Universidad de las Ciencias Informáticas
Facultad 1

Título: Sistema de manejo de llaves y certificados digitales en dispositivos criptográficos para soluciones de emisión de documentos de identificación electrónicos.

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autores: Guillermo Rodríguez González.

Adrián Rivera Correa.

Tutores: Ing. Jose Enrique Saura Guerra.

Msc. Jorge Landrián García.

La Habana, __ de ___de 2011. "Año 54 de la Revolución".

"La ignorancia afirma o niega rotundamente, la ciencia duda."

Voltaire

DEDICATORIA

Adrian:

Dedico este trabajo:

- A mi tío Rodolfo y a mi papá, porque sus vidas me inspiraron a perseguir este objetivo.
- A mis padres, que han dedicado sus vidas a cuidar de mí y mis hermanas y han hecho un gran sacrificio para que me pudiera graduar.
- A mi hermana Olga, que tanto me ha cuidado y me ha ayudado en estos 5 años.
- A Angeliquín, que ha tenido que privarse de la presencia de sus hermanos en estos 5 años.
- A mi tío Juan José, por formar parte de esta familia y,
- A toda la familia en general.
- A Marita, que sin saberlo, fue una de las razones que me dio fuerzas para seguir.
- A todos los profesores que de buena fe exigieron más de mí y dieron más de ellos para educarme. En especial a: Isel Batista Najarro, Yeneit Delgado Kios y Joviel Macías Ortiz.
- A mis amigas y amigos que con su presencia me facilitaron el camino. ¡Cuando piense en la Universidad… pensaré en ustedes, cuando hable de la Universidad… hablaré de ustedes, porque ustedes han sido mi Universidad!

Guillermo:

Dedico este trabajo a mi madre, tú has sido mi única fuerza, si he resistido y vencido todo en esta vida ha sido por tu amor, tus consejos, tu ayuda, la confianza que has depositado en mí. Porque has sacrificado tu vida en favor nuestro. Me siento privilegiado por tenerte por madre, tú eres mi inspiración.

AGRADECIMIENTOS

Agradecemos a Adonis Cesar Legon Campos, Mairelys Boeras Velázquez, Felix Alejandro Prieto Carratala y Alina Suros Vicente por la ayuda brindada para la elaboración de este trabajo.

Adrián:

Agradezco a mis padres por el sacrificio realizado. Agradezco a Guillermo por su empeño y dedicación a esta tesis. Agradezco a Olga, Frank, Ronaldo, Mairelys y Keytia por su ayuda en la realización de este documento. Agradezco a Roberto Carlos Cardero Pérez por estar cuando necesité de un amigo. Agradezco a todos los que de una forma u otra me han ayudado en estos 5 años: profesores, amigos, etc.

Guillermo:

Agradezco primero que nada a Dios, por haber creado con tanto amor este planeta, esta tierra hermosa, esta especie, por allanarme el camino hasta este momento, por colmar de lindas emociones este instante, que para mí, pasa de ser tan solo el fin de una carrera para convertirse en el inicio de una nueva vida. Gracias a mi titánica madre, porque guiarme en la vida ha pasado a ser su razón de existir y porque de un amor genuino me ha colmado, el amor de madre. A mi hermano por su ayuda, amor y admiración, sí, porque sentirse admirado por un hermano ayuda mucho en esta vida. A mi esposa, gracias a quien hoy por segunda vez en 5 años me gradúo, tu paciencia, ayuda y amor para conmigo, han marcado mi corazón. Gracias a mis amigos, porque han sido protagonistas de mis triunfos cada uno en su debido tiempo: Carlos Eduardo Roig González (Desde la secundaria), Carlos Adian Sosa Calvo (Preuniversitario), Jorge David Plasencia Hernández (Universidad). Gracias a mi compañero de tesis, persona a quien admiro en esta vida por su capacidad, intelecto, tenacidad, bondad y muy importante, su paciencia conmigo, poco te falta para ser perfecto.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y al Centro de Identificación y Seguridad Digital (CISED) a hacer uso de la misma en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año_____.

Autores: Guillermo Rodríguez González Adrián Rivera Correa

_____ _____

Tutores: Ing. Jose Enrique Saura Guerra Msc. Jorge Landrián García

_____ _____

RESUMEN

La presente investigación se centra en el desarrollo de un sistema para el manejo de llaves y certificados digitales (KMS por sus siglas en inglés) en dispositivos criptográficos, que pueda ser utilizado en soluciones para la emisión de documentos de identificación electrónicos en el Centro de Identificación y Seguridad Digital (CISED) cumpliendo con los estándares de criptografía de llave pública: PKCS#11, PKCS#8 y PKCS#10.

Con el resultado de esta investigación el CISED contará con un sistema KMS que podrá integrar a soluciones para la emisión de documentos de identificación electrónicos y a otras soluciones que hagan uso de una infraestructura de llave pública (PKI por sus siglas en inglés) con el fin de gestionar de forma segura las llaves y los certificados digitales que se utilizan en estas soluciones.

Palabras Claves:

Criptografía, dispositivo criptográfico, HSM, KMS, PKCS, X509.

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE

INTRODUCCIÓN

La ciencia (o el arte) de proteger la información, así como ponerla al descubierto, se denomina **criptografía** y su origen es tan antiguo como la historia misma. Los seres humanos, a través de la historia, han inventado mecanismos para proteger los mensajes y ponerlos a salvo de ataques de intrusos. No pocos esfuerzos se han invertido en esta tarea, pues muchas veces lo que se desea proteger es de gran valor, como la identidad de un ser humano o la seguridad de una transacción comercial.

La criptografía es una necesidad del mundo moderno. Como actividad sistemática, organizada y de alcance social, su pasado es fuertemente militar y diplomático; sin embargo, con la automatización de las comunicaciones, la hegemonía de la Internet y la globalización económica y social, la criptografía se ha hecho indispensable en casi toda comunicación.

Los métodos de criptografía actuales basan su uso en una o más **llaves**. La llave es una secuencia de caracteres, que puede contener letras, dígitos y símbolos, y es convertida en un número, utilizada por los métodos de criptografía para codificar y decodificar mensajes (1). De acuerdo a la cantidad de llaves utilizadas, los métodos criptográficos pueden ser divididos en dos grandes categorías: **criptografía simétrica** y **criptografía asimétrica**. Los que utilizan criptografía simétrica, usan la misma llave para codificar y decodificar mensajes. Sin embargo, los que utilizan criptografía asimétrica usan dos llaves distintas, una para codificar y otra para decodificar. De esta forma, cada persona o entidad mantiene dos llaves: una **pública**, que puede ser divulgada libremente, y otra **privada**, que debe ser mantenida en secreto por su dueño.

Para asegurar que una llave pública pertenece a un usuario en concreto se utilizan los **certificados digitales**. Un certificado digital es un documento digital mediante el cual un tercero confiable (llamado autoridad de certificación o CA por sus siglas en inglés) garantiza la vinculación entre la identidad de un sujeto o entidad y una llave pública. (2)

La administración de llaves y certificados es una parte fundamental de los sistemas criptográficos, la cual se encarga de la generación, almacenamiento, seguridad, uso, verificación y reemplazo de llaves y certificados, además implementa los estándares que sirven de soporte a las funcionalidades

anteriormente mencionadas.

Dentro de los sistemas encargados de este tipo de actividades se encuentran los Sistemas de Administración de Llaves (KMS, por sus siglas en inglés), los cuales proporcionan además un canal seguro para la comunicación con dispositivos criptográficos: Módulos de Seguridad Hardware (HSM, por sus siglas en inglés), tarjetas criptográficas, entre otros, los cuales tienen entre sus responsabilidades: el almacenamiento de los objetos criptográficos, la aceleración de operaciones criptográficas, la generación de llaves, el cifrado y firma de datos.

En los últimos tiempos, el desarrollo de este tipo de sistemas en el mundo ha presentado un auge significativo. En su mayoría estos software son privativos, de código cerrado, diseñados para cumplir con los estándares y administrar llaves y certificados de forma centralizada sobre ambientes distribuidos, con gestión de roles, capaces de comunicarse a través de diferentes medios y con interfaces gráficas y por línea de comandos. Algunos son específicos para determinadas plataformas y determinados proyectos, volviéndolos soluciones a la medida.

En el CISED se desarrollan soluciones para la emisión de documentos de identificación, que incluyen el sistema de personalización para este tipo de documentos. La inclusión de documentos electrónicos en el sistema de personalización implica que durante este proceso se precise la firma del objeto de seguridad del documento, como única medida de seguridad de carácter obligatorio. Sin embargo no existe forma segura de gestionar las llaves necesarias, así como los certificados. Ni se cuenta con mecanismos de generación de solicitudes para obtener los certificados que permitan firmar los documentos.

Luego de un estudio de la situación antes descrita se plantea como **problema de investigación:**

¿Cómo gestionar llaves y certificados digitales en soluciones para la emisión de documentos de identificación electrónicos en el Centro de Identificación y Seguridad Digital?

Definiéndose como **objeto de estudio** los procesos de gestión de llaves y certificados digitales.

Para dar solución a la problemática planteada se ha definido como **objetivo general** de la investigación: desarrollar un sistema para el manejo de llaves y certificados digitales que pueda ser

INTRODUCCIÓN

utilizado en las soluciones para la emisión de documentos de identificación electrónicos en el Centro de Identificación y Seguridad Digital cumpliendo con los estándares de criptografía de llave pública (PKCS por sus siglas en inglés). Y como **objetivos específicos**:

- Realizar un estudio de soluciones KMS existentes en el mundo y en nuestro país. [Responsable: Adrián Rivera Correa]
- Identificar y aplicar los estándares asociados a los procesos de gestión de llaves y certificados digitales. [Responsable: Adrián Rivera Correa]
- Definir las principales características y funcionalidades del sistema a desarrollar. [Responsable: Guillermo Rodríguez González]
- Analizar y seleccionar las herramientas a utilizar en el desarrollo del sistema. [Responsable: Guillermo Rodríguez González]
- Realizar el diseño e implementación del KMS. [Responsables: Guillermo Rodríguez González, Adrián Rivera Correa]
- Realizar pruebas para verificar el correcto funcionamiento del sistema. [Responsables: Guillermo Rodríguez González, Adrián Rivera Correa]

Enmarcándose como **campo de acción** los procesos de gestión de llaves y certificados digitales en soluciones para la emisión de documentos de identificación electrónicos.

Para darle cumplimiento a los objetivos mencionado se han trazado las siguientes **tareas investigativas**:

- Identificación y análisis de las características de algunos de los KMS existentes. [Responsable: Adrián Rivera Correa]
- Análisis y caracterización de los estándares PKCS#11 (interfaz de dispositivos criptográficos), PKCS#10 (estándar de solicitud de certificación) y X.509 (estándar para infraestructura de llaves públicas). [Responsable: Adrián Rivera Correa]

INTRODUCCIÓN

- Definición y descripción de las características del sistema. [Responsable: Guillermo Rodríguez González]
- Análisis y caracterización de las herramientas, tecnologías y metodología a utilizar en el desarrollo del sistema. [Responsable: Guillermo Rodríguez González]
- Definición de la arquitectura del sistema. [Responsable: Guillermo Rodríguez González]
- Diseño del modelo de datos del sistema. [Responsable: Adrián Rivera Correa]
- Diseño e implementación de la capa de acceso a datos del KMS. [Responsable: Adrián Rivera Correa]
- Diseño e implementación del módulo de gestión de usuario. [Responsable: Guillermo Rodríguez González]
- Diseño e implementación de la capa de servicios. [Responsable: Guillermo Rodríguez González]
- Diseño e implementación de la aplicación cliente del KMS. [Responsable: Adrián Rivera Correa]
- Diseño e implementación del módulo de gestión de llaves y certificados. [Responsables: Guillermo Rodríguez González, Adrián Rivera Correa]
- Realización de las pruebas al sistema. [Responsables: Guillermo Rodríguez González, Adrián Rivera Correa]

Para desarrollar estas tareas se utilizarán los siguientes **métodos científicos**:

Dentro de los **métodos teóricos**:

- **Analítico – sintético**: para descomponer los procesos de gestión de llaves y certificados digitales en pequeños subprocesos y descubrir relaciones y características generales que pudieran existir entre estos.
- **Modelación**: para representar por medio de diagramas los conceptos asociados a los

procesos de gestión de llaves y certificados digitales, teniendo como resultado un mejor entendimiento de la posible solución a implementar.

Dentro de los **métodos empíricos**:

- **Entrevistas**: para recopilar información sobre los procesos de gestión de llaves y certificados digitales. Las entrevistas previstas serán no estructuradas, es decir, serán entrevistas en las cuales se define un tema pero no llevan un cuestionario rígido.

La presente investigación se realiza debido a la necesidad del CISED de un KMS genérico, del cual él sea propietario y pueda así adaptarlo y configurarlo para los diversos ambientes de proyectos y soluciones que en él se generan. Con el resultado de esta investigación el CISED contará con un KMS el cual podrá integrar a soluciones para la emisión de documentos de identificación electrónicos y a otras soluciones que hagan uso de una infraestructura de llave pública (PKI por sus siglas en inglés) con el fin de gestionar de forma segura las llaves y los certificados digitales que se utilizan en estas soluciones.

Estructuración del contenido

Capítulo 1: Fundamentación teórica: En este capítulo se presenta una fundamentación teórica de Criptografía, se mencionan características de los sistemas para el manejo de llaves y certificados digitales existentes a nivel internacional y se analizan las tecnologías, herramientas y la metodología que se usarán para la solución del problema.

Capítulo 2: Características del sistema: En este capítulo se presenta un modelo general del sistema, la vista de la arquitectura del sistema, una lista de las características a desarrollar en la aplicación y un plan de desarrollo en base a las características a desarrollar.

Capítulo 3: Diseño del sistema: En este capítulo se describe cómo será construido el sistema a través de diversos diagramas y sus descripciones. Se muestran los diagramas de clases agrupados según la capa de la aplicación a la que pertenecen y se hace un análisis de los patrones de diseño utilizados en cada uno de ellos. Se describen algunas de las tablas más importantes del modelo de datos. Además se presentan diagramas de secuencias de algunas de las funcionalidades críticas.

INTRODUCCIÓN

Capítulo 4: Implementación y Prueba del sistema: En este capítulo se abordan aspectos importantes de la implementación y prueba del sistema. Se describe el estilo de codificación utilizado, se muestra la distribución física del sistema entre los diferentes nodos de cómputo a través del diagrama de despliegue, se presenta el diagrama de componente que permite comprender la estructura del sistema y se muestran los resultados de las pruebas realizadas.

CAPÍTULO 1. Fundamentación Teórica

1.1 Introducción

En este capítulo se presentan algunos conceptos básicos necesarios para el posterior entendimiento de los temas tratados en este trabajo. Además se abordan aspectos, tales como: tipos de criptosistemas, dispositivos criptográficos, algoritmos de cifrado y estándares PKCS y X509, los cuales forman parte de los principios o cimientos en los que se apoya y se desarrolla la tesis. También se realiza un estudio de algunos de los KMS existentes en el mundo y se define la metodología de desarrollo y herramientas a utilizar.

1.2 Conceptos básicos

Según el diccionario de la Real Academia, la palabra **criptografía** proviene del griego *cripto*, que significa oculto, y *grafía*, que significa escritura, y su definición es: “Arte de escribir con llave secreta o de un modo enigmático”. (3)

Obviamente la criptografía hace años que dejó de ser un arte para convertirse en una técnica, o más bien un conglomerado de técnicas, que tratan sobre la protección de la información. Entre las disciplinas que engloba cabe destacar la Teoría de la Información, la Teoría de Números y la Complejidad Algorítmica. (4)

Cuando se está hablando de criptografía se utilizan términos tales como:

- **cifrar**, se refiere a la acción de aplicar técnicas criptográficas con el objetivo de esconder un mensaje.
- **descifrar**, es la acción que permite hacer entendible un mensaje cifrado.
- **texto plano o texto en claro**, es el mensaje legible inicial que se quiere transmitir.
- **algoritmo de cifrado**, es el conjunto de pasos que se realizan para cifrar el texto plano.
- **llave o clave**, es una secuencia de caracteres, que puede contener letras, dígitos y símbolos (como una contraseña), y que es convertida en un número, utilizada por los algoritmos de

cifrado para codificar y decodificar mensajes.

La criptografía se representa a través de un modelo al cual se le llama **criptosistema**. Un criptosistema es una quíntupla (M, C, K, E, D), donde:

- M representa el conjunto de todos los mensajes sin cifrar.
- C representa el conjunto de todos los posibles mensajes cifrados, o **criptogramas**.
- K representa el conjunto de llaves que se pueden emplear en el criptosistema.
- E es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de M para obtener un elemento de C. Existe una transformación diferente E_k para cada valor posible de la llave k.
- D es el conjunto de transformaciones de descifrado, análogo a E.

Todo criptosistema ha de cumplir la siguiente condición: $D_k(E_k(m)) = m$. (4)

Existen dos tipos fundamentales de criptosistemas, los **criptosistemas simétricos o de llave privada** y los **criptosistemas asimétricos o de llave pública**, los cuales serán abordados en otro epígrafe de este capítulo.

1.3 Problemas de seguridad que resuelve la criptografía

Entre los problemas de seguridad que resuelve la criptografía están:

- **privacidad**, se refiere a que la información sólo pueda ser leída por personas autorizadas. Es tal vez la más importante de las metas.
- **integridad**, se refiere a que la información no pueda ser alterada en el transcurso de ser enviada.
- **autenticidad**, se refiere a que se pueda confirmar que el mensaje recibido haya sido mandado por quien dice lo mandó o que el mensaje recibido es el que se esperaba.

- **no rechazo**, se refiere a que no se pueda negar la autoría de un mensaje enviado. (5)

1.4 Tipos de criptosistemas

1.4.1 Criptosistemas simétricos

Los criptosistemas simétricos, también conocidos como de llave única, basan su fortaleza en el secreto de una llave k. De modo que, si esta llave se llega a descubrir, hallar la información inicial a partir de la cifrada sería una tarea teóricamente fácil.

La llave k es secreta y está compartida por los dos usuarios, el transmisor y el receptor, y es así como se consigue la confidencialidad e integridad en este tipo de sistemas: solamente el receptor autorizado conocerá la llave con la que ha sido cifrado el mensaje; otro cualquiera al no conocerla, no podría interpretarlo. (Ver **figura 1.1**)

Este tipo de criptosistema presenta el inconveniente de que para ser empleado en comunicaciones la llave k debe estar tanto en el emisor como en el receptor, por lo que surge la interrogante de cómo transmitir la llave de forma segura.

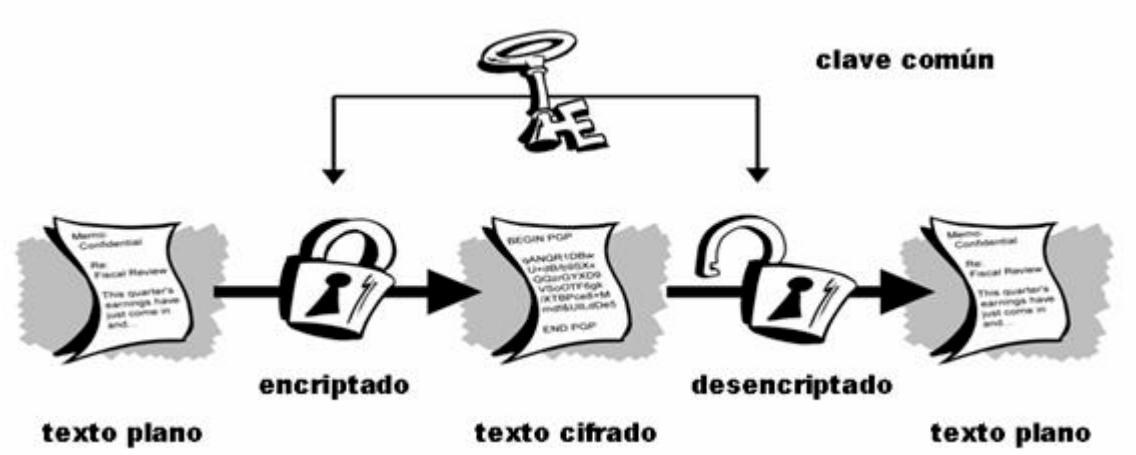


Figura 1.1 Criptosistema Simétrico. (6)

Para que un algoritmo de cifrado simétrico sea considerado fiable debe cumplir varios requisitos básicos:

1. Conocido el criptograma (texto cifrado) no se pueden obtener de él ni el texto en claro ni la llave.

2. Conocidos el texto en claro y el texto cifrado debe resultar más caro en tiempo o dinero descifrar la llave que el valor posible de la información obtenida por terceros.

Generalmente el algoritmo de cifrado es conocido, se divulga públicamente, por lo que la fortaleza del mismo dependerá de su complejidad interna y sobre todo de la longitud de la llave empleada, ya que una de las formas de criptoanálisis primario de cualquier tipo de sistema es la de prueba-ensayo o fuerza bruta, mediante la que se van probando diferentes llaves hasta encontrar la correcta. (7)

1.4.2 Criptosistemas asimétricos

Los criptosistemas asimétricos, también llamados de llave pública, se basan en el uso de dos llaves diferentes, llaves que poseen una propiedad fundamental: una puede descifrar lo que la otra ha cifrado. (Ver **figura 1.2**)

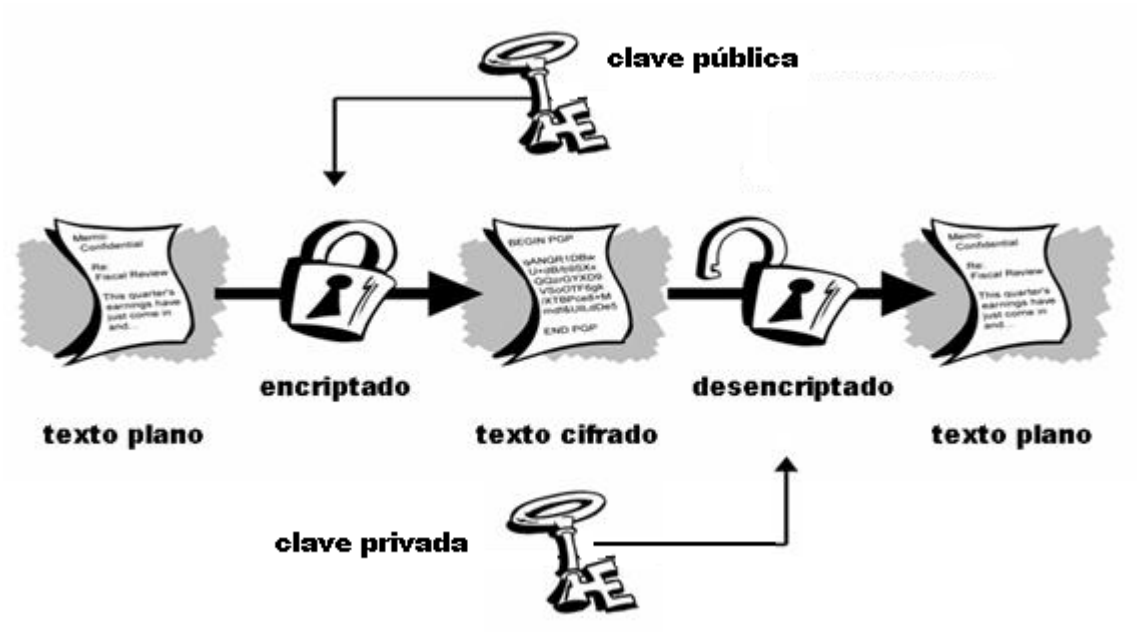


Figura 1.2 Criptosistema Asimétrico. (6)

Las llaves pública y privada tienen características matemáticas especiales. Estas se generan siempre a la vez, por parejas, estando cada una de ellas ligada intrínsecamente a la otra, de tal forma que si dos llaves públicas son diferentes, entonces sus llaves privadas asociadas también lo son, y viceversa. Los algoritmos asimétricos están basados en funciones matemáticas fáciles de resolver en un sentido, pero muy complicadas de realizar en sentido inverso, salvo que se conozca la llave privada, como la potencia y el logaritmo discreto.

La principal ventaja de los sistemas de llave pública frente a los simétricos es que la llave pública y el algoritmo de cifrado son o pueden ser de dominio público y que no es necesario poner en peligro la llave privada en tránsito por los medios inseguros, ya que esta siempre está oculta y en poder únicamente de su propietario. Además los sistemas asimétricos cuentan con una firma digital, esquema matemático que sirve para demostrar la autenticidad de un mensaje digital. Sin embargo, los sistemas de llave pública tienen como inconveniente que dificultan la implementación del sistema y son más lentos que los simétricos.

Para que un algoritmo de llave pública sea considerado seguro debe cumplir:

1. Conocido el texto cifrado no debe ser posible encontrar el texto en claro ni la llave privada.
2. Conocido el texto cifrado (criptograma) y el texto en claro debe resultar más caro en tiempo o dinero descifrar la llave que el valor posible de la información obtenida por terceros.
3. Conocida la llave pública y el texto en claro no se puede generar un criptograma correcto cifrado con la llave privada.
4. Dado un texto cifrado con una llave privada sólo existe una pública capaz de descifrarlo, y viceversa. (7)

1.4.3 Criptosistemas híbridos

En la práctica la mayoría de los sistemas utilizan un criptosistema híbrido. Un criptosistema híbrido no es más que la combinación de un criptosistema simétrico y uno asimétrico. Este usa la llave pública del receptor para cifrar una llave simétrica que se usará en el proceso de comunicación cifrada. De esta forma se aprovechan las ventajas de ambos sistemas, usando el sistema asimétrico para el

envío de la llave sensible y el simétrico, con mayor velocidad de proceso, para el envío masivo de datos.

Ejemplos que combinan la utilización de los métodos de criptografía de llave única y de llave pública y privada son las conexiones seguras, establecidas entre el navegador de un usuario y una web, en transacciones comerciales o bancarias vía web.

Estas conexiones seguras vía web utilizan el método de criptografía de llave única, implementado por el protocolo SSL (Secure Socket Layer). El navegador del usuario necesita informar a la web cual será la llave única utilizada en la conexión segura, antes de iniciar una transmisión de datos sigilosos.

Para esto, el navegador obtiene la llave pública del certificado de la institución que mantiene la web. Entonces, utiliza esta llave pública para codificar y enviar un mensaje a la web, conteniendo la llave única a ser utilizada en la conexión segura. La web utiliza su llave privada para decodificar el mensaje e identificar la llave única que será utilizada.

A partir de este punto, el navegador del usuario y la web pueden transmitir informaciones, de forma sigilosa y segura, a través de la utilización del método de criptografía de llave única. La llave única puede ser cambiada a intervalos de tiempo determinados, a través de la repetición de procedimientos descritos anteriormente, aumentando así el nivel de seguridad de todo el proceso. (1)

1.5 Dispositivos criptográficos

Las llaves criptográficas deben ser protegidas de la exposición. En aplicaciones reales, a menudo son protegidas por dispositivos criptográficos que emplean sofisticadas medidas de seguridad de hardware.

Los dispositivos criptográficos o módulos de seguridad hardware (HSM, por sus siglas en inglés) son una parte importante de muchas de las infraestructuras de seguridad que utilizan criptografía. Ellos protegen las llaves criptográficas en entornos hostiles usando medidas de protección de manipulación física. Los dispositivos criptográficos almacenan llaves y usan estas para operaciones criptográficas, pero las llaves generalmente no salen del perímetro de seguridad físico establecido por el hardware de seguridad. Estos dispositivos se utilizan porque se pueden controlar mejor que las computadoras

que llaman a las operaciones sobre ellos.

