UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SERVICIO DE FIRMA DIGITAL EN PÁGINAS PARA EL DEPÓSITO CENTRAL DE VALORES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

DANIEL EDUARDO ESTÉVEZ GARAY

PROFESOR GUÍA:
MARÍA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:
ALEJANDRO HEVIA ANGULO
ALEX BORQUEZ GRIMALDI

SANTIAGO DE CHILE
ENERO 2013

Resumen

El Depósito Central de Valores (DCV) es una Sociedad Anónima regulada por la Superintendencia de Valores y Seguros, está facultada para recibir en depósito valores de oferta pública y facilitar así las operaciones de transferencia de dichos valores entre los depositantes. Además, procesa y registra electrónicamente las operaciones de transferencia efectuadas en las bolsas de valores y en el mercado extrabursátil. También coordina y suministra la información necesaria para la liquidación financiera de las operaciones.

Por las facultades que implementa, el DCV provee un servicio para firmar digitalmente el contenido de las páginas web que publica. Con esto asegura autenticación, integridad y no repudio de las operaciones.

Actualmente la tecnología que se utiliza para implementar el servicio de firma digital en página web esta obsoleta. El objetivo de esta memoria es actualizar éste por uno que no sea obsoleto y sea igual o más eficiente que la solución actual.

Se desarrolló una aplicación orientada al servicio y utilizando estilo de arquitectura REST. Además se aplicó metodología iterativa identificando de forma temprana los actores internos del DCV que proveen requerimientos sobre el desarrollo.

La solución desarrollada es más eficiente que la solución anterior y es compatible con los navegadores y tecnologías actuales. Se resuelve un problema típico de seguridad en plataforma empresarial de forma simple y con una arquitectura bien definida. Actualmente la aplicación está en proceso de instalación en ambiente de producción siendo certificada por las áreas de calidad interna del DCV.

Como resultado adicional se obtiene una arquitectura de firma de páginas web la cual puede ser utilizada como base de solución de esquema de firma digital en futuros proyectos.

A mi familia.

Agradecimientos

Quiero agradecer a mi familia que me ha apoyado siempre en todos mis objetivos. Espero con mis logros retribuir con orgullo lo que me han entregado.

Agradezco a los profesores de la comisión por sus acertados consejos. Al profesor Alejandro por los detalles que elevaron el nivel de este trabajo. Al profesor Alex por su preocupación e interés constante.

En especial quiero agradecer a la profesora Cecilia que supo impulsarme y darme sensatez en los momentos críticos. Sin su ayuda este trabajo hubiera sido imposible.

A los funcionarios y académicos del departamento de ciencias de la computación que hacen que cada visita al departamento sea un agrado.

Al Depósito Central de Valores por permitir desarrollar mi memoria al interior de la empresa. Al área de Arquitectura a la cual pertenezco. Agradezco especialmente a Marcelo por su comprensión.

Finalmente quiero agradecer a mis amigos que comprendieron que la ausencia no significa despreocupación y siempre estuvieron ahí para darme una palabra de apoyo.

Tabla de contenido

Introducción

El Depósito Central de Valores S.A. (DCV) es una Sociedad Anónima constituida de acuerdo a la Ley 18.876 [7], su reglamento y a las instrucciones impartidas por la Superintendencia de Valores y Seguros (SVS). El DCV está facultado para recibir en depósito valores de oferta pública y facilitar así las operaciones de transferencia de dichos valores entre los depositantes, de acuerdo a los procedimientos contemplados en la citada Ley.

El DCV es una entidad, que en el cumplimiento de su objeto, procesa y registra electrónicamente las operaciones de transferencia efectuadas en las bolsas de valores y en el mercado extrabursátil. Adicionalmente coordina y suministra la información necesaria para la liquidación financiera de las operaciones.

Durante el año 2000, el DCV lanzó al mercado el servicio de Administración de Registros de Accionistas a través de su filial DCV Registros S.A. Las empresas que contratan este servicio delegan la administración del pago de dividendos de sus acciones. Lo anterior, permitió que las Sociedades Anónimas se libraran de un trabajo especializado y ajeno a su giro, y así reintegrar capacidades productivas a sus respectivas áreas de negocios.

Los clientes del DCV se dividen en dos:

- Depositante: Son todas aquellas entidades que tienen contrato de depósito vigente con el DCV y que han sido definidas por la Ley 18.876, articulo N°2 como Depositante.
- Emisores: Instituciones autorizadas por la Superintendencia de Valores y Seguros o por la Superintendencia de Bancos e Instituciones Financieras, para emitir valores de oferta pública, el Banco Central de Chile, el Estado y las demás empresas o entidades cuyos valores emitidos puedan ser objeto de depósito de conformidad a la Ley.

Los depositantes según la Ley 18.876 pueden ser:

- El Fisco de Chile, a través de la Tesorería General de la República.
- La Corporación de Fomento de la Producción.
- Los agentes de valores.
- Los corredores de bolsa.
- Las bolsas de valores.
- Los bancos, sociedades financieras y demás instituciones autorizadas para operar en Chile, de acuerdo a la Ley General de Bancos e Instituciones Financieras.
- Las administradoras de fondos mutuos.

- Las administradoras de fondos de pensiones.
- Las compañías de seguros y de reaseguros establecidos en Chile.
- Los fondos de inversión.
- Los fondos de inversión de capital extranjero.
- Las demás que autorice la empresa.

Cabe destacar que el 80 % de la propiedad y control del DCV la manejan sus propios depositantes participando como accionistas.

Problema

Para el DCV, que los servicios cumplan exigencias operativas de seguridad y tecnológicas es lo más importante, ya que sus servicios son críticos en el mercado chileno.

El cumplir con estas exigencias de seguridad es atribuir a las operaciones críticas de los servicios las siguientes características:

1. Autenticación
2. No repudio
3. Operación como transacción

Autenticación se refiere a verificar que las operaciones sean realizadas por quienes dicen ser. No repudio significa establecer un mecanismo en que el cliente no pueda objetar una operación ya enviada. Que la operación se realice como una transacción quiere decir que esta debe cumplir con las características de atomicidad, consistencia, aislamiento e integridad; estas cuatro características se denominan con el acrónimo ACID [10].

El no repudio de las operaciones es muy importante dado que es una defensa legal fuerte en caso de que un cliente objete alguna operación realizada en los sistemas del DCV.

Como solución a este problema se establece el uso de firma digital sobre las páginas de confirmación de operaciones críticas, estableciendo que los clientes firmen digitalmente la confirmación y la envíen al interior de una transacción.

Una firma digital es el análogo criptográfico de las firmas escritas presentando mayor nivel de seguridad que éstas. En Chile desde el año 2002 que es reconocida legalmente como mecanismo para certificar contratos, notificar documentos y autenticar corporaciones [3]. Una firma digital se compone por una entidad que firma y un conjunto de entidades verificadoras de la firma generada.

La entidad que firma comienza con la ejecución de un algoritmo de generación de claves. Este algoritmo genera una clave pública y una clave secreta o privada. La entidad distribuye la clave pública a las entidades verificadoras y, utilizando la clave secreta, aplica un algoritmo de firmado sobre los documentos. Las entidades verificadoras, teniendo el documento firmado, aplican un algoritmo de verificación utilizando la clave pública de la entidad que ha firmado

validando autenticación, integridad y no repudio sobre el contenido de este. [20]

Los servicios del DCV se ofrecen mediante una plataforma web por lo que la solución actual establece que la firma digital la debe efectuar el cliente utilizando su clave privada sobre la página de confirmación de la operación crítica y enviar la confirmación de la operación firmada con su clave privada. Los sistemas del DCV verifican la firma digital recibida mediante la clave pública del cliente antes de concretar la operación crítica.

Actualmente para implementar la solución de firma digital en páginas se utiliza la tecnología CAPICOM de Microsoft [14] que permite mediante un plugin ActiveX seleccionar la clave privada depositada en un almacén de claves en la máquina del cliente para luego realizar la firma e incrustar esta en el envío de confirmación de operación. En el sistema DCV existe un servicio que captura esta firma digital, la almacena y la relaciona con la operación efectuada. Estos módulos en conjunto se constituyen como un servicio de firma digital en páginas.

El problema a solucionar en esta memoria nace a partir del hecho que CAPICOM está obsoleto y Microsoft no le da soporte. Esto produce problemas de incompatibilidad con la mayor parte de los navegadores del mercado. El área de Mantención reporta que el servicio de firma digital en páginas no funciona en los navegadores actuales Internet Explorer 9, Chrome 18 y Firefox 11.

Es por esto que se hace perentorio actualizar el servicio de firma digital en páginas para que éste sea compatible con los navegadores actuales cumpliendo con las características de autenticación, no repudio y transaccionalidad que contempla la solución actual. El actualizar el servicio de firma digital en páginas puede implicar modificar la validación y almacenamiento del documento firmado en el servidor.

Se hace necesario investigar las normativas legales en Chile relacionadas al uso de firma digital en aplicaciones web con el fin de evitar resquicios legales que puedan generar un riesgo al DCV.

No se descarta la compra de tecnologías que implementen parte de la solución de firma digital. El evaluar tecnologías existentes en el mercado debe ser una etapa al interior del proyecto.

Relevancia

Las prestaciones del servicio de firma digital en páginas son clave dentro de los sistemas del DCV, ya que sin este no se podrían efectuar operaciones de transferencia o liquidación de valores, que es el propósito del DCV. Esto impactaría fuertemente al mercado nacional y de forma leve al internacional en relación a inversiones de custodia extranjera desde y hacia Chile.

La actualización de este servicio en los sistemas del DCV es un proyecto que se quiere lograr al mediano plazo, por lo que tomar esta tarea al interior de una memoria es una oportunidad interesante en cuanto a que se dejará evidencia de una solución que la mayoría

de los sistemas financieros actuales está obligado a implementar.

Alternativas analizadas y opción de solución elegida

Las alternativas varían en torno a la tecnología utilizada para desarrollar el servicio de firma digital en páginas. Si se llegan a modificar aplicaciones existentes en el sistema del DCV, se deben seguir los estándares actuales de desarrollo que existen en el DCV, los cuales en resumen exigen arquitectura orientada al servicio y que los servicios se implementen mediante estilo de arquitectura REST. Entonces, evaluando la implementación de firma por el lado del cliente, se tienen las siguientes alternativas:

- Microsoft .NET Framework
- Applet Java
- Plugin navegador
- ADSS GO>Sign Applet
- CRYPTBOT web-Sign

.NET es la opción recomendada por Microsoft para el reemplazo de CAPICOM, pero señala que no es una solución completa, ya que requiere desarrollar un control ActiveX para completar la solución[13]. Java ofrece su módulo criptográfico el cual, junto a implementaciones de algoritmos de seguridad, constituye una solución completa para firmado digital. La solución de plugin implica un desarrollo específico para cada navegador y desarrollar un componente binario común instalado en la máquina del cliente que implemente las funcionalidades de firma. Para comparar las diferentes alternativas se consideran los siguientes puntos:

- Portabilidad: Qué tanto varía la implementación de la solución entre distintos ambientes (navegadores y máquinas).
- Complejidad: Dificultad para implementar solución.
- Documentación: Bibliografía de apoyo existente.
- Costo: Valor monetario de solución según escala empresa.

Se presenta en la tabla 1 una comparativa con cada punto. Hecha esta se opta por desarrollar el servicio de firma digital en páginas mediante Applet de Java. Las otras soluciones sin costo presentan mayor complejidad y menor portabilidad no cumpliendo con los requisitos. De las soluciones comerciales sólo ADSS Go>Sign Applet cumple con los requisitos, pero es una herramienta que se centra en firmar formato PDF lo que agrega una sobrecarga al proceso de firmado, ya que exige conversión de la página HTML a PDF.

	Portabilidad	Complejidad	Documentación	Costo
Microsoft .NET Framework	Baja. Funciona sólo sobre MS Internet Explorer.	Media. Se desarrollan controles ActiveX y componentes .NET mediante MS Visual Studio.	Alta. Microsoft provee vía online documentación de sus módulos.	N/A
Applet Java	Alta. Dado que se distribuye como byte code se porta de un sistema a otro fácilmente responsabilizando al cliente que instale la máquina virtual Java.	Media. Se debe utilizar el módulo Java Security	Alta. Oracle provee vía online documentación de módulos.	N/A
Plugin navegador	Media. Los plugins varían entre navegadores. Los binarios deben implementarse para arquitecturas de 32 y 64 bits.	Alta. Desarrollar binarios para cada sistema y plugins para cada navegador existente en el mercado.	Media. Para desarrollo de binarios existe completa documentación. Documentación respecto al desarrollo de plugins no es clara.	N/A
ADSS GO>Sign Applet	Alta. La solución está implementada mediante Applet Java.	Media. La solución de firma en páginas se basa en convertir el formulario en PDF y luego firmarlo.	Alta. Como solución comercial presenta documentación completa.	Medio.
CRYPTBOT web-Sign	Baja. Solución compatible sólo con MS Internet Explorer.	Baja. Solución simple sobre navegador MS IE.	Media. No presenta documentación consistente en sitio web.	Bajo.

Tabla 1: Tabla comparativa alternativas de implementación

Objetivos

Migrar el actual servicio de firma digital en páginas por un Applet Java.

Objetivos específicos

Desarrollar un Applet Java para que cliente firme la página web donde confirma una operación crítica. Desarrollar prototipos para validar compatibilidad de Applet con los distintos navegadores. Generar la documentación del Applet para los clientes que lo utilicen. La arquitectura de la aplicación debe ser orientada al servicio y utilizar estilo REST. Capturar

requerimientos importantes desde las áreas involucradas del DCV. Otro objetivo es validar que la migración cumpla con los objetivos planteados.

Metodología

El trabajo de esta memoria se inscribe como proyecto al interior del DCV por lo que las horas de trabajo en la memoria serán reportadas como horas de trabajo real de proyecto. En específico el proyecto se inscribe al interior del proyecto llamado *Mejoras Arquitectura Aplicativa* como fase 5 llamada *Servicio de firma digital en páginas.*

El trabajo se divide lógicamente en tres partes: Análisis, Desarrollo y Validación. A continuación se explicarán las actividades a realizar en cada etapa.

En etapa de análisis se estudiarán los aspectos legales en profundidad con el fin de extraer posibles requerimientos. Luego se estudiarán los algoritmos y formatos existentes para firma digital y se crearán pruebas de concepto. Se estudiarán los componentes existentes que se deberán modificar y se definirá un alcance para las modificaciones. Como entregables de esta etapa se tendrán:

- Casos de Uso
- Documento de arquitectura de software
- Plan de pruebas

Luego, en etapa de implementación, se comenzará con tareas de configuración. En específico se configurará el ambiente de desarrollo, para luego dedicarse de lleno en el desarrollo de los componentes lógicos identificados que son:

- Acceso a almacén de claves
- Algoritmo de firma
- Algoritmo de verificación
- Applet Java
- Receptor de operaciones
- Filtro web de firma digital
- Actualización de componentes existentes

Al término de esta etapa se escribirá un manual de uso del servicio de firma digital en páginas, para ser entregado a los clientes y se escribirá la primera versión del documento de memoria.

En la etapa de validación se ejecutará el plan de pruebas y se efectuarán las correcciones correspondientes. Finalmente se escribirá la versión final de la memoria.

Estructura de la memoria

El trabajo comienza con un marco conceptual donde se presentan ámbitos técnicos y legales que afectan a la solución a implementar. Este capítulo se centra en preparar al lector en antecedentes que le sean útiles para el entendimiento del problema y de la solución a implementar. Parte importante de este capítulo es mostrar la arquitectura de la aplicación existente, el buen entendimiento de ésta es fundamental para comprender el desarrollo posterior.

Luego se muestra el diseño e implementación de la solución, donde se detalla el desarrollo de cada componente que forma parte de la aplicación. Comienza con una sección de requerimientos capturados desde las distintas áreas del DCV, la ubicación de éstos en este capitulo responde a que son requerimientos que se descubrieron durante el desarrollo de la solución. La estrategia utilizada para presentar el diseño e implementación de la solución es mostrar la arquitectura a bajo nivel de la aplicación, soportándose en su separación por componentes para permitir su buen entendimiento.

Se continua presentando los resultados obtenidos en cuanto a atributos de calidad y cumplimiento de los objetivos. Se presenta una breve historia de usuario para comprender lo que el cliente final verá en pantalla.

Finalmente se presentan las conclusiones en función de los resultados obtenidos y objetivos planteados.

Capítulo 1

Marco Conceptual

En este capítulo se analizan los antecedentes teóricos y prácticos que se manejan respecto a la solución actual de firma digital en la plataforma web existente en el DCV. Se contrasta la correctitud del esquema actual y se evalúan las decisiones existentes. Como resultado se buscan resolver los puntos críticos, definiendo para cada uno el alcance sobre la solución a lograr. Se detalla la arquitectura de la solución actual de firma digital en páginas web que es fundamental para comprender la solución desarrollada en el siguiente capítulo. Al final de este capítulo se realiza un estudio de las arquitecturas orientadas al servicio y estilo de

arquitectura REST con el fin de estandarizar la implementación de un servicio REST en cuanto a tecnologías y paradigmas a utilizar.

1.1. Firma Digital

Un esquema de firma digital se divide en tres algoritmos probabilísticos que se resuelven en tiempo polinomial:

1. Generador de clave pública y de la clave privada
2. Firmado de documento utilizando clave secreta
3. Verificación de documentos firmados mediante clave secreta

La generación de claves es responsabilidad de las entidades certificadoras las cuales distribuyen a sus clientes estas claves en formato de certificado digital.

Los clientes del DCV almacenan sus certificados en tokens de seguridad marca Aladdin modelo eToken PRO. Estos soportan algoritmos de seguridad RSA 1024-bit / 2048-bit, DES, 3DES y SHA1.

Actualmente la solución de firma digital en páginas utiliza firma digital RSA-SHA1. Esta consiste en aplicar el algoritmo de firma RSA y luego sobre los bytes resultantes aplica el algoritmo de hashing SHA1.

Al contenido a firmar siempre se le agrega la fecha de la operación a modo de *timestamp* con el fin de asegurar no repudio de la operación.

1.2. Normativa legal en Chile respecto a firma digital

En Chile existe la Ley 19.799 SOBRE DOCUMENTOS ELECTRONICOS, FIRMA ELECTRONICA Y SERVICIOS DE CERTIFICACION DE DICHA FIRMA que define los usos de esta firma para que judicialmente se pueda velar por su correcto uso. La primera versión de esta data del 12 de Abril del año 2002 y fue modificada por la Ley 20.217 MINISTERIO DE ECONOMIA; FOMENTO Y RECONSTRUCCION; SUBSECRETARIA DE ECONOMIA; FOMENTO Y RECONSTRUCCION el 12 de Noviembre del año 2007. La modificación a la ley es importante, ya que le da sentido judicial al concepto *Fecha electrónica* la cual se entiende como la fecha en que se realiza la firma electrónica.

Con este antecedente se promueve agregar al interior del contenido firmado la fecha en que se efectúa esta. En los sistemas computacionales esto se denomina timestamp.

Hasta la fecha ningún reglamento define cómo técnicamente crear un timestamp, ya que sólo hacen referencia a literatura relacionada en el reglamento asociado a la ley 19.799.

