia a un objeto de la clase *InvoiceModel*, que será asignada al correspondiente dato miembro.

Los métodos de la clase decorador son los siguientes:

Método getInvoiceModel()

Descripción	Método para acceder al objeto de la clase *InvoiceModel* contenido en el decorador.
Parámetros	Ninguno.
Valor de retorno	Referencia al dato miembro de la clase *InvoiceModel*.
Excepciones	Ninguna.

8.1.4 Factura recapitulativa

La factura recapitulativa añade el periodo de facturación. La clase *SummarizeInvoice* debe fijar de forma automática el valor del elemento *InvoiceTypeCode* al correspondiente a factura recapitulativa. Al añadir nuevas posibilidades a una factura, la clase será un decorador. Los métodos que tendrá son los siguientes:

Método setInvoicePeriod(Date startDate, Date endDate)

Descripción	Permite fijar el contenido del elemento *InvoicePeriod*.
Parámetros	Fecha con el contenido deseado para el elemento *StartDate*. Fecha con el contenido deseado para el elemento *EndDate*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

8.1.5 Factura rectificativa

La factura rectificativa, definida en el apartado 2.4.3 , tiene como objetivo la corrección de parte o el total de los datos de la factura original. Con este objetivo, y para facilitar la generación de este tipo de facturas, se define la clase *AmendmentInvoice*, derivada de *InvoiceModel*. La clase solo permitirá por simplicidad la generación de facturas que rectifiquen a una sola factura.

La clase proporciona un constructor que recibe como parámetro la factura a rectificar. Este constructor copia la información de la factura a la versión rectificada y fija el valor del elemento *BillingReference*. Asimismo, fijará el valor del elemento *InvoiceTypeCode* al valor correspondiente a factura rectificativa.

Una parte de los métodos que añade la clase *AmendmentInvoice* a los de la clase base tienen como objetivo corregir los datos de la factura original. Estos son los siguientes:

Método setAccountingSupplierPartyIdentification(String id, String name)

Descripción	Permite modificar los datos de identificación del proveedor presentes en la factura original.
Parámetros	String con el identificador del proveedor. String con la denominación del proveedor.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingSupplierPartyIdentification(String id, *EnumCodeType* idType, String name)

Descripción	Permite modificar los datos de identificación del proveedor presentes en en la factura original.
Parámetros	String con el identificador del proveedor. Tipo de documento identificador del proveedor escogido de una lista de códigos. String con la denominación del proveedor.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingSupplierPartyAddress(String addressLine, String cityName, String postalCode, String country, *EnumCodeType* countryCode)

Descripción	Permite modificar los datos relativos a la dirección del proveedor presentes en la factura original.
Parámetros	String con una línea de dirección del proveedor. String con el nombre de la ciudad del proveedor. String con el código postal del proveedor. String con el nombre del país del proveedor. Código del país seleccionado de la lista de códigos correspondiente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingCustomerPartyIdentification(String id, String name)

Descripción	Permite modificar los datos de identificación del cliente presentes en la factura original.
Parámetros	String con el identificador del cliente. String con la denominación del cliente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingCustomerPartyIdentification(String id, *EnumCodeType* idType, String name)

Descripción	Permite modificar los datos de identificación del cliente presentes en la factura original.
Parámetros	String con el identificador del proveedor. Tipo de documento identificador del cliente escogido de una lista de códigos. String con la denominación del proveedor.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setAccountingCustomerPartyAddress(String addressLine, String cityName, String postalCode, String country, *EnumCodeType* countryCode)

Descripción	Permite modificar los datos relativos a la dirección del cliente presentes en la factura original.
Parámetros	String con una línea de dirección del cliente. String con el nombre de la ciudad del cliente. String con el código postal del cliente. String con el nombre del país del cliente. Código del país seleccionado de la lista de códigos correspondiente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método addTaxes(double base, double percent, *EnumTypeCode* typeCode)