Un dispositivo criptográfico actúa como un dispositivo auxiliar para una aplicación que se ejecuta desde una computadora. Este realiza varias tareas criptográficas, tales como: la generación de llaves, la derivación de llaves, cifrado, descifrado, firma, verificación de la firma, etc. Los dispositivos criptográficos también aplican una política de seguridad en las llaves almacenadas y otros objetos criptográficos. Cuando una aplicación invoca un comando del dispositivo, esta operación puede acceder a objetos suministrados por la misma aplicación o a otros objetos almacenados en el dispositivo. Sin embargo, la aplicación no siempre puede obtener todos los resultados de la operación, como por ejemplo, cuando una llave es generada y almacenada sólo en el dispositivo. La política de seguridad de la interfaz del dispositivo define cuáles operaciones son permitidas y a quién. Los dispositivos por lo general tienen al menos dos niveles de seguridad, uno para la aplicación de carga, que tiene privilegios restringidos, y otro para el administrador, que puede acceder a todas las llaves en el dispositivo. (8)

Muchos sistemas HSM son también aceleradores criptográficos. Los dispositivos HSM no son sólo periféricos locales de un ordenador, algunas compañías ofrecen hardware HSM con conectividad de red para la protección de material residente en múltiples sistemas conectados.

Dispositivos criptográficos existen hoy en día de todo tipo y forma. Algunos como los ePass (ver **figura 1.3**) y BioPass se conectan a la computadora por puerto USB, otros se integran al hardware de la computadora por puerto PCI y otros como las tarjetas inteligentes necesitan lectores que las reconozcan.

Figura 1.3 Token ePass. (9)

1.5.1 Fabricantes de HSM

Algunas de las empresas fabricantes de HSM son:

- ARX, el producto que comercializa lo hace llamar **PrivateServer**, este dispositivo tiene una infraestructura flexible y segura, lo que permite a los clientes y socios desarrollar módulos personalizados (utilizando .NET) que se ejecutan dentro del HSM. (10)
- THALES, comercializa los productos **HSM 8000** y **nShield**. (11)
- SafeNet. (12)
- Futurex, comercializa **Excrypt™SSP9000 Enterprise**. (13)
- Oracle, comercializa el producto **Sun Crypto Accelerator 6000 PCIe Card**. (14)

1.6 La criptografía y sus algoritmos de cifrado

1.6.1 Algoritmos simétricos

1.6.1.1 Redes de Feistel

La red de Feistel es una estructura que se utiliza generalmente en los algoritmos de cifrado simétrico en bloque, es decir, algoritmos que dividen el texto plano en bloques de longitud fija. Este algoritmo se denomina simétrico por rondas, es decir, realiza siempre las mismas operaciones un número

determinado de veces (denominadas rondas), lo que permite que la implementación del algoritmo en hardware sea sencilla.

Este algoritmo tiene como entrada un bloque de m bits, donde m es par, y una llave K de longitud n. Es necesario que m sea par porque en la primera ronda del algoritmo, se divide el bloque de m bits en dos partes, a la parte izquierda se le denota como L y a la parte derecha como R (ver **figura 1.4**). A partir de esta etapa se realiza un cifrado de producto iterativo en el que la salida de cada ronda se usa como entrada para la siguiente según la relación de recurrencia que se muestra en la **ecuación 1.1**.

$$\left.\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f\left(R_{i-1}, K_i\right) \end{aligned}\right\} \quad si \quad i < n$$

$$\begin{aligned} L_n &= L_{n-1} \oplus f\left(R_{n-1}, K_n\right) \\ R_n &= R_{n-1} \end{aligned}$$

Ecuación 1.1 Red de Feistel.

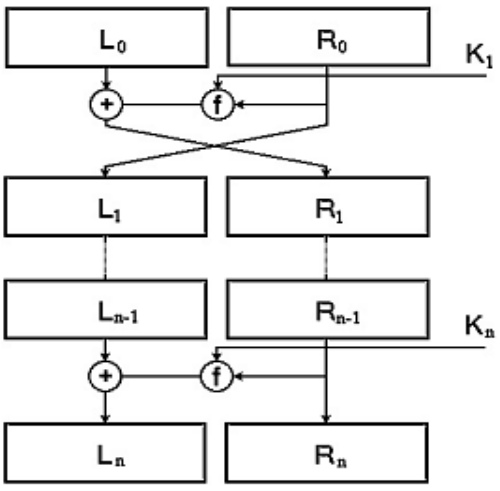


Figura 1.4 Estructura de una red de Feistel.

Este tipo de estructura tiene la interesante propiedad de ser reversible, independientemente de la función f que utilice, para ello basta con aplicar de nuevo el algoritmo al resultado, pero empleando

K_i en orden inverso. Esto permite utilizar el mismo mecanismo tanto para cifrar como para descifrar. (4)

Demostración de la reversibilidad de la red de Feistel

Lo que permite que la estructura de la red de Feistel sea reversible es el uso de la operación XOR, que se expresa con el símbolo $\oplus$. Esta operación tiene la siguiente tabla de verdad:

A	B	Resultado
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 1.1 Tabla de verdad XOR.

Y presenta la siguiente propiedad $(A \oplus B) \oplus B = A$. Utilizando esta propiedad y las ecuaciones de recurrencias planteadas anteriormente (ver **ecuación 1.1**), a continuación se demuestra la reversibilidad de la estructura. Se denota L', R' y K' como los bloques y la llave en el proceso de descifrado y L, R y K como los bloques y la llave en el proceso de cifrado. Además es importante destacar que la llave en el proceso de descifrado se invierte, es decir, $K_{n-i}' = K_{i+1}$.

Al comenzar el proceso de descifrado (1) $L_0' = L_n$ (2) $R_0' = R_n$.

De acuerdo con las ecuaciones de recurrencia para $i = n$:

(3) $L_n = L_{n-1} \oplus f(R_{n-1}, K_n)$ (4) $R_n = R_{n-1}$

Sustituyendo (3) en (1) y sustituyendo (4) en (2):

(5) $L_0' = L_{n-1} \oplus f(R_{n-1}, K_n)$ (6) $R_0' = R_{n-1}$

En la próxima iteración de descifrado:

(7) $L_1' = R_0'$ (8) $R_1' = L_0' \oplus f(R_0', K_1')$

(9) $K_1' = K_n$ por $K_{n-i}' = K_{i+1}$.

Sustituyendo (6) en (7) y sustituyendo (5), (6) y (9) en (8):

(10) $L_1' = R_{n-1}$ (11) $R_1' = \left[L_{n-1} \oplus f(R_{n-1}, K_n)\right] \oplus f(R_{n-1}, K_n) = L_{n-1}$

Repitiendo estas iteraciones, se puede llegar a una ecuación general:

(12) $\left.\begin{array}{l} L_i' = R_{n-i} \\ R_i' = L_{n-i} \end{array}\right\}$ $si \quad i < n$

Penúltima iteración de descifrado, $i = n-1$:

(13) $L_{n-1}' = R_{n-(n-1)} = R_1$ (14) $R_{n-1}' = L_{n-(n-1)} = L_1$

Última iteración de descifrado:

(15) $L_n' = L_{n-1}' \oplus f(R_{n-1}', K_n')$ (16) $R_n' = R_{n-1}'$

(17) $K_n' = K_1$ por $K_{n-i}' = K_{i+1}$.

Sustituyendo (13), (14) y (17) en (15) y sustituyendo (14) en (16):

(18) $L_n' = R_1 \oplus f(L_1, K_1)$ (19) $R_n' = L_1$

De acuerdo con las ecuaciones de recurrencia para $i = 1$:

(20) $L_1 = R_0$ (21) $R_1 = L_0 \oplus f(R_0, K_1)$

Sustituyendo (20) y (21) en (18) y sustituyendo (20) en (19):

$$L_n' = \left[L_0 \oplus f(R_0, K_1)\right] \oplus f(R_0, K_1) = L_0 \qquad R_n' = R_0$$

Como puede notar al final del proceso de descifrado se obtiene la cadena original, es decir, la cadena inicial en el proceso de cifrado.

1.6.1.2 DES (Data Encryption Standard)

Es el algoritmo simétrico más extendido mundialmente. Data de mediados de los setenta, cuando fue adoptado como estándar para las comunicaciones seguras por el Gobierno de los EE.UU. En realidad la Agencia de Seguridad Nacional del Gobierno de los EE.UU (NSA) lo diseñó para ser implementado por hardware, con la intención de mantenerlo en secreto, pero al parecer por un malentendido entre ellos y la Oficina Nacional de Estandarización, su especificación se hizo pública con suficiente detalle como para que cualquiera pudiera implementarlo por software. No fue casualidad que el siguiente algoritmo adoptado (Skipjack) fuera mantenido en secreto.

A mediados de 1998, se demostró que un ataque por la fuerza bruta a DES era viable, debido a la escasa longitud que emplea en su llave. No obstante, el algoritmo aún no ha demostrado ninguna debilidad grave desde el punto de vista teórico, por lo que su estudio sigue siendo plenamente interesante.

El algoritmo DES codifica bloques de 64 bits empleando llaves de 56 bits. Es una red de Feistel de 16 rondas, más dos permutaciones, una que se aplica al principio P_i y otra que se aplica al final P_f, tal que $P_i = P_f^{-1}$. (4)

La función F de Feistel

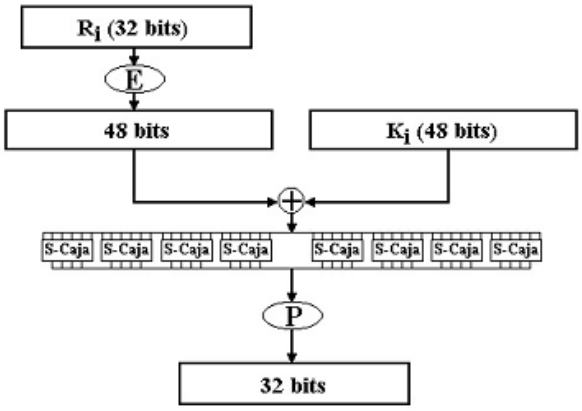


Figura 1.5 Esquema de la función f del algoritmo DES.

La función f que se utiliza en la red de Feistel (ver **ecuación 1.1**), representada en la **figura 1.5**, opera sobre medio bloque (32 bits) cada vez y consta de cuatro pasos:

1. **Expansión** – la mitad del bloque (32 bits) se expande a 48 bits mediante la permutación de expansión, denominada *E* en el diagrama, duplicando algunos de los bits.

2. **Mezcla** — el resultado se combina con una subllave utilizando una operación XOR. Dieciséis subllaves — una para cada ronda — se derivan de la llave inicial mediante la generación de subllaves descrita más abajo.

3. **Sustitución** — tras mezclarlo con la subllave, el bloque es dividido en ocho trozos de 6 bits antes de ser procesados por las S-cajas, o cajas de sustitución. Cada una de las ocho S-cajas reemplaza sus seis bits de entrada con cuatro bits de salida, de acuerdo con una trasformación no lineal, especificada por una tabla de búsqueda. Las S-cajas constituyen el núcleo de la seguridad de DES — sin ellas, el cifrado sería lineal, y fácil de romper.

4. **Permutación** — finalmente, las 32 salidas de las S-cajas se reordenan de acuerdo a una permutación fija; la P-caja.

Alternando la sustitución de las S-cajas, la permutación de bits de la P-caja y la expansión-E proporcionan las llamadas “confusión y difusión” respectivamente, un concepto identificado por Claude Shannon en los 40 como una condición necesaria para un cifrado seguro y práctico. (4)

Generación de subllaves

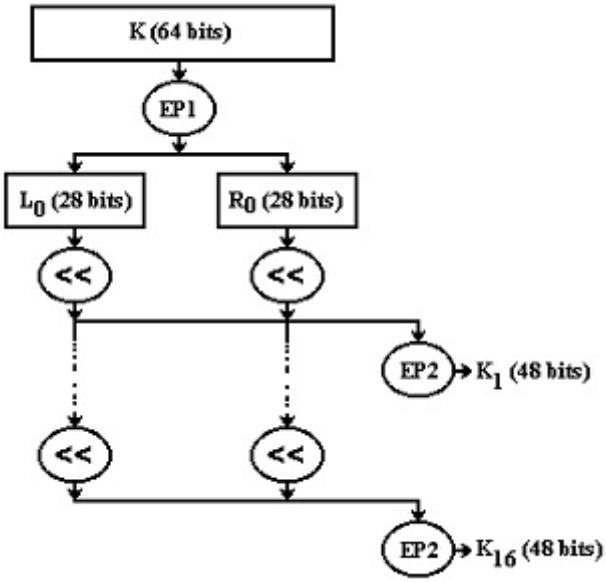


Figura 1.6 Generación de subllaves.

A partir de la llave K inicial se generan 16 subllaves, una para cada ronda de la red de Feistel. Primero, se seleccionan 56 bits de la llave de los 64 bits iniciales mediante la *Elección Permutada 1* (*EP1*), los otros 8 bits restantes se utilizan como bit de paridad, uno por cada byte de la llave inicial. Luego los 56 bits se dividen en dos mitades de 28 bits y se lleva a cabo desplazamientos a la izquierda de cada una de las dos mitades y se realiza una *Elección Permutada 2 (EP2)* de 48 bits en cada ronda, obteniendo así K_i. Los desplazamientos a la izquierda son de dos bits, salvo para las rondas 1, 2, 9 y 16, en las que se desplaza sólo un bit. (4)

1.6.1.3 IDEA

El algoritmo IDEA (International Data Encryption Algorithm) es más joven que DES, pues data de 1992. Para muchos constituye el mejor y más seguro algoritmo simétrico disponible en la actualidad.

Trabaja con bloques de 64 bits de longitud y emplea una llave de 128 bits. Como en el caso de DES, se usa el mismo algoritmo tanto para cifrar como para descifrar.

IDEA es un algoritmo bastante seguro, y hasta ahora se ha mostrado resistente a todos los ataques, entre ellos el criptoanálisis diferencial. La longitud de su llave hace imposible en la práctica un ataque por la fuerza bruta. El algoritmo hace uso de las siguientes operaciones elementales:

- XOR.
- Suma de módulo 2^{16}.
- Producto módulo $2^{16}+1$.

El algoritmo IDEA consta de ocho rondas. Divide el bloque X a codificar, de 64 bits, en cuatro partes X_1, X_2, X_3 y X_4 de 16 bits. Utiliza 52 subllaves de 16 bits que se denotarán como Z_i. Las operaciones que lleva a cabo en cada ronda son las siguientes:

1. Multiplicar X_1 por Z_1.
2. Sumar X_2 con Z_2.
3. Sumar X_3 con Z_3.
4. Multiplicar X_4 con Z_4.
5. Hacer XOR entre los resultados del paso 1 y el paso 3.
6. Hacer XOR entre los resultados del paso 2 y el paso 4.
7. Multiplicar el resultado del paso 5 por Z_5.
8. Sumar los resultados de los pasos 6 y 7.
9. Multiplicar el resultado del paso 8 por Z_6.
10. Sumar los resultados de los pasos 7 y 9.
11. Hacer un XOR entre los resultados de los pasos 1 y 9.
12. Hacer un XOR entre los resultados de los pasos 3 y 9.
13. Hacer un XOR entre los resultados de los pasos 2 y 10.
14. Hacer un XOR entre los resultados de los pasos 4 y 10.

La salida de cada iteración serán los cuatro sub-bloques obtenidos en los pasos 11, 12, 13 y 14, que serán la entrada del siguiente ciclo, en el que se empleará las siguientes seis subllaves, hasta un total de 48. Al final de todo se intercambiará los dos bloques centrales. Después de la octava iteración, se realizan las siguientes transformaciones:

1. Multiplicar X_1 por Z_{49}.
2. Sumar X_2 con Z_{50}.
3. Sumar X_3 con Z_{51}.
4. Multiplicar X_4 por Z_{52}. (4)

Generación de subllaves

Las primeras ocho subllaves se calculan dividiendo la llave de entrada en bloques de 16 bits. Las siguientes ocho se calculan rotando la llave de entrada 25 bits a la izquierda y volviendo a dividirla, y así sucesivamente. Las subllaves necesarias para descifrar se obtienen cambiando de orden las Z_i y calculando sus inversas para la suma o la multiplicación, según la tabla que se muestra en la **figura 1.7**.

Ronda	Subclaves de Cifrado						Subclaves de Descifrado					
1	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_{49}^{-1}	$-Z_{50}$	$-Z_{51}$	Z_{52}^{-1}	Z_{47}	Z_{48}
2	Z_7	Z_8	Z_9	Z_{10}	Z_{11}	Z_{12}	Z_{43}^{-1}	$-Z_{45}$	$-Z_{44}$	Z_{46}^{-1}	Z_{41}	Z_{42}
3	Z_{13}	Z_{14}	Z_{15}	Z_{16}	Z_{17}	Z_{18}	Z_{37}^{-1}	$-Z_{39}$	$-Z_{38}$	Z_{40}^{-1}	Z_{35}	Z_{36}
4	Z_{19}	Z_{20}	Z_{21}	Z_{22}	Z_{23}	Z_{24}	Z_{31}^{-1}	$-Z_{33}$	$-Z_{32}$	Z_{34}^{-1}	Z_{29}	Z_{30}
5	Z_{25}	Z_{26}	Z_{27}	Z_{28}	Z_{29}	Z_{30}	Z_{25}^{-1}	$-Z_{27}$	$-Z_{26}$	Z_{28}^{-1}	Z_{23}	Z_{24}
6	Z_{31}	Z_{32}	Z_{33}	Z_{34}	Z_{35}	Z_{36}	Z_{19}^{-1}	$-Z_{21}$	$-Z_{20}$	Z_{22}^{-1}	Z_{17}	Z_{18}
7	Z_{37}	Z_{38}	Z_{39}	Z_{40}	Z_{41}	Z_{42}	Z_{13}^{-1}	$-Z_{15}$	$-Z_{14}$	Z_{16}^{-1}	Z_{11}	Z_{12}
8	Z_{43}	Z_{44}	Z_{45}	Z_{46}	Z_{47}	Z_{48}	Z_7^{-1}	$-Z_9$	$-Z_8$	Z_{10}^{-1}	Z_5	Z_6
Final	Z_{49}	Z_{50}	Z_{51}	Z_{52}			Z_1^{-1}	$-Z_2$	$-Z_3$	Z_4^{-1}		

Figura 1.7 Tabla de subllaves para cifrado y descifrado.

Puesto que $2^{16}+1$ es un número primo, nunca se obtendrá cero como producto de dos números, por lo que no se necesitará representar dicho valor. Cuando se esté calculando productos, se utilizará el cero para expresar el número 2^{16}. (4)

1.6.2 Algoritmos asimétricos

1.6.2.1 RSA

De entre todos los algoritmos asimétricos, quizá RSA sea el más sencillo de comprender e implementar. Sus pares de llaves son duales, por lo que sirve tanto para codificar como para autentificar. Su nombre proviene de sus tres inventores: Ron Rivest, Adi Shamir y Leonard Adleman. Desde su nacimiento nadie ha conseguido probar o rebatir su seguridad, pero se le tiene como uno de los algoritmos asimétricos más seguros.

RSA se basa en la dificultad para factorizar grandes números. Las llaves pública y privada se calculan a partir de un número que se obtiene como producto de dos primos grandes. El atacante se enfrentará, si quiere recuperar un texto plano a partir del criptograma y la llave pública, a un problema de factorización.

Para generar un par de llaves privada y pública, en primer lugar se escogen aleatoriamente dos números primos grandes, p y q. Después se calcula el producto $n = pq$. Se escogerá ahora un número e primo relativo con $(p-1)(q-1)$. (e,n) será la llave pública. Nótese que e tiene inversa módulo $(p-1)(q-1)$ por lo que existirá un número d tal que, $de \equiv 1(\mathrm{mod}(p-1)(q-1))$ es decir, d es la inversa de e módulo $(p-1)(q-1)$. (d,n) será la llave privada. Esta inversa puede calcularse fácilmente empleando el algoritmo extendido de Euclides. Nótese que si se desconoce los factores de n, este cálculo resulta prácticamente imposible.

La codificación se lleva a cabo según la expresión: $c = m^e \,(\mathrm{mod}\, n)$, mientras que la descodificación se realiza de la siguiente forma $m = c^d \,(\mathrm{mod}\, n)$ (m es la representación binaria del texto en claro y c es el texto cifrado). (4)

Primos fuertes

Aunque p y q sean primos grandes, existen algunos casos en los que es relativamente fácil factorizar el número $n = pq$. Se proponen entonces una serie de condiciones para p y q que dificultan la factorización de n. Se dice que p y q son números primos fuertes si cumplen:

- El máximo común divisor de $p-1$ y $q-1$ debe ser pequeño.
- $p-1$ y $q-1$ deben tener algún factor primo grande p' y q'.
- Tanto $p'-1$ como $q'-1$ debe tener factores primos grandes.
- Tanto $p'+1$ como $q'+1$ deben tener factores primos grandes.

Las dos primeras condiciones se cumplen si tanto $\frac{(p-1)}{2}$ como $\frac{(q-1)}{2}$ son números primos. (4)

1.6.2.2 ElGamal

El algoritmo ElGamal fue diseñado en un principio para producir firmas digitales, pero posteriormente se extendió también para codificar mensajes. Se basa en el problema de los algoritmos discretos, que está íntimamente relacionado con el de la factorización de números enteros.

Para generar un par de llaves, se escoge un número primo p y dos números aleatorios g y x, tal que $1 < g < p$ y $1 < x < p$. Se calcula entonces $y = g^x \pmod p$. La llave pública será (g, y, p) y la privada x. (4)

Firma Digital de ElGamal

Para firmar un mensaje m basta con escoger un número k aleatorio, tal que $mcd(k, p-1) = 1$, y calcular $a = g^x \pmod p$, luego se emplea el algoritmo extendido de Euclides para resolver la ecuación $b = (H(m) - xa) * k^{-1} \pmod{p-1}$, donde la función H es la función de hash, Si $b = 0$ se vuelve a iniciar el proceso. La firma la constituye el par (a, b) y se verifica comprobando que $y^a a^b = g^m \pmod p$. (4)

Cifrado y descifrado

Para codificar el mensaje m se escoge primero un número aleatorio k primo relativo con $(p-1)$,

que también se mantendrá en secreto. Se calcula entonces las siguientes expresiones $a = g^k \pmod p$, $b = y^k m \pmod p$. El par (a,b) es el texto cifrado, de doble longitud que el texto original. Para decodificar se calcula $m = b/a^x \pmod p$. (4)

1.6.2.3 Criptografía con curvas elípticas

En los últimos años, la criptografía con curvas elípticas ha adquirido una creciente importancia, llegando a formar parte de estándares industriales. Si bien se han diseñado variantes con curvas elípticas de criptosistemas clásicos, como el RSA, su principal logro se ha conseguido en los criptosistemas basados en el problema del logaritmo discreto, como los del tipo ElGamal. En este caso, los criptosistemas elípticos garantizan la misma seguridad que los construidos sobre el grupo multiplicativo de un cuerpo finito primo (explicados anteriormente), pero con longitudes de llave mucho menores. (15)

1.7 Estándares PKCS y X.509

Los estándares de criptografía de llave pública (PKCS, por sus siglas en inglés) son las especificaciones elaboradas por los laboratorios RSA, en colaboración con los desarrolladores de sistemas de seguridad en todo el mundo con el propósito de acelerar la implementación de la criptografía de llave pública. Publicado por primera vez en 1991 como resultado de las reuniones con un pequeño grupo de los primeros en adoptar tecnologías de llave pública, los documentos PKCS han sido ampliamente referenciados. (16)

En este apartado se describirán algunos de los estándares que serán utilizados en el sistema a desarrollar.

1.7.1 PKCS#11 Estándar de dispositivo criptográfico

Esta norma especifica una API[1], llamada Cryptoki, para los dispositivos que contienen información criptográfica y realizan funciones criptográficas. Cryptoki sigue un enfoque orientado a objeto y tiene como fin la independencia de la tecnología (cualquier tipo de dispositivo) y el intercambio de recursos (múltiples aplicaciones tienen acceso a múltiples dispositivos), presentando una vista lógica, común

[1] Application Programming Interface (API), conjunto de funciones y procedimientos que ofrece cierta biblioteca de clases.

de los dispositivos criptográficos. (16)

1.7.2 PKCS#10 Estándar de solicitud de certificación

Esta norma especifica cómo debe ser la sintaxis para la solicitud de certificación. Una solicitud de certificación se compone de tres partes: la información de solicitud de certificación, un identificador del algoritmo de firma, y una firma digital de la información de la solicitud de certificación. La información de la solicitud de certificación está compuesta del nombre distinguido de la entidad, una llave pública de la entidad, y opcionalmente un conjunto de atributos. Las solicitudes de certificación son enviados a una autoridad de certificación, que transforma la solicitud en un certificado de llave pública X.509.

La intención de incluir un conjunto de atributos tiene un doble objetivo: ofrecer información sobre una determinada entidad o un "desafío de contraseña" por el cual la entidad más adelante puede pedir la revocación del certificado, y proporcionar atributos para su inclusión en los certificados X.509. (17)

1.7.3 X.509

X.509 especifica, entre otras cosas, el formato estándar para certificados de llaves públicas y un algoritmo de validación de la ruta de certificación. Su sintaxis, se define empleando el lenguaje ASN.1, y los formatos de codificación más comunes son DER o PEM. X.509 incluye también estándares para implementación de listas de certificados en revocación (CRLs). (18)

1.8 Análisis de KMS

En el mundo existen muchas empresas que utilizan sistemas KMS o se dedican a comercializarlos. En este epígrafe se caracteriza brevemente algunos KMS que se han tomado como referencia para el estudio de este tipo de software.

1.8.1 Bundes KMS

El KMS de la empresa alemana Bundes Druckerei es un software propietario fabricado para ser utilizado en el proceso de personalización de documentos de identificación electrónica. Este software permite gestionar las llaves y certificados digitales que se necesitan para cifrar y firmar los datos almacenados en el chip del documento de identificación electrónica. Además permite interactuar con

dispositivos criptográficos de diferentes fabricantes, fue desarrollado siguiendo los estándares PKCS y permite la gestión de roles.

1.8.2 Crypto Key Management Station

Este KMS es una solución propietaria desarrollada por la Sun Microsystems que se integra solamente con dispositivos desarrollados por ellos. Trabaja sobre el sistema operativo Solaris 10. Además presenta una interfaz por línea de comandos e interfaz gráfica de usuario. Permite la gestión de roles y usuarios y permite configurar otro sistema KMS (específicamente un Crypto Key Management Station) en forma de backup.

1.8.3 Oracle Key Manager

Este KMS es una solución propietaria desarrollada por Oracle que presenta control de acceso basado en roles, puede funcionar como un clúster de nodos KMS con sistema de backup lo que permite una alta disponibilidad en caso de falla. Este software fue desarrollado siguiendo los estándares PKCS, soporta dispositivos criptográficos de diferentes fabricantes y presenta una interfaz por línea de comandos y una interfaz gráfica.

1.8.4 KMS en Cuba

A partir de la investigación realizada, no se tiene referencia de que en Cuba se produzca este tipo de software y tampoco se conoce de la existencia a nivel mundial de alguna solución KMS libre. Sin embargo, en la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Identificación y Seguridad Digital (CISED), se utiliza el KMS fabricado por la empresa alemana Bundes Druckerei.

A partir del estudio realizado de soluciones KMS, se llega a la conclusión, de que ninguna de las soluciones estudiadas, cumple totalmente con las necesidades del CISED, ya que se requiere de una solución genérica, de código abierto y del cual el CISED pueda hacer uso, para adaptarla y configurarla a los diversos ambientes de proyectos y soluciones que en él se generan.

1.9 Metodologías de desarrollo de software

Para desarrollar un software se necesita una guía que permita controlar el proceso de construcción

del software, siempre teniendo en cuenta: las características del equipo, el entorno, los tiempos de entrega, etc. A estas guías se le denominan metodologías de desarrollo de software.

Las metodologías definen Quién debe hacer Qué, Cuándo y Cómo debe hacerlo (19). Por tanto, una metodología es un proceso. Estas engloban procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software.

Si se toma como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, se pueden clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. Por otra parte, considerando su filosofía de desarrollo, se pueden clasificar en dos grupos: Metodologías Tradicionales (o Pesadas) y Metodologías Ágiles.