Actualmente se encuentra aprobado el decreto 154 que modifica el reglamento de la ley

19.799. Fue aprobado el 11 de Agosto del 2012 y comienza a regir 180 días luego de su publicación. La modificación se refiere a las normas técnicas para la prestación del servicio por los Prestadores del servicio de certificación. Es por esto que no afecta a cómo se realiza actualmente la firma digital en el DCV y legalmente se puede utilizar cualquier esquema de firma digital válido.

La última modificación aprobada por el congreso tiene por objetivo aumentar el uso de la firma digital en Chile dando mayor poder a las entidades certificadoras de Chile. En Chile las entidades certificadoras más importantes son E-CertChile de la Cámara de Comercio de Santiago, Acepta y Certinet.

1.3. Proyectos similares en Chile

Actualmente existe sólo un proyecto similar al ejecutado en esta memoria que es el desarrollado por la empresa Agile llamado Agile Signer Web Integrator. Este es un Applet Java que implementa firma de archivos PDF que están en la web y se conecta directamente a los tokens de seguridad utilizando estándar PKCS#11.

El estándar PKCS#11[11] fue desarrollado en los laboratorios RSA. La versión vigente data de Junio del año 2004. Esta define una API denominada Criptoki basada en orientación de objetos y acceso a recursos de forma concurrente. Implementando esta API uno puede operar con tokens de seguridad que cumplan el estándar. El eToken PRO de Aladdin cumple con este estándar.

1.4. Historia sistema DCV

El sistema que soporta los servicios que el DCV provee a sus clientes inicialmente fue desarrollado bajo las siguientes plataformas:

- Cobol Open VMS: Procesamiento de lógica de negocio batch
- Sybase: Manejo de transacciones en modelo relacional
- Aplicación Powerbuilder: Interfaz administrativa interna
- Aplicación Visual Basic: Interfaz clientes

Esto cambió el año 2002 cuando bajo el proyecto de *Migración Web* la mayor parte de los servicios fueron desarrollados sobre la plataforma IBM Websphere el cual es un contenedor de Servlets Java. Luego durante el año 2007 bajo petición del directorio del DCV se exige el uso de certificados digitales, por lo que se modifican los servicios para que estos puedan ser visibles sólo por clientes que tengan inscritos en el sistema sus certificados (claves públicas). Es en este momento que se solicita la posibilidad de que los clientes puedan firmar digitalmente en sus navegadores web.

1.5. Firma digital en navegador web

El problema en el año 2007 era crear una firma digital en el navegador web mediante una interfaz amigable sobre el sistema operativo Microsoft Windows.

Para solucionar este problema se utilizó CAPICOM de Microsoft. Este componente se instala como plugin ActiveX en los navegadores Microsoft Internet Explorer y disponibiliza una API en la página HTML para firmar digitalmente.

Su uso a través del tiempo ha pasado a no ser una solución al problema, debido a que es una herramienta obsoleta y difícil de configurar en los navegadores Microsoft Internet Explorer, esto según registros históricos de la Mesa de Ayuda del DCV. Además este no funciona en navegadores distintos a Internet Explorer debido a que la API que expone para su uso utiliza lenguaje Visual Basic.

CAPICOM se complementa directamente con el almacén de claves de Windows utilizando nativamente bibliotecas comunes de Visual Basic de Windows. Estas bibliotecas potencialmente podrían cambiar en una futura versión de Microsoft Windows dejando completamente inutilizable la funcionalidad de firma digital en el navegador, por lo que el DCV además de exigir un navegador especifico, comenzaría a exigir un Sistema Operativo obsoleto.

1.6. Uso de navegadores por clientes del DCV

Estudios del área de Marketing indica que los clientes en el último trimestre del año 2012 hicieron las siguientes visitas con cada navegador.

Navegador	Número de visitas	Porcentaje de visitas
Internet Explorer	61.113	69,20 %
Firefox	22.468	25,44 %
Chrome	3.531	4,00 %
Safari	522	0,59 %
IE with Chrome Frame	405	0,46 %

Se visualiza que los clientes utilizan mayormente Internet Explorer, por esto el área de Marketing realiza un segundo estudio en el cual identifica la versión de Internet Explorer utilizada en el último trimestre del año 2012, teniendo los siguientes resultados.

Navegador	Número de visitas	Porcentaje de visitas
8.0	9.318	52,80 %
7.0	6.585	37,31 %
6.0	1.230	6,97 %
9.0	508	2,88 %
10.0	7	0,04 %

Existe una situación de uso de navegadores obsoletos por parte de los clientes que utilizan Internet Explorer. Sólo el 1 % de estos clientes utiliza la versión vigente de Internet Explorer.

1.7. Aplicación existente

El propósito de firma digital en el navegador es que se firme lo que el cliente visualiza en pantalla. De esta forma se establece marcar las páginas de confirmación de operaciones criticas como candidatas a firmar.

Por esto la aplicación existente se basa en interceptar los llamados sobre páginas de confirmación de operaciones crítica y no permitir su ejecución a no ser que exista una firma digital ya enviada. Esto lo consigue redireccionando cualquier requerimiento HTTP que se realice sobre estas páginas hacia una página template. Esta página template contiene por defecto la inicialización del plugin ActiveX de CAPICOM e incluye al interior de esta la página de confirmación.

Los componentes que conforman a la aplicación existente consisten en:

1. Filtro firma digital
2. Servicio de firma digital
3. Template de firma digital

Se pueden revisar los componentes en la figura 1.1. A continuación se explica cada uno de los componentes.

Filtro firma digital

Es un filtro de servlet web. Este recibe el requerimiento HTTP original y redirecciona hacia el sitio que contiene el template de firma digital. Junto con la redirección se le entrega al template la URL de la página de confirmación para que la incluya al interior de esta. Para presentarle una vista de buen aspecto al cliente, generalmente se prepara un JSP distinto al de la página de confirmación. Este JSP se denomina como página de firma y es una réplica del JSP de la página de confirmación con estilos modificados.

La redirección HTTP que se realiza incluye la totalidad de los parámetros y atributos del requerimiento HTTP original.

En caso de que el requerimiento HTTP venga con parámetros de firma digital, estos se envían al servicio de validación de firma, y en caso de ser válida la firma, se continúa con el procesamiento normal de la página de confirmación.

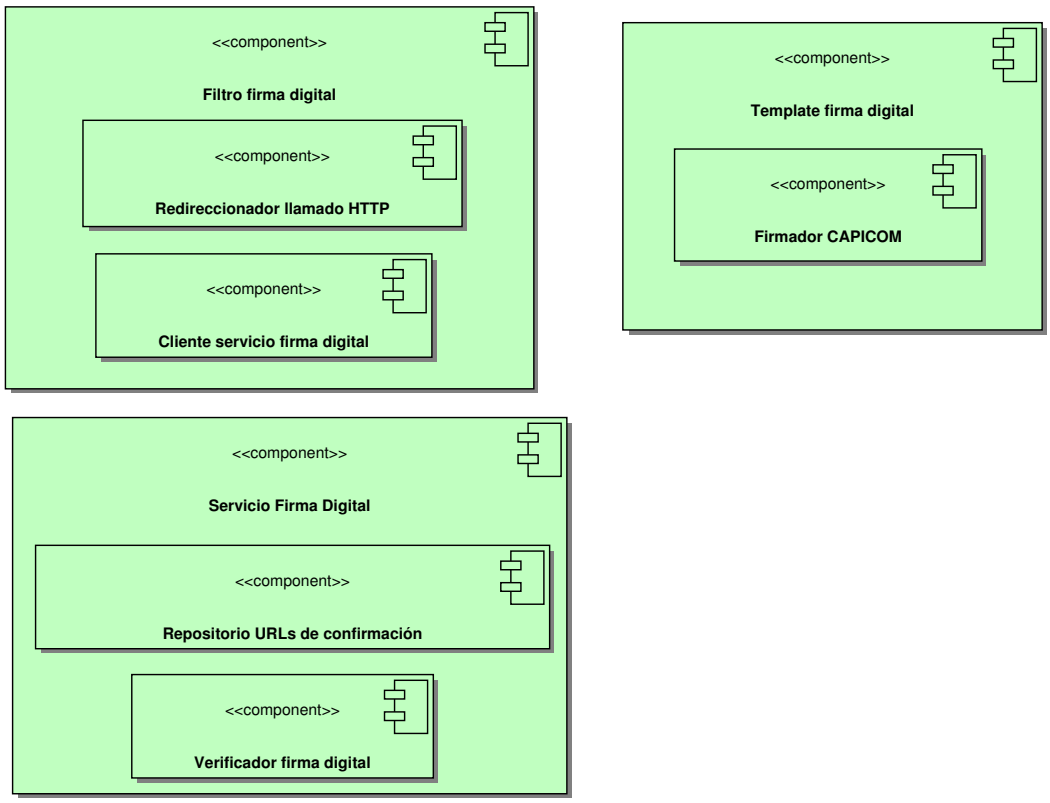

Figura 1.1: Componentes aplicación existente

Template firma digital

Este template contiene el código que inicializa el plugin ActiveX CAPICOM para firmar digitalmente el contenido de la página web desplegada al interior de esta. En caso de que se detecte la no presencia del plugin sobre el navegador web, entonces el navegador redirige de forma automática al sitio de descarga e instalación de este.

Luego de que el cliente firma digitalmente el contenido, se produce un nuevo requerimiento HTTP a la URL de la página de confirmación, con los parámetros originales de requerimiento inicial más los parámetros de la firma digital recién creada.

Servicio de firma digital

Este servicio provee el método para validar la firma digital realizada. Obtiene la clave pública asociada al cliente y valida la firma digital enviada. Es un servicio desarrollado bajo arquitectura de componentes Enterprise Java Beans (EJB), lo cual asegura transaccionalidad y escalabilidad.

1.8. Arquitectura orientada al servicio

El desarrollo de aplicaciones empresariales con arquitecturas orientadas al servicio ha ido en aumento en las últimas décadas. Se refleja esto en la existencia de una gran cantidad de estándares y paradigmas para la creación de estos servicios, predominando principalmente los basados en llamados remotos de procedimiento (RPC) como SOAP. Sin embargo existe una alta crítica sobre la arquitectura orientada al servicio en cuanto a que es muy compleja y rígida [12] .

La introducción de REST: Representational State Transfer, estilo de arquitectura creado por Roy Thomas Fielding [9] , trae consigo un desarrollo orientado a los recursos, filosofía que produce en sí un aumento en la eficiencia que históricamente los servicios basados en RPC no tienen [6]. Mientras los servicios basados en SOAP implementan una arquitectura interesada en la transmisión de los XML codificados mediante HTTP, REST se encarga de administrar los recursos. Por esto utiliza el poder que los métodos HTTP en sí tienen, pudiendo administrar operaciones del patrón CRUD de forma simple, mediante un subconjunto de métodos HTTP: GET, POST, PUT y DELETE.

Una comparación de tiempos de respuesta se encuentra en el trabajo de Mulligan [15] donde encuentra que para todos los casos los servicios REST tienen menos latencia que los basados en SOAP. En algunos casos la diferencia es más del 80 %.

Sobre los servicios RPC se construyen estándares para asegurar transaccionalidad, confiabilidad (totalidad) y seguridad entre otros. Estos estándares se denotan por WS-* en la literatura y si bien son capas necesarias en una arquitectura de aplicaciones empresariales, estas agregan más complejidad al servicio RPC implementado.

REST como estilo de arquitectura no implementa estándar alguno, se desarrolla un estándar para su implementación denominado JAX-RS (definido en JSR 311) el cual asegura la implementación de servicios REST que cumplen:

- Identificación del recurso mediante URI: La URI del servicio es autodescriptiva del mismo.
- Interface uniforme: Se utilizan sólo 4 métodos HTTP PUT, GET, POST, y DELETE, donde cada uno refleja una operación del patrón CRUD.
- Mensajes auto-descriptivos: Así como las URI se autodescriben los mensajes también.
- Interacciones sin estados: Los llamados no realizan cambios en el estado del servicio.

Para implementación de transaccionalidad sobre servicios REST, como punto de partida de análisis se puede utilizar el estudio de Da Silva [5] el cual en base a los estándares WS-* establece la necesidad de que el servicio contenga un servicio de compensación asociado (rollback). Concluye que mediante esta simple práctica y manejo de retornos HTTP específicos se puede abordar esta problemática.

Sobre seguridad REST es compatible con cualquier sistema en cuanto se respete la ausencia de estados; se comprueba esto en el estudio de Field et al. [8] donde extendiendo Spring Security agrega seguridad a servicios REST. Como requerimiento no funcional los servicios

REST pueden cumplir con los niveles de seguridad establecidos, donde la complejidad varía por cada solución.

1.9. Servicio REST

Se estandariza la implementación de un servicio con estilo de arquitectura REST en el DCV. Para esto se evalúan dos alternativas de alto nivel en cuanto a tecnología:

- JAX-RS: Java API for RESTful Services es el estándar vigente para el desarrollo de servicios REST. En caso de seleccionar esta opción se debe utilizar la implementación soportada por IBM Websphere 7.
- Spring Framework 3: El módulo web de Spring Framework permite el desarrollo de aplicaciones MVC (Modelo-Vista-Controlador). Utilizando este módulo se pueden construir servicios RESTful.

IBM Websphere 7 incluye soporte a JAX-RS incluyendo en su distribución las bibliotecas Java de Apache Wink. Es una implementación de JAX-RS versión 1.1. Actualmente se encuentra en trabajo la versión 2.0 [1] la cual identifica las siguientes falencias en la actual definición:

- MVC: No se implementa un patrón MVC[1] predominante. La nueva versión debe considerar la implementación actual sólo como el Controlador e integrarse con tecnologías de vista del mercado.
- Validación de parámetros: existe validación básica de los parámetros entrantes y no existe estándar de respuesta cuando la validación es fallida
- Inyección de dependencias: El estándar JSR 330: Dependency Injection for Java [18] para definir inyección de dependencias que promueve la reutilización, mantenibilidad y testing fue creado después de JAX-RS. Por esto el manejo de dependencias debe ser manejado mediante métodos tradicionales como constructores, factories o service locators.
- Requerimientos asíncronos: Servlet versión 3.0 permite requerimientos HTTP asíncronos los cuales no son soportados por JAX-RS
- Negociación de contenido: JAX-RS no implementa mecanismo por el lado del servidor para definir qué contenido responder en función del MIME Type requerido.

Se encuentra que Spring Framework 3 resuelve cada uno de estos puntos de la siguiente forma:

- MVC: Provee un módulo MVC especializado.
- Validación de parámetros: Es compatible con JSR-303 Bean Validation API [17].
- Inyección de dependencias: Spring Framework se centra en los patrones Inversión del Control e Inyección de dependencias. Se resume en la filosofía que es el framework quien controla la dependencia entre componentes inyectando la dependencia correspondiente

[1] Modelo-Vista-Controlador

en los componentes que conforman la aplicación. Por esto promueve que cada componente se relacione con otros mediante sus interfaces. Esto según los patrones de la GoF[2] es asumir un patrón Strategy para cada componente implementado.

- Requerimientos asíncronos: Spring Framework es compatible con Servlet 3 en su versión 3.2 en adelante
- Negociación de contenido: El módulo MVC implementa una capa denominada *resolutor de vista* el cual se encarga de gestionar a nivel de servidor el contenido a entregar al cliente HTTP.

Por esta comparación y la integración con tecnologías existentes en el mercado que presenta se opta por utilizar Spring Framework para el desarrollo de servicios REST en esta aplicación.

Capítulo 2

Diseño e implementación

En este capítulo se muestra cómo se diseñó e implementó la solución. Para esto se presentan primero los requerimientos más importantes obtenidos durante el proceso de desarrollo iterativo de la solución. Luego se presenta el proceso de firma digital desarrollado, mostrando un proceso de firma digital completo. Se presenta el proceso de desarrollo de software que se tuvo que cumplir. Finalmente se separa la solución en componentes lógicos y se presenta su implementación a bajo nivel.

2.1. Requerimientos

A continuación se presenta un extracto de los requerimientos más importantes de la solución a desarrollar. La mayoría de estos provienen de los estándares y normativas de las áreas de Arquitectura, Riesgo y Operaciones del DCV. Además de las peticiones del área de Servicio que provee la visión del cliente.

- Plataforma: La solución debe ser construida sobre la plataforma productiva vigente. Esta corresponde a IBM Websphere 7, IBM Websphere MQ Server 7 y Sybase 12.5. La

[2]Gang of Four

base de datos Sybase está en proceso de actualización a su versión 15, por lo que se debe asegurar que las sentencias SQL a utilizar sean compatibles con la nueva versión.

- Ausencia de UPDATES: En la jerga DCV la "Ausencia de UPDATES" se refiere a que la nueva aplicación no debe provocar que se realicen operaciones de actualización sobre la base de datos fuera de sistema como parte de su funcionamiento.
- Confidencialidad de la información: El contenido a firmar así como la firma deben transitar por un canal seguro (HTTPS) y ser tratados como información sensible. Además implica que se deben proteger las tablas de la base de datos donde se almacene la información.
- Mantener vistas: Las vistas y la forma en que se realiza la firma digital en las páginas se debe conservar.
- Mantener servicio antiguo: Como medida de respaldo se establece dejar el servicio antiguo de firma digital en páginas y que el nuevo servicio se pueda activar/desactivar sin tener que realizar una instalación en ambiente de producción para esto. Para esto se solicita que se active/desactive a nivel de empresa utilizando un parámetro del requerimiento HTTP llamado **codEmpresa**; este representa el código de la empresa.
- Documento a firmar debe ser HTML: Dado que ya existe un visualizador de firmas como traza de auditoría como HTML, se solicita que el documento a firmar sea texto HTML que se pueda insertar en un cuerpo HTML existente para su muestra.
- Intervención de filtro de firma digital no debe crear nuevas dependencias: Intervenir el filtro de firma digital existente no debe crear nuevas dependencias a componentes que estén fuera de la JVM de Java. Esto debido a que el filtro como biblioteca Java se instala como biblioteca de servidor, por lo que es visible por todas las aplicaciones instaladas en este. Existe el riesgo de que una dependencia externa agregada a nivel de servidor afecte a las aplicaciones existentes y limite los desarrollos futuros.
- Máximo de errores por operación: En caso de que una operación de firma presente más de 3 errores ésta se debe anular y marcar como fallada.
- Firma digital de Applet Java: Como parte de la construcción del Applet Java se requiere que éste sea firmado digitalmente por el certificado digital DCV. Esto se denomina *Code Signing* y lo que se busca es que al momento en que se ejecute el Applet en la máquina del cliente, se verifique que el Applet fue efectivamente desarrollado por el DCV y no existió intervención de terceros.

Como requerimientos adicionales se deben resolver los siguiente problemas preexistentes:

- Error en esquema de validación de firma: La firma digital no se valida mediante la clave pública que está almacenada localmente en el DCV, sólo se valida que el hash generado a partir de la firma sea válido. Esto es un error detectado en el sistema original.
- Ausencia de mantenedores asociados a firma digital: No existen mantenedores asociados al proceso de firma, por lo que no se pueden visualizar y validar las firmas ya efectuadas.
- Falta de protección de página que efectúa lógica de negocio: La página que realiza la lógica de negocio que requiere firma digital no está asociada a la página de confirmación que se firma. Por esto, si se realiza el requerimiento fuera de la página de confirmación, entonces se efectúa la lógica de negocio sin existir una firma digital asociada.