Descripción	Añade un elemento *TaxSubtotal* a la factura original correspondiente a los datos de un impuesto. El cálculo de la cantidad resultante de impuestos es realizada por esta función a partir de la base y el porcentaje. Este resultado debe ser añadido al valor del elemento *TaxAmount*, de forma que el valor de este elemento al finalizar la ejecución de la función sea igual a la suma del valor antes de su ejecución más el resultado del producto de la base por el porcentaje. El valor del atributo *currencyID* toma el mismo valor fijado en el elemento *DocumentCurrencyCode*.
Parámetros	Cantidad base donde aplicar el impuesto. La cantidad debe estar expresada en la moneda indicada en el elemento. *DocumentCurrencyCode*. Porcentaje a aplicar. Código del impuesto. Código del impuesto escogido de la lista de códigos correspondiente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método addIVA(double base, double percent)

Descripción	Añade un elemento *TaxSubtotal* a la factura original correspondiente a los datos de un impuesto cuyo código es IVA. El cálculo de la cantidad resultante de impuestos es realizada por esta función a partir de la base y el porcentaje. El valor del atributo *currencyID* toma el mismo valor fijado en el elemento *DocumentCurrencyCode*.
Parámetros	Cantidad base donde aplicar el impuesto. La cantidad debe estar expresada en la moneda indicada en el elemento. *DocumentCurrencyCode*. Porcentaje a aplicar.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método removeTax(int i)

Descripción	Elimina el impuesto presente en la posición indicada (zero-based). Debe actualizar adecuadamente el elemento *TaxAmount*, de forma que se reste la cantidad correspondiente al impuesto eliminado.
Parámetros	Posición que ocupa el impuesto a eliminar.
Valor de retorno	Ninguno.
Excepciones	*IndexOutOfBoundsException* en el caso de indicar una posición que esté fuera de rango.

Método addInvoiceLine(*EnumTypeCode* unitCode, double quantity, double unitaryAmount, String itemDescription)

Descripción	Añade una nueva línea a la factura original. La función realizará el cálculo del precio total como producto de la cantidad por el precio unitario. El valor del atributo *currencyID* toma el mismo valor fijado en el elemento *DocumentCurrencyCode*.
Parámetros	Unidades en las que se expresa la cantidad escogidas de la lista de códigos correspondiente. Cantidad de items. Precio unitario. Descripción que aparecerá en la línea.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método removeInvoiceLine(int i)

Descripción	Elimina la línea presente en la posición indicada (zero-based). Debe actualizar adecuadamente los totales, de forma que se reste la cantidad correspondiente a la línea eliminada.
Parámetros	Posición que ocupa la línea a eliminar.
Valor de retorno	Ninguno.
Excepciones	*IndexOutOfBoundsException* en el caso de indicar una posición que esté fuera de rango.

Por otro lado, esta clase es la responsable de realizar los cálculos necesarios para actualizar el elemento *LegalMonetaryTotal*. El valor del atributo *currencyID* de los hijos de *LegalMonetaryTotal* es, al igual que con los otros elementos, igual al fijado en el elemento *DocumentCurrencyCode*.

Por otro lado, la clase tiene métodos para fijar el valor de los elementos propios de la factura rectificativa:

Método addNote(String note)

Descripción	Añade una nota a la factura. Cabe recordar que una factura rectificativa debe tener como mínimo una nota descriptiva del motivo de la rectificación.
Parámetros	Mensaje a añadir como nota.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

8.1.6 PaymentData

La clase *PaymentData* tiene como objetivo decorar la factura añadiéndole la información de pago. La clase actúa como superclase de las distintas implementaciones correspondientes a cada una de las distintas formas de pago.

Los métodos implementados en esta clase son:

Método setPaymentMeansID(String id)

Descripción	Permite fijar el valor del identificador de la forma de pago.
Parámetros	Identificador de la forma de pago.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setPaymentDueDate(Date paymentDueDate)

Descripción	Fija el valor del elemento *PaymentDueDate*.
Parámetros	Nuevo valor para el elemento *PaymentDueDate*.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

8.1.6.1 PaymentInCashData

Esta clase decoradora será la encargada de añadir la información necesaria para indicar el pago al contado. La clase de forma automática fijará el código de la forma de pago con el valor indicado en 2.4.4 .