Es esta última clasificación la que se tiene en cuenta en el desarrollo del presente trabajo. Las metodologías tradicionales son aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos, tienen una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Sin embargo, las metodologías ágiles están más orientadas a la generación de código con ciclos cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

1.9.1 Selección de la metodología de desarrollo de software

En el **anexo 1** se realiza un análisis entre una metodología tradicional y dos ágiles, en el primer caso se analizó el Proceso Unificado de Rational, (RUP por sus siglas en inglés), en el segundo caso se analizaron Programación Extrema, *eXtreme Programming* (XP), y Desarrollo Guiado por la Funcionalidad, *Feature Driven Development* (FDD).

Se decide usar FDD debido a que la cantidad de modelos y documentación que genera son aceptables teniendo en cuenta el tiempo de desarrollo del sistema, así como los recursos, y los cambios constantes en las funcionalidades. Además es a medio camino entre el desarrollo y la organización, no define explícitamente la obtención de requisitos, sino el proceder a partir de que se

han capturado dichos requisitos de la forma que el equipo de desarrollo haya estimado y propone implementar en grupo y no en pareja con un solo ordenador como propone XP. Todos estos aspectos hicieron decidirse por FDD, ya que permitirá desarrollar y documentar lo necesario en tiempo relativamente corto.

1.10 Análisis y selección de herramientas a utilizar en el proceso de desarrollo

1.10.1 Lenguaje de programación

Se analizaron los lenguajes de programación C# y Java. Ambos lenguajes adoptan mucho de la sintaxis de C y C++, pero tienen un modelo de objetos más sencillo eliminando herramientas de bajo nivel, como la manipulación directa de punteros y memoria, siendo esta última gestionada por el lenguaje y no por el programador de forma explícita.

C# cuenta con las bibliotecas de clases “NCryptoki” y “pkcs11.net” que son implementaciones del estándar PKCS#11 que facilitan la interacción con los dispositivos criptográficos; pero en el momento de esta investigación sólo se encontró la posibilidad de descargar versiones de prueba de estas bibliotecas. Sin embargo, Java tiene incluido una implementación nativa de PKCS#11 disponible como parte de la Arquitectura Criptográfica de Java (JCA por sus siglas en inglés) y la Extensión Criptográfica de Java (JCE por sus siglas en inglés) desde la versión 5, característica que determinó la selección de este lenguaje para el desarrollo de la aplicación.

Existen otras características que convierten a Java en una buena opción: Java es software libre, es un lenguaje ideado para sistemas distribuidos y cuenta con un conjunto de bibliotecas de clases para el trabajo en red. Además Java es especialmente conocido por ser muy robusto y seguro, bueno para crear software altamente fiable ya que ha sido implementado con numerosas barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real. Es independiente de la plataforma y la arquitectura sobre la que se ejecute, facilitando así que un programa escrito usando este lenguaje, pueda ser ejecutado en cualquier arquitectura sin necesidad de modificar el código fuente ni recompilar la aplicación, gracias a la presencia de su máquina virtual.

1.10.2 Frameworks

1.10.2.1 Hibernate

El trabajo con bases de datos relacionales en software programado usando el paradigma de orientación a objetos, suele ser una tarea compleja. Hibernate resuelve este problema en ambientes Java. Cuenta con herramientas que permiten generar automáticamente los archivos de mapeo y las clases persistentes a partir de un esquema de base de datos existente, eliminando el proceso de crearlos manualmente por parte del programador. (20)

Hibernate tiene su lenguaje de consultas Hibernate Query Language (HQL) el cual se basa en las clases y sus atributos generando consultas SQL (siglas en inglés de Lenguaje de Consulta Estructurado) para cada sistema gestor de base de datos (SGBD) soportado (Oracle, PostgreSQL, MySQL, entre otros) garantizando así la portabilidad. Proporciona manejo de transacciones, gestión de sesiones, cache, logs, entre otras características que alivian al desarrollador del 95% de las tareas comunes de persistencia de datos. (21)

Además Hibernate está licenciado bajo la LGPL[2] v2.1 que permite utilizar este framework en el desarrollo de programas libres o propietarios. (22)

1.10.2.2 Axis2

El proyecto Apache Axis2 es una implementación basada en Java tanto para el lado del cliente como para el lado del servidor. Provee un modelo de objetos y una arquitectura modular que hacen fácil la adición de módulos plug-ins que extienden sus funcionalidades, para características tales como seguridad e incremento de su fiabilidad y el soporte de nuevas especificaciones de servicios web.

Axis2 permite ejecutar fácilmente tareas tales como:

- Enviar mensajes SOAP.
- Recibir y procesar mensajes SOAP.

[2] GNU LESSER GENERAL PUBLIC LICENSE(LGPL) puede profundizar más en: http://www.gnu.org/licenses/

- Crear un servicio web a partir de una clase java.
- Crear implementaciones de clases tanto para cliente como servidor usando un WSDL.
- Fácil recuperación del WSDL a partir de un servicio.
- Enviar y recibir mensajes SOAP con adjuntos.
- Crear y utilizar servicios web basados en REST.
- Crear o utilizar servicios web que tomen ventaja de: WS-Reliable Messaging, WS-Addressing, WS-Coordination y WS-Atomic Transaction recomendations. (23)

Axis2 viene equipado con poderosas herramientas de generación tanto de clientes de servicios web a partir de WSDL (wsdl2java) como WSDL a partir de clases java (java2wsdl) aspectos que liberan al programador de una buena parte del trabajo manual. Se integra muy bien con los entornos de desarrollo de Java, dígase Eclipse y NetBeans que son los más populares, mediante plug-ins, así como con servidores de aplicaciones o contenedores de servlet como también se les conoce, se está hablando de JBoss y Apache Tomcat, por sólo mencionar algunos, además de contar con una distribución autónoma y diversas formas de exponer servicios web de forma programática, lo cual es muy útil en tiempo de desarrollo a la hora de depurar servicios.

1.10.2.3 Plataforma NetBeans

Una plataforma de aplicaciones clientes ricas (RCP, por sus siglas en inglés), es un entorno que sirve de base al ciclo de vida de una aplicación de escritorio. La mayoría de las aplicaciones de escritorio tienen características similares: menú, barra de progreso, barra de estado, configuración de personalización, pantalla de inicio, caja de diálogo, internacionalización y ayuda de usuario. Además permiten cargar, guardar datos y configuraciones del usuario. Para estas y otras funcionalidades típicas de RCP, una plataforma RCP provee un framework con el cual estas características pueden ser juntadas fácilmente.

Entre las ventajas que ofrece una plataforma RCP, se encuentran: reducción del tiempo de desarrollo, alta consistencia de las interfaces de usuario, actualización, independencia de plataforma,

reusabilidad y fiabilidad.

La plataforma NetBeans, como plataforma RCP, ofrece todas estas ventajas genéricas y en adición a esto, brinda otras características que resultan de gran utilidad para el desarrollo de la aplicación cliente que se desea realizar, entre las más importantes se encuentran:

- Pantalla de personalización.
- Biblioteca de clases para la construcción de asistentes visuales.
- Internacionalización.
- Sistema de ayuda. (24)

Las características y funcionalidades antes mencionadas convierten a la plataforma NetBeans en el candidato ideal para el desarrollo de la aplicación cliente para el sistema de manejo de llaves y certificados digitales que se pretende desarrollar.

1.10.3 Entorno de desarrollo integrado

Los entornos de desarrollo integrado (IDEs, por sus siglas en inglés) son herramientas pensadas para escribir, compilar, depurar y ejecutar programas. Los más populares para escribir programas Java son el NetBeans y el Eclipse. A partir de la elección de la plataforma NetBeans como marco de trabajo para la construcción de la aplicación cliente de escritorio, se descarta la elección del IDE Eclipse dado que tiene un bajo soporte para la creación de aplicaciones con dicha plataforma, todo lo contrario del IDE NetBeans, el cual se integra totalmente con dicho marco de trabajo, contando con un largo número de módulos y asistentes visuales para el desarrollo sobre la plataforma NetBeans, liberando al programador de gran parte de la implementación del código fuente así como de las tareas de internacionalización y configuración del sistema.

Existe un gran número de plug-ins para extender las funcionalidades del NetBeans. Este IDE es un producto libre, gratuito y de código abierto, sin restricciones de uso, escrito usando la plataforma NetBeans de la cual se habló anteriormente. Soporta todos los estándares java (J2EE, JavaME, JavaSE, EJB, etc.). Entre sus características más notables se encuentran:

- Sistema de proyectos basado en Ant.
- Soporte para control de versiones (CVS, SVN, Mercurial).
- Refactorización.
- Soporte para persistencia.
- Soporte para JAX-WS.
- En su versión Enterprise Pack, soporta desarrollo de aplicaciones empresariales con Java EE 5, incluye herramientas de desarrollo visual de SOAP, herramientas de orientación a servicios web (BPEL) y modelado UML.
- Licenciado bajo la Common Development and Distribution License (CDDL), licencia basada en la Mozilla Public License (MPL).
- Muy buena integración con herramientas tales como: Axis2, Hibernate, Plataforma NetBeans y Tomcat, con los cuales se trabajará en el desarrollo de la aplicación.

1.10.4 Gestor de base de datos

Para la selección de la base de datos se analizaron los tres gestores de base de datos más populares en la actualidad. Primeramente se analizó Oracle que presenta 4 ediciones de base de datos pero una sola gratuita (Express Edition). Esta edición tiene grandes limitaciones respecto a los otros gestores de base de datos que se analizaron posteriormente, ya que presenta limitaciones de tamaño y de uso de la RAM. Otra de las opciones que se analizó fue MySQL, el cual es un gestor de base de datos libre con características que compiten con los gestores más usados en la actualidad pero presenta un esquema de licenciamiento dual: por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para incorporarlo en productos privativos se debe comprar a la empresa una licencia específica que permita su uso. Y finalmente se analizó y seleccionó PostgreSQL el cual tiene licencia BSD la cual permite el uso de este en software privativos. Además PostgreSQL incluye características de la orientación a objetos, como son los tipos de datos, funciones,

restricciones, disparadores, reglas e integridad transaccional. Es hoy, el sistema gestor de bases de datos libre más avanzado que existe, contando con características tales como:

- Soporte del estándar SQL92/SQL99.
- Soporte de distintos tipos de datos: tipos básicos, tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP,...), cadenas de bits, etc. También permite la creación de tipos propios.
- Cuenta con una amplia biblioteca de funciones para el trabajo con fecha, redes, geometría, etc.
- Soporta el uso de índices, reglas y vistas.
- Permite la gestión de usuarios, y permisos.
- Soporta transacciones distribuidas en sistemas distribuidos.
- Multiplataforma, disponible para Linux, Solaris, MS Windows y Mac OS X.
- Soporta concurrencia mediante el sistema MVCC (Acceso concurrente multi-versión), el cual permite que mientras un proceso modifica una tabla otro proceso pueda acceder a la misma tabla sin necesidad de bloqueos.

PostgreSQL, cuenta con una intuitiva herramienta de administración con interfaz gráfica llamada pgAdmin, la cual proporciona una forma fácil para ejecutar tareas de configuración, creación de bases de datos, tablas, usuarios, etc. (25)

1.10.5 Apache Tomcat como servidor de aplicaciones

Apache Tomcat es una implementación de código abierto de las tecnologías Java Servlet y Java Server Pages. Tomcat es un servidor web y contenedor de servlet desarrollado por la fundación Apache, licencia bajo la cual está liberado. Incluye herramientas de configuración y administración, además de ofrecer la posibilidad de editar las configuraciones directamente en archivos XML. Tomcat es más rápido que sus homólogos contenedores de servlet: JBoss y Glass Fish, se puede integrar

fácilmente con el servidor web Apache mediante el modulo mod_jk y en tiempo de desarrollo, es muy conveniente ya que se integra muy bien con el IDE NetBeans, dando la oportunidad de depurar aplicaciones que están corriendo en el servidor, así como desplegar y replegar aplicaciones hacia y desde el servidor respectivamente.

1.10.6 UML como lenguaje de modelado

UML se define como: “lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software…” (19). La elegancia, flexibilidad y expresión de este lenguaje lo han convertido en el estándar de factor de la industria. Prescribe una notación estándar y semánticas esenciales para modelar sistemas orientados a objetos, notación que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa.

1.10.7 Herramienta CASE

Una herramienta CASE (Ingeniería de Software Asistida por Ordenador, de la traducción del inglés *Computer Aided Software Engineering*) es una aplicación informática que permite la realización de un buen diseño para el proyecto, implementando a partir de él, parte del código automáticamente.

Para la selección de la herramienta CASE a utilizar en el desarrollo del proyecto se analizó la herramienta BoUML la cual es una herramienta de software libre y tiene como principales características que es gratis, es multiplataforma, permite especificar y generar código en C++, Java y Php, es rápida y no necesita mucho espacio de memoria. Sin embargo, está limitada en funcionalidades en comparación con el Visual Paradigm; ya que no permite la creación de diagramas Entidad – Relación y generación de script SQL, características necesarias para facilitar el desarrollo de la aplicación. Por lo que se escogió Visual Paradigm como herramienta ya que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones. Tiene como principales características:

- Soporte de UML versión 2.1

- Ingeniería inversa – código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, etc.
- Generación de código – modelo a código, diagrama a código.
- Soporte ORM – Generación de objetos Java desde la base de datos.
- Generación de bases de datos – Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Generación de la documentación del proyecto automáticamente en varios formatos como web o pdf.
- Herramienta multiplataforma. (26)

1.11 Conclusiones

A partir del estudio realizado de soluciones KMS, se llega a la conclusión, de que ninguna cumple con las necesidades del CISED, debido a que se requiere un software KMS que pueda adaptarse a los diversos ambientes de proyectos que allí se generan, y que pueda ser modificado fácilmente sin tener que incurrir en gastos monetarios. Por lo que se hace necesario desarrollar un sistema que cuente con las características de los sistemas estudiados y cumpla con las necesidades del CISED.

Se analizaron y escogieron las herramientas y tecnologías a utilizar en el desarrollo del sistema de acuerdo a las características deseadas en este. Lo que permitirá crear una solución multiplataforma y con el menor costo de construcción posible.

CAPÍTULO 2. Características del sistema

2.1 Introducción

A partir de este capítulo se comenzará a abordar la propuesta que dará solución al problema de la investigación planteado utilizando la metodología de desarrollo FDD. Esta metodología se divide en 5 fases y en este capítulo se abordarán las tres primeras fases: Desarrollo de un modelo general, Construcción de la lista de características y Plan de liberaciones en base a las funcionalidades a implementar, las cuales ocupan gran parte del tiempo en las primeras iteraciones.

2.2 Modelo General

El modelo general es un diagrama de clases que representa los conceptos más importantes dentro del dominio del problema y las relaciones entre ellos. Este modelo se realiza con el fin de proveer un marco general dentro del cual evoluciona el proyecto. (Ver **figura 2.1**)

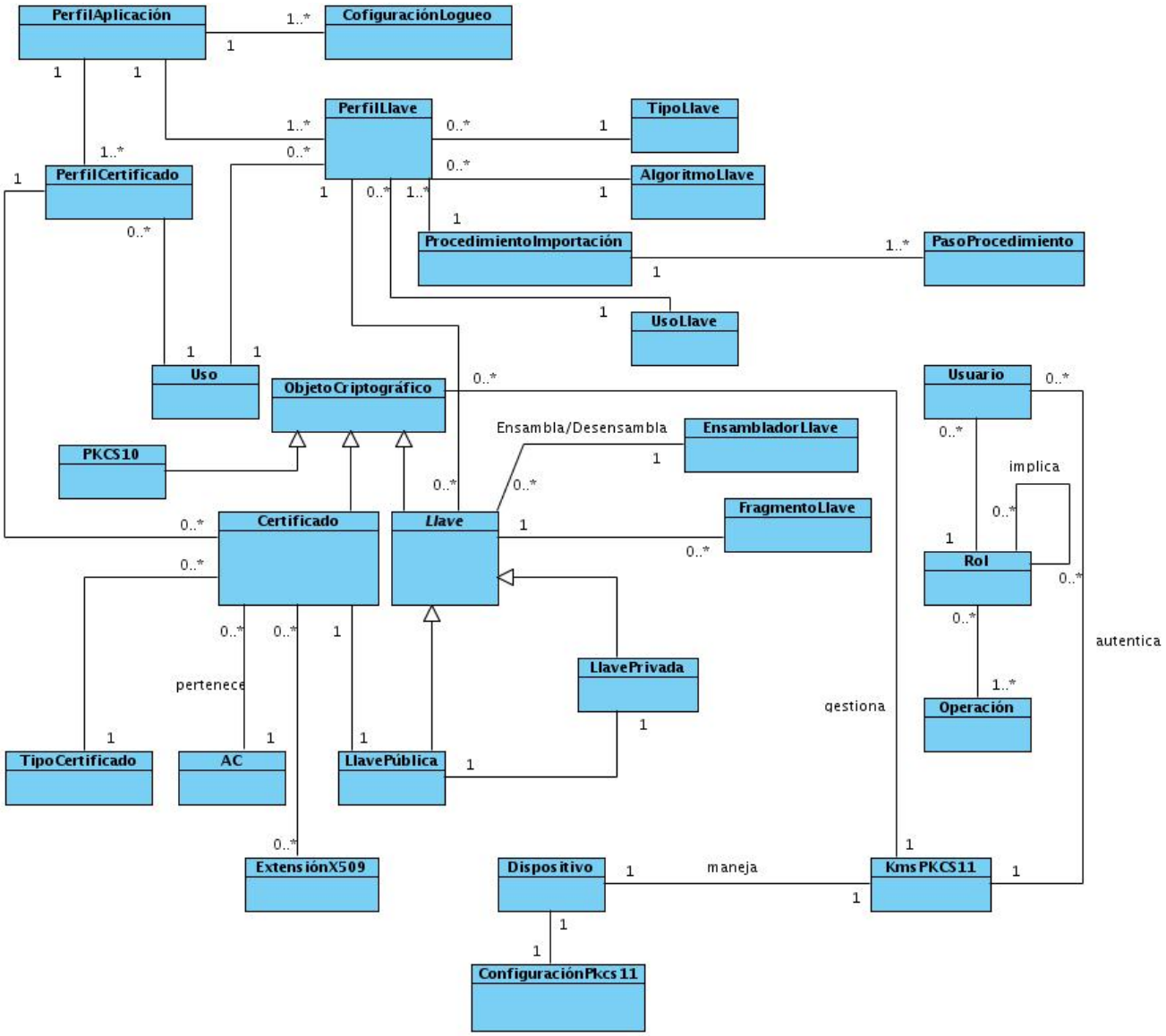


Figura 2.1 Modelo General.

Conceptos tratados en el modelo

- **PerfilAplicación**: Plantilla que define las características del sistema que pueden ser usadas: protocolos, algoritmos, procedimientos de importación, forma de autenticación, operaciones criptográficas habilitadas, perfiles de llaves y certificados.
- **PerfilLlave**: Plantilla que define las características de una llave: algoritmo, tamaño, uso, procedimiento de importación, forma de ensamblado y desensamblado.

- **PerfilCertificado**: Plantilla que define las características de un certificado: Autoridad Certificadora (AC), fecha de inicio de validez, fecha de fin de validez, uso, algoritmo de firma, versión, etc.
- **ConfiguraciónLogueo**: Define un procedimiento de autenticación en el sistema.
- **ProcedimientoImportación**: Define la forma en la que se puede importar una llave.
- **PasoProcedimiento**: Define un paso en el procedimiento de importación de una llave especificando, entre otras cosas, el protocolo de importación a utilizar.
- **ObjetoCriptográfico**: Información principal que maneja el KMS.
- **Llave**: es una secuencia de caracteres, que puede contener letras, dígitos y símbolos, y que es convertida a un número, utilizada por los algoritmos de cifrado para codificar y decodificar mensajes.
- **LlavePública**: Tipo de llave criptográfica utilizada en los criptosistemas asimétricos que puede ser de dominio público.
- **LlavePrivada**: Tipo de llave criptográfica que debe permanecer en secreto.
- **FragmentoLlave**: Sección de una llave lógicamente separada por un ensamblador de llave.
- **TipoLlave**: Clasificación de una llave. Ejemplo: pública, privada, secreta.
- **AlgoritmoLlave**: Mecanismo criptográfico con el cual fue generada la llave.
- **UsoLlave**: Uso que se le da a la llave a nivel de operaciones criptográficas. Ejemplo: firma digital, no repudio, cifrado de llave, cifrado de datos, firma de certificado, descifrado, etc.
- **EnsambladorLlave**: Encargado de seccionar las llaves de forma lógica y unir sus fragmentos en un proceso inverso.
- **Certificado**: Tipo de objeto criptográfico que asocia una llave pública con la identidad de su propietario.
- **ExtensiónX509**: Parte del certificado que es opcional y define información extra que se quiera almacenar en el certificado.
- **AC**: Autoridad certificadora, es una entidad de confianza, responsable de emitir y revocar los

certificados.

- **TipoCertificado**: Estándar que cumple el certificado. Ejemplo: X509.
- **Uso**: Uso que se le da al objeto criptográfico. Ejemplo: comunicación con otra entidad, firma digital, control de acceso extendido, etc.
- **Usuario**: Persona, entidad o sistema que interactúa con el KMS.
- **Rol**: Clasificación del usuario según su nivel de acceso a las funcionalidades del KMS.
- **Operación**: Funcionalidad del KMS.
- **Dispositivo**: Dispositivo criptográfico: HSM, SmartCard, etc.
- **ConfiguraciónPkcs11**: Define los parámetros de inicialización y uso del dispositivo.
- **PKCS10**: Objeto criptográfico que define una solicitud de certificado.
- **KmsPKCS11**: Biblioteca de clases que implementa el estándar PKCS#11.

2.3 Vista de la arquitectura del sistema

Aunque no es requerida por la metodología FDD, se diagrama la vista de la arquitectura del sistema, para mostrar la organización de los elementos más importantes de la solución. (Ver **figura 2.2**)

La arquitectura definida es una mezcla de los estilos arquitectónicos n-capas y componentes. Está compuesta por varias capas: **presentación**, **middleware**, **sesión**, **lógica de negocio**, **acceso a dato** y **pkcs11**.

- La capa de **presentación** contiene los servicios web que expone el KMS, donde se encuentran 2 componentes: KMS (servicios de las operaciones criptográficas y de control de acceso) y Utilidades.
- La capa **middleware** es la encargada de interceptar y monitorear todas las peticiones y las respuestas entre el cliente y el servidor. Además prepara los datos entrantes y las respuestas.
- La capa de **sesión** realiza el control de acceso.

- La capa de **lógica de negocio** se divide en los componentes: administración de objetos criptográficos (contiene las operaciones criptográficas) y administración del KMS (contiene las operaciones de gestión de usuarios y perfiles) y objetos de dominio.

- La **capa de acceso a datos** es la encargada de mapear los datos de los objetos de dominio a las tablas de la base de datos.

- La capa **pkcs11** contiene la interfaz para la comunicación con el dispositivo criptográfico.

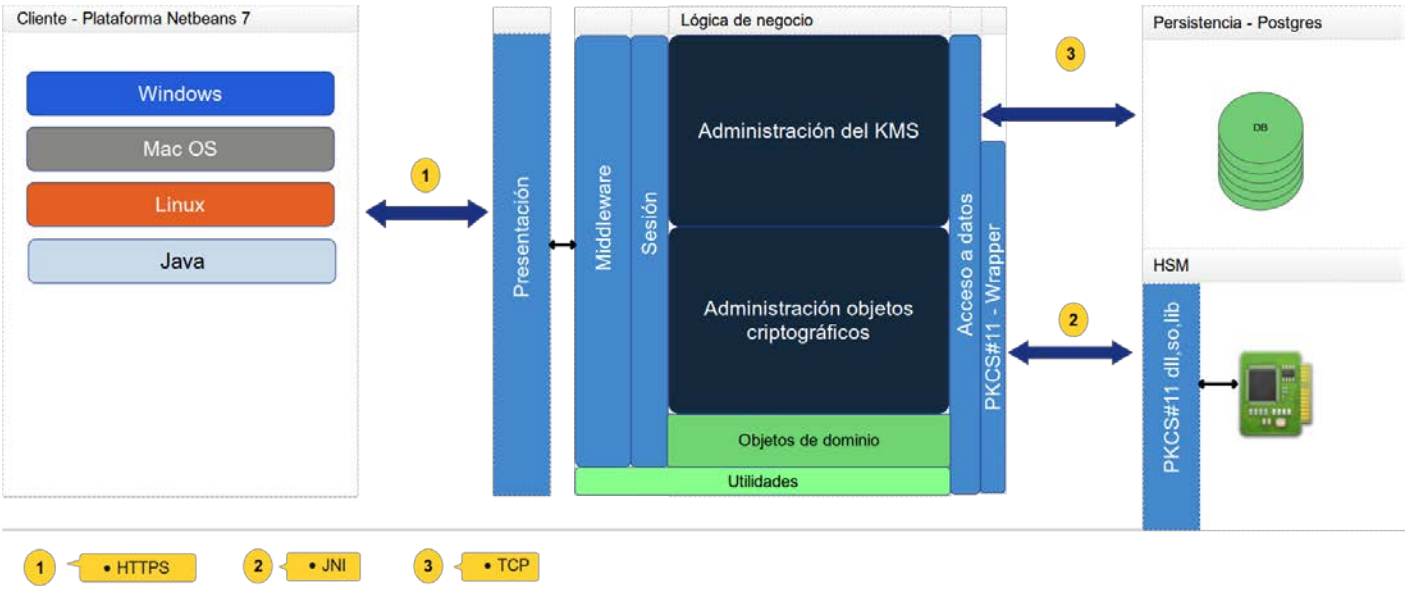


Figura 2.2 Diagrama de arquitectura.

2.4 Lista de características

FDD se estructura alrededor de la definición de características que representan las funcionalidades que debe contener el sistema y tienen un alcance lo suficientemente corto como para ser implementadas en un par de semanas. Estas características, a su vez, se agrupan en conjuntos de características, según la relación que exista entre ellas, para una mejor organización. A continuación en la **tabla 2.1** se listan las características del software que se propone realizar.

CAPÍTULO 2. Características del sistema.

Tabla 2.1 Listado de características.

Conjunto de características	Características
Gestionar rol.	C1. Crear un rol del sistema. C2. Modificar un rol del sistema. C3. Eliminar un rol del sistema. C4. Buscar rol por nombre.
Gestionar usuario.	C5. Registrar un usuario en el sistema. C6. Modificar un usuario en el sistema. C7. Eliminar un usuario del sistema. C8. Buscar un usuario del sistema.
Configuración de operaciones.	C9. Habilitar operación del sistema. C10. Deshabilitar operación del sistema. C11. Buscar operaciones del sistema.
Autenticación y sincronización.	C12. Autenticación en un HSM. C13. Autenticación en el KMS. C14. Sincronizar HSM con KMS.
Gestión de objetos criptográficos.	C15. Generar llave simétrica. C16. Generar par de llaves y certificado auto-firmado. C17. Generar solicitud de certificado (PKCS#10). C18. Exportar certificado. C19. Exportar solicitud de certificado (PKCS#10). C20. Exportar llave pública.

	C21. Importar llave. C22. Importar certificado. C23. Eliminar objetos criptográficos. C24. Eliminar solicitud de certificado. C25. Activar llave de firma. C26. Desactivar llave de firma. C27. Adicionar soporte para dispositivo criptográfico. C28. Buscar objetos criptográficos por categoría.

Una vez definido el listado de características, se procede a describir las restricciones del sistema.

- **Restricciones de Software**

El sistema se podrá ejecutar sobre los sistemas operativos: Linux, Windows XP/Vista/7.

Para que el sistema se ejecute correctamente la versión de la Máquina Virtual de Java requerida será JVM 1.6u26.

El sistema utilizará para el manejo de datos el SGBD PostgreSQL.

- **Restricciones de Hardware**

El sistema podrá interactuar con dispositivos criptográficos: HSM/SmartCard/Token de cualquier fabricante con interfaz PKCS#11.