2.2. Proceso de firma digital desarrollado

En un proceso de firma digital en páginas se ven involucradas dos URLs:

1. URL o página de confirmación: Despliega la página que contiene la información de la operación a realizar y disponibiliza al cliente los controles para continuar con la operación.
2. URL o página de acción: Es la URL que recibe el requerimiento "confirmado" de la operación y que ejecuta la lógica de negocio que implica la operación en curso.

A este par de URLs se les asocia además un conjunto de nombres de parámetros especiales que definen como única a la operación de firma. Mediante estos nombres se busca, ya sea en la página HTML de confirmación o en el requerimiento sobre la URL de acción, su valor con el fin de calcular un hash interno de seguridad denominado *hash de verificación*.

Como ya existe un filtro servlet web configurado en las aplicaciones, que despliegan URLs que requieren firma digital, se opta por modificar este. Cabe destacar que la biblioteca Java que contiene las clases de este filtro, es una biblioteca común expuesta en el classpath del contenedor web. Por esto actualizar este servicio no implica modificar cada uno de los servicios existentes que requieren de firma digital en sus páginas. Sólo implica reemplazar la biblioteca y reiniciar el contenedor web.

Previo a la ejecución del proceso de firma, el filtro verifica mediante el código de empresa si el servicio está activo o no. El código de empresa se recibe como parámetro del requerimiento HTTP llamado **codEmpresa**.

Cada vez que se inicia un proceso de firma digital se asigna un *identificador único de operación* el cual es un valor numérico que en combinación con el *hash de verificación* permite identificar una operación de firma digital en el servicio de firma digital en páginas. Este identificador único se entrega al cliente en el código fuente HTML de las páginas de confirmación y acción, al contrario del hash de verificación que se maneja de forma interna.

2.2.1. Proceso de firma digital

A continuación se explican los pasos de un proceso de firma digital en páginas:

1. Cliente solicita mediante requerimiento HTTP una URL de confirmación.
2. Filtro servlet web de firma intercepta el requerimiento e identifica que no existe una firma digital válida asociado al requerimiento.
3. Filtro resuelve internamente el contenido HTTP a mostrar y lo modifica agregando el Applet Java al final del cuerpo HTTP de la respuesta y eliminando los controles HTTP originales que permiten en envío del requerimiento a la URL de acción.
4. Filtro a partir del cuerpo HTTP de respuesta original prepara el documento a firmar y calcula el hash de verificación. Con el documento a firmar y el hash de verificación realiza un llamado REST interno al servicio de firma e inicia una operación de firma

digital en páginas con lo cual el servicio le retorna un identificador único de operación. Además persiste la fecha de la operación de firma digital.

5. Filtro agrega como información de inicialización del Applet el identificador único de operación y retorna la página HTTP modificada.
6. Cliente inicializa Applet en su navegador. Mediante un llamado REST obtiene el documento a firmar asociado al identificador único de operación.
7. Cliente mediante un control mostrado por el Applet, selecciona un certificado digital de su almacén de claves y el Applet firma digitalmente el documento.
8. Applet mediante un llamado REST envía el documento firmado y el identificador único de operación.
9. Servicio de firma mediante el identificador único obtiene el documento a firmar. Desde la sesión HTTP obtiene el usuario principal y mediante este obtiene su clave pública almacenada. Con la clave pública, el documento a firmar y la firma digital recibida verifica la validez de esta.
10. Servicio almacena la firma válida recibida y responde de forma afirmativa al Applet que ejecuta el cliente en su navegador.
11. Applet al recibir respuesta afirmativa realiza el requerimiento HTTP original sobre la URL de confirmación agregando el identificador único de la operación al requerimiento.
12. Filtro de firma intercepta el llamado a la URL de confirmación y al verificar que existe una firma digital válida asociado al identificador único de operación recibido, procede a mostrar la página HTTP de confirmación original con el identificador único de operación y la fecha de la operación de firma como parámetros adicionales.
13. Cliente visualiza página de confirmación y confirma operación.
14. Filtro de firma intercepta llamado sobre URL de acción. A partir de los datos recibidos en el requerimiento calcula el hash de verificación y junto al identificador único de operación de firma recibido, consulta la existencia de una firma digital válida.
15. Filtro verifica existencia y permite ejecución de lógica de negocio asociada a la página de acción.

En caso de existir error en cualquier parte del proceso el filtro de firma marca la operación como fallida informando al cliente la situación de excepción. Esto incluye la ejecución fallida de la lógica de negocio, ya que el filtro interviene la URL de acción.

2.3. Proceso de desarrollo de Software DCV

El proceso de desarrollo existente en el DCV es uno basado en RUP [19] el cual implica un desarrollo iterativo en capas de la aplicación. El estado de una aplicación se define mediante una separación de ambientes bien definida.

Los ambientes son:

- Desarrollo: Es donde se efectúa el desarrollo de la aplicación.

- Integración: En este se realizan las pruebas integradas de la aplicación. Se evalúa el comportamiento de esta con el resto del sistema DCV.
- Aceptación: En este se realizan pruebas de negocio sobre la aplicación.
- Producción: Es el ambiente donde operan los clientes del DCV.

Cabe destacar que los ambientes previos a producción son similares a este en cuanto a plataforma.

En la aplicación desarrollada en esta memoria intervienen las siguientes áreas que aportan según su especialización:

- Desarrollo y Arquitectura: Define la arquitectura de la aplicación y la implementa. Utiliza el ambiente de desarrollo y promueve su instalación en ambiente de Integración.[1]
- Calidad: Realiza la certificación de la aplicación y valida su paso entre los ambientes de Aceptación y Producción.
- Operaciones: Instala y administra la aplicación en los distintos ambientes.
- Mantención: Corrige y mejora la aplicación en función de incidentes declarados por los clientes de la aplicación.

Luego de que es promovida la aplicación al ambiente de Producción por el área de Calidad, se debe realizar un traspaso de conocimiento al área de Mantención. Esta consiste en reuniones y entrega de documentación que estos aprueban previo a la instalación en Producción.

2.4. Estados operación de firma digital

Los estados que puede tener una operación de firma digital según la figura 2.1 son:

- Creada: Operación iniciada de forma correcta
- Validada: Operación firmada por el cliente de forma correcta
- Confirmada: Operación firmada ha sido notificada al cliente
- Ejecutada: Lógica de negocio de operación firmada ha sido ejecutada de forma correcta
- Fallada: Operación anulada debido a que ha superado el número máximo de errores permitidos
- Cancelada: Operación cancelada por cliente que inició operación. Sólo puede ser cancelada una operación en estado Creada.

[1]Para esta aplicación la definición e implementación fue realizada en su totalidad por el autor de la memoria.

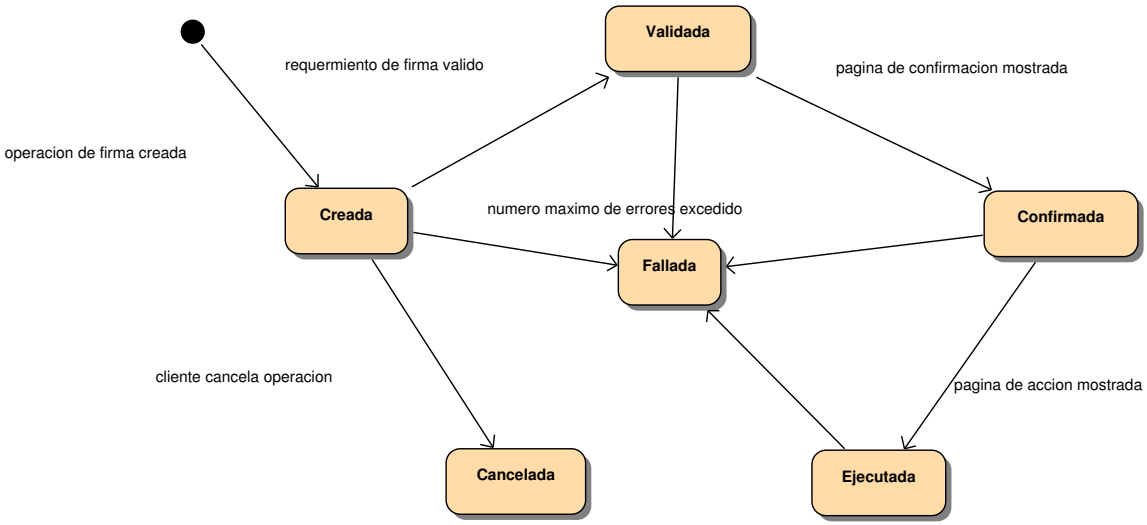

Figura 2.1: Estados operación firma digital

2.5. Componentes desarrollados

La aplicación construida se puede separar en los siguientes componentes:

- Filtro firma digital
- Componentes comunes
- Servicio Firma Digital
- Applet Firma Digital

Se pueden revisar los componentes en la figura 2.2.

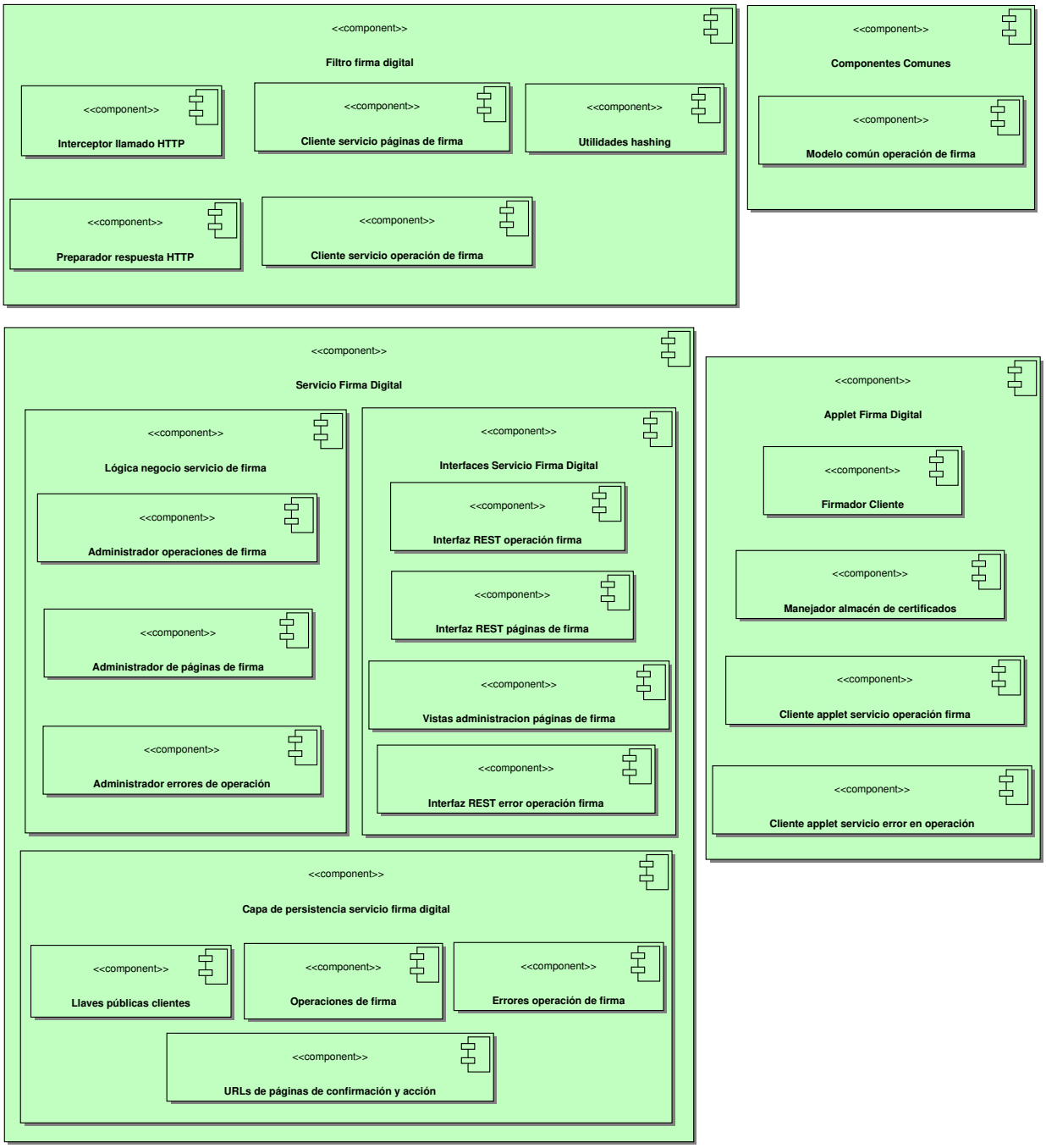

Figura 2.2: Componentes aplicación nueva

2.6. Filtro Firma Digital

El filtro de firma digital está compuesto por los siguientes componentes:

- Interceptor llamado HTTP
- Cliente servicio páginas de firma
- Cliente servicio operación de firma
- Utilidades hashing
- Preparador respuesta HTTP

Cabe recordar que por requerimiento, las clases que componen este componente no pueden utilizar bibliotecas externas debido a la ubicación común que poseen. Estos componentes se resuelven en las clases de la figura 2.3.

2.6.1. Interceptor llamado HTTP

El interceptor de llamados se resuelve utilizando la clase FiltroFirmaDigital la cual extiende a la clase de Java que representa un filtro servlet web. De esta forma esta clase puede intervenir en la cadena de procesamiento normal de una aplicación web.

Verifica si el servicio está activo extrayendo el código de empresa desde el requerimiento HTTP y consultando al servicio de firma digital. Luego accede al requerimiento HTTP y extrae la URL a la cual se pretende acceder. En caso de que identifique una URL de confirmación o acción orquesta las acciones necesarias según el estado de la operación.

2.6.2. Cliente servicio páginas de firma

Este componente se resuelve en la clase RESTSignService. Este cliente mediante peticiones de estilo REST se conecta al servicio de firma digital y permite realizar las siguiente operaciones:

- Mediante el código de empresa del cliente, obtener si el servicio de firma digital está activo. Esto permite conocer si se debe ejecutar la lógica de firma digital nueva o no.
- Mediante la URL, conocer si corresponde a una página de confirmación o acción.
- Mediante la URL de acción o confirmación, obtener el conjunto de nombres de los parámetros que hacen única la operación para calcular el hash de verificación.

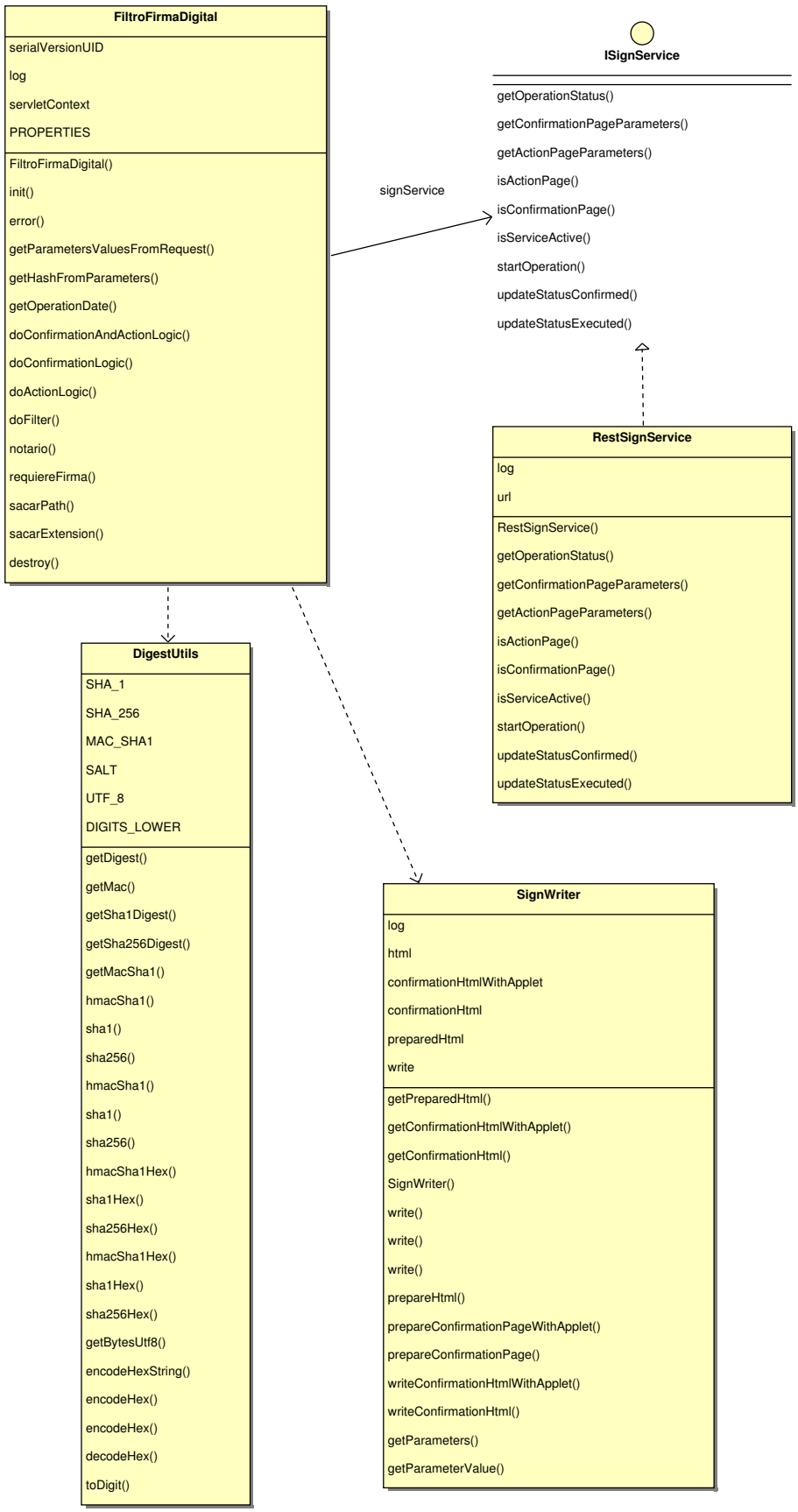

Figura 2.3: Clases Filtro Firma Digital

2.6.3. Cliente servicio operación de firma

Este componente se resuelve en la clase RESTSignService y permite realizar peticiones al estilo REST al servicio de operaciones de firma digital que constituyen las siguientes operaciones:

- Comenzar operación de firma enviando el contenido a firmar y el hash de verificación calculado a partir de los parámetros especiales
- Obtener estado de la operación mediante el identificador único de operación y el hash de verificación
- Actualizar el estado de una operación de firma a confirmado
- Actualizar el estado de una operación de firma a ejecutado

2.6.4. Utilidades hashing

Este componente se resuelve en la clase DigestUtils que contiene el método para calcular el hash de verificación mediante algoritmo HMAC-SHA1. Esta clase contiene al interior suyo la clave que utiliza para el algoritmo HMAC.

El hash de verificación es calculado por el filtro de firma digital en dos ocasiones:

1. Al identificar el comienzo de una operación de firma itera sobre el conjunto de nombres de parámetros especiales y busca el valor asociado a estos parámetros en el HTML de la página de confirmación. Luego concatena los valores de estos parámetros y calcula el hash de verificación.
2. Al interceptar un llamado sobre una URL de acción desde el requerimiento HTTP obtiene el valor de los parámetros especiales. Concatena los valores obtenido y calcula el hash de verificación.