8.1.6.2 PaymentByTransferData

Esta clase decoradora será la encargada de añadir la información necesaria para indicar el pago mediante transferencia. La clase de forma automática fijará el código de la forma de pago con el valor indicado en 2.4.4 . Asimismo, la clase implementa el siguiente método:

Método setPayeeFinancialAccountID(String id)

Descripción	Método para fijar el valor del número de cuenta del proveedor.
Parámetros	Identificador del número de cuenta del proveedor.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

8.1.6.3 PaymentByDirectDebitData

Esta clase decoradora será la encargada de añadir la información necesaria para indicar el pago mediante domiciliación. La clase de forma automática fijará el código de la forma de pago con el valor indicado en 2.4.4 . Asimismo, la clase implementa el siguiente método:

Método setPayerFinancialAccountID(String id)

Descripción	Método para fijar el valor del número de cuenta del cliente.
Parámetros	Identificador del número de cuenta del cliente.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

8.1.6.4 FactoringData

Esta clase decoradora será la encargada de añadir la información necesaria para indicar la cesión del cobro a un tercero. La clase de forma automática fijará el código de la factura y de la forma de pago con el valor indicado en 2.4.4 . Asimismo, la clase implementa los siguientes métodos:

Método setPayeePartyIdentification(String id, String name)

Descripción	Permite fijar los datos de identificación del beneficiario.
Parámetros	String con el identificador del beneficiario. String con la denominación del beneficiario.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setPayeePartyAddress(String addressLine, String cityName, String postalCode, String country)

Descripción	Permite fijar los datos relativos a la dirección del beneficiario.
Parámetros	String con una línea de dirección del beneficiario. String con el nombre de la ciudad del beneficiario. String con el código postal del beneficiario. String con el nombre del país del beneficiario.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método addNote(String note)

Descripción	Permite añadir una nota al elemento *PaymentTerms* de la factura. Cabe recordar que se debe poner una nota con la cláusula para la cesión de cobro.
Parámetros	String con la nota deseada.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setPayeeFinancialAccountID(String id)

Descripción	Método para fijar el valor del número de cuenta del beneficiario.
Parámetros	Identificador del número de cuenta del beneficiario.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

Método setPaymentID(String id)

Descripción	Método para fijar la referencia del pago.
Parámetros	Referencia del pago.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

8.2 Nuevos documentos

8.2.1 ApplicationResponse

La clase *ApplicationResponse* tiene como objetivo almacenar toda la información de un documento como el definido en el apartado 3.1 . Está compuesta por un conjunto de elementos CBC, CAC y extensiones. De todos los componentes del documento *ApplicationResponse* existen unos que son obligatorios y otros opcionales según lo definido por el *xml-schema*. De la misma manera para cada componente se establece una multiplicidad que puede variar entre una única instancia o múltiples apariciones (constituyendo una lista de elementos del mismo nombre).

La clase *ApplicationResponse* como todas las de la librería que representan un componente xml de UBL derivará del objeto *XmlObject* y extenderá sus métodos *validate* y *serialize.* Implementará métodos para acceder y modificar cualquier componente CBC, CAC o extensión que la constituye siguiendo estas reglas:

Para elementos obligatorios (CBCs o CACs) de multiplicidad uno:
Método get*ElementNameX*()
Método set*ElementNameX*(*ElementType* x)

Para elementos obligatorios (CAC) de multiplicidad mayor a uno:
Método get*ElementNameX*Array()
Método get*ElementNameX*ArrayAsList()
Método get*ElementNameX*Array(int i)
Método set*ElementNameX*Array(*ElementType[]* xx)
Método set*ElementNameX*Array(int i, *ElementType* x)
Método add*ElementNameX*Array(*ElementTypeX* x)
Método inset*ElementNameX*Array(int i, *ElementType* x)
Método remove*ElementNameX*Array(int i)
Método sizeOf*ElementNameX*Array(int i)

Para elementos opcionales (CBCs o CACs) de multiplicidad uno:
Método get*ElementNameX*()
Método set*ElementNameX*(*ElementType* x)
Método has*ElementNameX*()
Método remove*ElementNameX*()

Para elementos opcionales (CBCs, CACs o extensiones) de multiplicidad

mayor a uno:
Método has*ElementNameX*Array()
Método remove*ElementNameX*Array()

Todos los métodos anteriores tienen los mismos nombres y comportamiento que los especificados en el apartado 6.2 .