- **Restricciones en el diseño y la implementación**

El sistema será implementado usando el lenguaje de programación Java y los frameworks Hibernate, Axis2 y la plataforma NetBeans.

- **Restricciones de Seguridad**

El sistema contará con dos niveles de seguridad: autenticación SSL y Servicio de autenticación y autorización de java (JAAS).

- **Restricciones de Usabilidad**

CAPÍTULO 2. Características del sistema.

El sistema utilizará asistentes visuales para la mayoría de las funcionalidades.

2.5 Descripción de las características del sistema

Para una mejor comprensión de las características del sistema se describirán, en esta sección, algunas de las que fueron identificadas como críticas. La metodología FDD no contempla en sus fases la descripción del listado de características, por lo que se creó una plantilla para la descripción de estas.

Tabla 2.2 "**Descripción de característica.** Buscar objetos criptográficos por categoría".

Nombre de la característica	Buscar objetos criptográficos por categoría.
Precondiciones	El usuario debe estar autenticado en el KMS con el rol de operador o administrador.
Características asociadas	C15, C16, C18, C19, C20, C21, C22, C23, C24, C25, C26, C28.
Conceptos tratados	Llave, Uso, Algoritmo de llave, Certificado, Llave Privada, Llave Pública, Objeto Criptográfico, Fragmento de llave, Tipo de certificado, PKCS#10, Usuario.
Descripción básica	1. El sistema muestra la interfaz principal que contiene: • Menú vertical, que presenta tres categorías: Uso: Opción: - Llaves de plataforma. - Objetos de transporte. - Objetos de firma. - Objetos de EAC. Tipo: Opción: - Llaves privadas. - Llaves públicas. - Certificados.

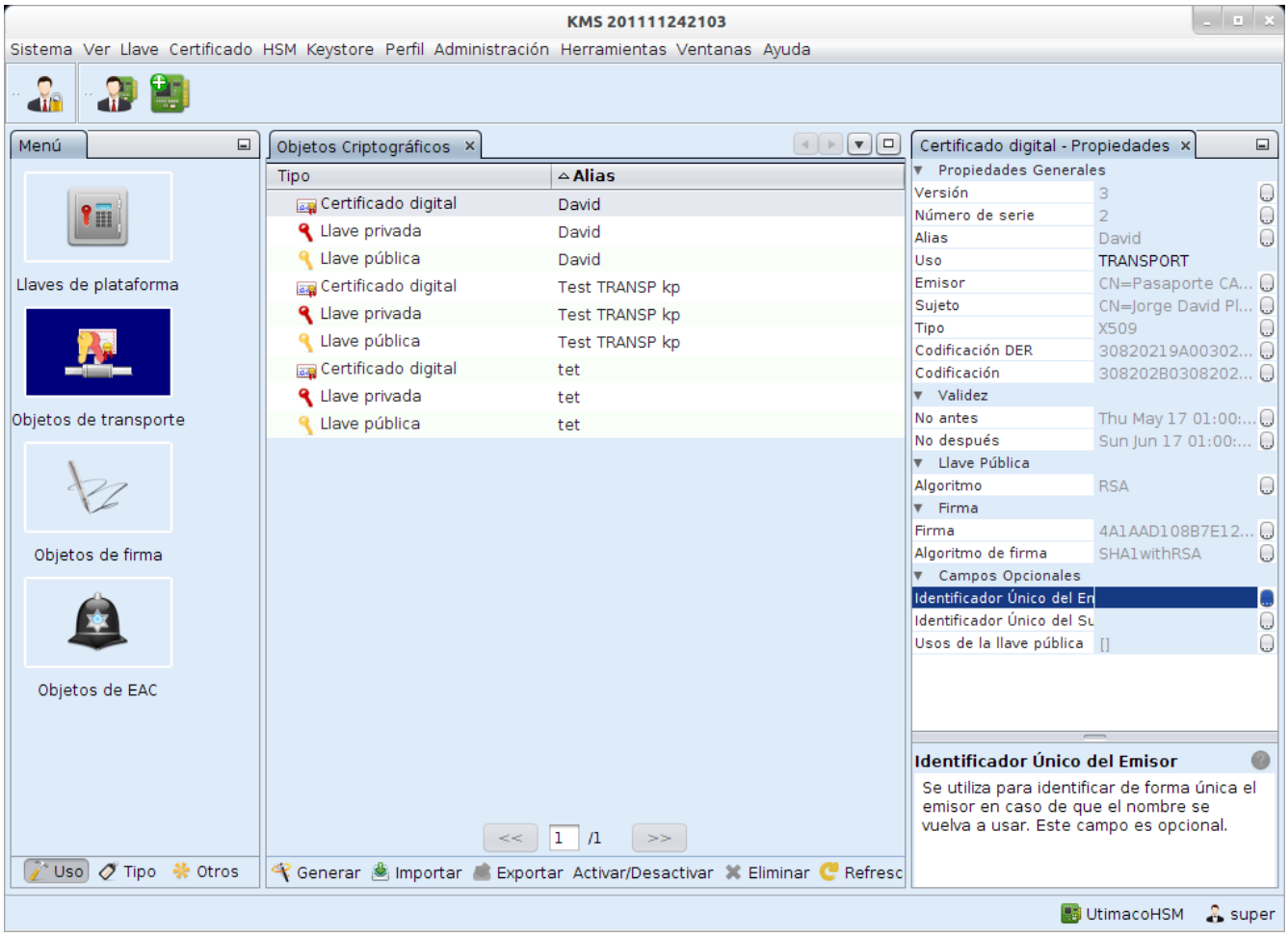
	- Solicitudes PKCS#10. Otros: Opción: - Autoridades Certificadoras. - Objetos SSL. • Listado de objetos criptográficos con los siguientes datos: - Tipo. - Alias. - Estado. • Ventana de propiedades con los siguientes datos: Para una llave: - Alias. - Versión. - Uso. - Algoritmo. - Cantidad de fragmentos. - Tipo. - Formato. - Longitud. - Objeto de dispositivo. - Sensible. - Extraíble. - Procedimiento de importación. - Ensamblador. Para un certificado: - Alias. - Versión. - Uso. - Número de serie. - Emisor. - Sujeto.

	- Tipo. - Codificación DER. - Codificación. - No antes. - No después. - Algoritmo. - Algoritmo de firma. - Firma. - Identificador único del emisor. - Identificador único del sujeto. - Usos de la llave pública. Para una solicitud de certificado: - Versión. - Sujeto. - Algoritmo de firma. - Estado (Generado o Exportado). - Alias. Opción: • Generar. • Importar. • Exportar (Deshabilitado). • Activar/Desactivar (Deshabilitado). • Eliminar (Deshabilitado). • Refrescar. 1.1 Si el usuario selecciona la opción del menú **Llaves de plataforma**, el sistema muestra un listado de las llaves de plataforma registradas. 1.2 Si el usuario selecciona la opción del menú **Objetos de transporte**, el sistema muestra un listado de los objetos de transporte registrados. 1.3 Si el usuario selecciona la opción del menú **Objetos de**

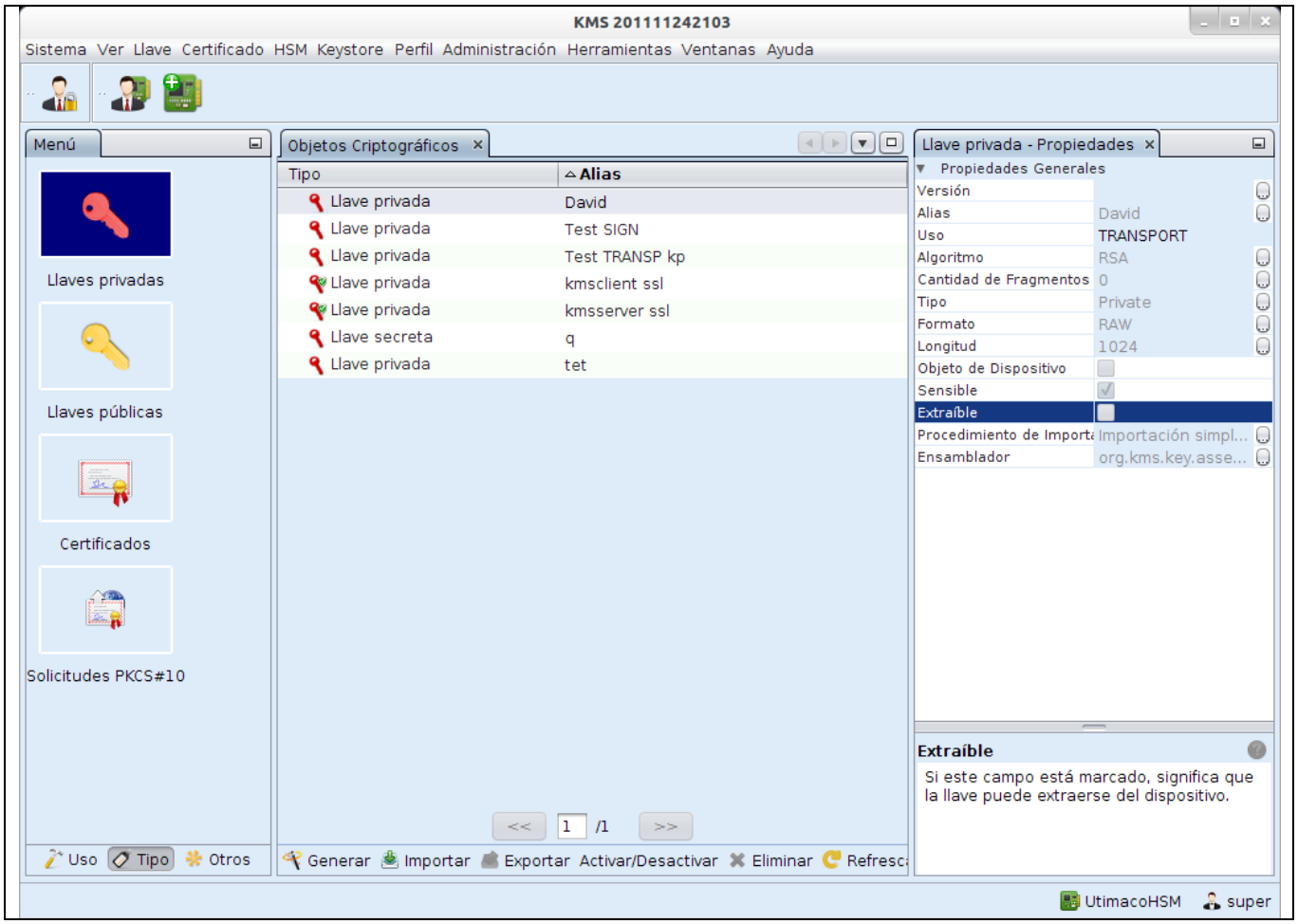
	firma, el sistema muestra un listado de los objetos de firma registrados. 1.4 Si el usuario selecciona la opción del menú **Objetos de EAC**, el sistema muestra un listado de los objetos de EAC registrados. 1.5 Si el usuario selecciona la opción del menú **Llaves privadas**, el sistema muestra un listado de las llaves privadas registradas. 1.6 Si el usuario selecciona la opción del menú **Llaves públicas**, el sistema muestra un listado de las llaves públicas registradas. 1.7 Si el usuario selecciona la opción del menú **Certificados**, el sistema muestra un listado de los certificados registrados. 1.8 Si el usuario selecciona la opción del menú **Solicitudes PKCS#10**, el sistema muestra un listado de las solicitudes de certificado PKCS#10 registradas. 1.9 Si el usuario selecciona la opción del menú **Autoridades Certificadoras**, el sistema muestra un listado de los certificados raíces de las autoridades certificadoras de confianza. 1.10 Si el usuario selecciona la opción del menú **Objetos SSL**, el sistema muestra un listado de los objetos que se utilizarán en la comunicación SSL. 1.11 Si el usuario presiona el botón **Refrescar**, el sistema actualiza el listado de objetos criptográficos. 1.12 Si el usuario selecciona un objeto en el listado de objetos criptográficos, el sistema mostrará sus propiedades en la ventana de propiedades y habilitará el botón **Eliminar**. 1.12.1 Si el elemento seleccionado es una llave de firma, el sistema habilita el botón **Activar/Desactivar**. 1.12.1.1 Si el usuario presiona el botón **Activar/Desactivar**,

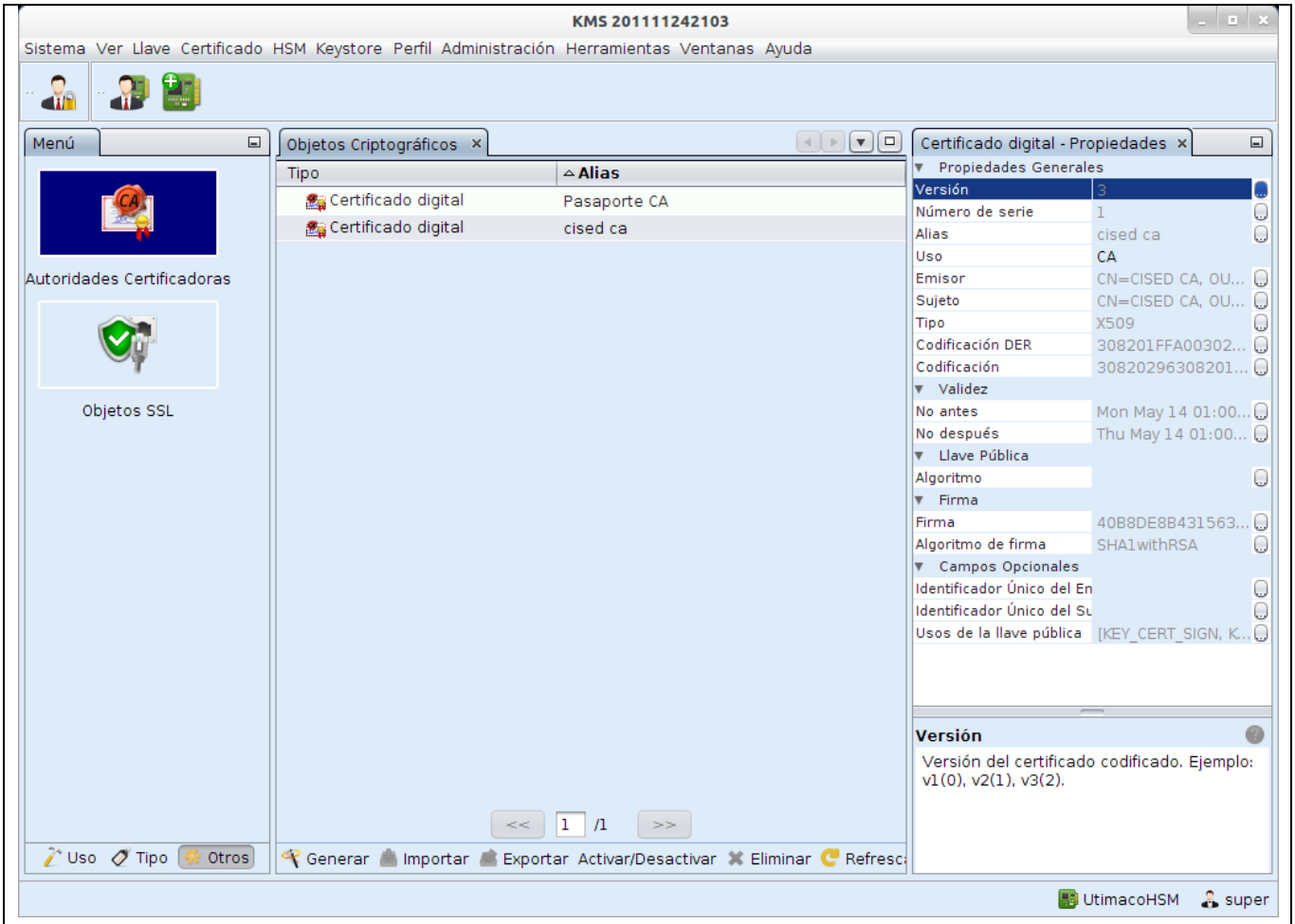
	1.12.1.1.1 Si la llave está activa, *(ver descripción de característica "Desactivar llave de firma").* 1.12.1.1.2 Si la llave está desactivada, *(ver descripción de característica "Activar llave de firma").* 1.12.2 Si el elemento seleccionado es un certificado, una llave pública o una solicitud de certificado PKCS#10, el sistema habilita el botón **Exportar**. 1.12.2.1 Si el usuario presiona el botón **Exportar**, *(ver descripción de característica "Exportar certificado", "Exportar solicitud de certificado (PKCS#10)" o "Exportar llave pública"* según el objeto criptográfico seleccionado por el usuario). 1.12.3 Si el usuario presiona el botón **Eliminar**, 1.12.3.1 Si el elemento seleccionado es una solicitud de certificado PKCS#10, *(ver descripción de característica "Eliminar solicitud de certificado").* 1.12.3.2 Si el elemento seleccionado es un certificado, una llave privada o una llave pública *(ver descripción de característica "Eliminar objetos criptográficos").* 1.13 Si el usuario presiona el botón **Importar**, 1.13.1 Si la opción seleccionada en el menú es **Autoridades Certificadoras** (*ver descripción de característica "Importar certificado"*). 1.13.2 Si la opción seleccionada en el menú es **Llaves de plataforma** (*ver descripción de característica "Importar llave"*). 1.14 Si el usuario presiona el botón **Generar**, 1.14.1 Si la opción seleccionada en el menú es **Llaves de plataforma** *(ver descripción de característica "Generar llave simétrica").* 1.14.2 Si la opción seleccionada en el menú no es **Llaves de**

	plataforma *(ver descripción de característica "Generar par de llaves y certificado auto-firmado").*



CAPÍTULO 2. Características del sistema.





Descripción alterna	No tiene.
Poscondiciones	No tiene.

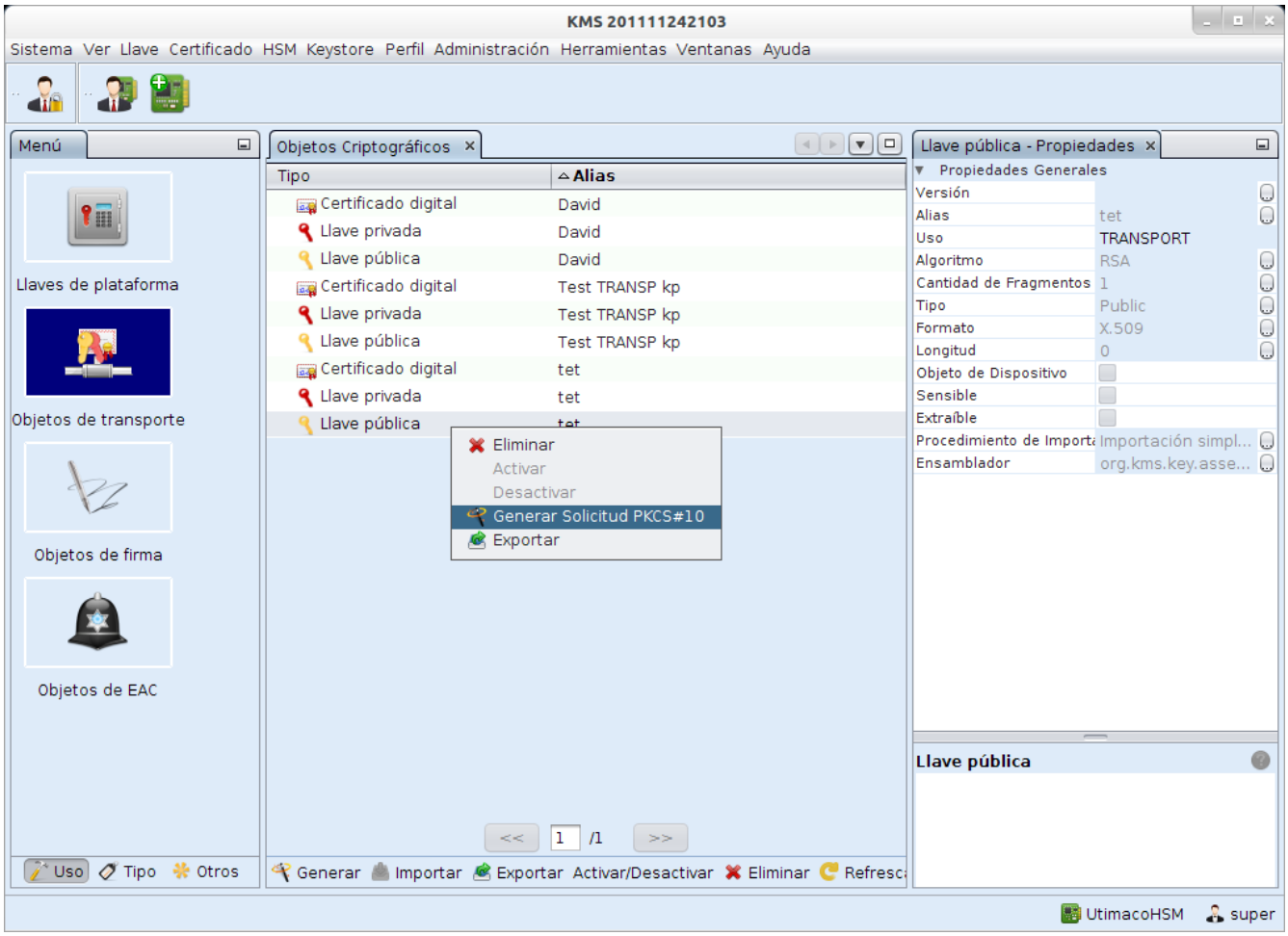
Tabla 2.3 "Descripción de característica. Generar solicitud de certificado (PKCS#10)".

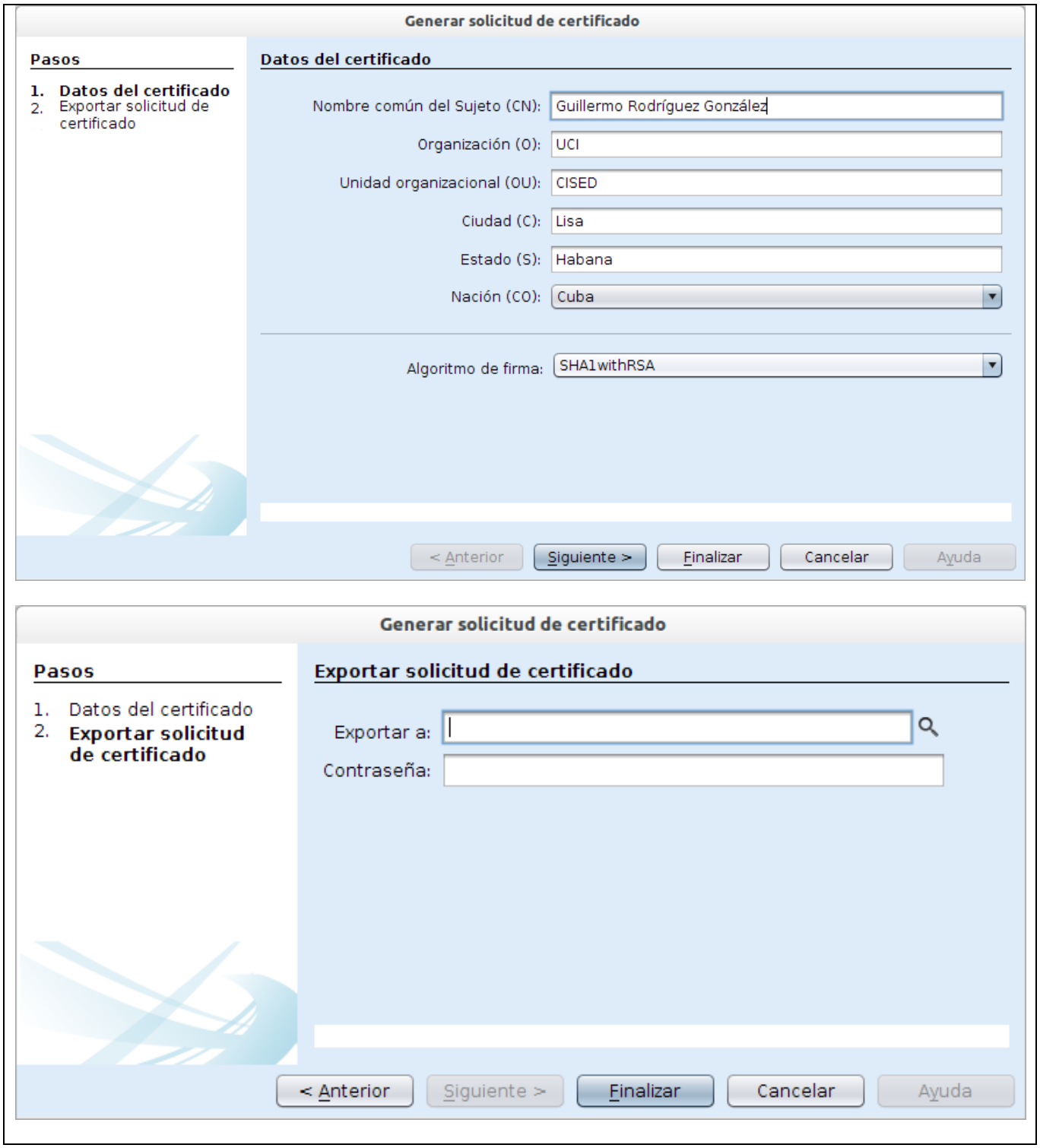
Nombre de la característica	Generar solicitud de certificado (PKCS#10).
Precondiciones	El usuario debe estar autenticado en el KMS y HSM con el rol de operador o administrador. El usuario debe haber seleccionado una llave asimétrica.

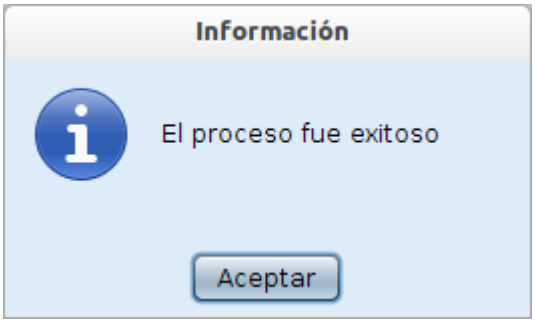
Características asociadas	C17.
Conceptos tratados	PKCS#10, Usuario, Llave pública.
Descripción básica	1. El usuario selecciona la opción **Generar Solicitud PKCS#10** del menú contextual del objeto. 2. El sistema muestra la interfaz "**Datos del certificado**" que contiene los siguientes campos: • Nombre común del sujeto (CN). • Organización (O). • Unidad organizacional (OU). • Ciudad (C). • Estado (S). • Nación (CO). • Algoritmo de firma. Opción: • Anterior (Deshabilitado). • Siguiente (Deshabilitado). • Finalizar (Deshabilitado). • Cancelar. • Ayuda. 2.1 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.2 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.3 Si el usuario introduce los datos, el sistema habilita las opciones **Siguiente** y **Finalizar.** 2.3.1 Si el usuario selecciona la opción **Finalizar**, el sistema registra la solicitud de certificado en la base de datos, muestra un mensaje informando si el proceso tuvo éxito o no y regresa a la interfaz principal.

	2.3.2 Si el usuario selecciona la opción **Siguiente**, el sistema muestra la interfaz "**Exportar solicitud de certificado**" que contiene los siguientes campos: • Exportar a. • Contraseña. Opción: • Navegar. • Anterior. • Siguiente (Deshabilitado). • Finalizar. • Cancelar. • Ayuda. 2.3.2.1 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.3.2.2 Si el usuario selecciona la opción **Anterior**, el sistema regresa a la interfaz "**Datos del certificado**". 2.3.2.3 Si el usuario presiona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.3.2.4 Si el usuario selecciona la opción **Navegar**, el sistema muestra la interfaz "**Exportar a**". 2.3.2.4.1 Si el usuario selecciona una ruta y presiona el botón **Aceptar**, el sistema muestra la interfaz "**Exportar solicitud de certificado**" y llena el campo "Exportar a" con la ruta seleccionada. 2.3.2.4.2 Si el usuario presiona el botón **Cancelar**, el sistema regresa a la interfaz "**Exportar solicitud de certificado**". 2.3.2.5 Si el usuario selecciona la opción **Finalizar**, el sistema genera la solicitud de certificación PKCS#10 a partir de la llave pública asociada al objeto seleccionado. 2.3.2.5.1 Si el campo "**Exportar a**" está vacío el sistema almacena

	la solicitud PKCS#10 en la base de datos del KMS y muestra un mensaje informando si el proceso tuvo éxito o no.
	2.3.2.5.2 Si el campo "**Exportar a**" no está vacío el sistema almacena la solicitud PKCS#10 en la base de datos del KMS, lo exporta a la ruta especificada y muestra un mensaje informando si el proceso tuvo éxito o no.