En caso de que se trate de la misma operación de firma digital los hash de verificación son iguales.

2.6.5. Preparador respuesta HTTP

Esto se resuelve en la clase SignWriter. Esta clase extiende de la clase Java PrintWriter con el fin de que se reemplace la escritura de la respuesta HTTP original existente en la aplicación por esta clase.

Lo que se realiza de forma concreta es crear un wrapper sobre la respuesta HTTP y entregar a esta clase para que intercepte la escritura sobre la respuesta HTTP y la almacene. Luego de que tiene almacenada la respuesta HTTP original, permite que se pueda modificar preparando las siguientes páginas:

- Página de confirmación con código de inicialización de Applet Java agregado.

- Página de confirmación con parámetros de operación de firma en curso agregados.
- Página preparada que constituye el documento a firmar.

Además permite la extracción de los valores de los parámetros especiales desde el HTML original de la página de confirmación.

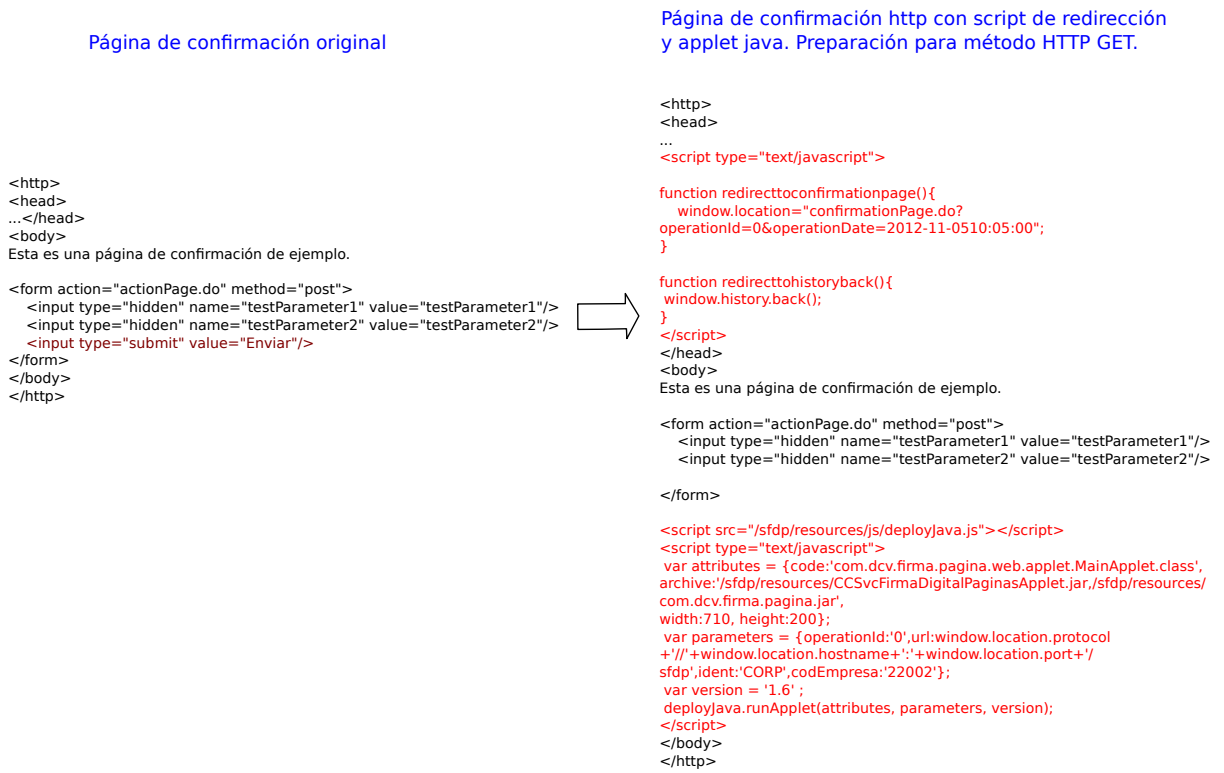

Figura 2.4: Modificación página de confirmación original agregando Applet Java

Página de confirmación con código de inicialización de Applet Java

La página de confirmación con código de inicialización de Applet Java consiste en la página de confirmación original con el script de inicialización del Applet Java al final de esta. Los controles originales de formularios presentes se eliminan y se agregan scripts con funciones javascript para redirigir la petición en función de si el requerimiento original es POST o GET.

Ejemplo de la modificación realizada se puede visualizar en la figura 2.4.

Página de confirmación con parámetros de operación de firma en curso

La página de confirmación con parámetros de operación de firma agregados se genera luego de que la operación de firma ha sido validada de forma exitosa. Entonces se agrega a

los formularios existentes en la página el identificador único de la operación y la fecha de la operación. Esto con el fin de que cuando se ejecute la acción se incluyan estos parámetros en el requerimiento.

Ejemplo de esta modificación se puede visualizar en la figura 2.5.

Página preparada que constituye el documento a firmar

A partir de la página de confirmación se crea el documento a firmar con el cual se inicia la operación de firma. Esta consiste en extraer el cuerpo del HTML de la página de confirmación y remover los controles que permitan manipular la página. Esto tiene por objetivo cumplir con el requerimiento de que el documento sea HTML y se pueda insertar al interior de un cuerpo HTML ya existente. Ejemplo de esta operación se puede ver en la figura 2.6.

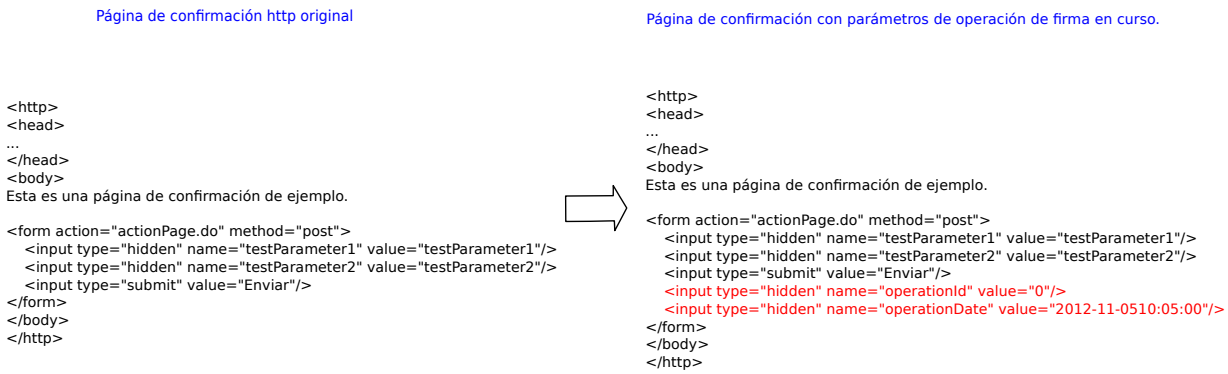

Figura 2.5: Modificación a página de confirmación HTTP con parámetros de operación de firma agregados

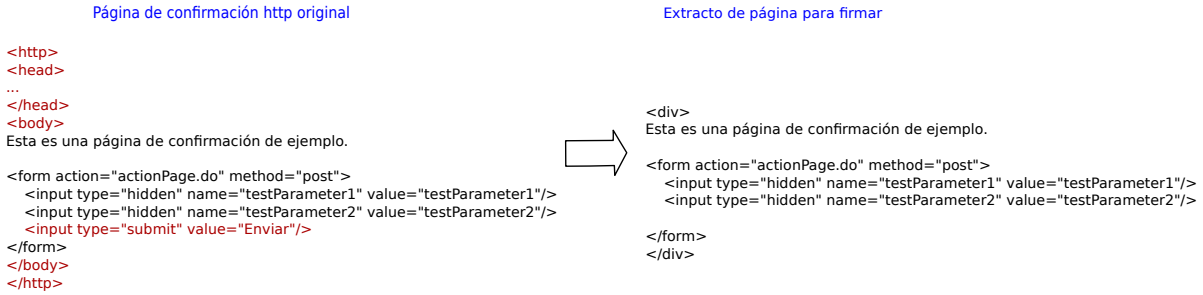

Figura 2.6: Creación de documento a firmar a partir de HTML de confirmación

Extracción de valores de parámetros especiales

La extracción de parámetros especiales se realiza identificando los formularios existentes y buscar el campo cuyo nombre sea igual al buscado, para luego extraer su valor. Ejemplo de este proceso se puede ver en la figura 2.7.

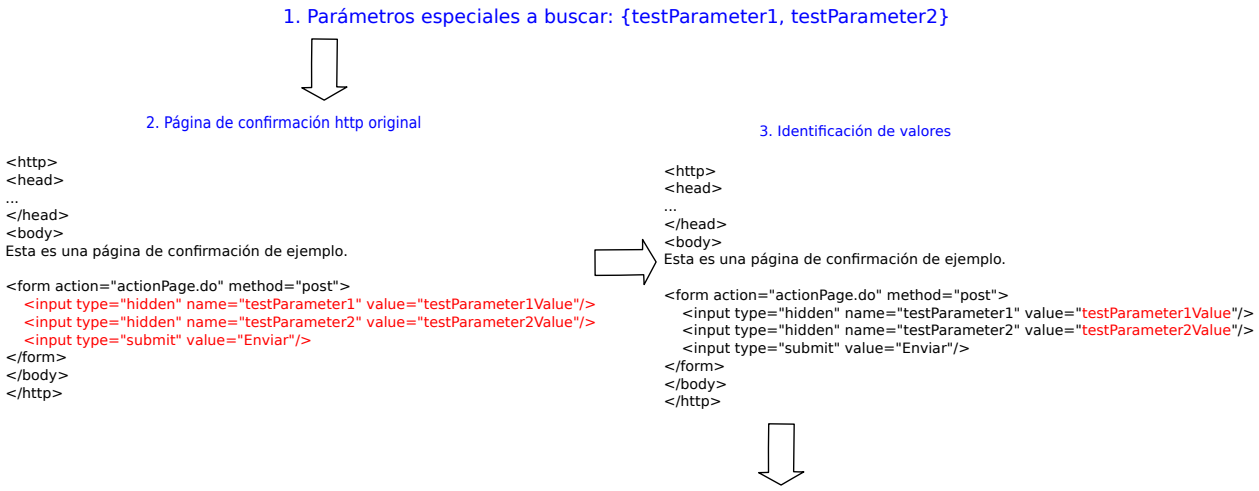

Figura 2.7: Extracción valor parámetros especiales

2.7. Componentes comunes

Se establece un componente común para que sea utilizado por el Applet de firma y el servicio de firma digital.

Contiene el modelo de clases que representa una operación de firma digital y el modelo de requerimiento y respuesta de validación de firma digital. Todas las clases están anotadas mediante JAXB con el fin de que los requerimientos y respuestas sean enviados y recibidos como xml. Las clases que definen estos componentes se pueden visualizar en la figura 2.8.

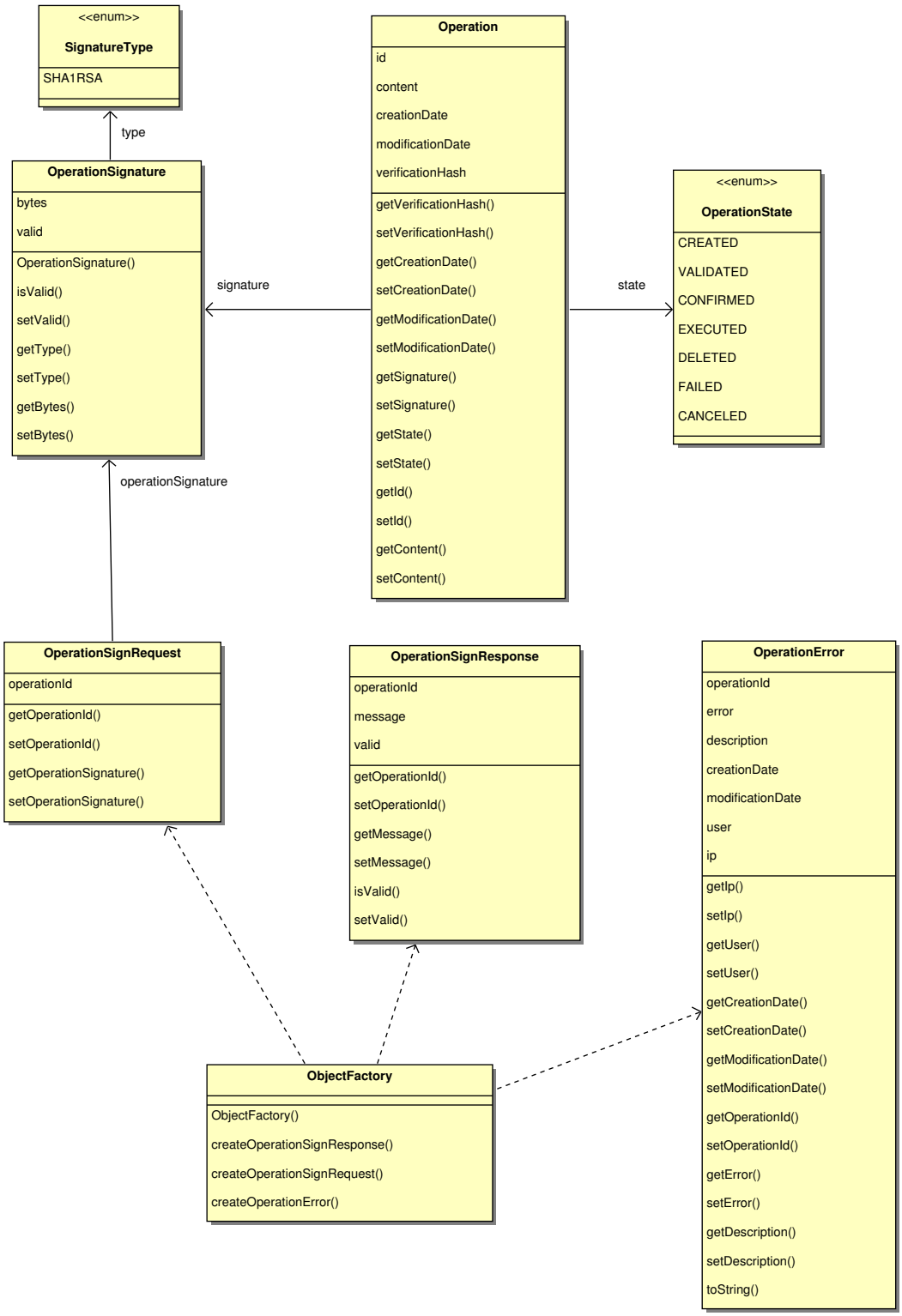

Figura 2.8: Clases Comunes Aplicación

2.8. Servicio de Firma Digital en Páginas

Este componente se divide en tres capas lógicas:

- Interfaces
- Lógica de negocio
- Capa de persistencia

2.8.1. Interfaces

Se distinguen 4 interfaces desarrolladas que son:

- Interfaz REST operación firma
- Interfaz REST páginas de firma
- Vistas administración páginas de firma
- Interfaz REST error operación firma

Las clases controladoras que definen las interfaces REST de la aplicación se pueden visualizar en la figura 2.9

Códigos de estado HTTP

Se deben definir los códigos de estado HTTP a retornar en caso exitoso o fallido. Es por esto que se establece el siguiente esquema por método HTTP utilizado que aplica a todas las firmas de servicios REST implementadas

Método HTTP	**Estado aplicación**	**Código de estado HTTP a responder**
POST,GET,PUT,DELETE	Acceso a recurso no autorizado	403 - Forbidden
POST,GET,PUT,DELETE	Error temporal en el sistema. Error recuperable.	503 - Service Unavailable
POST,GET,PUT,DELETE	Error no temporal en el sistema. Error no recuperable	500 - Internal Server Error
POST,GET,PUT,DELETE	Conflicto en recursos de servidor producto de requerimiento	409 - Conflict
GET,PUT,DELETE	Requerimiento procesado de forma exitosa	200 - OK
POST	Requerimiento procesado de forma exitosa	201 - Created

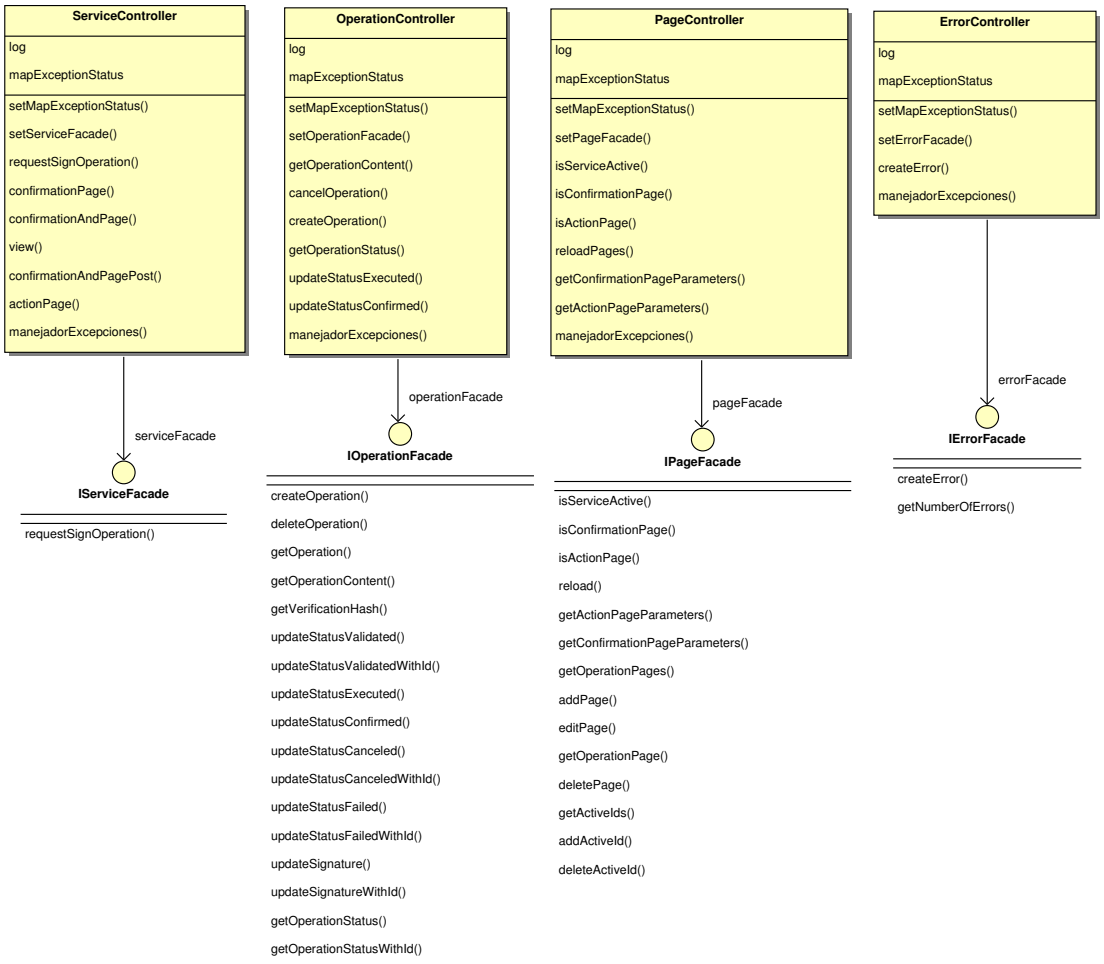

Figura 2.9: Controladores de interfaces REST

Intefaz REST de operación de firma

La interfaz REST de operación de firma resuelta por clases ServiceController y OperationController provee los siguientes métodos.