9 ESPECIFICACIÓN DE SERVICIOS

9.1 Especificación de clases asociadas a los servicios de serialización/deserialización

9.1.1 Serialización

El servicio de serialización proporciona los mecanismos para volcar la clase *Invoice* a su representación xml. Mediante los métodos del servicio de serialización es posible controlar tanto el destino del documento (un archivo, pantalla o un stream) como el formato de éste.

La clase *XmlInvoiceSerializer* implementará los siguientes métodos:

Método serialize(Invoice i, OutputStream os)

Descripción	Serializa el xml contenido en un objecto factura CCI UBL a un stream.
Parámetros	El objeto que contiene los datos de la factura y el stream sobre el que se realizará el volcado.
Valor de retorno	Ninguno.
Excepciones	Lanzará una *SerializationException* en caso de error.

Método serializeToStdout(Invoice i)

Descripción	Serializa el xml contenido en un objecto factura CCI UBL y lo muestra por la salida estándar.
Parámetros	El objeto que contiene los datos de la factura.
Valor de retorno	Ninguno.
Excepciones	Lanzará una *SerializationException* en caso de error.

Método serializeToFile(Invoice i, String filename)

Descripción	Serializa el xml contenido en un objecto factura CCI UBL a un fichero.
Parámetros	El objeto que contiene los datos de la factura.
Valor de retorno	Ninguno.
Excepciones	Lanzará una *SerializationException* en caso de error.

Método setSerializerOptions(XmlOptions opts)

Descripción	Fija los parámetros que controlan el formato de salida del serializador.
Parámetros	Contenedor de parámetros de formato.
Valor de retorno	Ninguno.
Excepciones	Ninguna.

El objeto *XmlOptions* contendrá los siguientes métodos para especificar el formato de la salida cuando se realice un volcado:

Método setSavePrettyPrint()

Descripción	El resultado de la serialización se volcará con saltos de linea para una lectura más fácil.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	Ninguno.

Método setSaveXmlDecl()

Descripción	El resultado de la serialización contendrá la declaración xml.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	Ninguno.

Si se quieren añadir nuevas opciones de formato a la serialización (tipo de indentación, declaración de prefijos de espacios de nombres, etc...) se deberán extender los objetos *XmlObject* y *XmlInvoiceSerializer*.

En los casos en que se serialicen facturas firmadas se debe prestar especial atención en no alterar el contenido de la factura en el proceso de serialización, ya que esto podría invalidar la firma digital. Se debe volcar la factura con el mismo formato de serialización que utiliza la herramienta de firma, por lo que es recomendable no serializar nunca la factura una vez firmada, sinó guardar el documento xml de la salida de la función de firma.

9.1.2 Deserialización

El servicio de deserialización tiene por objetivo la obtención de un objeto de la clase *Invoice* a partir de su representación xml. Los métodos de la clase facilitan la lectura del documento xml a partir de distintos orígenes.

La clase *InvoiceFactory* tendrá los siguientes métodos:

Método parse(String f)

Descripción	Lee el documento xml a partir de un fichero y produce un objeto del tipo *Invoice*.
Parámetros	Nombre del fichero del cual leerá.
Valor de retorno	*Invoice* correspondiente al xml leído.
Excepciones	Lanzará una *DeserializationException* en caso de error.

Método parse(InputStream s)

Descripción	Lee el documento xml a partir de un flujo y produce un objeto del tipo *Invoice*.
Parámetros	Flujo del cual leerá.
Valor de retorno	*Invoice* correspondiente al xml leído.
Excepciones	Lanzará una *DeserializationException* en caso de error.

9.2 Especificación de clases asociadas a los servicios de validación

Los servicios de validación tienen como objetivo comprobar la forma y el contenido de un documento xml generado o recibido. La comprobación se realiza contra documentos schema, schematron y genericodes. Mediante los schema se realizará la comprobación de la forma del xml mientras que mediante los schematron y los archivos genericodes permitirán comprobar su contenido.El detalle del proceso sistémico de validación queda descrito en la referencia [UBLVAL].

