

Funcionalidades

Funcionalidades y características

Anonymous · 08/10/2026

Funcionalidades y características

Estructura general

El proyecto se organiza en **paquetes** (Package). Cada paquete agrupa **componentes** (Component) relacionados con un dominio. Cada componente expone su funcionalidad a través de **workers** (Worker), que son los servicios principales. Los workers pueden tener estrategias intercambiables para distintas variantes de una misma operación.

El núcleo de LibreDTE tiene solo el paquete **Billing**, con los siguientes componentes:

Componente	Descripción
<code>billing.trading_parties</code>	Entidades participantes: emisor, receptor, contribuyente, mandatario.
<code>billing.identifier</code>	Autorización de folios (CAF).
<code>billing.document</code>	Ciclo de vida completo de los DTE: parseo, normalización, construcción, validación, renderizado, envío.
<code>billing.integration</code>	Cliente para los servicios web del SII (API SOAP).
<code>billing.exchange</code>	Intercambio de documentos entre partes vía email o SII.
<code>billing.book</code>	Libros y registros tributarios.
<code>billing.ownership_transfer</code>	Cesión electrónica (factoring/AEC).

Servicios externos usados por LibreDTE Core

XML (`derafu/xml`)

- ✓ `XmlDocument`: extiende `DomDocument` con funcionalidades adicionales, incluyendo `query()`.
- ✓ `XmlEncoder` y `XmlDecoder`: conversión entre arreglo PHP y string XML.
- ✓ `XmlValidator`: validación de esquema XSD.

Certificados digitales (`derafu/certificate`)

1. ✓ `Certificate`: representa un certificado digital con clave pública, privada y metadatos (RUN, nombre, período de validez).
2. ✓ `CertificateFaker`: genera certificados falsos para pruebas.

Firma electrónica (`derafu/signature`)

1. ✓ `SignatureGenerator`, `SignatureValidator`, `XmlSignatureNode`: firma y validación XML DSIG estándar, pero preparada para lo que el SII requiere.

Funcionalidades de LibreDTE Core

Partes comerciales (`billing.trading_parties`)

1. ✓ `Contribuyente`, `Emisor`, `Receptor`, `Mandatario`, `PersonaNatural`: entidades que representan a los participantes de un DTE.
2. ✓ `ContribuyenteFactory`, `EmisorFactory`, `ReceptorFactory`, `MandatarioFactory`: fábricas para construir cada entidad.
3. ✓ `MandatarioManagerWorker`: gestión del ciclo de vida de mandatarios.
4. ✓ `AutorizacionDte`: representación de la autorización de un emisor para emitir DTE.

Autorización de folios — CAF (`billing.identifier`)

1. ✓ `Caf`: representa un archivo CAF con clave pública, privada, tipo de documento, rango de folios y período de validez.
2. ✓ `CafFaker` y `CafFakerWorker`: generación de CAF falsos para pruebas.
3. ✓ `CafLoaderWorker`: carga de archivos CAF desde disco u otras fuentes.
4. ✓ `CafValidatorWorker`: validación de un CAF (vigencia, rango, firma).
5. ✓ `CafProviderWorker`: proveedor de CAF que combina carga y validación.
6. ✓ `CafBag`: colección de CAF para múltiples tipos de documento.

Documentos tributarios — DTE (`billing.document`)

1. ✓ `TipoDocumento` y `TipoDocumentoFactory`: representa un tipo de DTE con sus atributos (código, nombre, categoría, operación, tag XML, etc.).
2. ✓ Entidades de documento (`FacturaAfecta`, `FacturaExenta`, `FacturaCompra`, `FacturaExportacion`, `BoletaAfecta`, `BoletaExenta`, `NotaCredito`, `NotaDebito`, `NotaCreditoExportacion`, `NotaDebitoExportacion`, `GuiaDespacho`, `LiquidacionFactura`): heredan de `AbstractDocument`.
3. ✓ `SobreEnvio`: representa el XML `EnvioDTE` o `EnvioBOLETA` con métodos para firmar y validar.
4. ✓ `ImpuestoAdicionalRetencion`: soporte para impuestos adicionales y retenciones (excepto combustibles).
5. ✓ `ParserWorker`: parseo de datos de entrada en formato JSON, XML, YAML y formulario HTML.

6. ✓ **NormalizerWorker**: normalización de datos por tipo de documento, con estrategias específicas para cada uno y traits de normalización (detalle, descuentos/recargos, IVA, exportación, transporte, boletas, impuestos adicionales).
7. ✓ **BuilderWorker**: construcción del XML del DTE (timbre, firma, validación de esquema), con estrategia por tipo de documento.
8. ✓ **SanitizerWorker**: sanitización de los datos del DTE por tipo de documento.
9. ✓ **ValidatorWorker**: validación de datos, esquema XSD y firma electrónica del DTE por tipo de documento.
10. ✓ **RendererWorker** con **EstandarRendererStrategy**: generación de PDF en formato estándar (hoja carta).
11. ✓ **LoaderWorker**: carga de un DTE desde XML.
12. ✓ **DispatcherWorker**: despacho de documentos al siguiente paso del proceso.
13. ✓ **BatchProcessorWorker**: procesamiento en lote de múltiples documentos.
14. ✓ **DocumentBagManagerWorker**: gestión del **DocumentBag** (contenedor de datos de un DTE en proceso).