- Crear operación de firma: Este recibe como entrada el hash de verificación y el contenido de la operación a firmar. Retorna el identificador único de operación de firma digital creado.
- Cancelar operación: Cancela operación de firma digital asociada al identificador único de operación y el hash de verificación recibido.
- Obtener estado de operación: Retorna el estado de la operación de firma digital asociada al identificador y hash de verificación recibido.
- Actualizar estado de operación a confirmado: Actualiza a Confirmada el estado de la operación asociada al identificador único y hash de verificación recibido.

- Actualizar estado de operación a ejecutado: Actualiza a Ejecutada el estado de la operación asociada al identificador único y hash de verificación recibido.
- Procesar requerimiento de firma digital: Este recibe un requerimiento de firma consistente en el identificador único de operación y una firma digital. Retorna al cliente si la firma es correcta. Cambia a estado "validada" a la operación en caso de que sea válida la firma.

Interfaz REST páginas de firma

Esta interfaz se resuelve mediante la clase PageController. Provee los métodos de consulta de URLs de confirmación y acción. Además indica si el servicio se encuentra activo para el cliente que consulta. Los métodos que provee son los siguientes:

- Consulta URL confirmación: Responde si la URL enviada corresponde a una página de confirmación o no.
- Consulta URL acción: Responde si la URL enviada corresponde a una página de acción o no.
- Consulta servicio para código de empresa: Responde si para el código de empresa está activo el servicio de firma digital o no.
- Obtener parámetros especiales página de confirmación: Retorna el nombre de los parámetros especiales con los cuales utilizando el valor de estos, se calcula el hash de verificación.
- Obtener parámetros especiales página de acción: Retorna el nombre de los parámetros especiales con los cuales utilizando el valor de estos, se calcula el hash de verificación.

Vistas administración páginas de firma

Estas vistas HTML proveen los controles para ejecutar las siguientes funcionalidades:

- Inscripción de páginas de operación: Permite inscribir URLs de confirmación, acción y parámetros especiales asociados.
- Eliminar página de operación: Elimina una inscripción de URLs de confirmación, acción y parámetros especiales asociados.
- Modificar página de operación: Modifica una inscripción de URLs de confirmación, acción y parámetros especiales asociados.
- Inscripción de código de empresa a servicio: Agrega un código de empresa al servicio de firma digital.
- Eliminar código de empresa de servicio: Elimina código de empresa del servicio.

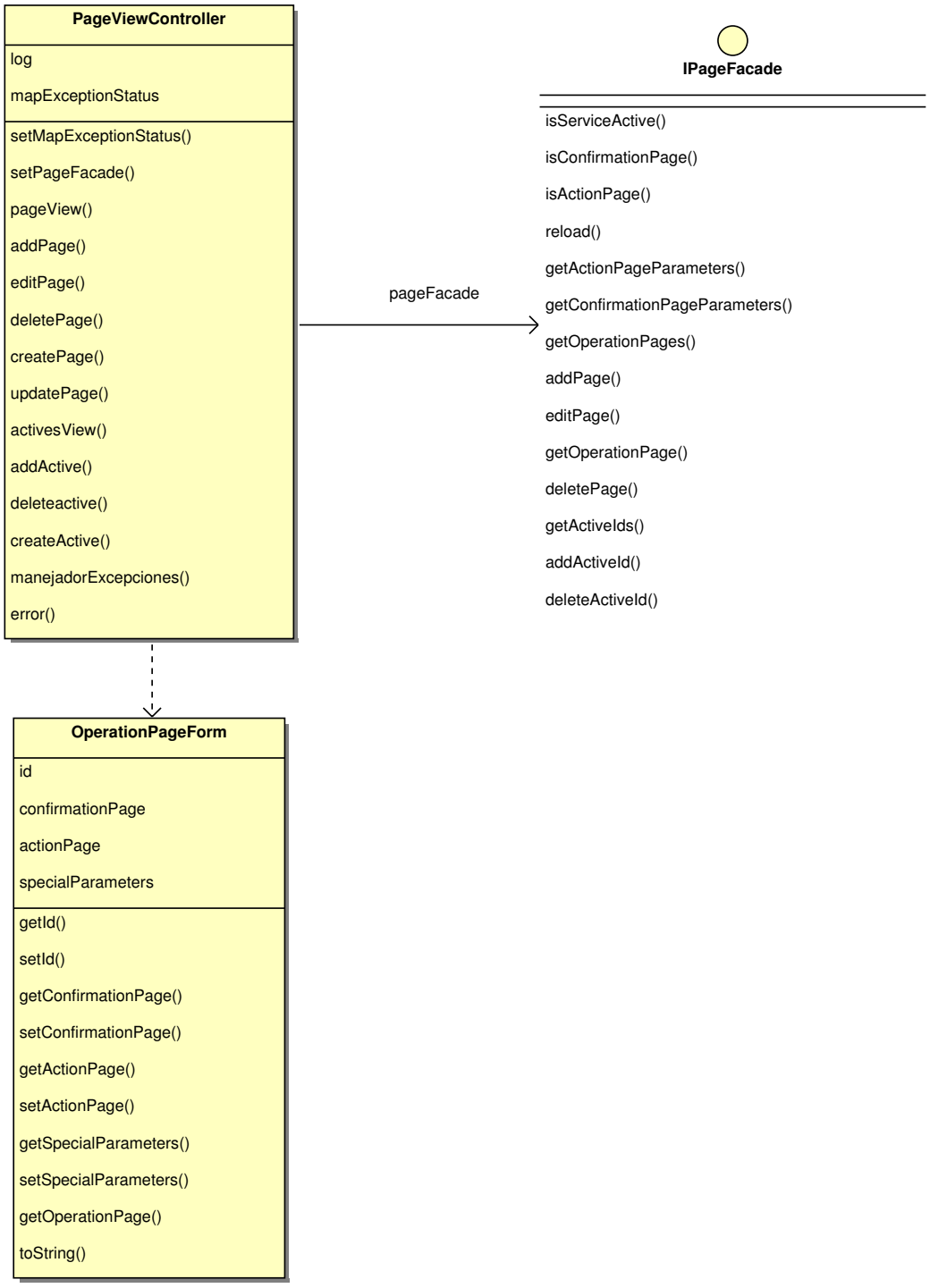

Figura 2.10: Controladores de interfaces HTML

Estas vistas se resuelven en la clase PageViewController mostrada en la figura 2.10. El formulario para el manejo de las páginas de operación se resuelve en la clase OperationPageForm.

Interfaz REST error operación firma

Esta se resuelve en la clase ErrorController y provee un único método que permite la inscripción de un error asociado a una operación de firma digital en curso.

2.8.2. Lógica de negocio

Esta capa es la encargada de ejecutar la lógica de la aplicación. Recibe las peticiones desde las interfaces y se conecta con la capa de datos en caso de ser necesario.

Esta capa se divide en tres componentes:

- Administrador operaciones
- Administrador páginas
- Administrador errores de operación

Cada componente se implementa como uno o más Facades dependiendo del tamaño de la capa. Se relacionan estos Facades siempre mediante su interfaz a modo de reducir el acoplamiento.

Administrador operaciones

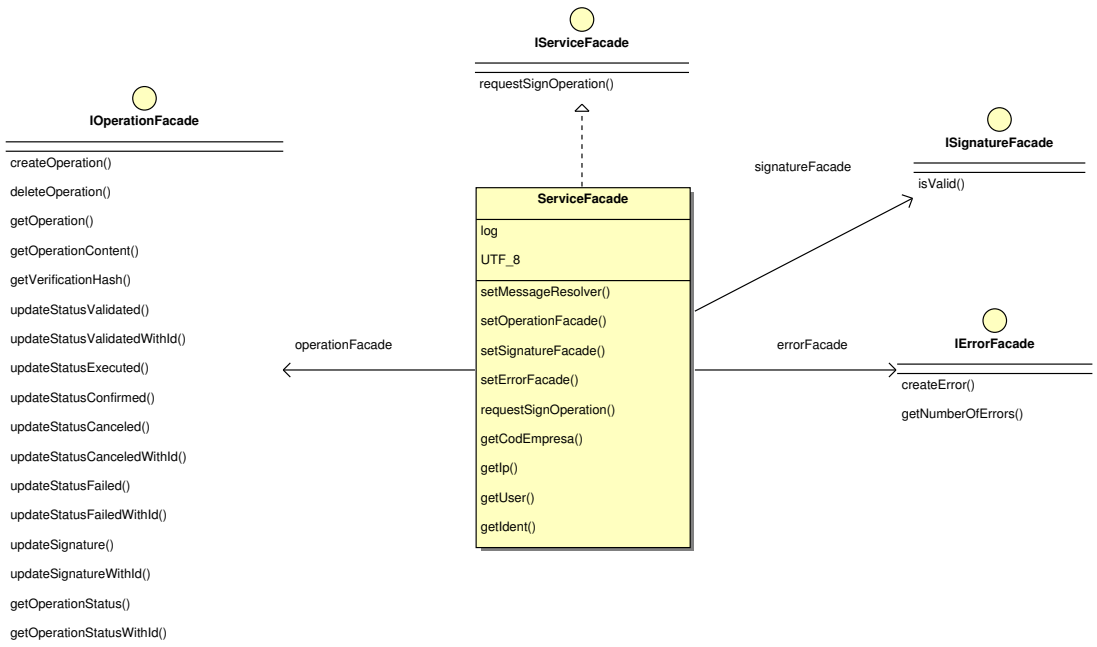

Figura 2.11: Administrador de operaciones de firma digital

Contiene la lógica que administra las operaciones de firma digital. Implementa y relaciona la lógica de administración de persistencia de operación de firma digital y validación de firma digital.

La lógica de administración de validación de las firmas digitales recibidas la realiza la clase facade ServiceFacade relacionando el facade administrador de persistencia de la operación de firma digital y el facade validador de firma digital. En la figura 2.11 se puede visualizar el facade y sus relaciones.

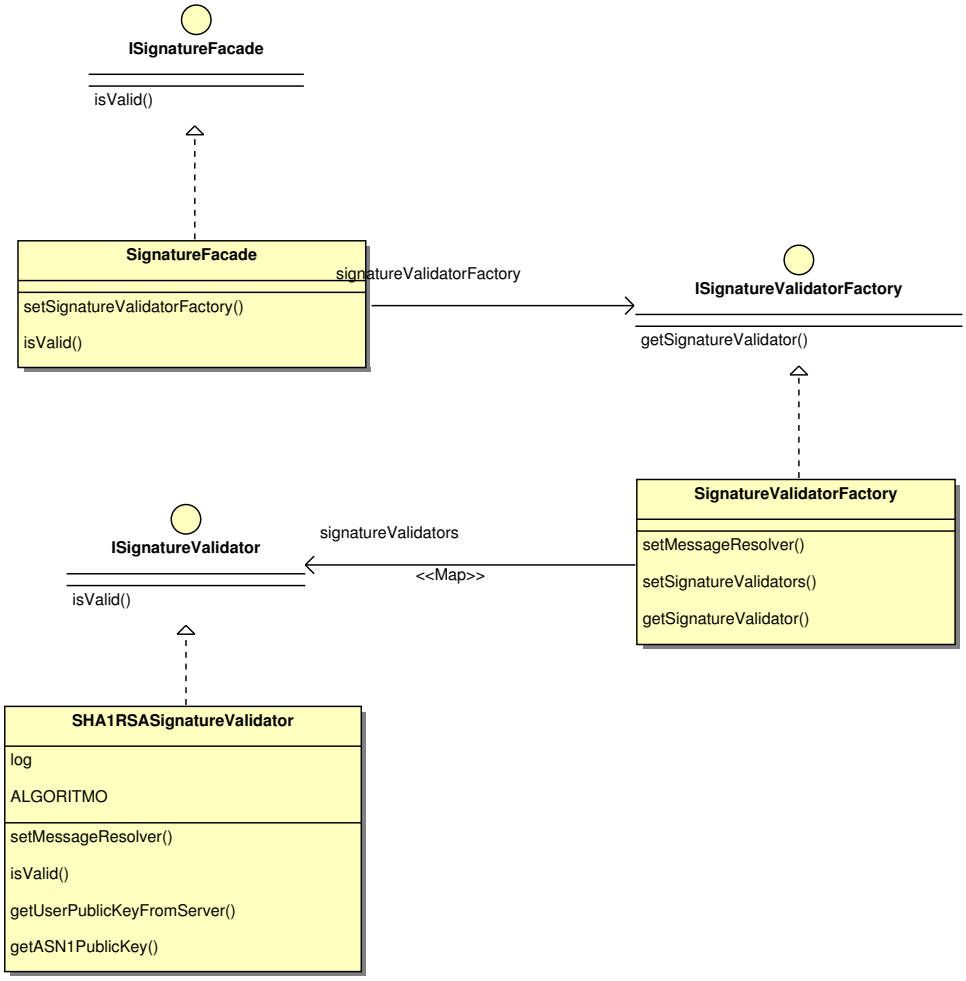

Figura 2.12: Facade validador de firma digital

El facade validador de firma digital utiliza un patrón de fábrica abstracta como se puede observar en la figura 2.12. Al instanciar el facade, resuelto en la clase SignatureFacade, este carga en memoria todos las implementaciones de validación de firma que son las implementaciones de la clase ISignatureValidator. Por el momento se implementa la validación de firma RSA-SHA1 implementada en la clase SHA1RSASignatureValidator.

La implementación concreta de validación de firma es responsable de buscar la clave pública desde el repositorio interno de los clientes del DCV y recrear la firma en lenguaje

Java.

Se almacenan claves públicas en formato RSA de 1024 y 2048 bits sólo, ya que existen aplicaciones distintas a la desarrollada en esta memoria que las manejan con este formato. Una clave pública RSA se compone de dos enteros: módulo y exponente. Las claves públicas se almacenan con estructura ASN.1:Abstract Syntax Notation One y sus bytes transformados a texto hexadecimal. La decisión histórica de almacenar mediante este formato, es debido a que Windows muestra con formato ASN.1 las claves públicas en su aplicación de manejo de certificados personales como se puede ver en la figura 2.13

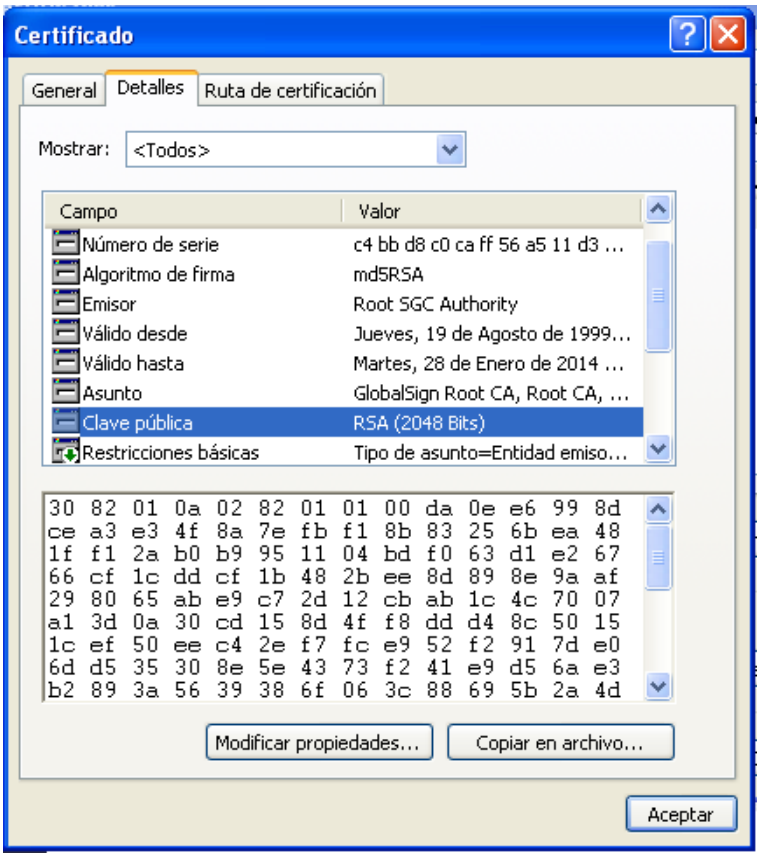

Figura 2.13: Vista clave pública en almacén de certificados de Microsoft Windows

Para claves públicas RSA de 1024 y 2048 bits la estructura ASN.1 provista por Windows siempre obedece a una de tipo secuencial en la cual se puede identificar el comienzo del módulo por el primer hexadecimal 00 y su término por la definición del exponente al final de este 02 03 el cual indica siempre que son 3 bytes que son 01 00 01 igual al entero 65537 que es el exponente.

De esta forma para cada cliente al momento de verificar la firma digital enviada, se obtiene la clave pública almacenada y se genera la clave pública RSA en Java para validar la firma digital enviada.

El facade administrador de la validación de firma utiliza el facade de error cuya interfaz es IErrorFacade para obtener el número de errores vigente previo a la validación. Esto con el

fin de verificar que no existan más de 3 errores existentes en la operación. Además lo utiliza por si falla la validación para inscribir un error nuevo.

El facade de administración de persistencia de operaciones de firma provee métodos para crear y modificar operaciones de firma digital y se resuelve en la clase OperationFacade descrita en la figura 2.14. En la mayoría de los métodos se requiere del hash de verificación y el identificador único de operación para realizar la lógica. Los métodos que no requieren el hash de verificación son exclusivamente para uso interno y nunca accedidos por las interfaces de la aplicación.

Figura 2.14: Administrador de persistencia de operación de firma digital

Administrador páginas

Este administra la persistencia de las URLs inscritas como páginas de confirmación o acción, además del nombre de los parámetros especiales para el cálculo del hash de verificación. Se resuelve en la clase PageFacade descrita en la figura 2.15.