9.2.1 Comprobación de schema

La factura CCI UBL está especificada mediante un conjunto de schemas. Este conjunto determina los nombres, espacios de nombres, hijos y atributos de cada uno de los elementos de la factura. Con el servicio que se define se ofrece la posibilidad de validar que una factura esté acorde con lo especificado en los schemas.

Para la comprobación de una factura xml contra los schemas correspondientes se dispone de la clase *SchemaValidator* que ofrece el servicio. Esta clase ofrece los siguientes métodos para solicitar sus servicios:

Método addSchema(InputStream xsd)

Descripción	Permite añadir un schema contra el cual realizar la validación.
Parámetros	Flujo con el schema que forma parte del conjunto de schemas a utilizar para realizar la validación.
Valor de retorno	Ninguno.
Excepciones	*SchemaException* en el caso de que el schema no sea un documento válido.

Dado que los schemas son documentos fijos que previsiblemente no deben cambiar, debe ser posible especificarlos en la configuración de la API. Con esta finalidad, el constructor de la clase deberá cargar los ficheros que indique la configuración mediante una invocación al siguiente método.

Método loadSchemas()

Descripción	Carga los schemas indicados en la configuración de la API, si hay alguno.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	*SchemaException* en el caso de que alguno de los schemas indicados en la configuración de la API no sea un documento válido.

Método validate(Invoice i)

Descripción	Valida la factura contra los schemas previamente añadidos.
Parámetros	Factura a validar.
Valor de retorno	Booleano con un valor igual a cierto si la factura es válida o falso en caso contrario.
Excepciones	*SchemaException* en el caso de producirse un error de proceso durante el procesado.

Método getError()

Descripción	Permite obtener el conjunto de los errores de validación de la factura encontrados por el método validate, si se han dado.
Parámetros	Ninguno.
Valor de retorno	Lista de mensajes que describen los errores detectados. Mediante estos mensajes se puede conocer los motivos por los cuales la factura no es correcta.
Excepciones	Ninguna.

9.2.2 Comprobación de las reglas de negocio

Además de una comprobación estructural del documento XML mediante una validación contra schemas, resulta interesante poder realizar una comprobación del contenido de la factura. Esta comprobación puede ser realizada mediante el uso de documentos schematron y listas de códigos genericode. Para la realización de esta comprobación se dispone de la clase *BusinessRulesValidator*. A continuación se describen los métodos que deberá tener dicha clase.

9.2.2.1 Documentos Schematron

Mediante los documentos schematron es posible comprobar el contenido de los elementos que conforman la factura.

A diferencia del procesado de un schema, los documentos que especifican las reglas de validación debe ser procesados previamente. Por lo tanto, posteriormente a la adición de los documento schematron, deberán ser procesados adecuadamente. El procesado consiste en la aplicación de transformaciones XSLT. Los documentos que especifican las transformaciones XSLT previsiblemente serán siempre los mismos, por lo tanto se debe poder fijar los mismos en la configuración de la API.

Una vez los documentos schematron estén preparados, pueden ser aplicados a la factura xml a comprobar. El procesado consiste en aplicar a la factura una transformación XSLT que tiene como documento XSLT el resultado de procesar los documentos schematron.

Los métodos que dispone la clase de validación de reglas de negocio para la gestión de documentos schematron son los siguientes:

Método addSchematron(InputStream schm)

Descripción	Permite añadir un documento schematron contra el cual realizar la validación.
Parámetros	Flujo con el schematron a utilizar para realizar la validación.
Valor de retorno	Ninguno.
Excepciones	En el caso de que el schematron no sea un documento válido o se haya producido un error al prepararlo se producirá una excepción *SchematronException*.

Método loadSchematrons()

Descripción	Carga los schematron indicados en la configuración de la API, si hay alguno.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	*SchematronException* en el caso de que alguno de los schemantron indicados en la configuración de la API no sea un documento válido.

9.2.2.2 Documentos genericode

Mediante documentos genericode es posible especificar conjuntos de valores posibles para un elemento o atributo.

El procesado de los documentos genericode tiene una parte igual al procesado de documentos schematron. Los documentos genericode son transformados mediante una transformación XSLT para obtener un documento schematron con unas reglas que equivalen a las listas especificadas. Este documento se añade al conjunto de documentos schematron que conforman las reglas de negocio.