Generar solicitud de certificado
Pasos
1. Datos del certificado
2. Exportar solicitud de certificado
Datos del certificado
Nombre común del Sujeto (CN): Guillermo Rodríguez González
Organización (O): UCI
Unidad organizacional (OU): CISED
Ciudad (C): Lisa
Estado (S): Habana
Nación (CO): Cuba
Algoritmo de firma: SHA1withRSA
< Anterior
Siguiente >
Finalizar
Cancelar
Ayuda
Generar solicitud de certificado
Pasos
1. Datos del certificado
2. Exportar solicitud de certificado
Exportar solicitud de certificado
Exportar a:
Contraseña:
< Anterior
Siguiente >
Finalizar
Cancelar
Ayuda

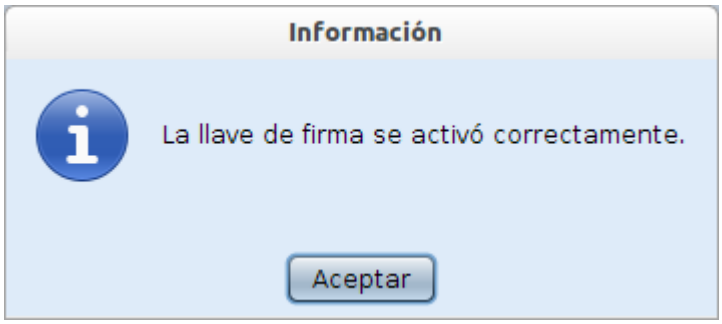


Descripción alterna	No tiene.

Poscondiciones	Queda registrada en el KMS una solicitud de certificado PKCS#10.

Tabla 2.4 "**Descripción de característica.** Activar llave de firma".

Nombre de la característica	Activar llave de firma.
Precondiciones	El usuario debe estar autenticado en el KMS con el rol de operador o administrador. El usuario debe haber seleccionado una llave de firma desactivada.
Características asociadas	C25.
Conceptos tratados	Usuario, Llave.
Descripción básica	1. El usuario selecciona la opción **Activar/Desactivar**, el sistema desactiva el par de llaves de firma activo y activa el par de llaves de firma con el mismo alias de la llave seleccionada. 1.1 Si el proceso fue exitoso, el sistema muestra el mensaje: "La llave de firma se activó correctamente". 1.2 Si el proceso no se pudo completar, el sistema muestra el mensaje: "La llave de firma no se pudo activar".



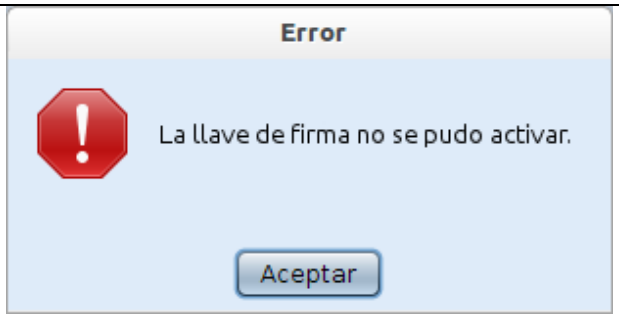
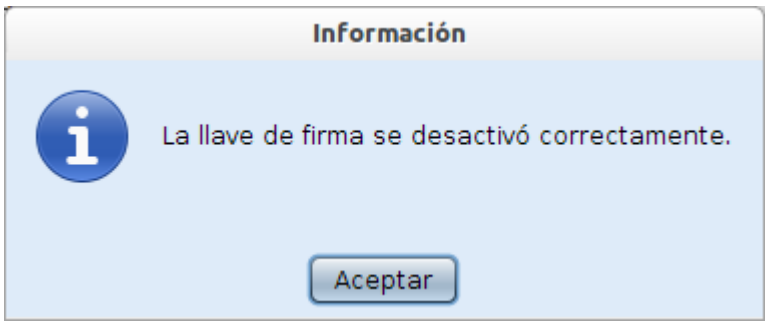
	
Descripción alterna	No tiene.
Poscondiciones	Quedan activadas el par de llaves de firma seleccionadas por el usuario y las llaves de firma restantes quedan desactivadas.

Tabla 2.5 "**Descripción de característica.** Desactivar llave de firma".

Nombre de la característica	Desactivar llave de firma.
Precondiciones	El usuario debe estar autenticado en el KMS con el rol de operador o administrador. El usuario debe haber seleccionado una llave de firma activa.
Características asociadas	C26.
Conceptos tratados	Usuario, Llave.
Descripción básica	1 El usuario selecciona la opción **Activar/Desactivar**, el sistema desactiva el par de llaves de firma que esté en ese momento activo. 1.2 Si el proceso fue exitoso, el sistema muestra el mensaje: "La llave de firma se desactivó correctamente". 1.3 Si el proceso no se pudo completar, el sistema muestra el mensaje: "La llave de firma no se pudo desactivar".

<table>
<tr><td colspan="2"> 

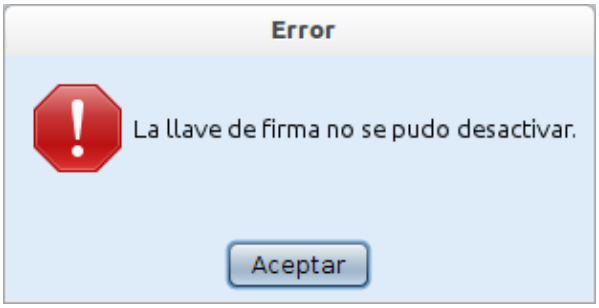
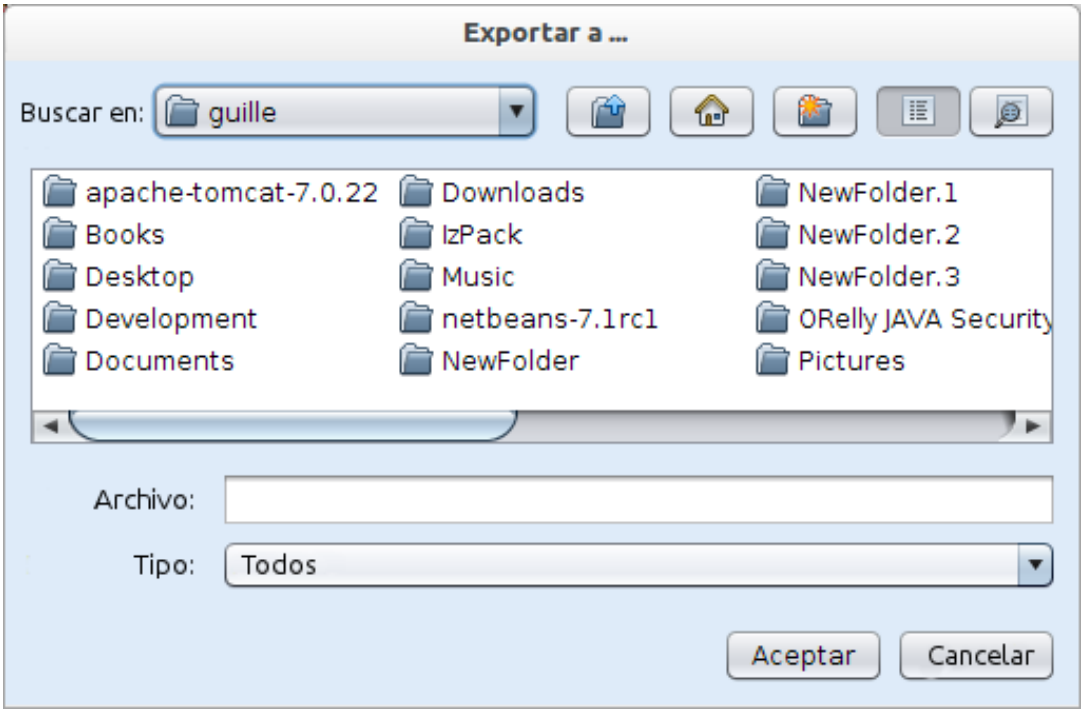
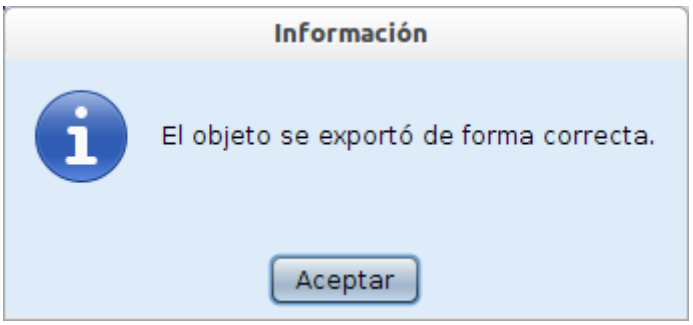

</td></tr>
<tr><td>Descripción alterna</td><td>No tiene.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Poscondiciones</td><td>Quedan desactivadas el par de llaves de firma seleccionadas por el usuario.</td></tr>
</table>

Tabla 2.6 "**Descripción de característica.** Exportar certificado".

Nombre de la característica	Exportar certificado.
Precondiciones	El usuario debe estar autenticado en el KMS y HSM con el rol de operador o administrador.
Características asociadas	C18.
Conceptos tratados	Llave, Usuario.
Descripción básica	1. El usuario selecciona el certificado que desea exportar y presiona el botón **Exportar** de la interfaz principal.

	2. El sistema muestra un cuadro de diálogo para que el usuario seleccione la ruta donde desea exportar el certificado. 2.1 Si el usuario selecciona la ruta y presiona el botón **Exportar**, el sistema generará en esta dirección un archivo .cer con la codificación del certificado y un archivo XML con el perfil de la llave pública asociada al certificado. 2.1.1 Si el proceso se realizó correctamente, el sistema muestra el mensaje: “El objeto se exportó de forma correcta”. 2.1.2 Si en el proceso de exportación ocurrió un error, el sistema muestra el mensaje: “El objeto no se pudo exportar correctamente”. 2.2 Si el usuario selecciona la opción **Cancelar**, el sistema aborta la operación y regresa a la interfaz principal.



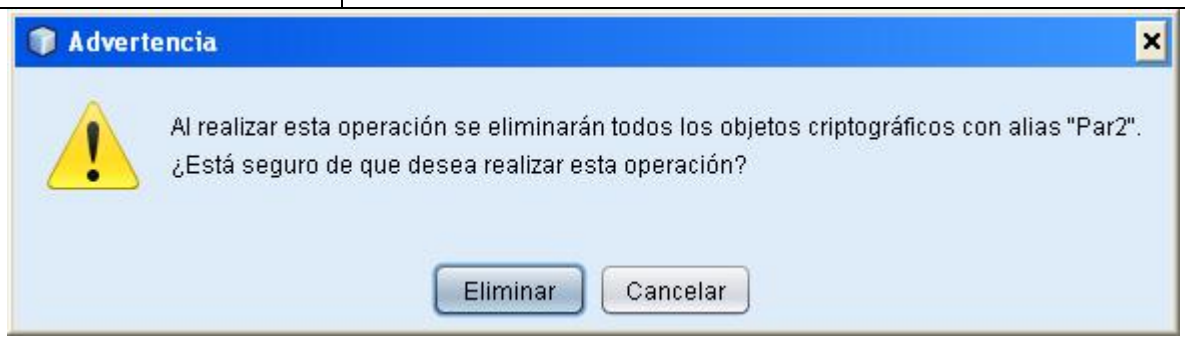
<table>
<tr><td colspan="2"> 

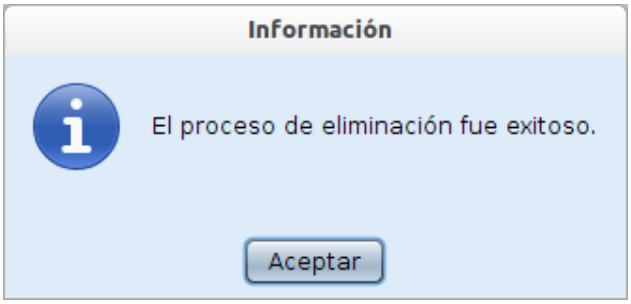

</td></tr>
<tr><td>Descripción alterna</td><td>No tiene.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Poscondiciones</td><td>Queda generado un archivo, que contiene la codificación del certificado y un archivo XML con el perfil de la llave pública asociada al certificado.</td></tr>
</table>

Tabla 2.7 "Descripción de característica. Eliminar objetos criptográficos".

Nombre de la característica	Eliminar objetos criptográficos.
Precondiciones	El usuario debe estar autenticado en el KMS y en el HSM con el rol de operador o administrador y debe existir al menos un objeto criptográfico en el HSM.
Características asociadas	C23.
Conceptos tratados	Llave, Certificado, Usuario.

Descripción básica	1. El usuario selecciona una llave o un certificado y presiona el botón **Eliminar**, de la interfaz principal. 2. El sistema muestra un cuadro de confirmación con las opciones **Eliminar** y **Cancelar**. 2.1 Si el usuario presiona el botón **Eliminar**, el sistema busca todos los objetos criptográficos con el alias del objeto seleccionado y los elimina. 2.1.1 Si el proceso de eliminación tuvo éxito, el sistema muestra el mensaje: “El proceso de eliminación fue exitoso”. 2.1.2 Si el proceso de eliminación no tuvo éxito, el sistema muestra el mensaje: “No se pudo eliminar correctamente”. 2.2 Si el usuario presiona el botón **Cancelar**, el sistema regresa a la interfaz principal.





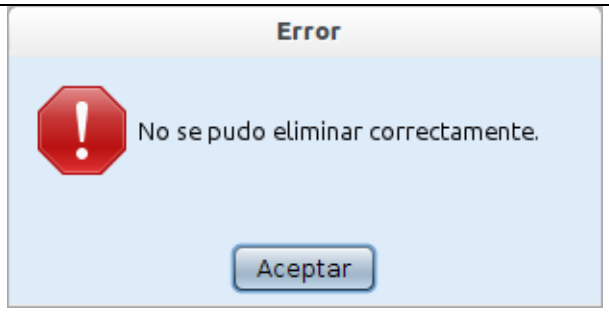
	
Descripción alterna	No tiene.
Poscondiciones	Se eliminan del KMS y HSM todos los objetos criptográficos con el alias del objeto seleccionado.

Tabla 2.8 "**Descripción de característica:** Eliminar solicitud de certificado".

Nombre de la característica	Eliminar solicitud de certificado.
Precondiciones	El usuario debe estar autenticado en el KMS y HSM con el rol de operador o administrador. El usuario debe haber seleccionado una solicitud de certificado del listado de objetos criptográficos.
Características asociadas	C24.
Conceptos tratados	Solicitud de certificado (PKCS#10), Usuario.
Descripción básica	1. El usuario selecciona la opción **Eliminar.** 2. El sistema muestra un diálogo de confirmación con las opciones **Eliminar** y **Cancelar.** 2.2 Si el usuario selecciona la opción **Eliminar**, el sistema elimina la solicitud de certificado seleccionada. 2.2.1 Si el proceso de eliminación tuvo éxito, el sistema muestra el mensaje: "El proceso de eliminación fue exitoso".

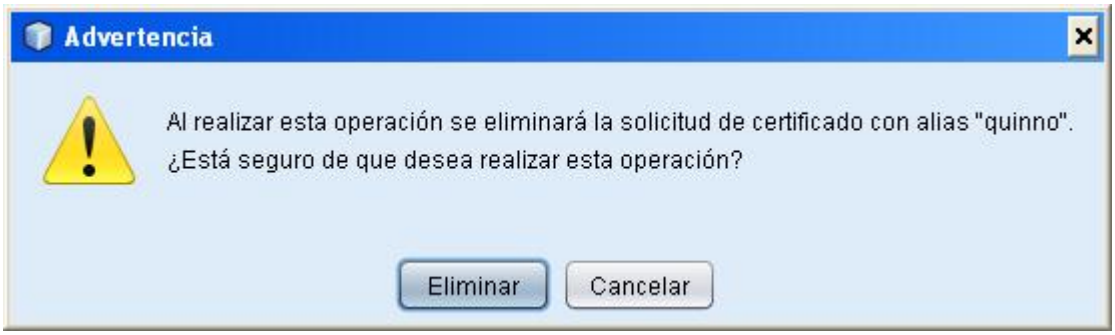
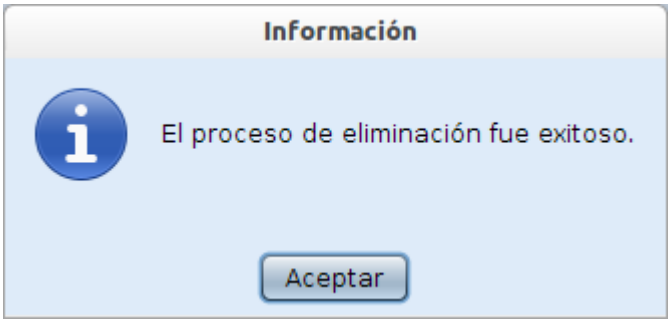
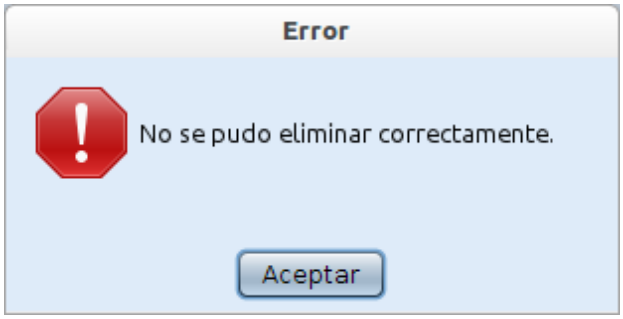
	2.2.2 Si el proceso de eliminación no tuvo éxito, el sistema muestra el mensaje: “No se pudo eliminar correctamente”. 2.3 Si el usuario selecciona la opción **Cancelar**, el sistema aborta la operación y regresa a la interfaz principal.
  	
Descripción alterna	No tiene.
Poscondiciones	Se elimina la solicitud de certificado seleccionada por el usuario.

Tabla 2.9 "**Descripción de característica:** Importar llave".

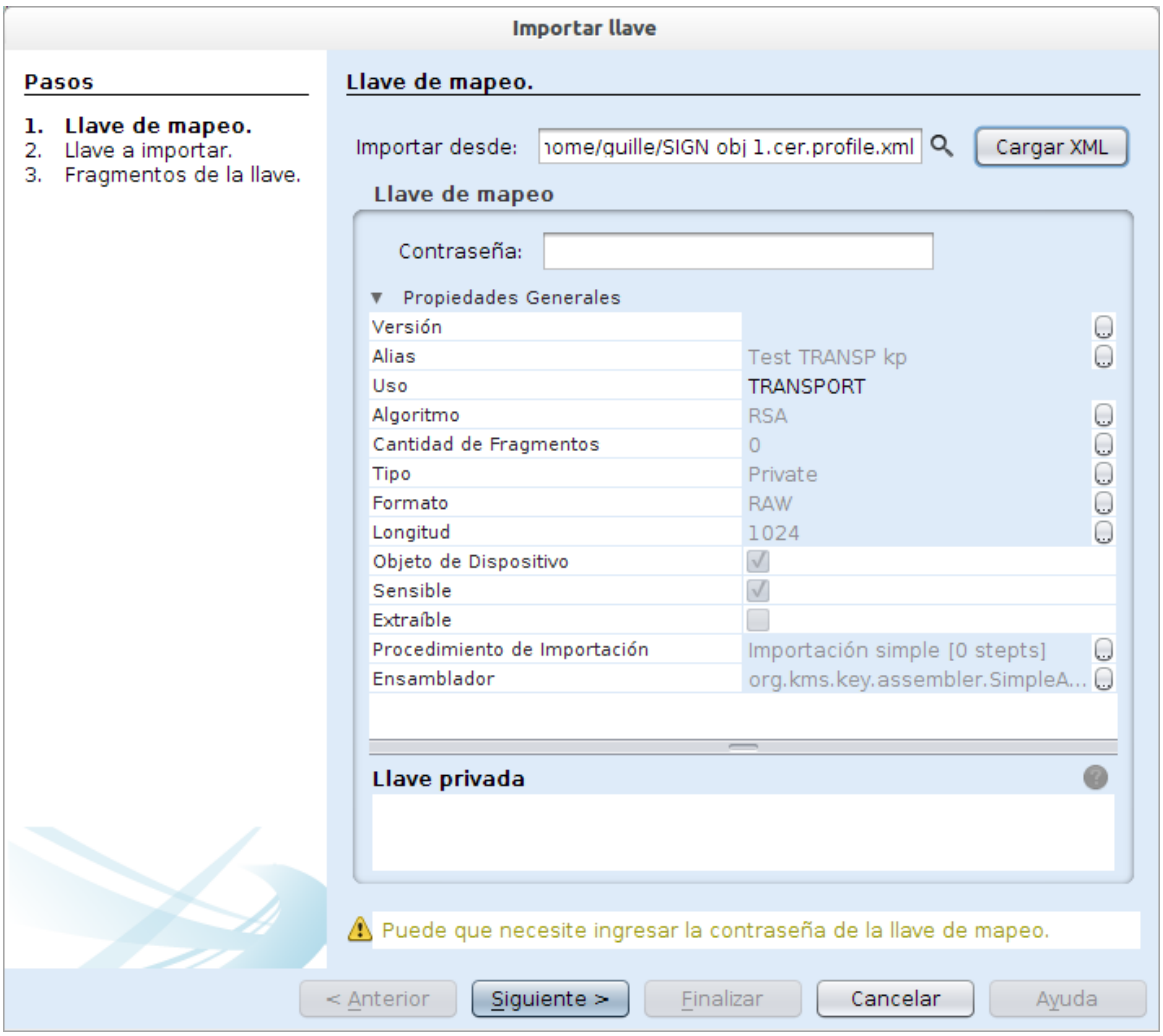
Nombre de la característica	Importar llave.
Precondiciones	El usuario debe estar autenticado en el KMS y HSM con el rol de operador o administrador.
Características asociadas	C21.
Conceptos tratados	Llave, Dispositivo criptográfico, Usuario.
Descripción básica	1. El usuario presiona el botón **Importar** de la interfaz principal. 2. El sistema muestra la interfaz "**Llave de mapeo**" que contiene: • Campos: - Contraseña. - Importar desde. • Ventana de propiedades que contiene los siguientes datos: - Alias. - Versión. - Uso. - Algoritmo. - Cantidad de fragmentos. - Tipo. - Formato. - Longitud. - Objeto de dispositivo. - Sensible. - Extraíble. - Procedimiento de importación. - Ensamblador. Opción: • Navegar. • Cargar XML.

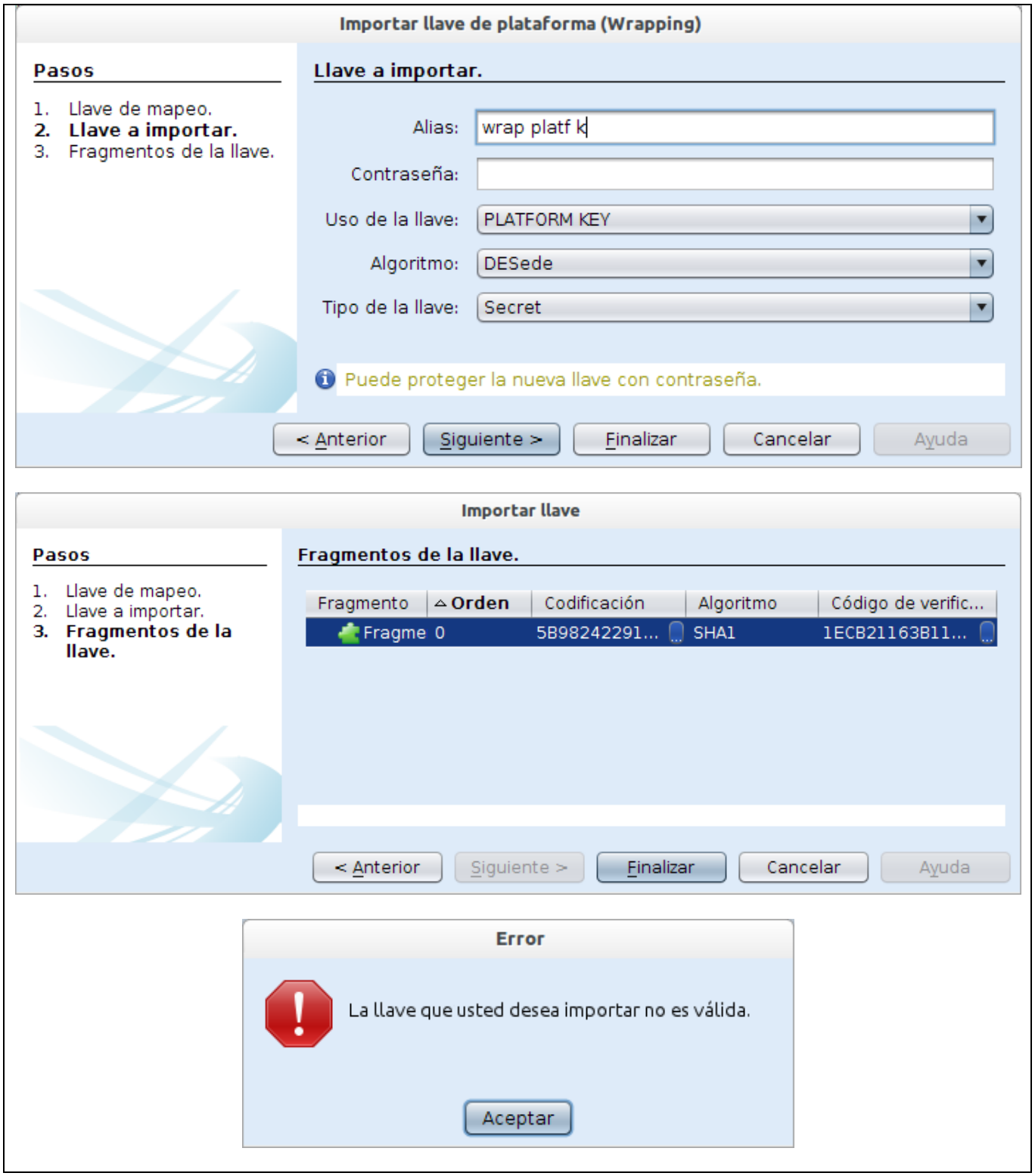
	• Anterior. (Deshabilitado) • Siguiente. (Deshabilitado) • Finalizar. (Deshabilitado) • Cancelar. • Ayuda. 2.1 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.2 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.3 Si el usuario selecciona la opción **Navegar**, el sistema muestra la interfaz "**Abrir**". 2.3.1 Si el usuario selecciona la ruta del archivo XML de la llave y presiona el botón **Abrir**, el sistema muestra la interfaz "**Llave de mapeo**" y llena el campo "Importar desde" con la ruta seleccionada. 2.3.2 Si el usuario presiona el botón **Cancelar**, el sistema regresa a la interfaz "**Llave de mapeo**". 2.4 Si el campo "Importar desde" no está vacío y el usuario presiona el botón **Cargar XML**, el sistema carga los datos del archivo XML en la ventana de propiedades y habilita la opción **Siguiente**. 2.4.1 Si el usuario selecciona la opción **Siguiente**, el sistema muestra la interfaz "**Llave a importar**" que contiene los campos: • Alias. • Contraseña. • Uso de la llave. • Algoritmo. • Tipo de la llave. Opción: • Anterior.