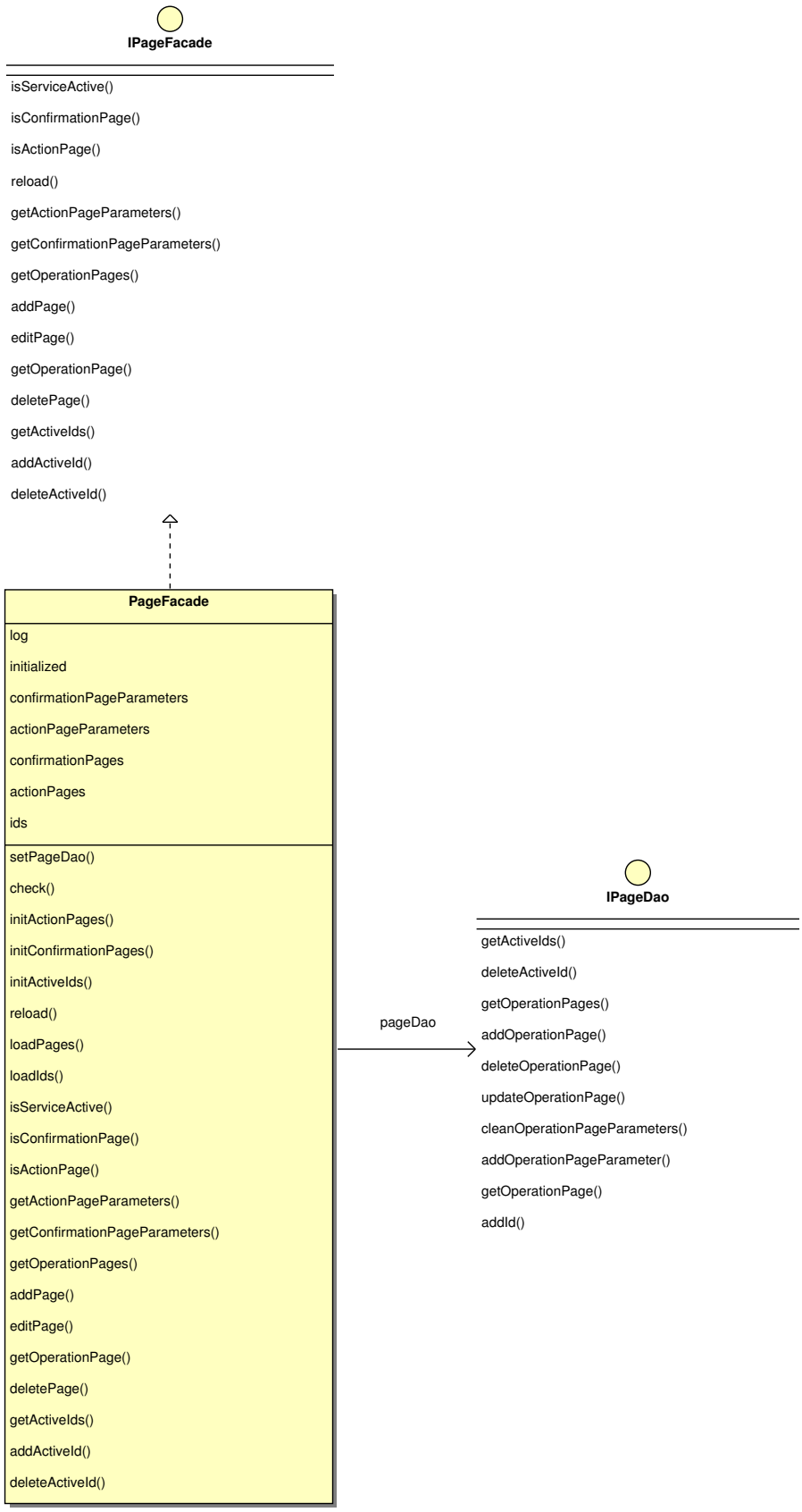

Figura 2.15: Administrador de persistencia de páginas de firma digital

Un punto importante en la lógica de esta clase es que al momento de crearse mantiene en memoria las URLs de confirmación, acción y parámetros especiales asociados con el fin de que las consultas sean más rápidas. Como es este mismo quien administra los cambios sobre los datos que mantiene en memoria, entonces se preocupa además de mantener actualizada esta información.

Administrador errores de operación

Este módulo se encarga de persistir los errores asociados a una operación de firma digital y de entregar la cuenta de errores asociados a una operación específica. Se resuelve en la clase ErrorFacade descrita en la figura 2.16.

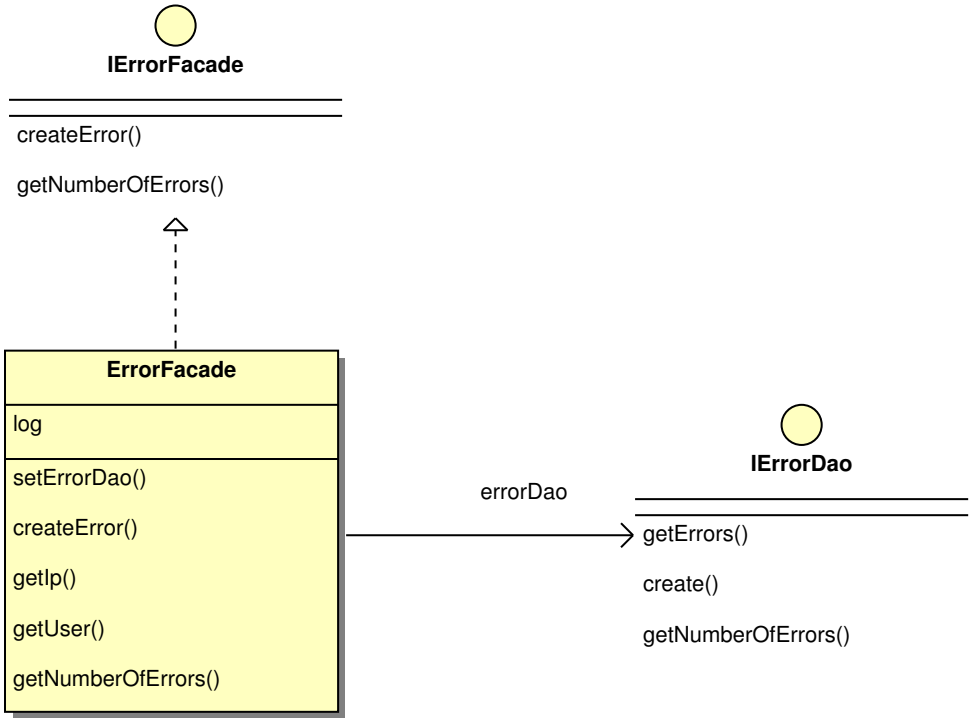

Figura 2.16: Administrador de persistencia de errores de operación

Además se maneja una jerarquía de excepciones conforme a la definición de código de estado HTTP de servicios REST para estados de falla definida en la figura 2.18. Es por esto que cada facade y controlador tiene la capacidad de generar estas excepciones para situaciones de error conocidas. Esto se puede apreciar en la figura 2.17.

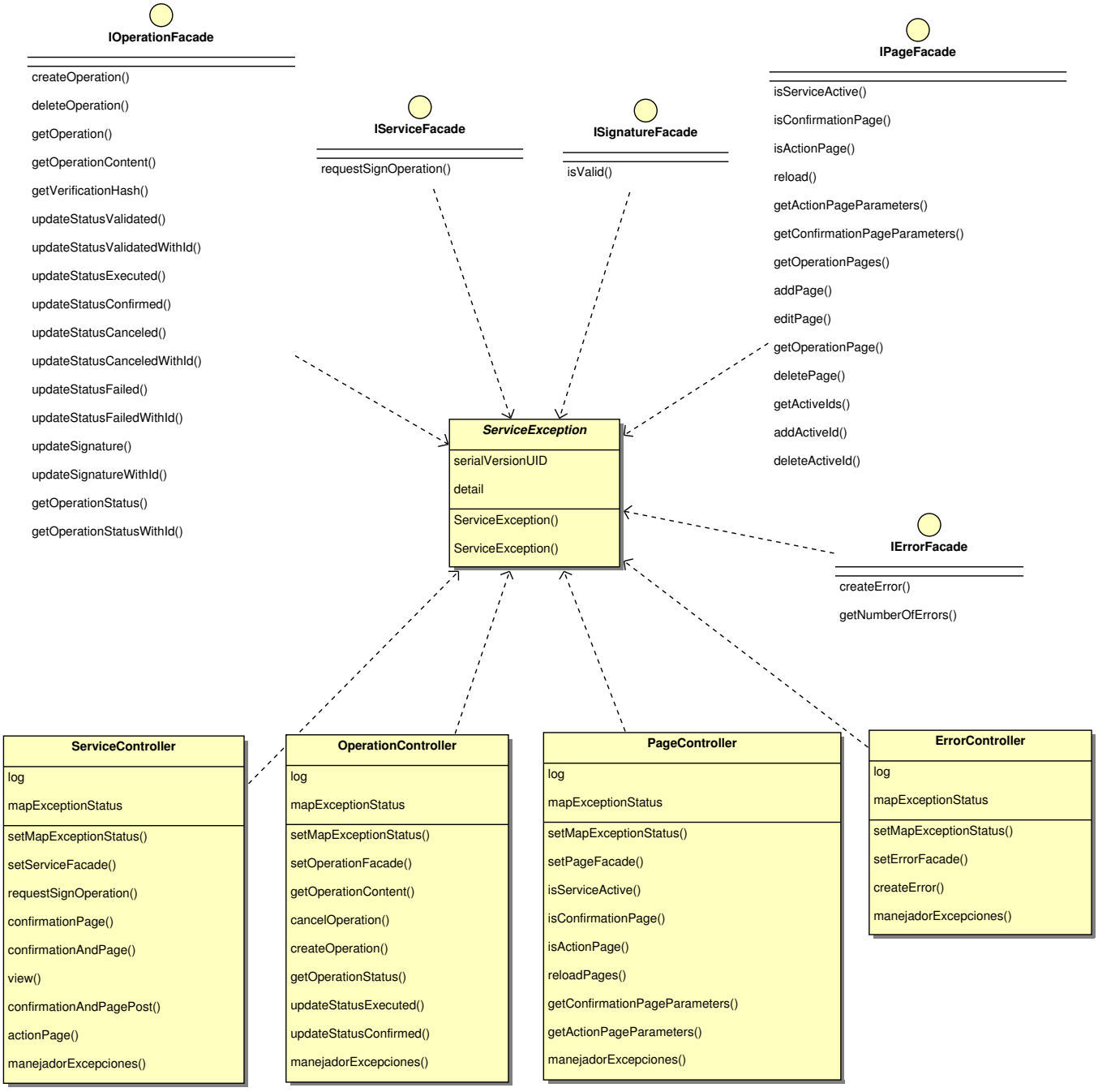

Figura 2.17: Dependencia facades y controladores sobre clase de excepción

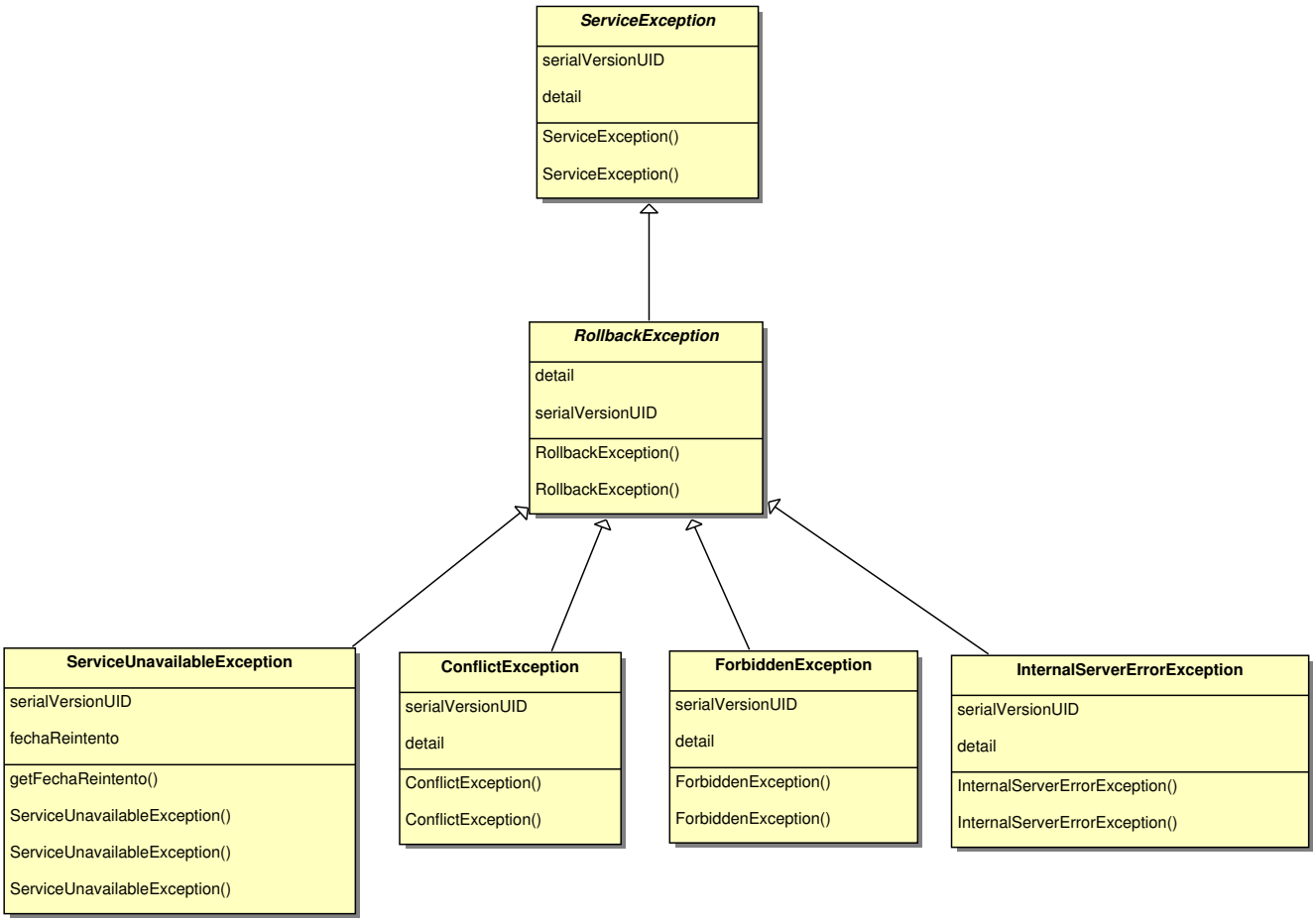

Figura 2.18: Excepciones servicio

Para el manejo de excepciones no controladas como por ejemplo las provenientes desde la capa de persistencia, se utiliza un transformador de excepciones. Este es llamado desde los controladores al identificar una excepción no controlada en el método llamado manejadorExcepciones y utiliza la implementación ExceptionHandler la cual retorna una excepción conocida por la aplicación. Figura 2.19.

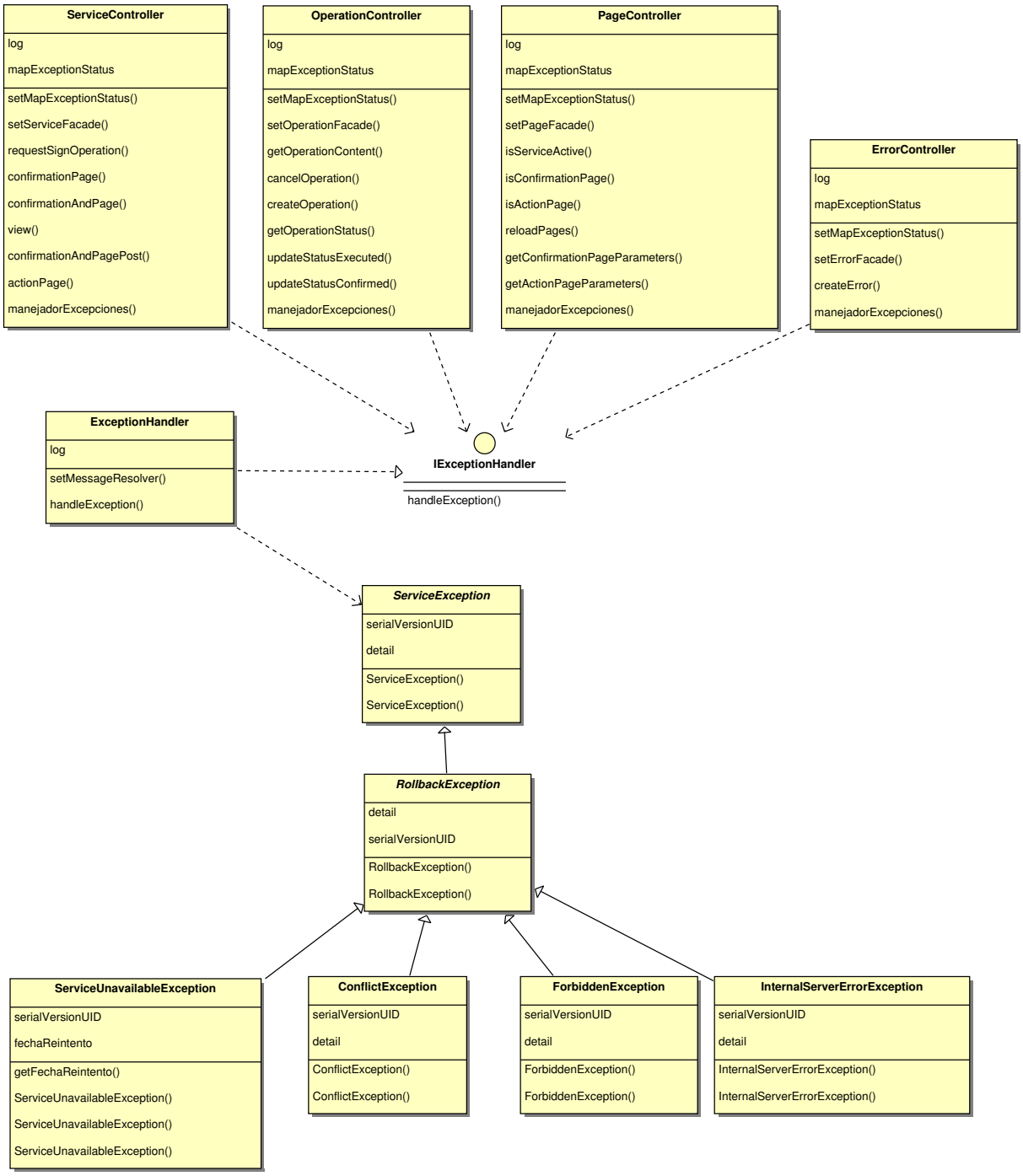

Figura 2.19: Transformación de excepciones desconocidas por la aplicación

2.8.3. Capa de persistencia

La capa de persistencia está formada por 4 componentes:

- Claves públicas cliente
- Operaciones firma digital
- Errores operación de firma
- URLs páginas de confirmación y acción

Las claves públicas se obtienen fuera de la aplicación mediante llamado EJB a un servicio existente, por lo que la implementación de su búsqueda no se tuvo que realizar.

Por otro lado el manejo de los otros 3 componentes sí se tuvo que implementar y para esto se utilizó el patrón DAO: Data Access Objects, el cual se refiere a separar la lógica de persistencia en módulos independientes. Se utilizó el framework Mybatis para implementar los DAOs debido a que se integra con Spring Framework y permite separar el código SQL del código Java.