Los métodos que dispone la clase de validación de reglas de negocio para la gestión de documentos genericode son los siguientes:

Método addGenericode(InputStream gc)

Descripción	Permite añadir un documento genericode con una nueva lista de códigos.
Parámetros	Flujo con el documento genericode a utilizar para realizar la validación.
Valor de retorno	Ninguno.
Excepciones	En el caso de que el documento genericode no sea un documento válido o se haya producido un error al prepararlo se producirá una *GenericodeException*.

Método loadGenericodes()

Descripción	Carga los documentos genericode indicados en la configuración de la API, si hay alguno.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	*GenericodeException* en el caso de que alguno de los genericodes indicados en la configuración de la API no sea un documento válido.

9.2.2.3 Procesado

Las reglas de negocio especificadas mediante documentos genericode o schematron serán procesadas simultáneamente. Dada una factura, esta clase se encargará de validarla contra todas las reglas especificadas. Para ello, se implementarán los siguientes métodos:

Método prepare()

Descripción	Prepara los documentos schematron y genericode con el fin de obtener el documento XSLT usado para la validación.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	*SchematronException* si se produce algún error al preparar alguno de los schematron. *GenericodeException* sise produce algún error al preparar alguno de los genericode. *BusinessRulesException* si se produce algún error en el proceso de preparación no distinguible si es debido a un schematron o a un genericode.

Método validate(Invoice i)

Descripción	Valida la factura contra las reglas de negocio previamente procesadas.
Parámetros	Factura a validar.
Valor de retorno	Booleano con un valor igual a cierto si la factura es válida o falso en caso contrario.
Excepciones	*BusinesRulesException* en el caso de producirse un error durante el procesado.

Método getError()

Descripción	Permite obtener el conjunto de los errores de validación, si se han dado. Mediante estos mensajes se puede conocer los motivos por los cuales la factura no es correcta.
Parámetros	Ninguno.
Valor de retorno	Lista de mensajes que describen los errores detectados.
Excepciones	Ninguna.

9.3 Especificación de clases asociadas a los servicios de seguridad

Los servicios de seguridad que proporciona la librería son los de integridad, confidencialidad y autenticidad de origen. Con esta finalidad, la librería debe permitir la firma y/o cifrado de una factura electrónica. En este apartado se describen las clases e interfaces que ofrecerán estos servicios.

9.3.1 Servicio de firma

El proceso de generación y verificación de una firma electrónica puede ser llevado a cabo de distintas formas e incluso mediante distintos dispositivos. La librería debe quedar abierta a distintas implementaciones mediante distintos motores de firma. Por este motivo para le diseño de la arquitectura se ha utilizado el patrón Strategy [GOF]. De esta forma, cada cliente de la librería puede personalizarla para que se adapte a su motor de firma electrónica..

Para proporcionar el servicio de firma se define el interfaz *SignerEngine*. Este interfaz deberá ser implementado para cada uno de los motores de firma concretos que se desee utilizar. Las funciones definidas en el interfaz son las siguientes:

Método initSigner()

Descripción	El método será invocado para inicializar el motor de firma. La clase que implemente el interfaz deberá insertar el código adecuado, si es que necesita alguna inicialización.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	Puede lanzar una excepción de seguridad *SignatureException* en el caso de producirse algún error en la inicialización.

Método sign(Invoice i, SignParams p)

Descripción	Método para firmar la factura dada.
Parámetros	Factura a firmar. Parámetros destinados al motor de firma.
Valor de retorno	Array de bytes con la factura ya firmada.
Excepciones	Puede lanzar una excepción de seguridad *SignatureException* en el caso de producirse algún error en el proceso de firma.

Método verify(InputStream i, VerifyParams p)

Descripción	Método para verificar la firma de la factura dada.
Parámetros	Flujo con la factura que contiene la firma a verificar. Parámetros destinados al motor de firma.
Valor de retorno	Ninguno.
Excepciones	Puede lanzar una excepción de seguridad *SignatureException* en el caso de producirse algún error en el proceso de verificación.