Intercambio de documentos (**billing.exchange**)

1. ✓ **SenderWorker**: envío de documentos con los siguientes handlers y estrategias:
 - ✓ **EmailSenderHandler** + **SmtpSenderStrategy**: envío por correo electrónico vía SMTP.
 - ✓ **SiiSenderHandler** + **DteSenderStrategy**: envío de DTE al SII.
 - ✓ **SiiSenderHandler** + **BoletaSenderStrategy**: envío de boletas al SII.
2. ✓ **ReceiverWorker**: recepción de documentos con:
 - ✓ **EmailReceiverHandler** + **ImapReceiverStrategy**: recepción de documentos desde correo electrónico vía IMAP.
3. ✓ **DocumentResponseWorker**: generación y validación de XML de respuesta al intercambio de DTE:
 - ✓ **buildEnvioRecibos()**: genera el XML **EnvioRecibos** con recibos de mercaderías (acción ERM). Cada Recibo se firma individualmente con ID **LibreDTE_T{tipo}F{folio}** y el **SetRecibos** se firma con ID **LibreDTE_SetDteRecibidos**.
 - ✓ **buildRespuestaEnvio()**: genera el XML **RespuestaDTE** con **RecepcionEnvio** (acuse del sobre) o **ResultadoDTE** (resultado comercial por DTE). Se firma con ID **LibreDTE_ResultadoEnvio**.
 - ✓ **validateSchema()**: valida el esquema XSD del documento, retorna **XmlDocumentInterface**.
 - ✓ **validateSignature()**: valida todas las firmas electrónicas, retorna **array<SignatureValidationResultInterface>**.

Libros tributarios (**billing.book**)

1. ✓ **TipoLibro**, **TipoOperacion**: enumeraciones de tipos de libro y operación.
2. ✓ **LibroComprasVentas**, **LibroGuias**, **LibroBoletas**, **ResumenVentasDiarias**: entidades que heredan de **AbstractBook**.
3. ✓ **LoaderWorker** con estrategias **ArrayLoaderStrategy** por tipo de libro.

4. ✓ `BuilderWorker` con estrategias `BuilderStrategy` por tipo de libro.
5. ✓ `ValidatorWorker`: validación de esquema XSD (`validateSchema()`) y firma electrónica (`validateSignature()`).
6. □ `RendererWorker`: generación del PDF de los libros.
7. ✓ `BookBag`: contenedor de datos del libro en proceso.

Cesión electrónica / Factoring (`billing.ownership_transfer`)

1. ✓ `AbstractOwnershipTransferDocument`: clase base para las entidades del AEC, con `getXml()`, `getId()`, `getSchema()` y `getSignatureNamespace()` (retorna `null` para usar namespace `xmldsig` estándar).
2. ✓ Entidades `DteCedido`, `Cesion`, `Aec`: heredan de `AbstractOwnershipTransferDocument`. Cada una corresponde a una capa de firma del AEC.
3. ✓ `AecBag`: contenedor de datos que acepta `DocumentInterface` (primera cesión) o `Aec` existente (re-cesión). El `seq` es auto-calculado si se omite.
4. ✓ `AecWorker`: generación y validación del Archivo Electrónico de Cesión (AEC):
 - ✓ `build()`: construye el AEC completo con tres capas de firma. Soporta primera cesión (desde un `DocumentInterface`) y re-cesión (desde un `Aec` existente, añadiendo una nueva `Cesion` al mismo `DTECedido`). El SII permite hasta 40 cesiones sobre el mismo DTE.
 - ✓ `validateSchema()`: valida el esquema XSD del AEC, retorna `XmlDocumentInterface`.
 - ✓ `validateSignature()`: valida todas las firmas del AEC (`DTECedido`, `Cesion(es)` y `AEC`), retorna `array<SignatureValidationResultInterface>`.

Integración con SII (`billing.integration`)

1. ✓ `SiiDteWorker`: cliente para los servicios web del SII con las siguientes operaciones:
 - ✓ `authenticate()`: obtención del token de sesión.
 - ✓ `sendXmlDocument()`: envío de un XML al SII, retorna Track ID.
 - ✓ `checkXmlDocumentSentStatus()`: consulta del estado del envío sin detalles de errores.
 - ✓ `requestXmlDocumentSentStatusByEmail()`: solicita al SII el correo con estado del envío y detalles de errores.
 - ✓ `validateDocument()`: verifica que un DTE es legalmente válido (folio, total, etc.).
 - ✓ `validateDocumentSignature()`: verificación avanzada de la firma del DTE en el SII.
 - ✓ `consumeWebservice()`: consumo genérico de un servicio SOAP del SII.
2. ✓ `SiiRcvWorker`: cliente para el Registro de Compra y Venta (RCV) del SII con las siguientes operaciones:
 - ✓ `submitDocumentAcceptance()`: ingresa aceptación o reclamo de un DTE (acciones: ERM, ACD, RCD, RFP, RFT).
 - ✓ `listDocumentEvents()`: lista el historial de eventos de un DTE.
 - ✓ `checkDocumentAssignability()`: consulta si un DTE puede ser cedido (factoring/AEC).
 - ✓ `getDocumentSiiReceptionDate()`: consulta la fecha en que el SII recibió un DTE.
3. ✓ `SiiRtcWorker`: cliente para el Registro de Transferencias de Créditos (RTC) del SII:
 - ✓ `sendAec()`: envía un AEC al RTC. Recibe `SiiRequestInterface`, `XmlDocumentInterface` (el AEC), RUT del cedente y email de contacto. Retorna Track ID. Usa endpoint

`cgi_rtc/RTC/RTCAnotEnvio.cgi` (distinto al de DTE).

Última actualización el 08/10/2026