	• Siguiente. • Finalizar. • Cancelar. • Ayuda. 2.4.1.1 Si el usuario selecciona la opción **Anterior**, el sistema regresa a la interfaz "**Llave de mapeo**". 2.4.1.2 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.4.1.3 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.4.1.4 Si el usuario selecciona la opción **Finalizar**, el sistema importa los fragmentos de la llave cargados al KMS. 2.4.1.5 Si el usuario selecciona la opción **Siguiente**, el sistema muestra la interfaz "**Fragmentos de la llave**" que muestra los fragmentos de la llave cargados hasta el momento y da la posibilidad de cargar otros. Esta interfaz contiene una tabla con los siguientes campos: • Fragmento. • Orden. • Codificación. • Algoritmo. • Código de verificación. Opción: • Anterior. • Siguiente. (Deshabilitado) • Finalizar. • Cancelar. • Ayuda. 2.4.1.5.1 Si el usuario selecciona la opción **Anterior**, el sistema

	regresa a la interfaz “**Llave a importar**”. 2.4.1.5.2 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.4.1.5.3 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz “**Ayuda**”, con información de ayuda sobre el proceso. 2.4.1.5.4 Si el usuario selecciona la opción **Finalizar**, 2.4.1.5.4.1 Si no se cargaron todos los fragmentos de la llave, el sistema muestra el mensaje: “La llave que usted desea importar no es válida”. 2.4.1.5.4.2 Si se cargaron todos los fragmentos de la llave, el sistema importa los fragmentos de la llave cargados al KMS.

CAPÍTULO 2. Características del sistema.




Importar llave de plataforma (Wrapping)
Pasos
1. Llave de mapeo.
2. Llave a importar.
3. Fragmentos de la llave.
Llave a importar.
Alias:
wrap platf k
Contraseña:
Uso de la llave:
PLATFORM KEY
Algoritmo:
DESede
Tipo de la llave:
Secret
Puede proteger la nueva llave con contraseña.
< Anterior
Siguiente >
Finalizar
Cancelar
Ayuda
Importar llave
Pasos
1. Llave de mapeo.
2. Llave a importar.
3. Fragmentos de la llave.
Fragmentos de la llave.
Fragmento
Orden
Codificación
Algoritmo
Código de verific...
Fragme 0
5B98242291...
SHA1
1ECB21163B11...
< Anterior
Siguiente >
Finalizar
Cancelar
Ayuda
Error
La llave que usted desea importar no es válida.
Aceptar

Descripción alterna	No tiene.
Poscondiciones	Queda almacenada en el KMS y HSM la llave importada.

Tabla 2.10 "**Descripción de característica.** Importar certificado".

Nombre de la característica	Importar certificado.
Precondiciones	El usuario debe estar autenticado en el KMS y HSM con el rol de operador o administrador.
Características asociadas	C22.
Conceptos tratados	Certificado, Usuario, AC.
Descripción básica	1. Si el usuario presiona el botón **Importar** de la interfaz principal, el sistema muestra la interfaz "**Importar certificado**" que contiene los siguientes campos: • Alias. • Dirección. Opción: • Navegar. • Importar. (Deshabilitado) • Cancelar. 2. Si el usuario llena los campos, el sistema habilita la opción **Importar**. 2.1 Si el usuario selecciona la opción **Navegar**, el sistema muestra la interfaz "**Abrir**" para que el usuario escoja la ruta del archivo a importar. 2.1.1 Si el usuario escoge la ruta del archivo y selecciona la opción **Importar**, el sistema verifica que el archivo seleccionado sea un certificado bien estructurado y que esté validado por una

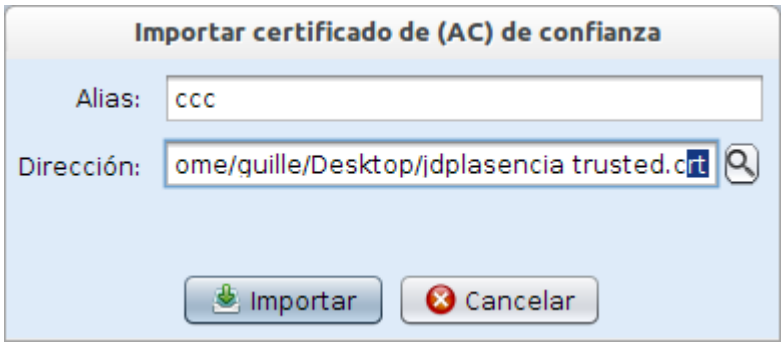
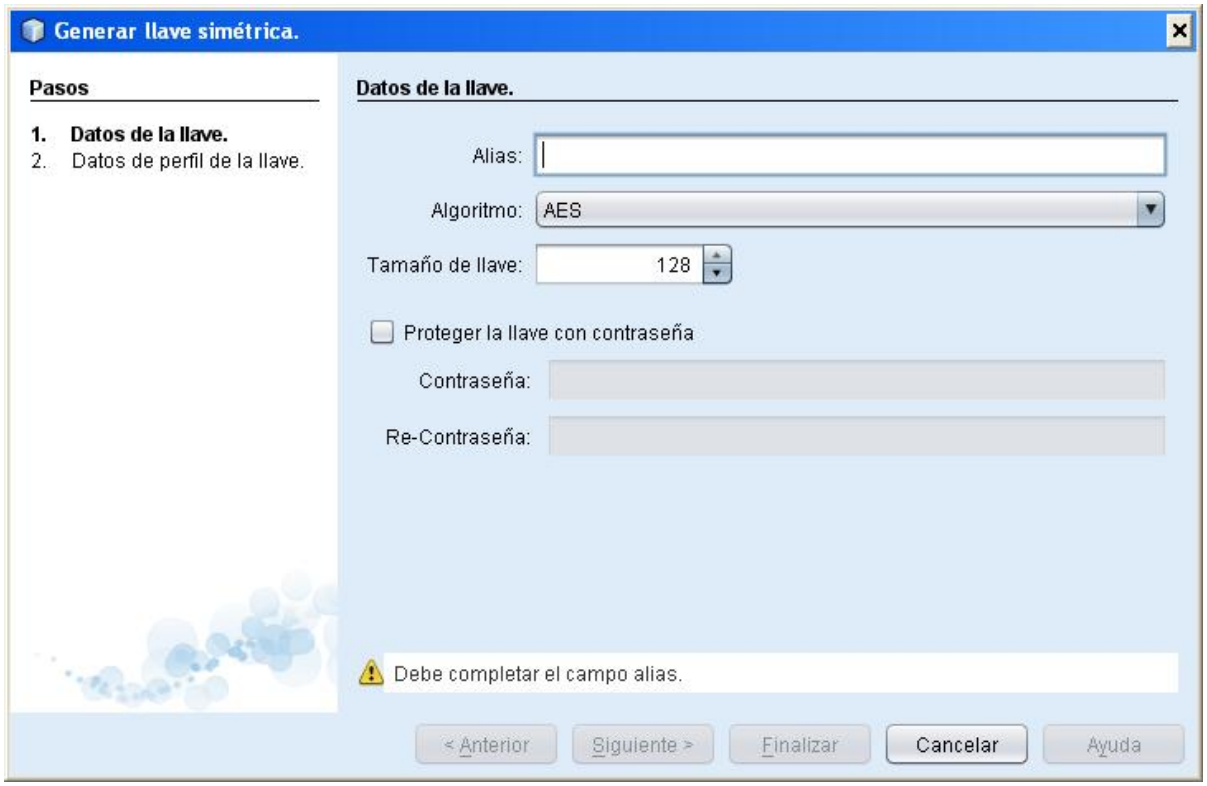
	Autoridad de Certificación (CA) de confianza. 2.1.2 Si el certificado es válido, el sistema almacena el certificado especificado y regresa a la interfaz principal. 2.1.3 Si el certificado no es válido, el sistema muestra el mensaje de error: “Certificado no válido”. 3. Si el usuario selecciona la opción **Cancelar**, el sistema aborta la operación y regresa a la interfaz principal.
 	
Descripción alterna	No tiene.
Poscondiciones	Queda almacenado en el KMS y HSM el certificado importado.

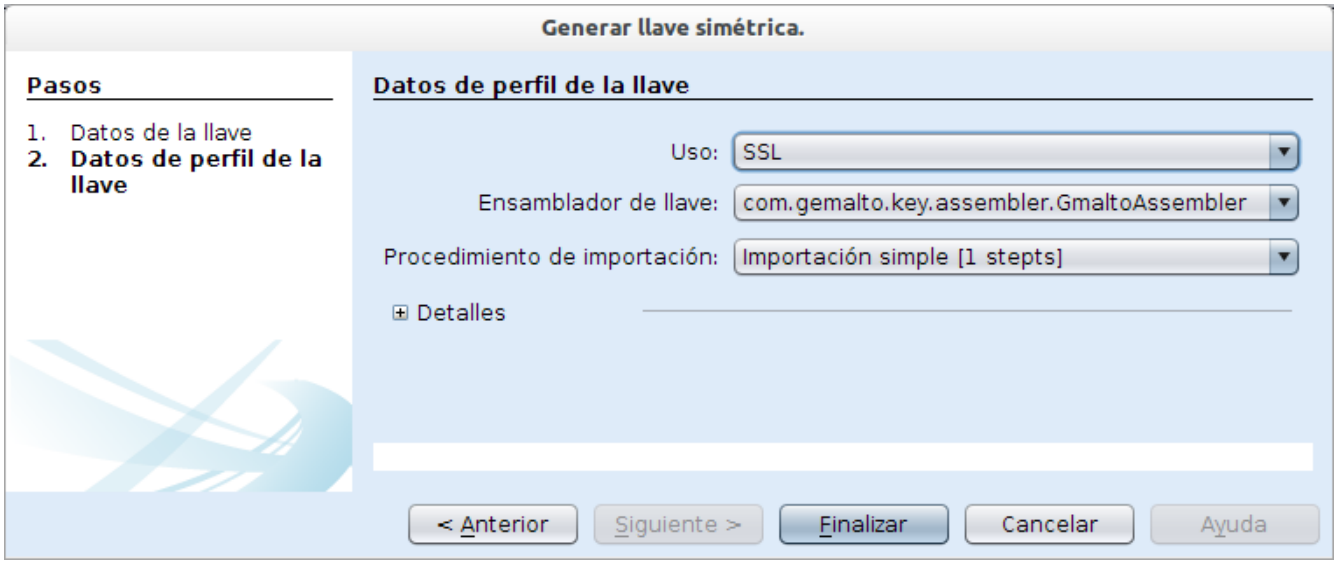
Tabla 2.11 “**Descripción de característica.** Generar llave simétrica”.

Nombre de la característica	Generar llave simétrica.
Precondiciones	El usuario debe estar autenticado en el KMS y HSM con el rol de operador o administrador.

Características asociadas	C15.
Conceptos tratados	Llave, Perfil de llave, Uso, Algoritmo de llave, Usuario.
Descripción básica	1. El sistema muestra la interfaz principal con la opción **Generar llave simétrica**. 2. Si el usuario selecciona la opción **Generar llave simétrica**, el sistema muestra la interfaz "**Datos de la llave**" que contiene: • Alias. • Algoritmo. • Tamaño de la llave. • Proteger la llave con contraseña. • Contraseña (Deshabilitado). • Re-Contraseña (Deshabilitado). Opción: • Anterior (Deshabilitado). • Siguiente (Deshabilitado). • Finalizar (Deshabilitado). • Cancelar. • Ayuda. 2.1 Si el usuario selecciona proteger la llave con contraseña, el sistema habilita los campos para introducir la contraseña. 2.2 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.3 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.4 Si el usuario introduce los datos, el sistema habilita la opción **Siguiente**. 2.4.1 Si el usuario selecciona la opción **Siguiente**, el sistema muestra la interfaz "**Datos de perfil de la llave**" que contiene: • Ensamblador de llave.

	• Uso. • Procedimiento de importación. Opción: • Anterior. • Siguiente (Deshabilitado). • Finalizar. • Cancelar. • Ayuda. 2.4.1.1 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.4.1.2 Si el usuario selecciona la opción **Anterior**, el sistema regresa a la interfaz "**Datos de la llave**". 2.4.1.3 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.4.1.4 Si el usuario selecciona la opción **Finalizar**, el sistema genera la llave y la almacena en el HSM y en el KMS.





Descripción alterna	No tiene.

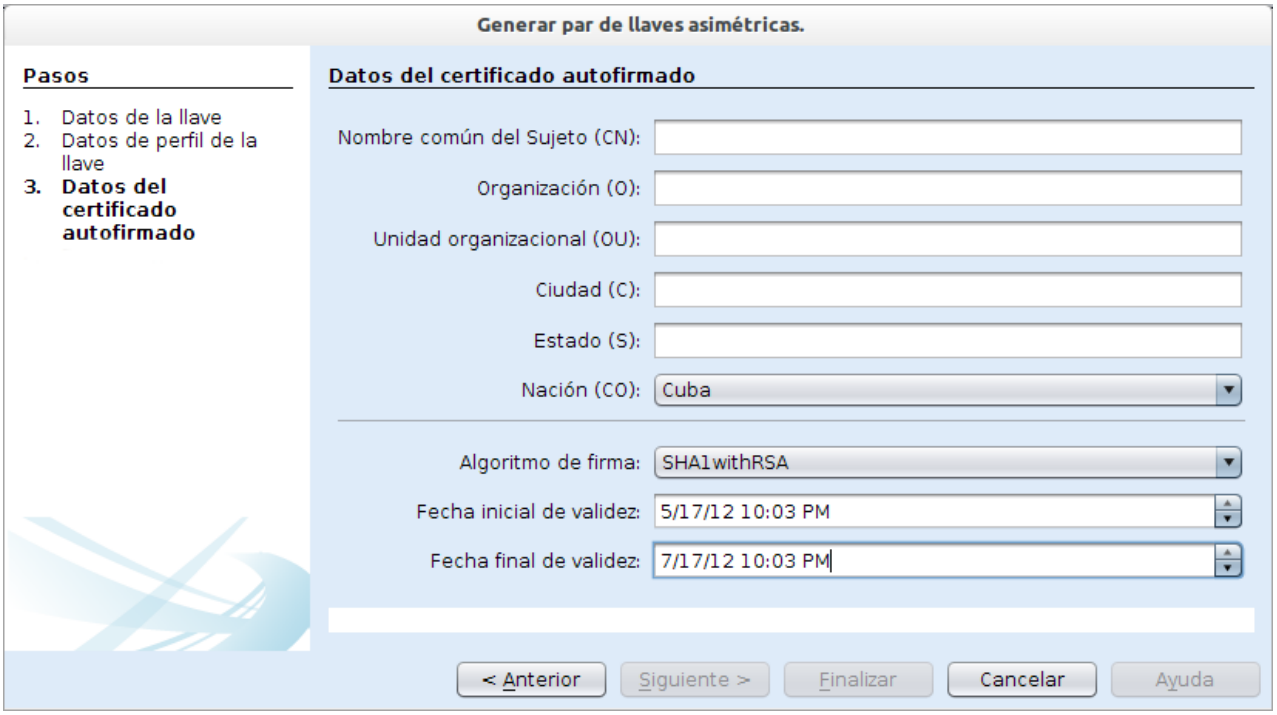
Poscondiciones	Que registrada en el KMS y HSM una llave simétrica.

Tabla 2.12 "**Descripción de característica.** Generar par de llaves y certificado auto-firmado".

Nombre de la característica	Generar par de llaves y certificado auto-firmado.
Precondiciones	El usuario debe estar autenticado en el KMS y HSM con el rol de operador o administrador.
Características asociadas	C16.
Conceptos tratados	Llave, Perfil de llave, Uso, Algoritmo de llave, Certificado, Usuario.
Descripción básica	1. El sistema muestra la interfaz principal de la aplicación de escritorio con la opción **Generar par de llaves asimétricas**. 2. Si el usuario selecciona la opción **Generar par de llaves asimétricas**, el sistema muestra la interfaz "**Datos de la llave**" que contiene: • Alias. • Algoritmo. • Tamaño de llave. • Proteger la llave con contraseña. • Contraseña (Deshabilitado). • Re-Contraseña (Deshabilitado). Opción: • Anterior (Deshabilitado). • Siguiente (Deshabilitado). • Finalizar (Deshabilitado). • Cancelar. • Ayuda. 2.1 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.2 Si el usuario selecciona proteger la llave con contraseña, el

	sistema habilita los campos para introducir la contraseña. 2.3 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.4 Si el usuario introduce los datos, el sistema habilita la opción **Siguiente**. 2.4.1 Si el usuario selecciona la opción **Siguiente**, el sistema muestra la interfaz "**Datos de perfil de la llave**" que contiene: • Ensamblador de llave. • Uso. • Procedimiento de importación. Opción: • Anterior. • Siguiente. • Finalizar (Deshabilitado). • Cancelar. • Ayuda. 2.4.1.1 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.4.1.2 Si el usuario selecciona la opción **Anterior**, el sistema regresa a la interfaz "**Datos de la llave**". 2.4.1.3 Si el usuario presiona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.4.1.4 Si el usuario selecciona la opción **Siguiente**, el sistema muestra la interfaz "**Datos del certificado auto-firmado**" que contiene: • Nombre común del sujeto (CN). • Organización (O). • Unidad organizacional (OU). • Ciudad (C).

	• Estado (S). • Nación (CO). • Algoritmo de firma. • Fecha inicial de validez. • Fecha final de validez. Opción: • Anterior. • Siguiente (Deshabilitado). • Finalizar (Deshabilitado). • Cancelar. • Ayuda. 2.4.1.4.1 Si el usuario selecciona la opción **Anterior**, el sistema regresa a la interfaz "**Datos de perfil de la llave**". 2.4.1.4.2 Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 2.4.1.4.3 Si el usuario selecciona la opción **Ayuda**, el sistema muestra la interfaz "**Ayuda**", con información de ayuda sobre el proceso. 2.4.1.4.4 Si el usuario introduce los datos del certificado, el sistema habilita la opción **Finalizar**. 2.4.1.4.4.1 Si el usuario selecciona la opción **Finalizar**, el sistema genera el par de llaves asimétricas y su certificado y los almacena en el HSM y en el KMS.

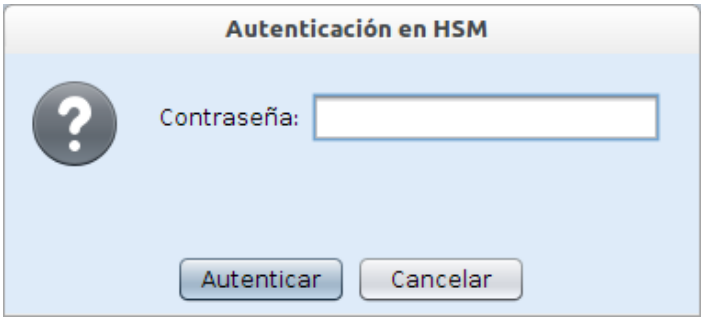


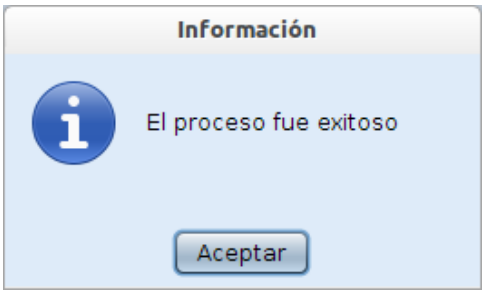
Descripción alterna	No tiene.
Poscondiciones	Se genera y almacena en el KMS y HSM un par de llaves asimétricas y el certificado auto-firmado correspondiente.

Tabla 2.13 "**Descripción de característica.** Autenticación en un HSM".

Nombre de la característica	Autenticación en un HSM.
Precondiciones	El usuario debe estar autenticado en el KMS con rol operador o administrador y debe haber añadido soporte para el dispositivo.
Características asociadas	C12.
Conceptos tratados	Dispositivo criptográfico, Usuario.

Descripción básica	1. El sistema muestra un formulario con los siguientes campos y opciones: Campos: • Contraseña. Opción: • Autenticar. • Cancelar. 2. Si el usuario selecciona la opción **Cancelar**, el sistema regresa a la interfaz principal. 3. Si el usuario introduce los datos y selecciona la opción **Autenticar**, el sistema trata de autenticarse en el HSM y muestra un cuadro de diálogo con el resultado de la operación.





	
Descripción alterna	No tiene.
Poscondiciones	El usuario queda autenticado en el HSM.

2.6 Planificación

Con el fin de realizar una planificación de la construcción de la solución propuesta, en la **tabla 2.14** se clasifican las características identificadas anteriormente, en críticas y secundarias.

Las características críticas son las que tienen mayor importancia para los clientes porque cubren las principales tareas o funciones que el sistema ha de realizar, por lo que deben implementarse en las primeras iteraciones. Las características secundarias aunque tienen un impacto más modesto sobre la aplicación, no deben dejar de implementarse en esta versión del producto.

Tabla 2.14 Clasificación de las características según su impacto.

Módulo	Características	Clasificación
Administración.	Autenticación en el KMS.	Secundaria.
	Crear un rol del sistema.	Secundaria.
	Modificar un rol del sistema.	Secundaria.
	Eliminar un rol del sistema.	Secundaria.
	Buscar rol por nombre.	Secundaria.

	Registrar un usuario en el sistema.	Secundaria.
	Modificar un usuario en el sistema.	Secundaria.
	Eliminar un usuario del sistema.	Secundaria.
	Buscar un usuario del sistema.	Secundaria.
	Habilitar operación del sistema.	Secundaria.
	Deshabilitar operación del sistema.	Secundaria.
	Buscar operaciones del sistema.	Secundaria.
Gestión de objetos criptográficos.	Autenticación en un HSM.	Crítica.
	Sincronizar HSM con KMS.	Secundaria.
	Generar llave simétrica.	Crítica.
	Generar par de llaves y certificado auto-firmado.	Crítica.
	Generar solicitud de certificado (PKCS#10).	Crítica.
	Exportar certificado.	Crítica.
	Exportar solicitud de certificado (PKCS#10).	Crítica.
	Exportar llave pública.	Crítica.
	Importar llave.	Crítica.
	Importar certificado.	Crítica.
	Eliminar objetos criptográficos.	Crítica.

	Eliminar solicitud de certificado.	Crítica.
	Activar llave de firma.	Crítica.
	Desactivar llave de firma.	Crítica.
	Adicionar soporte para dispositivo criptográfico.	Secundaria.
	Buscar objetos criptográficos por categoría.	Crítica.

2.6.1 Cronograma de diseño y construcción

En la **tabla 2.15** se muestran las fechas de ejecución de las características en las fases de diseño y construcción además de la iteración a la que pertenecen.

Tabla 2.15 Planificación de la solución.

Iteración	Módulo	Característica	Fecha de ejecución	
			Diseño	Construcción
1	Gestión de objetos criptográficos.	Autenticación en un HSM.	9/01/12 - 11/01/12	12/01/12 - 24/01/12
		Generar llave simétrica.	9/01/12 - 11/01/12	12/01/12 - 24/01/12
2	Gestión de objetos criptográficos.	Generar par de llaves y certificado.	25/01/12 - 28/01/12	29/01/12 - 9/02/12
		Generar solicitud de certificado (PKCS#10).	25/01/12 - 28/01/12	29/01/12 - 9/02/12
3	Gestión de objetos criptográficos.	Buscar objetos criptográficos por categoría.	10/02/12 - 12/02/12	13/02/12 - 23/02/12
		Eliminar objetos	10/02/12 - 12/02/12	13/02/12 - 23/02/12

		criptográficos.		
		Eliminar solicitud de certificado.	10/02/12 - 13/02/12	14/02/12 - 25/02/12
		Exportar solicitud de certificado (PKCS#10).	10/02/12 - 13/02/12	14/02/12 - 25/02/12
4	Gestión de objetos criptográficos.	Exportar certificado.	26/02/12 - 29/02/12	1/03/12 - 12/03/12
		Exportar llave pública.	26/02/12 - 29/02/12	1/03/12 - 12/03/12
5	Gestión de objetos criptográficos.	Activar llave de firma.	13/03/12 - 15/03/12	16/03/12 - 26/03/12
		Desactivar llave de firma.	13/03/12 - 15/03/12	16/03/12 - 28/03/12
6	Gestión de objetos criptográficos.	Importar llave.	29/03/12 - 31/03/12	1/04/12 - 11/04/12
		Importar certificado.	29/03/12 - 31/03/12	1/04/12 - 11/04/12
		Adicionar soporte para dispositivo criptográfico.	29/03/12 - 31/03/12	1/04/12 - 11/04/12
7	Administración.	Crear un rol del sistema.	29/03/12 - 31/03/12	1/04/12 - 13/04/12
		Modificar un rol del sistema.	29/03/12 - 31/03/12	1/04/12 - 13/04/12
		Eliminar un rol del sistema.	14/04/12 - 17/04/12	18/04/12 - 29/04/12
		Buscar rol por nombre.	14/04/12 - 17/04/12	18/04/12 - 29/04/12
8	Administración.	Registrar un usuario en el sistema.	30/04/12 – 2/05/12	3/05/12 - 10/05/12
		Modificar un usuario en el	30/04/12 - 2/05/12	3/05/12 - 10/05/12

		sistema.		
9	Administración.	Eliminar un usuario del sistema.	30/04/12 - 2/05/12	3/05/12 - 10/05/12
		Buscar un usuario del sistema.	30/04/12 - 2/05/12	3/05/12 - 10/05/12
		Habilitar operación del sistema.	30/04/12 - 2/05/12	3/05/12 - 10/05/12
		Deshabilitar operación del sistema.	30/04/12 - 2/05/12	3/05/12 - 10/05/12
		Buscar operaciones del sistema.	30/04/12 - 2/05/12	3/05/12 - 10/05/12
		Autenticación en el KMS.	5/05/12 - 7/05/12	7/05/12 - 10/05/12
		Sincronizar HSM con KMS.	5/05/12 - 7/05/12	7/05/12 - 10/05/12

2.7 Conclusiones

Con la realización de este capítulo, quedó definido un modelo general del sistema, la arquitectura del sistema y una lista de características con sus descripciones; artefactos que permitirán guiar a los desarrolladores en la implementación de la solución propuesta. Además quedó establecido un plan que garantiza la organización del trabajo y permitirá definir el estado en que se encuentra el desarrollo de la solución.

CAPÍTULO 3. Diseño del sistema

3.1 Introducción

En este capítulo se muestra cómo será construido el sistema a través de diversos diagramas y sus descripciones. En primer lugar se describen algunas de las tablas más importantes del modelo de datos. Luego se muestran los diagramas de clases agrupados según la capa de la aplicación a la que pertenecen y se hace un análisis de los patrones de diseño utilizados en cada uno de ellos. Por último se presentan diagramas de secuencias de algunas de las funcionalidades críticas, los cuales modelan la interacción entre los objetos del sistema.

3.2 Descripción de las tablas de la base de datos

El modelo de datos utilizado en la aplicación contiene 42 tablas. A continuación se describirán algunas de las más importantes, el resto de las descripciones de las tablas se pueden encontrar en el **anexo 2**.

Tabla 3.1 Descripción de la tabla “crypto_object”.

Nombre	crypto_object.	
Descripción	Esta es la tabla base de los objetos criptográficos, en ella persisten los datos comunes de los objetos criptográficos.	
Atributos	**Tipo**	**Descripción**
+id.	varchar (255).	Identificador del objeto criptográfico. Es un identificador único universal (UUID).
alias.	varchar (255).	Nombre por el cual se identifica un conjunto de objetos criptográficos: el par de llaves, el certificado y la solicitud de certificado.

Tabla 3.2 Descripción de la tabla “key”.