Método finalizeSigner()

Descripción	El método será invocado para finalizar el motor de firma. La clase que implemente el interfaz deberá insertar el código adecuado, si es que necesita realizar algún proceso.
Parámetros	Ninguno
Valor de retorno	Ninguno.
Excepciones	Puede lanzar una excepción de seguridad *SignatureException* en el caso de producirse algún error en la finalización del motor de firma.

Así como existen distintos motores de firma, cada uno de ellos puede recibir parámetros de distinto tipo tanto para la generación como para la verificación. Dada la variedad en los parámetros que puede recibir un motor de firma, se define un interfaz común a todos para la generación y otro para la verificación. Los dos interfaces no tendrán ningún método, simplemente tienen como objetivo actuar como clase base. Un posible ejemplo de implementación de los parámetros sería:

```
class PKCS12Params implements SignParams{
 private InputStream pkcs12; //PKCS12 con la clave privada
 private String password; //Password para el acceso al PKCS12
 //Metodos para el acceso a los datos...
}
```

Ejemplo 11: Parámetros para la generación de una firma extrayendo el certificado y su clave asociada de un archivo PKCS#12

```
class CertParams implements VerifyParams{
 private List certChain; //Lista con los certificados que forman la
          //cadena de certificacion
 //Metodos para el acceso a los datos...
}
```

Ejemplo 12: Parámetros para la verificación de una firma digital a partir de una cadena de certificación hasta el certificado del firmante.

Para cada motor de firma se deberá generar unas clases de parámetros que implementen los interfaces descritos. El siguiente diagrama de clases muestra un ejemplo de jerarquía para un motor de firma.

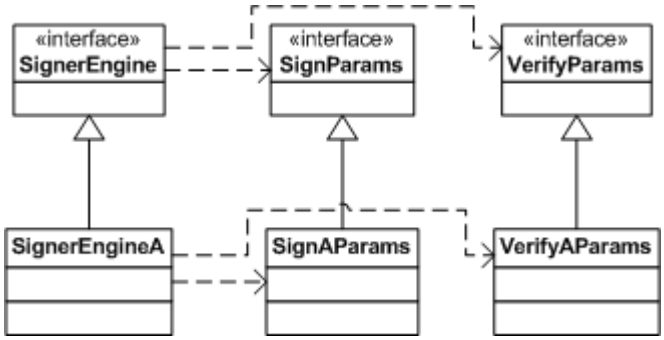


Figura 13: Diagrama de clases para el servicio de firma

9.3.2 Servicio de cifrado

El servicio de cifrado permite garantizar la confidencialidad de la factura generada. Solamente los destinos autorizados podrán obtener el contenido. Al igual que la firma electrónica, el cifrado y descifrado de un documento puede ser realizado de distintas formas o con distintos dispositivos. Por este motivo para le diseño de la arquitectura se ha utilizado el patrón Strategy [GOF]. De esta forma, se definirá el interfaz *CipherEngine* similar al caso de la firma que deberá ser implementado para cada motor de cifrado deseado.

Las funciones definidas en el interfaz son las siguientes:

Metodo initCipher()

Descripción	El método será invocado para inicializar el motor de cifrado. La clase que implemente el interfaz deberá insertar el código adecuado, si es que necesita alguna inicialización.
Parámetros	Ninguno.
Valor de retorno	Ninguno.
Excepciones	Puede lanzar una excepción de seguridad *CipherException* en el caso de producirse algún error en la inicialización.

Método encrypt(Invoice i, CipherParams p)

Descripción	Método para cifrar la factura dada.
Parámetros	Factura a cifrar. Parámetros destinados al motor de cifrado.
Valor de retorno	Array de bytes con el resultado de cifrar la factura.
Excepciones	Puede lanzar una excepción de seguridad *CipherException* en el caso de producirse algún error en el proceso de cifrado.

Método decrypt(byte[] b, DechipherParams p)

Descripción	Método para descifrar una factura previamente cifrada.
Parámetros	Array de bytes que contiene la factura cifrada. Parámetros destinados al motor de cifrado.
Valor de retorno	Array de bytes con el resultado de descifrar la factura. Pueden ser utilizados para construir un objeto factura.
Excepciones	Puede lanzar una excepción de seguridad *CipherException* en el caso de producirse algún error en el proceso de descifrado.