Tablas

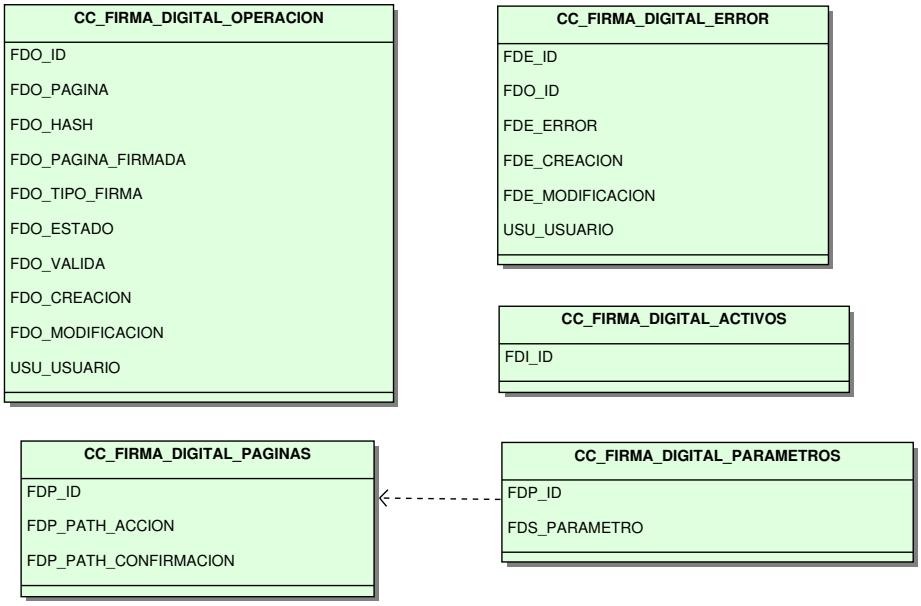

Figura 2.20: Tablas capa de persistencia

Se crearon las tablas de la figura 2.20.

La tabla CC_FIRMA_DIGITAL_OPERACION almacena las operaciones de firma digital efectuadas en la aplicación sus atributos son los siguientes:

- FDO_ID: Identificador único de operación de firma digital
- FDO_PAGINA: Documento a firmar
- FDO_HASH: Hash de verificación asociado
- FDO_PAGINA_FIRMADA: Firma digital de documento

- FDO_TIPO_FIRMA: Tipo de firma digital
- FDO_ESTADO: Estado de firma digital
- FDO_VALIDA: Flag que indica si firma digital es válida
- FDO_CREACION: Fecha y hora de creación
- FDO_MODIFICACION: Fecha y hora de modificación
- USU_USUARIO: Nombre de usuario en sesión HTTP dueño de operación de firma digital

La tabla CC_FIRMA_DIGITAL_ERROR almacena los errores asociados a una operación de firma. Sus atributos son:

- FDE_ID: Identificador único del error.
- FDO_ID: Identificador único de operación de firma digital.
- FDE_ERROR: Texto descriptor del error.
- FDE_CREACION: Fecha y hora de creación.
- FDE_MODIFICACION: Fecha y hora de modificación.
- USU_USUARIO: Usuario en sesión HTTP dueño del error.

La tabla CC_FIRMA_DIGITAL_ACTIVOS almacena los identificadores únicos de las empresas de los clientes que tienen activos los servicios de firma digital que se implementan en esta memoria. Tiene un único atributo FDI_ID que es el identificador antes señalado.

La tabla CC_FIRMA_DIGITAL_PAGINAS almacena las URLs de acción y confirmación, sus atributos son:

- FDP_ID: Identificador único de registro.
- FDP_PATH_ACCION: URL relativa de acción.
- FDP_PATH_CONFIRMACION: URL relativa de confirmación.

La tabla CC_FIRMA_DIGITAL_PARAMETROS almacena los nombres de los parámetros asociados a las páginas de acción y confirmación. Estos parámetros son utilizados por el filtro para calcular el hash de verificación. Sus atributos son:

- FDP_ID: Identificador único de registro en tabla CC_FIRMA_DIGITAL_PAGINAS.
- FDS_PARAMETRO: Nombre de parámetro.

Data Access Objects

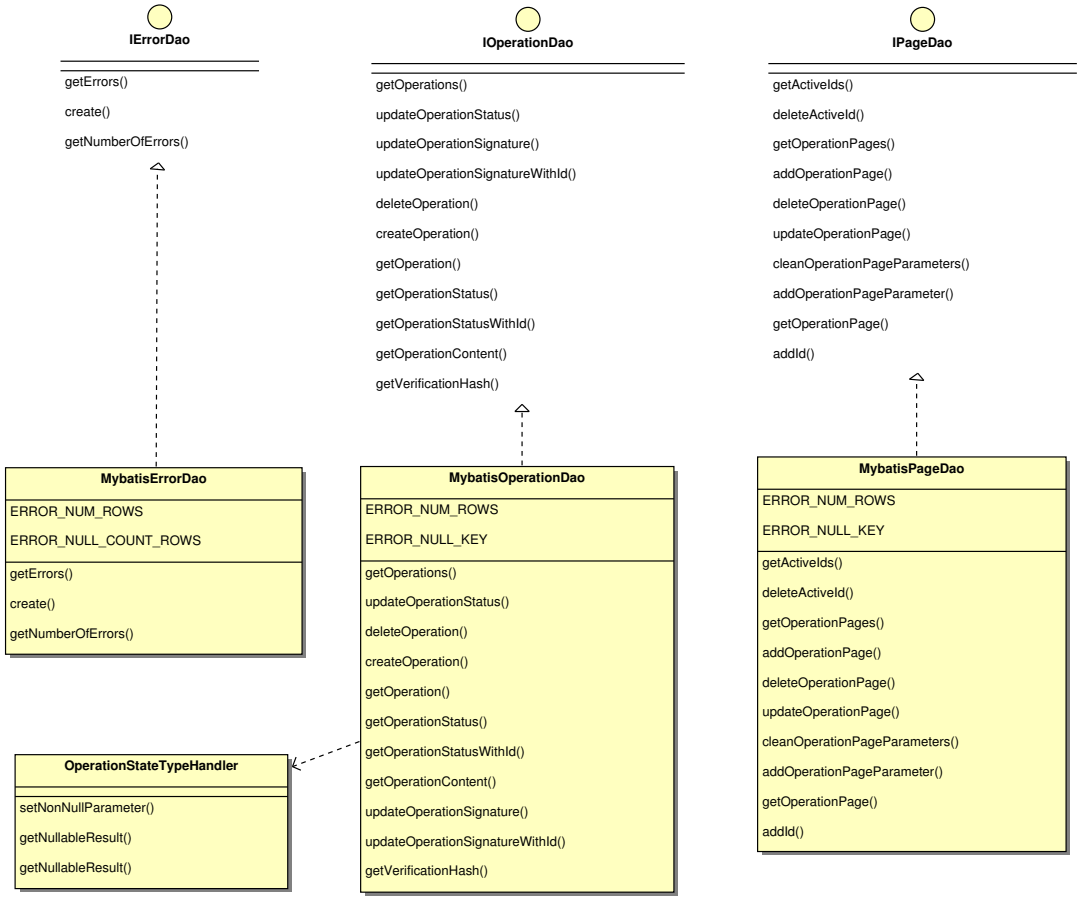

Figura 2.21: DAOs capa de persistencia

La figura 2.21 muestra los DAOs implementados que conforman la capa de persistencia. MybatisOperationDao representa el DAO que administra la tabla CC_FIRMA_DIGITAL_OPERACION. MybatisErrorDao crea nuevos registros en la tabla CC_FIRMA_DIGITAL_ERROR y retorna la cantidad de errores asociados a un identificador único de operación. Finalmente MybatisPageDao administra y consulta las tablas CC_FIRMA_DIGITAL_PAGINAS, CC_FIRMA_DIGITAL_PARAMETROS y CC_FIRMA_DIGITAL_ACTIVOS.

2.9. Applet Firma Digital

El Applet de firma digital es el componente principal de la solución. Corre en el navegador del cliente utilizando la JVM que este tiene instalada en su máquina.

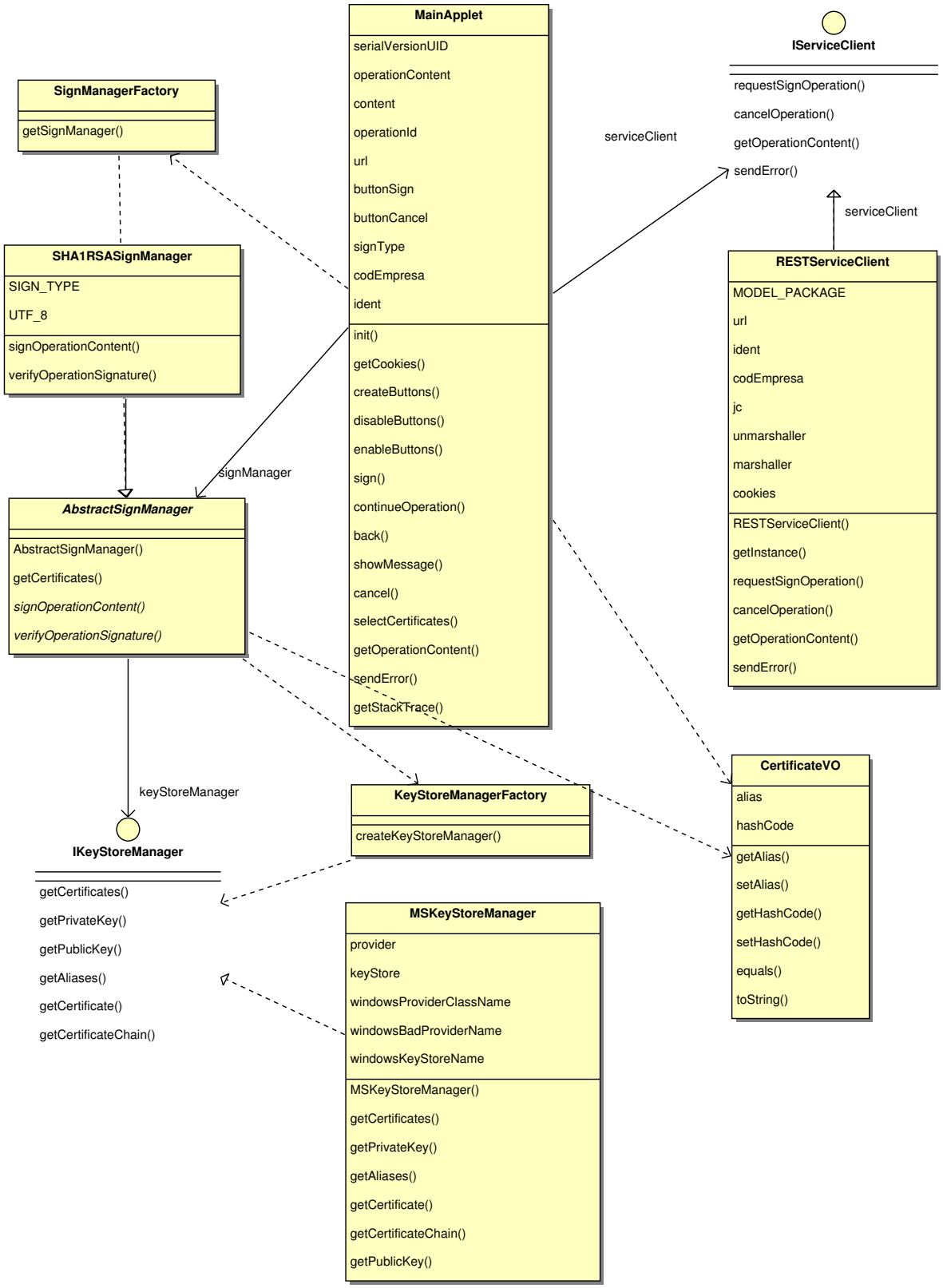

Figura 2.22: Applet Firma Digital

Para que este se inicialice de forma correcta requiere que se le entreguen los siguientes parámetros:

- identificador único de operación
- URL desde donde obtener los servicios REST

Sus componentes son los siguientes:

- Firmador
- Manejador almacén de certificados
- Cliente Applet servicio operación firma
- Cliente Applet servicio error en operación

Estos se ven implementados en las clase de la figura 2.22.

2.9.1. Firmador

Este componente tiene la lógica para firmar el documento que representa la operación desplegada en el navegador web. En la figura 2.23 se muestran las clases que implementan este componente.

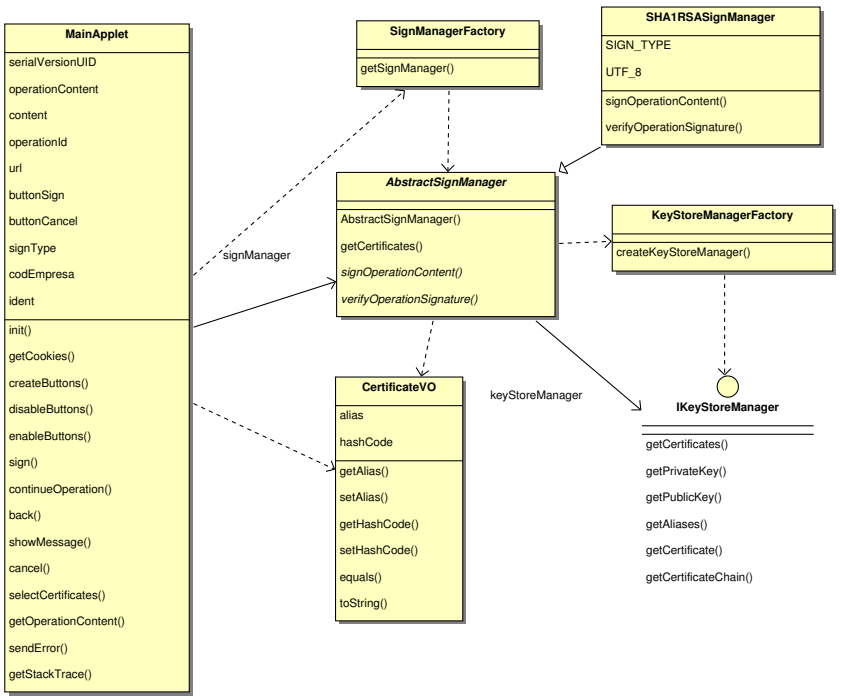

Figura 2.23: Firmador

Se utiliza el factory SignManagerFactory para crear al administrador de firma utilizando el tipo de firma vigente que para esta solución es RSA-SHA1 implementado en la clase SHA1RSASignManager. Esta clase implementa los métodos de firma y verificación.

Además el firmador inicializa mediante el factory KeyStoreManagerFactory el almacén de claves dependiendo del sistema operativo identificado. Esto para entregarle a la interfaz principal del Applet MainApplet la lista de certificados del almacén de claves del cliente. Esta lista la entrega en un formato amigable utilizando la clase CertificateVO la cual tiene el alias del certificado y el código hash del certificado.

2.9.2. Manejador almacén de certificados

Como almacén de certificados [2] se define el lugar físico donde se encuentran los certificados digitales.

Según Java Cryptography Architecture (JCA) [16] los almacenes de claves en Java se definen mediante la implementación de la clase Keystore. A continuación se describen las implementaciones de almacenes de claves que Java provee en sus distribuciones dependiendo del sistema operativo.

Sistema operativo	Proveedor Java	Almacén de claves
Microsoft Windows	sun.security.mscapi.SunMSCAPI	Windows-MY certificados personales de MS Windows CAPI
Mac	Apple	KeystoreChain certificados personales de OSX
Todos	sun.security.pkcs11.SunPKCS11	Almacenes de claves que cumplan estándar PKCS#11 y Mozilla NSS

En sistema operativo Microsoft Windows existen dos alternativas para acceder a los certificados personales de los clientes en sus máquinas que son:

- Almacén de certificados de Microsoft Windows: Los tokens Aladdin traen consigo un driver llamado PKI Client que se instala en el sistema operativo. Este hace que los certificados personales del token de seguridad se encuentren disponibles en el almacén de certificados de Windows.
- Conexión directa a token de seguridad: Utilizando la API del estándar PKCS#11 se puede crear una conexión a los almacenes de claves del token de seguridad.

El estándar PKCS#11 está afecto a múltiples ataques [2], por lo que el uso directo de Criptoki para acceder a los certificados del token de seguridad no es una idea aceptable. Por esto se asume la existencia del driver PKI Client instalado que los token Aladdin incluyen y utilizar MSCAPI para acceder al almacén de claves de Windows que incluye los certificados del token.

Esto a su vez simplifica la aplicación en cuanto a amigabilidad ya que evita que el cliente deba ingresar los parámetros de conexión del token de seguridad. Además la hace más mantenible en cuanto a que la seguridad de conexión entre el token y la aplicación es de absoluta

[2]Keystore en Inglés

responsabilidad del driver.

Las clases que representan este componente implementado se ven en la figura 2.24. Se utiliza el factory KeyStoreManagerFactory para instanciar el almacén de claves en función del sistema operativo identificado. Como la solución actual contempla ser ejecutada sólo en sistemas operativos Microsoft Windows se tiene una sola implementación MSKeyStoreManager basada en el uso de la implementación de almacén de claves Java sun.security.mscapi.SunMSCAPI.

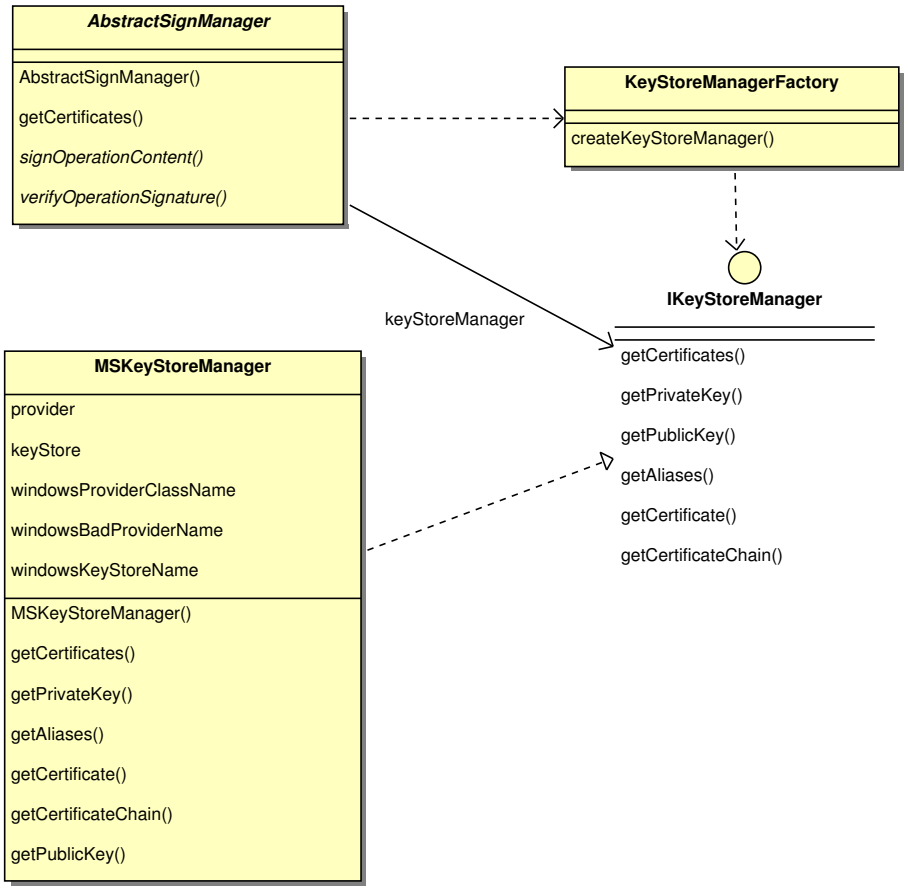

Figura 2.24: Manejador almacén de certificados

2.9.3. Cliente Applet servicio operación firma

Este componente se implementa en la clase RESTServiceClient como se visualiza en la figura 2.25. Esta provee las siguientes funcionalidades:

- Obtener contenido operación: Mediante el identificador de operación único y la URL de servicio con el que es inicializado el Applet se realiza un llamado estilo REST al servicio de operación de firma digital y se obtiene el contenido con el que fue iniciada la operación. Esto porque el contenido a firmar no se puede extraer desde el navegador, ya que el código fuente que muestran los navegadores siempre es una interpretación de lo que se recibe desde los servidores web.