Nombre	key.	
Descripción	En esta tabla persisten los datos comunes de las llaves.	
Atributos	**Tipo**	**Descripción**
+#crypto_objectid.	varchar (255).	Identificador del objeto criptográfico que representa.
tokenObject.	bool.	Si este campo es true, significa que las llaves se almacenarán en el dispositivo (HSM).
sensitive.	bool.	Si este campo es true, las partes sensibles de la llave permanecerán ocultas.
extractable.	bool.	Si este campo es true, significa que la llave puede extraerse del dispositivo.
length.	int4.	Longitud de la llave.
activated.	bool.	Si este campo es true indica que la llave se usará en el proceso de firma.

Tabla 3.3 Descripción de la tabla “certificate”.

Nombre	certificate.	
Descripción	En esta tabla persisten los datos del certificado.	
Atributos	**Tipo**	**Descripción**
+#crypto_objectid.	varchar (255).	Identificador del objeto criptográfico que representa.
encoded.	varchar.	Codificación del certificado en base 64.

serialNumber.	int8.	Es un número entero asignado por la CA a cada certificado.
subjectName.	varchar (255).	Datos del sujeto al cual le pertenece el certificado.
subjectUniqueID.	varchar (255).	Se utiliza para identificar de forma única un sujeto en el caso de que subjectName sea re-usado.
derEncoded.	varchar.	Codificación DER (Reglas de Codificación Distinguida) del certificado.
signature.	varchar.	Firma del certificado.
notBefore.	date.	Fecha de inicio de validez del certificado.
notAfter.	date.	Fecha de fin de validez del certificado.
#certificate_profileid.	numeric (19,0).	Identificador del perfil del certificado.
#issuerid.	int4.	Identificador de la Autoridad Certificadora (CA).
#ncl_certificate_typeid.	int4.	Identificador del nomenclador del tipo de certificado.
#public_key.	varchar (255).	Identificador de la llave pública que posee el certificado.

Tabla 3.4 Descripción de la tabla “certifícate_request”.

Nombre	certificate_request.	
Descripción	En esta tabla persisten los datos de la solicitud de certificado.	
Atributos	**Tipo**	**Descripción**
+#crypto_objectid.	varchar (255).	Identificador del objeto criptográfico que representa.

version.	int4.	Número de versión para compatibilidad con el estándar PKCS#10.
subjectDN.	varchar (255).	Datos del sujeto al cual le pertenece la solicitud de certificado.
#public_key.	varchar (255).	Identificador de la llave pública al cual se le generará el certificado.
#ncl_signature_algorithmid.	int4.	Identificador del algoritmo de firma a utilizar.
#stateid.	int4.	Identificador del estado de la solicitud (exportada, generada).

3.3 Diseño de la capa de acceso a datos

3.3.1 Diagrama de clases

En la capa de acceso a datos existe una clase entidad asociada a cada tabla de la base de datos, excepto las tablas que se generan de las relaciones mucho – mucho. Estas clases tienen asociadas un archivo de mapeo el cual le dice a Hibernate cómo transformar los datos de estas clases a los campos de la tabla asociada y viceversa. Debido a que el diseño de estas clases es muy parecido al del modelo de datos, sólo se muestra el diseño de una de estas clases y su archivo de mapeo en el **anexo 3**.

A continuación, en la **figura 3.1**, se muestra el diseño de las clases DAO (Data Access Object), el cual se realizó siguiendo el patrón *Fachada* (Facade) y el patrón *DAO*.

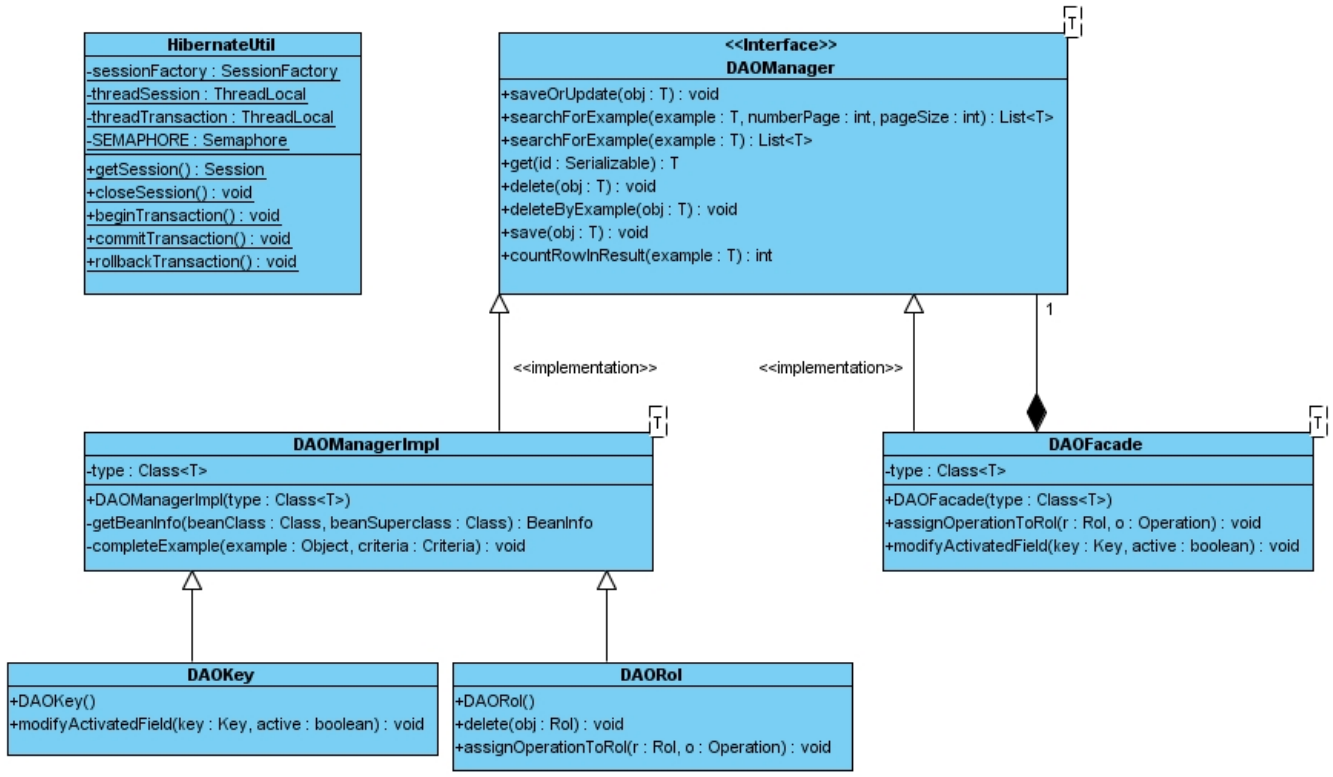


Figura 3.1 Diagrama de clases DAO.

El patrón fachada se utiliza para proporcionar una interfaz unificada de alto nivel para la capa de acceso a datos, haciendo más fácil su uso. Este simplifica el acceso a dicho conjunto de clases, ya que el que requiera utilizarla sólo se comunica con ellas a través de una única interfaz.

El patrón DAO se utiliza para encapsular la forma de acceder a la fuente de datos.

2.4.2 Descripción de las clases

Tabla 3.5 Descripción de clase “HibernateUtil”.

Nombre: HibernateUtil.	
Tipo de clase: controladora. Está diseñada para manejar las sesiones y las transacciones a la base de datos.	
Atributo	**Tipo**

sessionFactory.	SessionFactory.
threadSession.	ThreadLocal.
threadTransaction.	ThreadLocal.
SEMAPHORE.	Semaphore.
Para cada responsabilidad:	
Nombre:	**getSession()**
Descripción:	Obtiene la sesión existente, de no existir la crea.
Nombre:	**closeSession()**
Descripción:	Cierra la sesión existente.
Nombre:	**beginTransaction()**
Descripción:	Obtiene la transacción existente, de no existir la crea.
Nombre:	**commitTransaction()**
Descripción:	Termina la transacción existente.
Nombre:	**rollbackTransaction()**
Descripción:	Hace retroceder la última transacción realizada.

Tabla 3.6 Descripción de clase "DAOManager".

Nombre: DAOManager<T>[3].
Tipo de clase: interfaz.

[3] T es el parámetro genérico de la clase.

Define el comportamiento general de una clase de acceso a dato.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**saveOrUpdate(T obj)**
Descripción:	Si el objeto que se le pasa por parámetro no existe en la base de datos, lo guarda; sino lo actualiza.
Nombre:	**searchForExample(T example)**
Descripción:	Devuelve una lista de los objetos cuyos atributos tienen el mismo valor que los atributos del objeto que se pasa por parámetro (ignorando los atributos id y los atributos que tienen valor null).
Nombre:	**searchForExample(T example, int numberPage, int pageSize)**
Descripción:	Devuelve una lista de los objetos que se obtienen de la misma forma que en el método anterior, pero esta vez el resultado se devuelve paginado, comenzando por el valor de la variable **numberPage** y con la cantidad de elementos de **pageSize**.
Nombre:	**get(Serializable id)**
Descripción:	Se obtiene de la base de datos el objeto con identificador igual al que se pasa por parámetro.
Nombre:	**delete(T obj)**
Descripción:	Elimina el objeto que coincida con el objeto que se pasa por parámetro.
Nombre:	**deleteByExample(T obj)**

Descripción:	Elimina todos los objetos que cumplan con el ejemplo que se pasa por parámetro.
Nombre:	**save(T obj)**
Descripción:	Guarda el objeto que se pasa por parámetro.
Nombre:	**countRowInResult(T example)**
Descripción:	Devuelve la cantidad de tuplas que se retornarían al hacer una búsqueda por ejemplo con el objeto que se pasa por parámetro.

Tabla 3.7 Descripción de clase "DAOManagerImpl".

Nombre: DAOManagerImpl<T>.	
Tipo de clase: controladora. Implementa el comportamiento por defecto de las clases de acceso a dato.	
Atributo	**Tipo**
type.	Class<T>.
Para cada responsabilidad:	
Nombre:	**DAOManageImpl(Class<T> type)**
Descripción:	Construye un objeto de la clase.
Nombre:	**getBeanInfo(Class beanClass, Class beanSuperclass)**
Descripción:	Devuelve un objeto con información sobre la clase que se le pasa por parámetro: atributos y métodos de acceso.
Nombre:	**completeExample(Object example, Criteria criteria)**
Descripción:	Completa el Criteria que se le pasa por parámetro formando el árbol de Criterion

	con las subclases que contiene el objeto **example**.

Tabla 3.8 Descripción de clase “DAOKey”.

Nombre: DAOKey.	
Tipo de clase: controladora. Hereda el comportamiento por defecto de las clases de acceso a dato y adiciona un comportamiento más para los objetos de tipo llave.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**DAOKey()**
Descripción:	Construye un objeto de la clase.
Nombre:	**modifyActivatedField(Key key, boolean active)**
Descripción:	Modifica el campo “activated” de la tabla “key” en la base de datos.

Tabla 3.9 Descripción de clase “DAORol”.

Nombre: DAORol.	
Tipo de clase: controladora. Hereda el comportamiento por defecto de las clases de acceso a dato, sobrescribe el comportamiento **delete** y adiciona un comportamiento más para los objetos de tipo rol.	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	**DAORol()**
Descripción:	Construye un objeto de la clase.
Nombre:	**delete(Rol obj)**
Descripción:	Elimina un rol de la base de datos.
Nombre:	**assignOperationToRol(Rol r, Operation o)**
Descripción:	Asigna una operación a un rol.

Tabla 3.10 Descripción de clase "DAOFacade".

Nombre: DAOFacade<T>.	
Tipo de clase: controladora. Brinda una fachada con los comportamientos de las clases de acceso a dato.	
Atributo	**Tipo**
type.	Class<T>.
manager.	DAOManager<T>.
Para cada responsabilidad:	
Nombre:	**DAOFacade(Class<T> type)**
Descripción:	Crea una instancia de la clase, instanciando a su vez el DAOManager perteneciente al tipo pasado por parámetro.
Nombre:	**assignOperationToRol(Rol r, Operation o)**
Descripción:	Asigna una operación a un rol.

Nombre:	**modifyActivatedField(Key key, boolean active)**
Descripción:	Modifica el campo “activated” de la tabla “key” en la base de datos.

2.5 Diseño de la capa lógica de negocio

2.5.1 Diagrama de clases

El diagrama de clases de la capa lógica de negocio se muestra en las **figuras 3.2**, **3.3** y **3.4**. En la **figura 3.2** se esquematiza la estructura de clases de los objetos de transferencia de datos (DTO por sus siglas en inglés), los cuales son objetos que se utilizan para transportar la información desde un objeto de negocio a un objeto de tipo entidad. Cada clase entidad de acceso a dato tiene asociado una clase DTO. En este diagrama sólo se representan algunas de las clases DTO.

Las clases de la **figura 3.4** que presentan en su nombre el sufijo "Manager" son las clases controladoras. En la mayoría de estas clases se puede observar el patrón *Instancia Única* (Singleton), este se utiliza con el fin de restringir la creación de objetos de estas clases a una sola instancia.

En el diseño de la clase DataStore se puede observar el patrón *Creador* y el patrón *Método Fábrica* (Factory Method), ya que a esta clase se le ha asignado la responsabilidad de crear los objetos DTO y decidir, según la clase de dominio, qué clase de la jerarquía de clases DTO le corresponde instanciar.

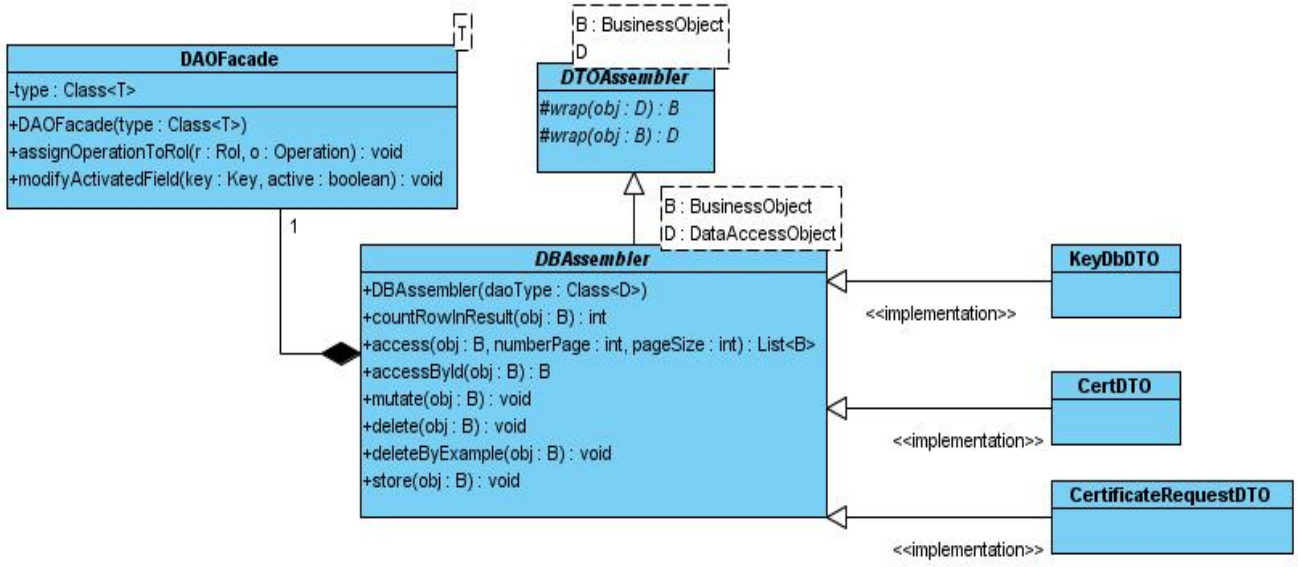


Figura 3.2 Segmento 1 del diagrama de clases “Lógica del negocio”.

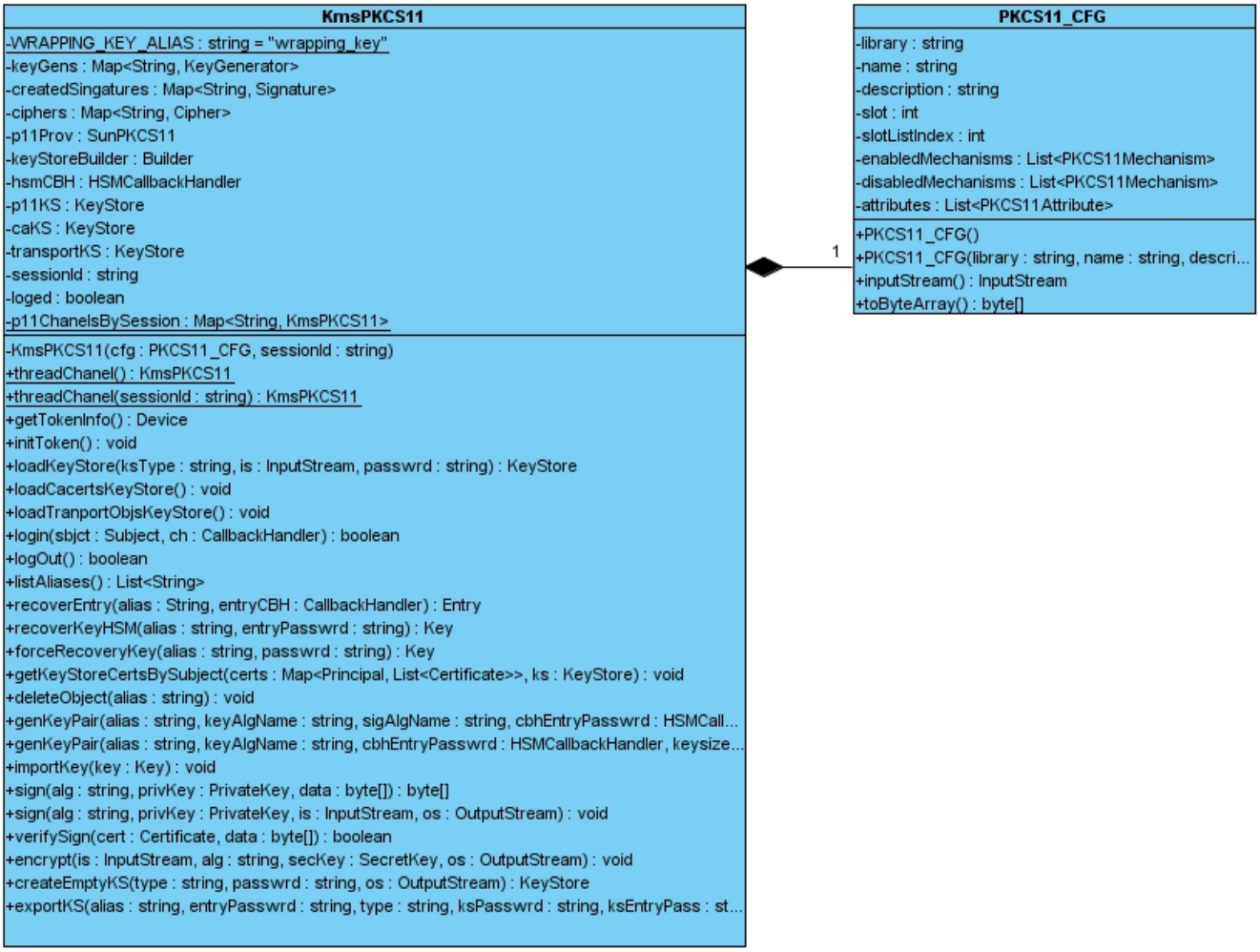


Figura 3.3 Segmento 2 del diagrama de clases “Lógica del negocio”.

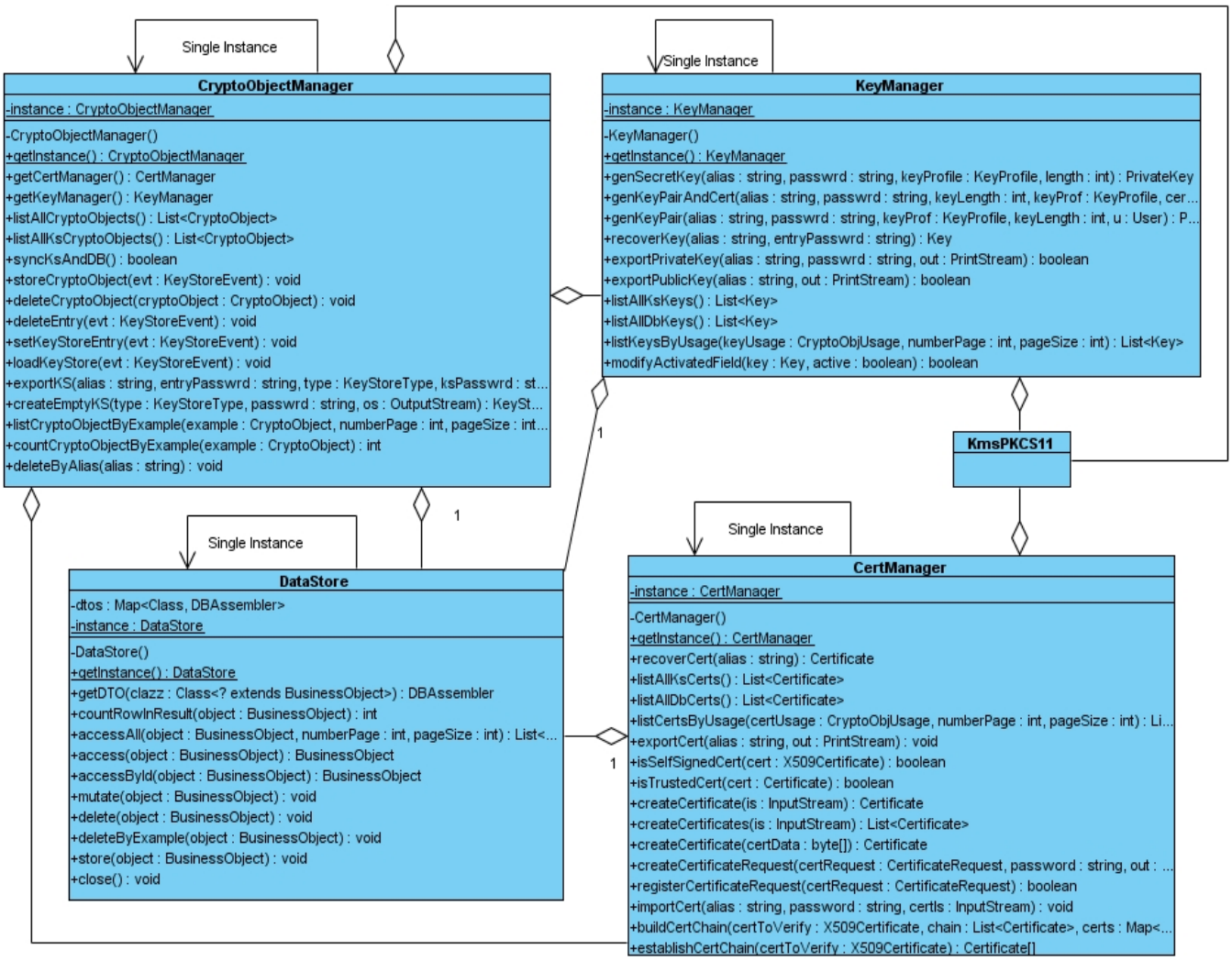


Figura 3.4 Segmento 3 del diagrama de clases “Lógica del negocio”.

2.5.2 Descripción de las clases

Tabla 3.11 Descripción de clase “DTOAssembler”.

Nombre: DTOAssembler<B extends BussinesObject, D>.
Tipo de clase: interfaz. Clase abstracta que define el comportamiento de las clases DTO.

Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**wrap(D obj)**
Descripción:	Define el comportamiento de convertir un objeto determinado al tipo de objeto de negocio.
Nombre:	**wrap(B obj)**
Descripción:	Define el comportamiento de convertir un objeto de negocio a otro tipo de objeto.

Tabla 3.12 Descripción de clase “DBAssembler”.

Nombre: DBAssembler<B extends BussinesObject, D extends DataAccessObject>.	
Tipo de clase: controladora. Esta clase contiene comportamientos genéricos que se realizan sobre los objetos del sistema.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**countRowInResult(B obj)**
Descripción:	Devuelve la cantidad de tuplas que se retornarían al hacer una búsqueda por ejemplo con el objeto que se pasa por parámetro.
Nombre:	**access(B obj, int numberPage, int pageSize)**
Descripción:	Devuelve una lista con el resultado de buscar por ejemplo el objeto que se pasa

	por parámetro.
Nombre:	**accessById(B obj)**
Descripción:	Devuelve un objeto de negocio con el identificador del ejemplo que se pasa por parámetro.
Nombre:	**mutate(B obj)**
Descripción:	Hace persistir o actualiza el objeto que se pasa por parámetro.
Nombre:	**delete(B obj)**
Descripción:	Elimina de la base de datos el objeto que se pasa por parámetro.
Nombre:	**deleteByExample(B obj)**
Descripción:	Elimina todos los objetos que coincidan con el ejemplo que se pasa por parámetro.
Nombre:	**store(B obj)**
Descripción:	Hace persistir el objeto que se pasa por parámetro.

El resto de las descripciones de las clases se encuentran en el **anexo 4.**

3.5 Diseño de la capa sesión

3.5.1 Diagrama de clases

El diagrama de clases de la capa sesión se dividió en tres segmentos y se muestran en la **figura 3.4**, **3.5** y **3.6**. En el diseño de la clase CompositePolicy se puede apreciar el patrón *Compuesto* (Composite) el cual permite que se construyan políticas de acceso (Policy) complejas a partir de otras más simples.

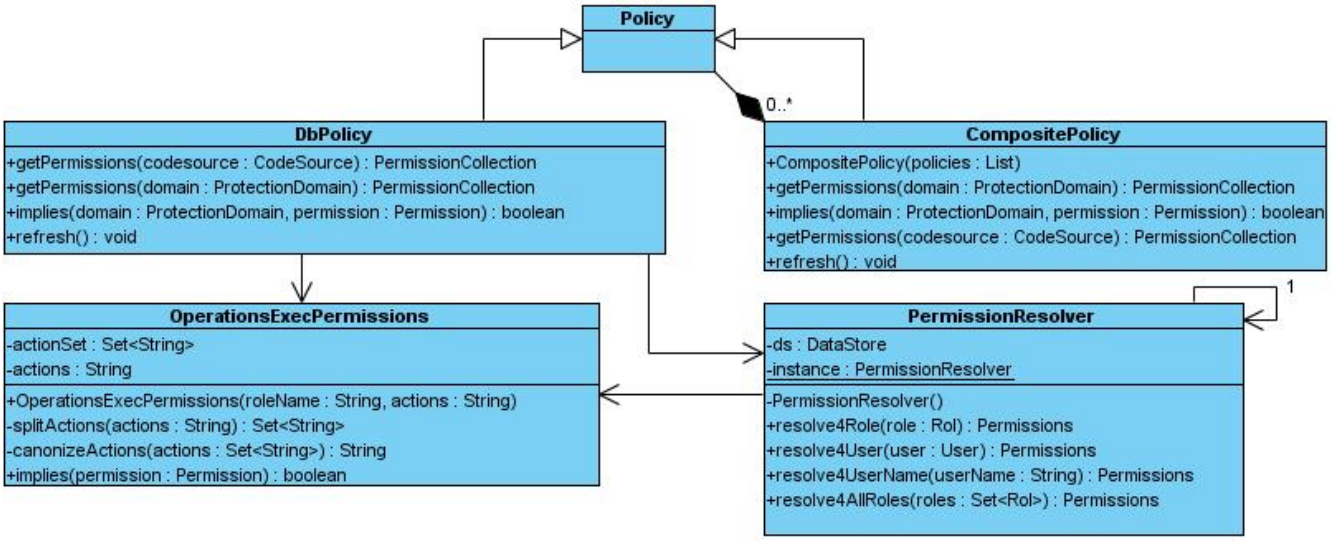


Figura 3.4 Segmento 1 del diagrama de clase “Capa de sesión”.