Método finalizeCipher()

Descripción	El método será invocado para finalizar el motor de cifrado. La clase que implemente el interfaz deberá insertar el código adecuado, si es que necesita realizar algún proceso.
Parámetros	Ninguno
Valor de retorno	Ninguno.
Excepciones	Puede lanzar una excepción de seguridad *CipherException* en el caso de producirse algún error en la finalización del motor de cifrado.

Así como existen distintos motores de cifrado, cada uno de ellos puede recibir parámetros de distinto tipo tanto para el cifrado como para el descifrado. Dada la variedad en los parámetros que puede recibir un motor de cifrado, se define un interfaz común a todos para el proceso de cifrado y otro para el proceso de descifrado. Los interfaces no tendrán ningún método, simplemente tienen como objetivo actuar como clase base. Un posible ejemplo de implementación de los parámetros sería:

```
class PKCS12Params implements DecipherParams{
 private InputStream pkcs12; //PKCS12 con la clave privada para
                        //descifrar
 private String password; //Password para el acceso al PKCS12   //Metodos para el acceso
a los datos...
}
```

Ejemplo 13: Parámetros para el descifrado de una factura utilizando la clave privada de un certificado contenida en un archivo PKCS#12.

```
class CertParams implements CipherParams{
 private Certificate cert; //Certificado del destinatario usado para
            //cifrar la factura
 //Metodos para el acceso a los datos...
}
```

Ejemplo 14: Parámetros para el cifrado de una firma extrayendo la clave pública del destinatario de un certificado digital.

Para cada motor de cifrado se deberán generar unas clases de parámetros que implementen los interfaces descritos. El siguiente diagrama de clases muestra un ejemplo de jerarquía para un motor de cifrado.

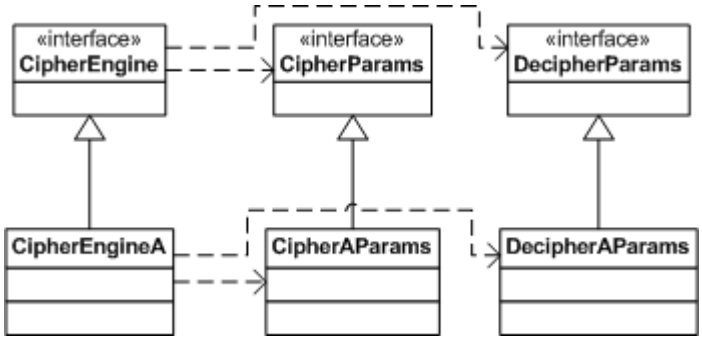


Figura 14: Diagrama de clases para el servicio de cifrado

9.4 Especificación de clases asociadas a los servicios de transformación

Las clases descritas en el presente apartado tienen como objetivo la prestación de servicios de transformaciones. Se dispone de un interfaz *Transformer* a implementar para cada transformación deseada.

Los métodos especificados en el interfaz son los siguientes:

Método transform(InputStream transform, Invoice i)

Descripción	Método para aplicar una transformación una factura dada.
Parámetros	Flujo con los parámetros que especifica la transformación. Factura de entrada a transformar.
Valor de retorno	Array de bytes resultantes después de aplicar la transformación.
Excepciones	Puede lanzar una excepción de transformación en el caso de producirse algún error en el proceso de transformación.

Un ejemplo de transformación es XSLT. Las transformaciones XSLT tienen por objetivo la obtención de todos o algunos de los datos de la factura y su traslado a otro formato. Pueden ser usadas para una mejor visualización o para obtener un formato de documento que pueda entender otro aplicativo. Para llevar a cabo transformaciones XSLT será necesario implementar el interfaz *Transformer*. La clase debe ser capaz de procesar una transformación XSLT cualquiera.

10 REFERENCIAS

[XSD] http://www.w3.org/TR/xmlschema-2/#built-in-datatypes

[UBLCCI] Guía de Implementación Factura CCI UBL, versión 2.0, Marzo 2007.

[GOF] Gamma H., Helm R., Johnson R., Vlissides J., "Patrones de diseño", Addison Wesley.

[UBLVAL] UBL Methodology for Code List and Value Validation, working draft 0.8, 23 April 2007.