- Enviar requerimiento de firma digital: Mediante JAXB crea un xml con una petición de firma representada por la clase OperationSignRequest. Luego procesa el retorno xml desde el servicio creando una clase OperationSignResponse.

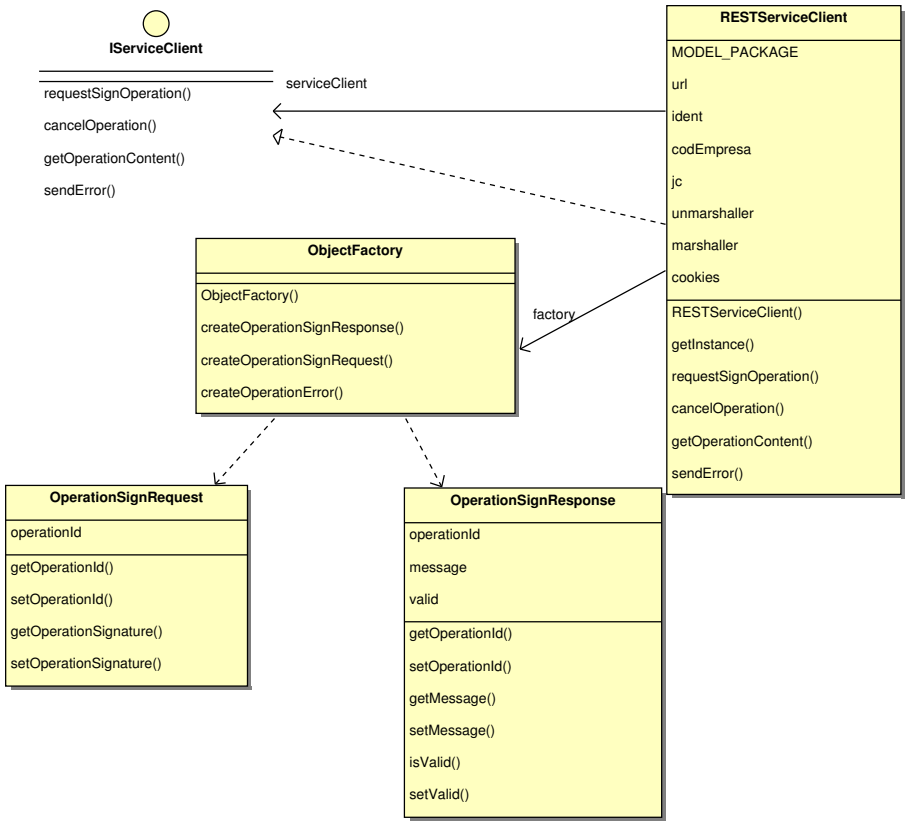

Figura 2.25: Cliente servicio operación de firma

2.9.4. Cliente Applet servicio error en operación

Cada vez que ocurre algún estado de excepción en la aplicación se extrae la información de esta y se realiza un intento de envío al servicio de error de operaciones de firma. Como se puede ver en la figura 2.26 esto se implementa en la clase RESTServiceClient enviando un xml generado a partir de la clase OperationError mediante JAXB. Por parte del cliente no se espera respuesta exitosa, ya que no se puede producir un estado de excepción sobre otro.

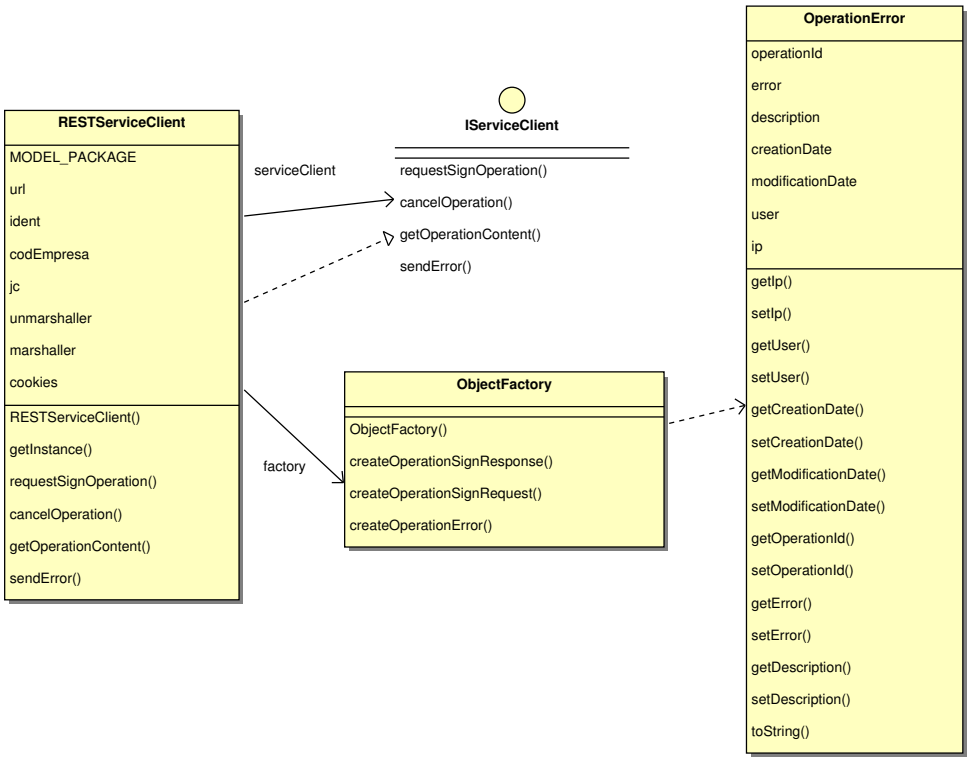

Figura 2.26: Cliente servicio error de operación

Capítulo 3

Resultados

En este capítulo se muestran los resultados obtenidos luego de instalar la aplicación en el sistema DCV existente en ambiente de integración. Se presenta una historia de usuario con detalle técnico con el fin de que se comprenda qué ocurre a bajo nivel.

3.1. Historia de usuario

Se presenta cómo al usuario se le presenta el nuevo servicio. Esto sobre una simple aplicación de pruebas que presenta un formulario simple en la url relativa /sfdp/confirmationpage.do y que realiza un POST sobre la url relativa /sfdp/action.do. Esta tiene como parámetros ocultos de formulario los nombres testParameter1 y testParameter2.

La página de confirmación original es la presentada en la figura 3.1.

Figura 3.1: Página HTML original de confirmación

Luego activando el servicio para el código de empresa 22002 y agregando las url de confirmación y acción se accede nuevamente y se inicializa el Applet presentando la vista mostrada en la figura 3.2.

Figura 3.2: Página HTML de confirmación intervenida con Applet Java

En este momento la operación de firma ya fue iniciada y viendo el código fuente de la página se aprecia que el identificador único de operación es 107 como se aprecia en la figura 3.3

Figura 3.3: Código fuente página HTML de confirmación intervenida con Applet Java

Luego se selecciona el certificado con el cual se desea firmar el contenido como se muestra en las figuras 3.4 y 3.5.

Figura 3.4: Selección de certificado mediante Applet Java

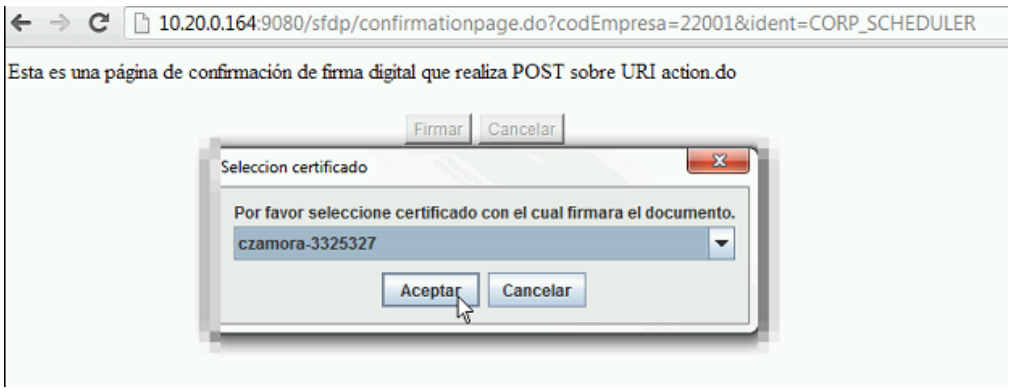

Figura 3.5: Selección de certificado mediante Applet Java

A continuación se redirige a la página de confirmación original, pero con parámetros adicionales en su requerimiento como se aprecia en la figura 3.6.

Figura 3.6: Página de confirmación con parámetros adicionales

Luego se envía el requerimiento a la página de acción, que mediante los parámetros de firma enviados, valida que existe una operación de firma digital validada, por lo que procesa de manera exitosa el requerimiento como se puede ver en la figura 3.7

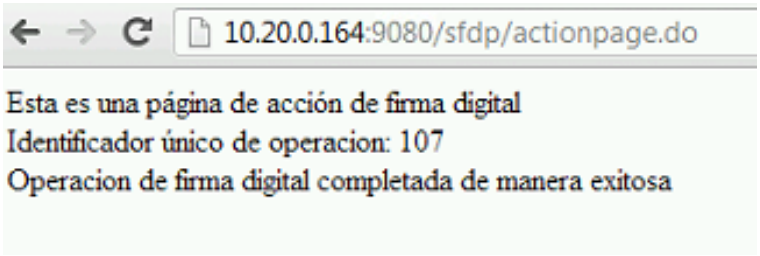

Figura 3.7: Página de acción ejecutada de forma correcta

3.2. Atributos de calidad

A continuación se presentan los atributos de calidad logrados con la nueva aplicación.

- Correctitud: Para validar la aplicación se compró un token eToken Aladdin PRO y un certificado digital válido emitido a nombre del autor de la memoria. Luego utilizando

este token se realizó el proceso de firma digital en los navegadores web actuales. En todos el resultado fue exitoso.

- Seguridad: El nuevo proceso de firma digital hace imposible que se ejecute la lógica de negocio sin que su información sea firmada de forma correcta. Esto debido a que hash de verificación se compone de los parámetros críticos que componen la operación. Además los Applets Java en los navegadores actuales no corren a no ser que el plugin Java esté actualizado.
- Portabilidad: Es una solución construida en Java que corre sobre la JVM de las máquinas cliente y servidor. La única implementación dependiente del sistema operativo es la del almacén de claves de Microsoft Windows en el Applet. Este tiene un patrón factory que identifica el sistema operativo, por lo que la dificultad de extender este acceso al almacén de claves en otro sistema operativo es baja.
- Tiempo de respuesta: El DCV maneja un servicio interno para la contabilización del tiempo de respuesta. Realizado un benchmark de 1000 operaciones de firma digital se encuentra que los tiempos mejoran en un 30 % aproximadamente. Esto principalmente porque no redirige el requerimiento HTTP.
- Mantenibilidad: La nueva aplicación permite detectar de forma temprana errores tanto en el cliente como en el servidor. Esto gracias a su interfaz de errores que persiste en base de datos cualquier error asociado a una operación específica. Además la arquitectura que presenta la aplicación sigue patrones de desarrollo comprobados, que son principalmente los patrones factory y strategy.

La nueva aplicación de firma interviene de forma muy simple la página de confirmación agregando los botones para firmar o cancelar la operación. Esto reduce la cantidad de acciones previas a la firma digital.

La solución antigua implicaba realizar 5 acciones antes de concretar la operación:

1. Aceptar ejecución de plugin ActiveX CAPICOM
2. Aceptar ingreso en almacén personal de certificados de CAPICOM
3. Selección de certificado
4. Ingreso de PIN de token de seguridad
5. Aceptar ingreso en almacén personal de certificados de CAPICOM

La nueva aplicación basada en Applet Java maneja sólo 2 acciones:

1. Selección de certificado
2. Ingreso de PIN de token de seguridad

Esto porque el Applet Java está firmado digitalmente con el certificado del DCV donde la página se publica. Por esto el navegador web por defecto no pide confirmación para ejecutar el Applet Java.

Conclusión

Los objetivos de esta memoria han sido cumplidos. Se ha migrado el actual servicio de firma digital en páginas implementando un Applet Java para que el cliente firme digitalmente el contenido parcial de la página web HTML que visualiza.

Se comenzó desarrollando prototipos para validar el acceso al almacén de claves principalmente. Luego a partir de estos prototipos se hizo un desarrollo incremental que terminó en la solución presentada en esta memoria.

La arquitectura de la aplicación es orientada al servicio y utiliza estilo de arquitectura REST. Este requerimiento provocó la creación de nuevos estándares al interior del DCV para la creación de futuros servicios. Estos estándares definen las tecnologías a utilizar y la arquitectura base.

La metodología fue la adecuada, pero hubiera sido útil haber elaborado un plan de riesgo del proyecto con el fin de mitigar posibles retrasos.

El marco conceptual del proyecto fue amplio, ya que cualquier implementación en el DCV debe estar respaldada por un marco teórico fuerte. Además al ser una modificación y no un desarrollo nuevo, se tuvo que analizar el servicio existente, el cual en su mayoría no estaba documentado y presentaba errores de concepto muy graves.

El concretar esta memoria contempla un hito importante en cuanto a que facilita el uso de la firma digital de una forma simple al usuario final. Hay que considerar que el usuario por lo general tiene un nivel de conocimiento de firma digital que no va más allá del ámbito legal existente en Chile.

De los proyectos existentes en el mercado ninguno servía para el propósito que desarrolla esta memoria, ya que se centran en el firmado de PDF. El gran número de soluciones basados en firma de PDF se debe a que los visores contienen validadores de firma que al usuario final le son más amigables. En la medida que se instruya a la gente en materia de firma digital se comenzarán a proponer más soluciones como esta, en las cuales el nivel de rendimiento es muchísimo más alto que las soluciones basadas en firma de PDF. De todas formas cabe destacar que extender esta solución para firma de PDF es muy simple debido a la arquitectura que el Applet Java presenta.

Se promueve una arquitectura simple para implementar un esquema de firma digital en plataformas web que puede ser replicado en cualquier organización. La separación lógica de

componentes fue clave para lograr esto. Esto nos indica que realizar una solución empresarial de firma digital no tiene por qué ser complicada y su orientación al servicio con estilo REST nos permite afirmar que la solución es independiente de la plataforma, ya que sólo se requiere conexión bajo el protocolo HTTP.

El proceso de desarrollo de esta memoria implicó desarrollar un trabajo ingenieril extenso que involucró a múltiples actores al interior del DCV representado por las siguientes áreas que tuvieron peticiones particulares.

- Arquitectura: Solicitud de estándares y cumplimiento de métricas de arquitectura.
- Desarrollo: Solicitud de estándares y presentaciones de tecnologías.
- Riesgo: Trazas y barreras de confidencialidad en determinadas ubicaciones de la aplicación.
- Servicio: Solicitudes desde el punto de vista del cliente que intervienen sobre la interfaz.

Habiendo identificado y extraído los requerimientos de los actores, se desarrolló un trabajo ingenieril iterativo que concluyó en validaciones que justificaron el inicio de este proyecto.

Este proyecto presenta un esquema base de solución de firma digital efectivo y otros proyectos podrían tomar esta solución e implementarla utilizando el contenido que en esta memoria se expone.

Los tiempos de respuestas son una mejora considerable producto de este trabajo. En ambiente de Producción se verán reflejados estos resultados en cuanto que son 4000 operaciones diarias que tendrán sus tiempos de respuesta mejorados.

Como trabajo futuro se propone mejorar la portabilidad del Applet Java haciendo que este funcione bajo plataformas basadas en Unix como Mac y Linux. Como antecedentes para esta solución, se adelanta que el mayor problema a enfrentar será el definir cuál será el almacén de claves en sistema operativos Linux, debido a que este no presenta un componente común para la administración de certificados como lo presenta Mac.

Bibliografía

[1] Oracle and/or its affiliates. Java API for RESTful Services (JAX-RS). disponible en: `http://jax-rs-spec.java.net/`, 2012.

[2] Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi, and Graham Steel. Attacking and fixing PKCS#11 security tokens. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 260–269, New York, NY, USA, 2010. ACM.

[3] Chile. Ley 19.799 SOBRE DOCUMENTOS ELECTRONICOS, FIRMA ELECTRONICA Y SERVICIOS DE CERTIFICACION DE DICHA FIRMA. disponible en: `http://www.leychile.cl/Navegar?idLey=19799`, June 2012.

[4] Chile. Ley 20.217 MINISTERIO DE ECONOMIA;FOMENTO Y RECONSTRUCCION;SUBSECRETARIA DE ECONOMIA;FOMENTO Y RECONSTRUCCION. disponible en: `http://www.leychile.cl/Navegar?idLey=20217`, June 2012.

[5] Luiz Alexandre Hiane da Silva Maciel and Celso Massaki Hirata. An optimistic technique for transactions control using REST architectural style. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 664–669, New York, NY, USA, 2009. ACM.

[6] Dan Davis and Manish P. Parashar. Latency Performance of SOAP Implementations. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '02, pages 407–, Washington, DC, USA, 2002. IEEE Computer Society.

[7] DCV. Depósito Central de Valores S.A. `http://www.dcv.cl`, June 2012.

[8] John P. Field, Stephen G. Graham, and Tom Maguire. A framework for obligation fulfillment in REST services. In *Proceedings of the Second International Workshop on RESTful Design*, WS-REST '11, pages 59–66, New York, NY, USA, 2011. ACM.

[9] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. AAI9980887.

[10] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317, December 1983.

[11] RSA Security Inc. PKCS#11: Cryptographic Token Interface Standard, v2.20. disponible en: http://www.rsa.com/, June 2004.

[12] Kostas Kontogiannis, Ying Zou, Chris Brealey, and Michael Athanasopoulos. Issues and challenges leveraging REST architectural style in enterprise service systems. In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '10, pages 368–370, Riverton, NJ, USA, 2010. IBM Corp.

[13] Microsoft. Alternatives to Using CAPICOM. disponible en: http://msdn.microsoft.com/en-us/library/windows/desktop/cc778518(v=vs.85).aspx, June 2012.

[14] Microsoft. Platform SDK Redistributable: CAPICOM. disponible en: http://www.microsoft.com/es-es/download/details.aspx?id=25281, June 2012.

[15] Gavin Mulligan and Denis Gračanin. A comparison of SOAP and REST implementations of a service based interaction independence middleware framework. In *Winter Simulation Conference*, WSC '09, pages 1423–1432. Winter Simulation Conference, 2009.

[16] Oracle. Java Cryptography Architecture (JCA) for Java Platform Standard Edition 6. disponible en: http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html, June 2011.

[17] Java Community Process. JSR 303: Bean Validation, version 1.0. disponible en: http://jcp.org/en/jsr/detail?id=303, November 2009.

[18] Java Community Process. JSR 330: Dependency Injection for Java, version 1.0. disponible en: http://www.jcp.org/en/jsr/detail?id=330, October 2009.

[19] Rational Software Corporation. Rational Unified Process (RUP). Technical report, Cupertino, May 2000.

[20] Moti Yung and Jonathan Katz. *Digital Signatures (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.