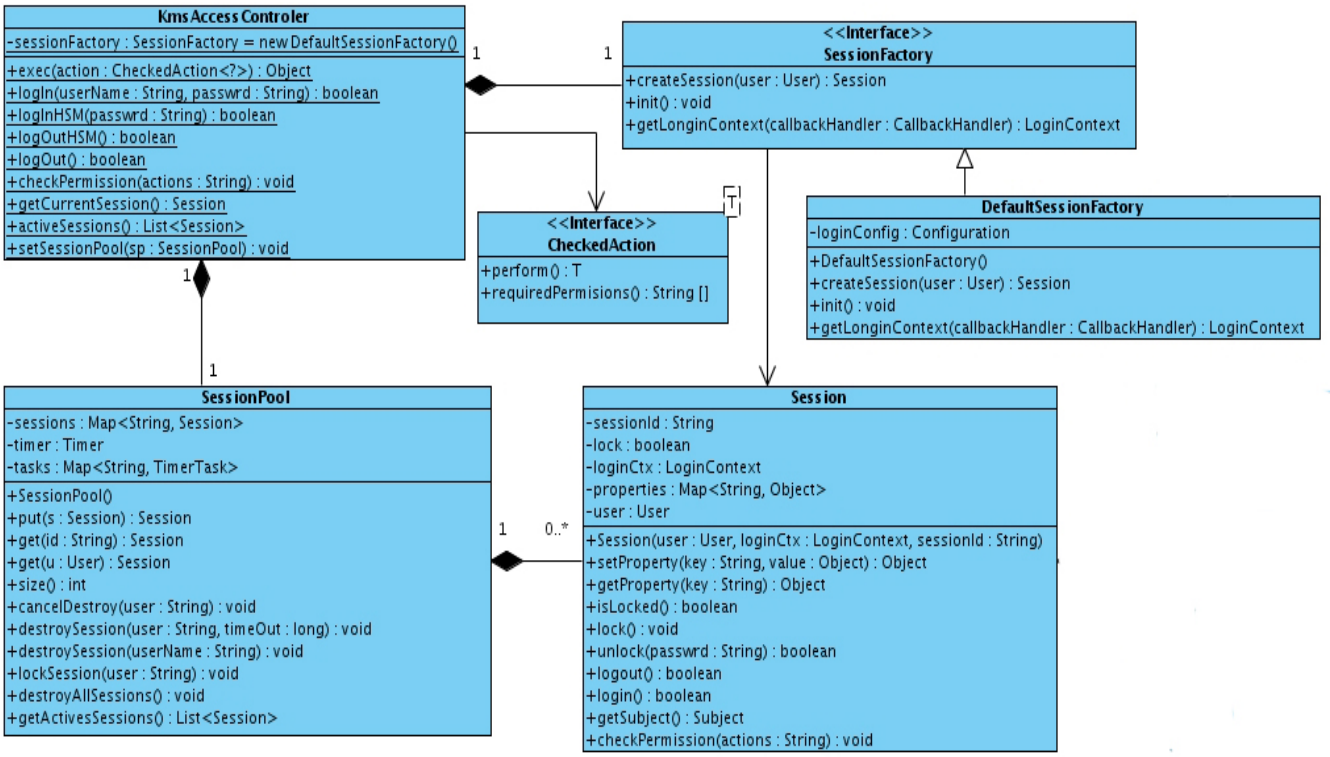


Figura 3.5 Segmento 2 del diagrama de clase “Capa de sesión”.

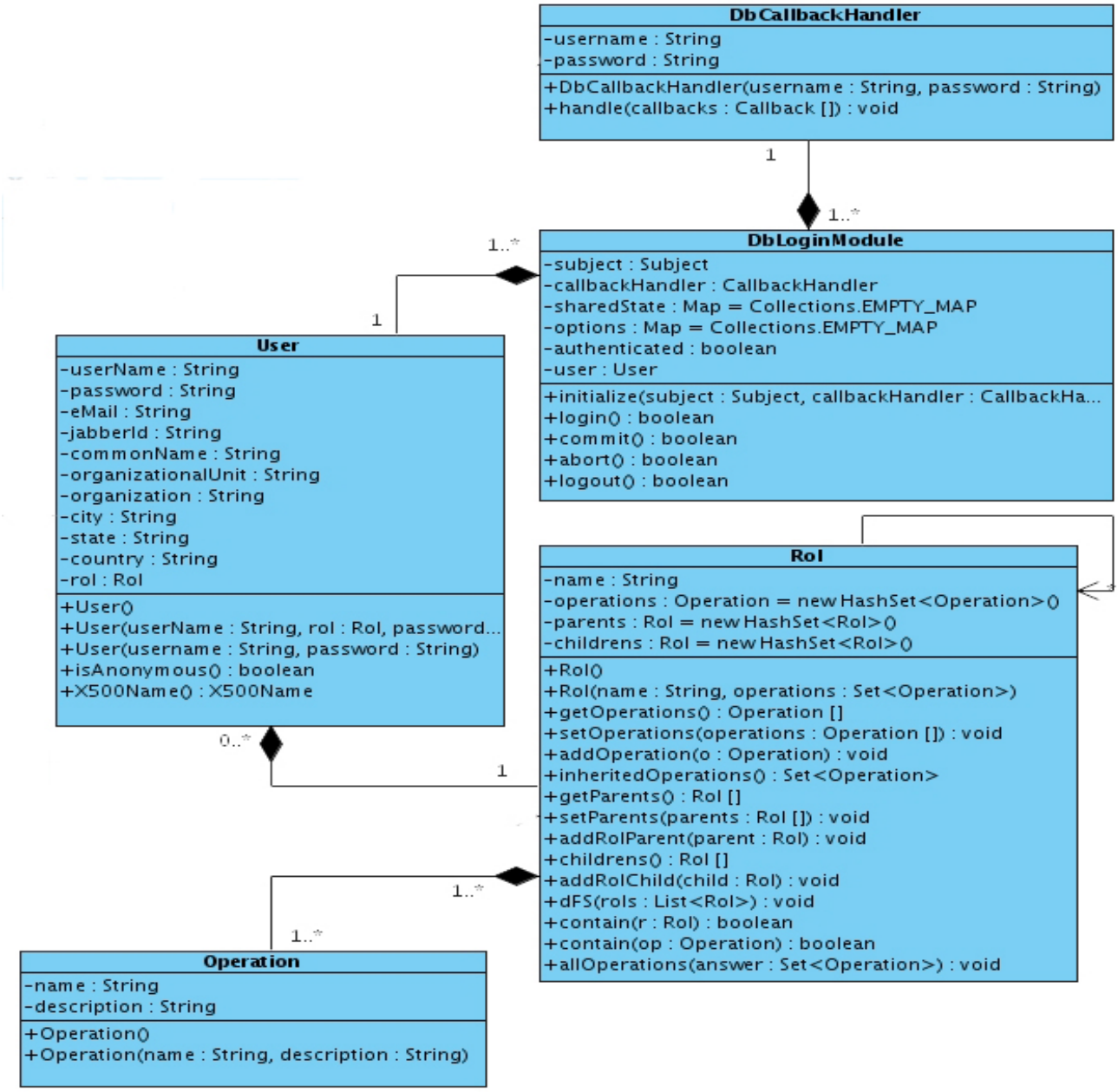


Figura 3.6 Segmento 3 del diagrama de clase “Capa de sesión”.

3.5.2 Descripción de las clases

Las descripciones de las clases de la capa sesión se encuentran en el **anexo 5**.

3.6 Diseño de la capa middleware

3.6.1 Diagrama de clases

En la **figura 3.7** se muestra el diagrama de clases de la capa middleware. Para el diseño de este diagrama se aplicó el patrón *Observador* (Observer), con el objetivo de mantener desacoplado a los objetos que les interese obtener información de las solicitudes y respuestas que llegan al middleware. También se aplicó el patrón *Instancia Única* y el patrón *Adaptador* (Adapter). El patrón *Adaptador* se utilizó para convertir la interfaz general "MiddlewareEventListener" en la interfaz deseada por los objetos que escuchan los eventos que ocurren en el middleware, evitando así que estos tengan que implementar funcionalidades que no le interesan realizar.

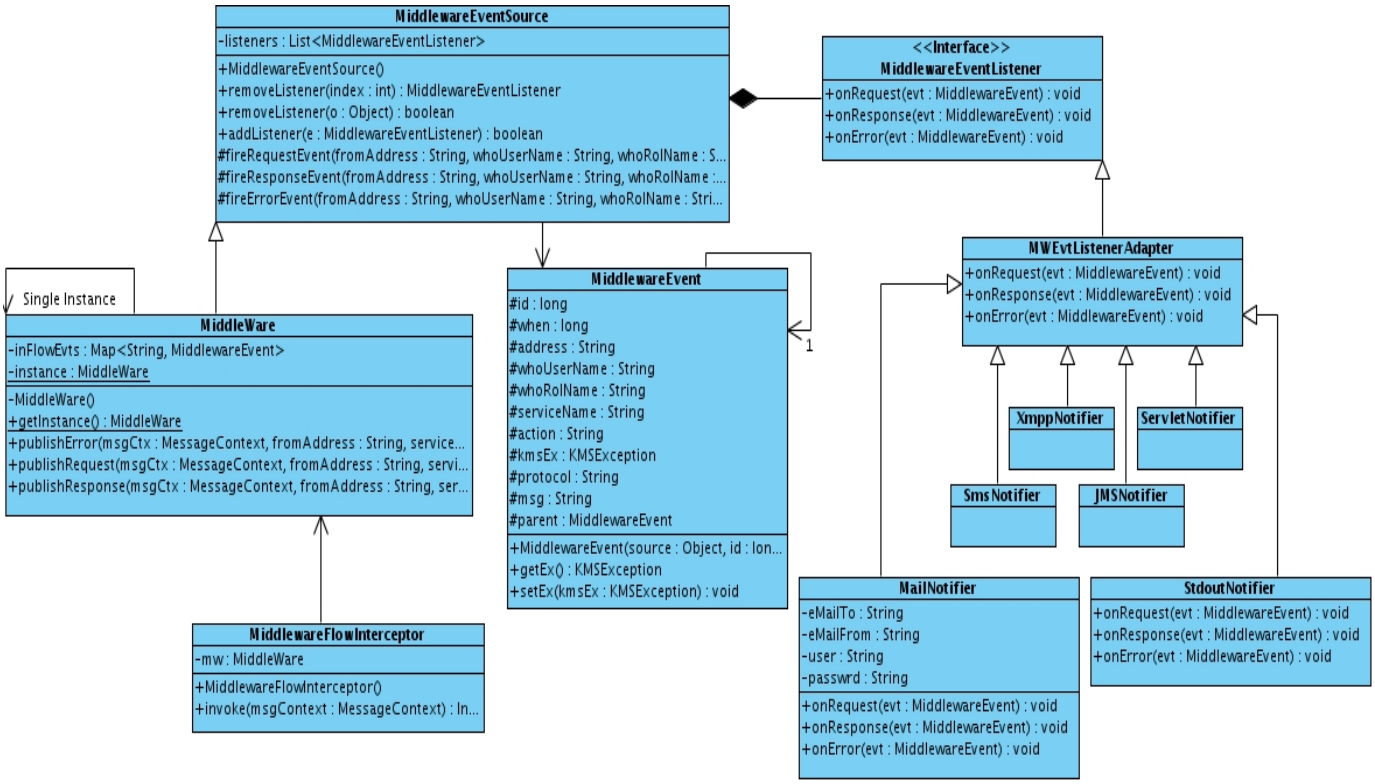


Figura 3.7 Diagrama de clase de la capa middleware.

3.6.2 Descripción de las clases

Las descripciones de las clases de la capa middleware se encuentran en el **anexo 6**.

3.7 Diseño de la capa de presentación

3.7.1 Diagrama de clases

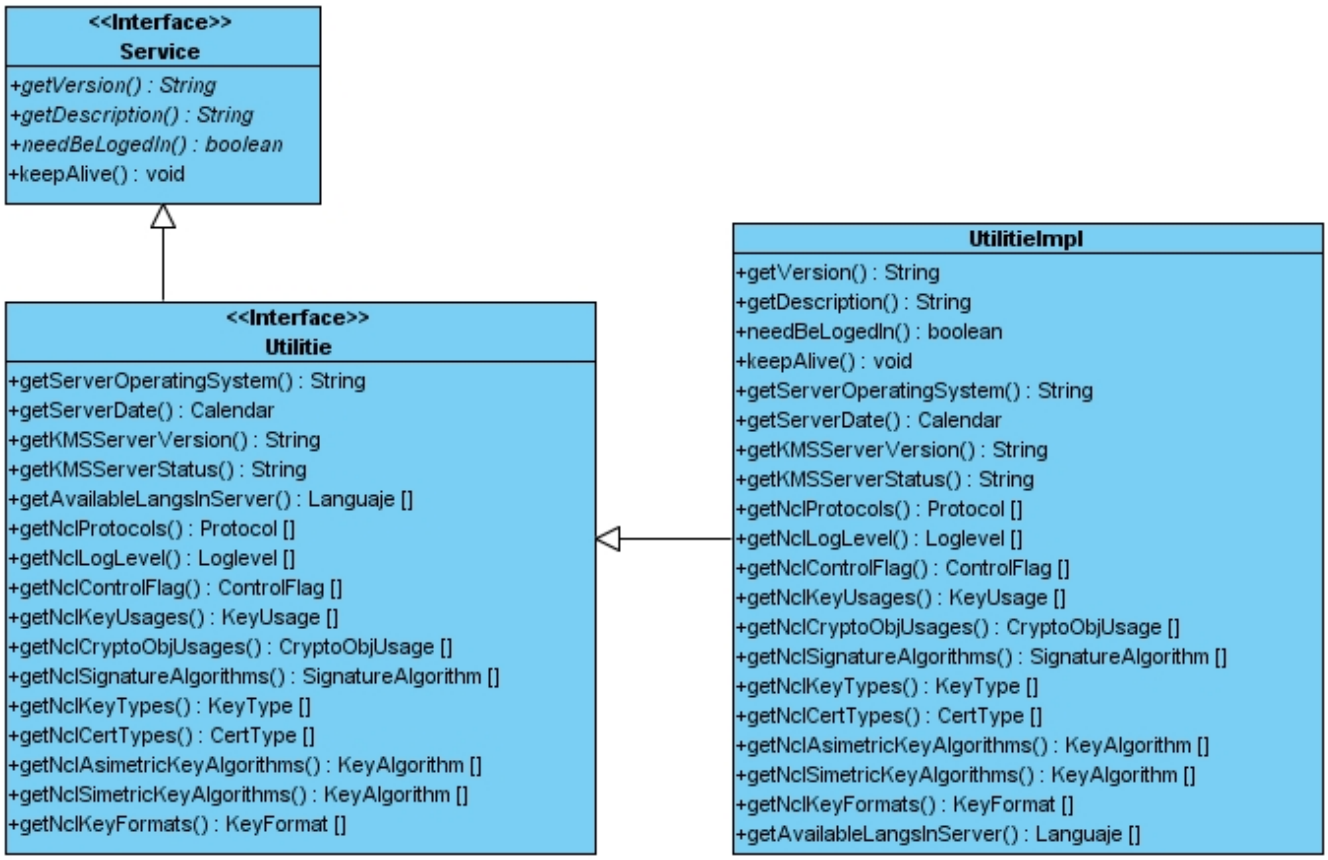


Figura 3.8 Segmento 1 del diagrama de clase “Capa de presentación”.

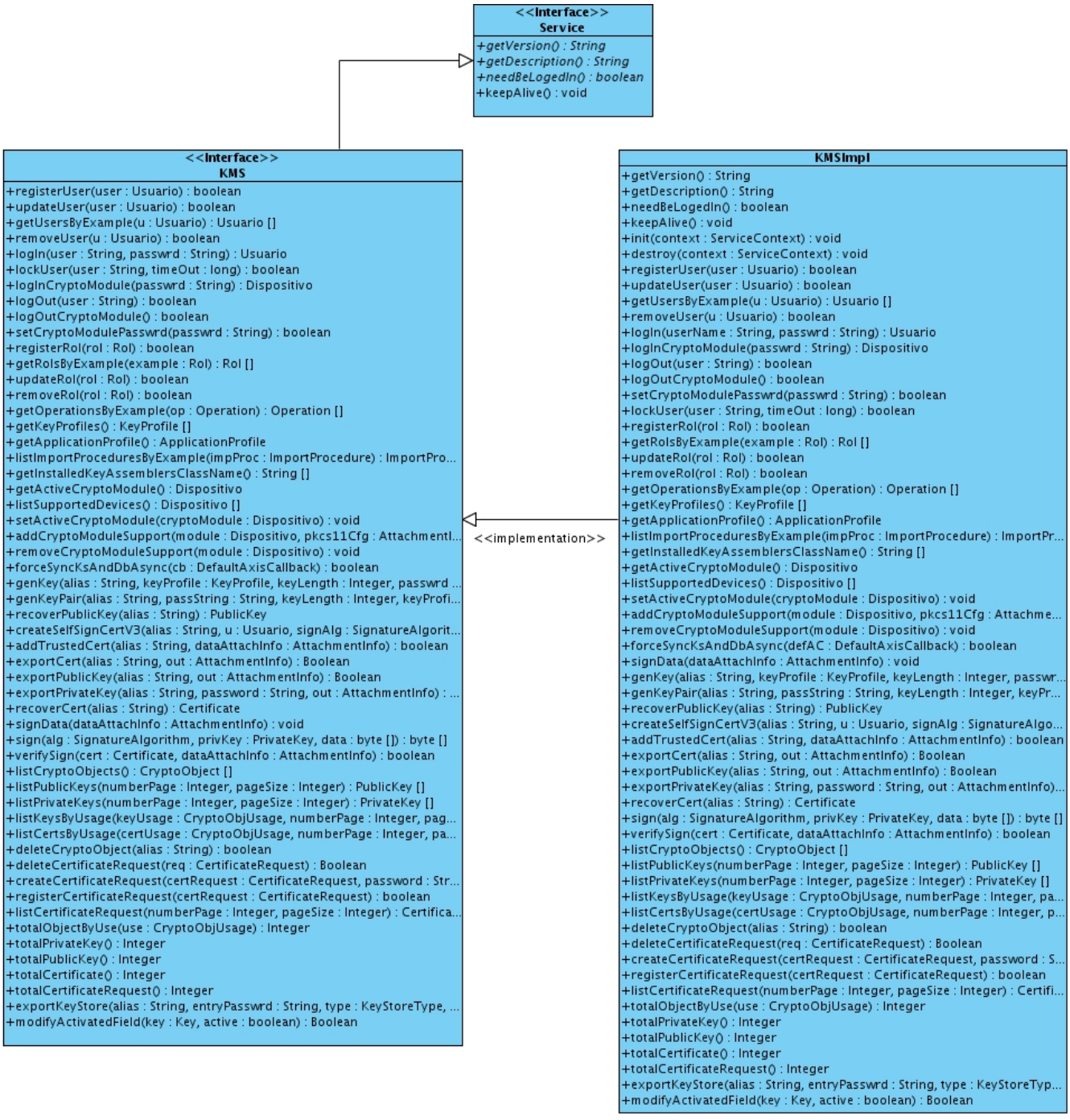


Figura 3.9 Segmento 2 del diagrama de clase “Capa de presentación”.

3.7.2 Descripción de las clases

La descripción de las clases de la capa presentación se encuentra en el **anexo 7**.

3.8 Diagramas de secuencias

A continuación se muestran diagramas de secuencia de algunas de las funcionalidades críticas, los cuales permiten comprender la interacción entre los objetos que integran la solución.

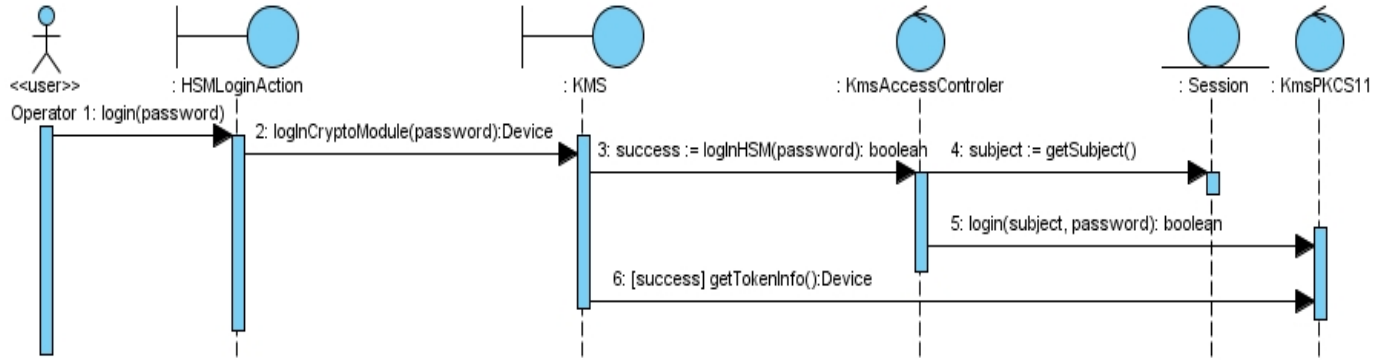


Figura 3.10 Diagrama de secuencia “Autenticación en un HSM”.

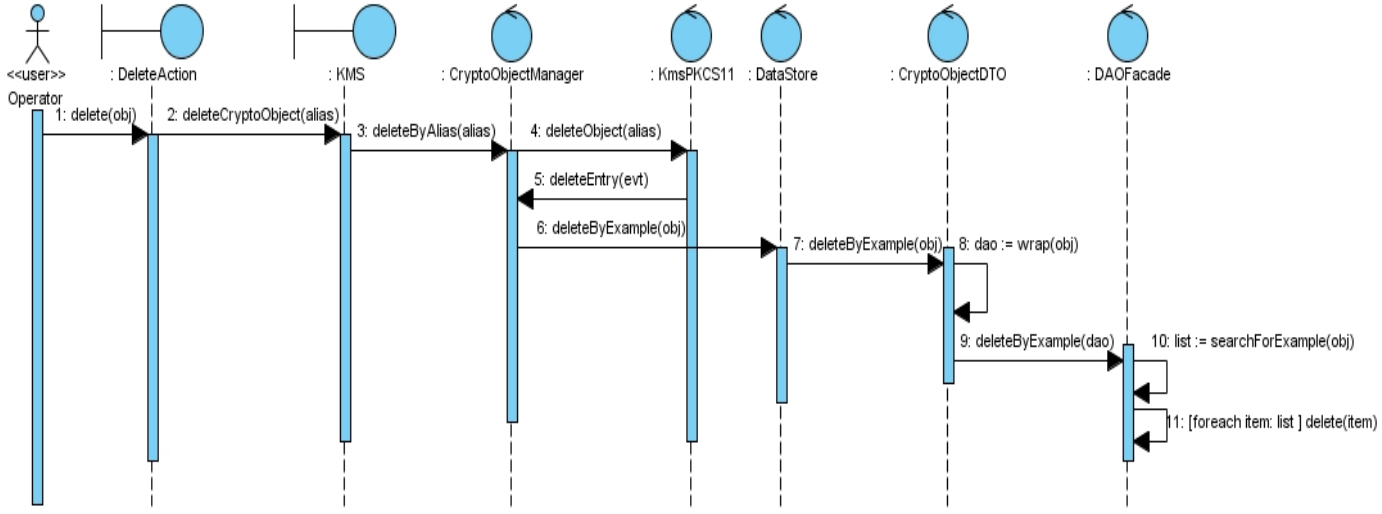


Figura 3.11 Diagrama de secuencia “Eliminar objetos criptográficos”.

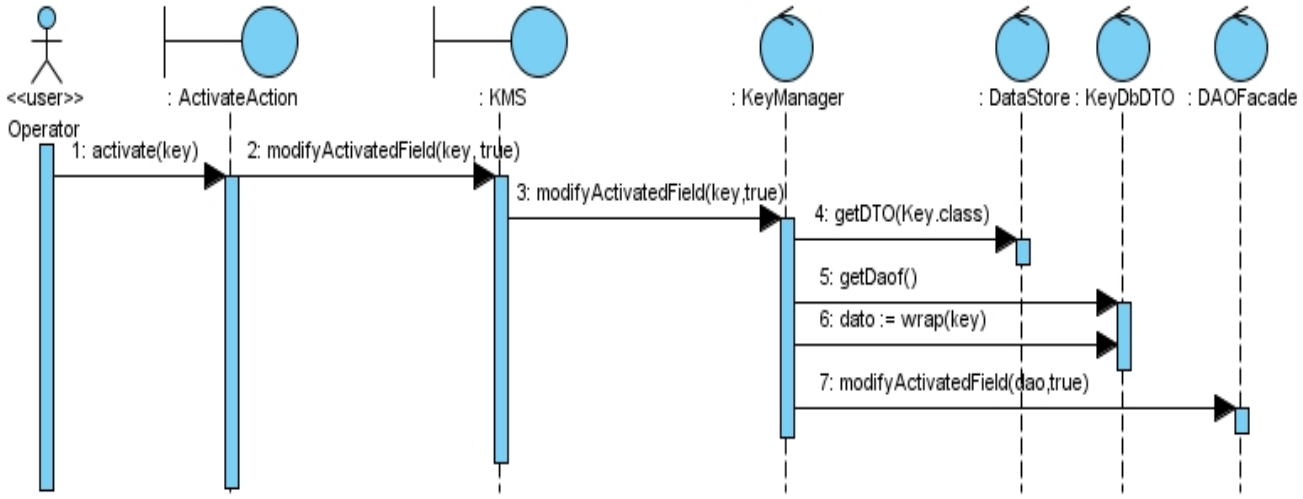


Figura 3.12 Diagrama de secuencia “Activar llave de firma”.

3.9 Conclusiones

Con la realización de este capítulo se mostró cómo puede ser construido el sistema, sirviendo esto como una abstracción del modelo de implementación y el código fuente. A través de los diagramas de clases y sus descripciones se logró visualizar, especificar y documentar cada capa de la solución propuesta. Y mediante los diagramas de secuencia se logró modelar la interacción entre los objetos del sistema.

CAPÍTULO 4. Implementación y Prueba del sistema

4.1 Introducción

En este capítulo se abordan aspectos importantes de la implementación y prueba del sistema. Primeramente se describe el estilo de codificación utilizado. Luego se muestra la distribución física del sistema entre los diferentes nodos de cómputo a través del diagrama de despliegue. Se presenta el diagrama de componente que permite comprender la estructura del sistema y se muestran los resultados de los casos de prueba realizados.

4.2 Estándar de codificación

Seguir un estándar en la codificación es importante, ya que mejora la lectura del software, permitiendo entender el código más rápidamente. A continuación se describe el estándar utilizado en la implementación de la solución.

Reglas de codificación

- El código fuente debe ser escrito en inglés.
- Se debe evitar las líneas de más de 80 caracteres, ya que no son bien manejadas por muchas herramientas.
- Cada línea debe contener cuando más una sentencia.
- Se debe seguir las convenciones de nombre mostradas en la **tabla 4.1**.

Tabla 4.1 Convenciones de nombre.

Tipos de identificadores	Reglas de nombre	Ejemplos
Paquetes.	El nombre del paquete debe empezar obligatoriamente con el prefijo “org.kms” seguido preferentemente por el nombre del módulo en minúscula.	org.kms.data_access.
Clases o	Los nombres de las clases o interfaces deben tener la	ApplicationProfile.

Interfaces.	primera letra de cada palabra en mayúscula.	
Métodos.	Los nombres de los métodos deben reflejar la acción a realizar y cuando sean compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras en mayúscula.	signData.
Variables.	Todas las variables empezarán con minúscula y la primera letra de las siguientes palabras en mayúscula.	logger.
Constantes.	Cada carácter que pertenezca al nombre de la constante se escribirá en mayúscula y en caso de ser un nombre compuesto, cada palabra se separará por un guión bajo "_".	MALFORMED_USER_MSG.

4.3 Diagrama de componentes

El diagrama de componentes representa cómo el sistema de software es dividido en componentes y muestra las dependencias entre estos. Los componentes se usan para agrupar elementos en estructuras lógicas. En la **figura 4.1** se muestra el diagrama de componentes de la aplicación y en la **tabla 4.2** se define el propósito de cada componente.

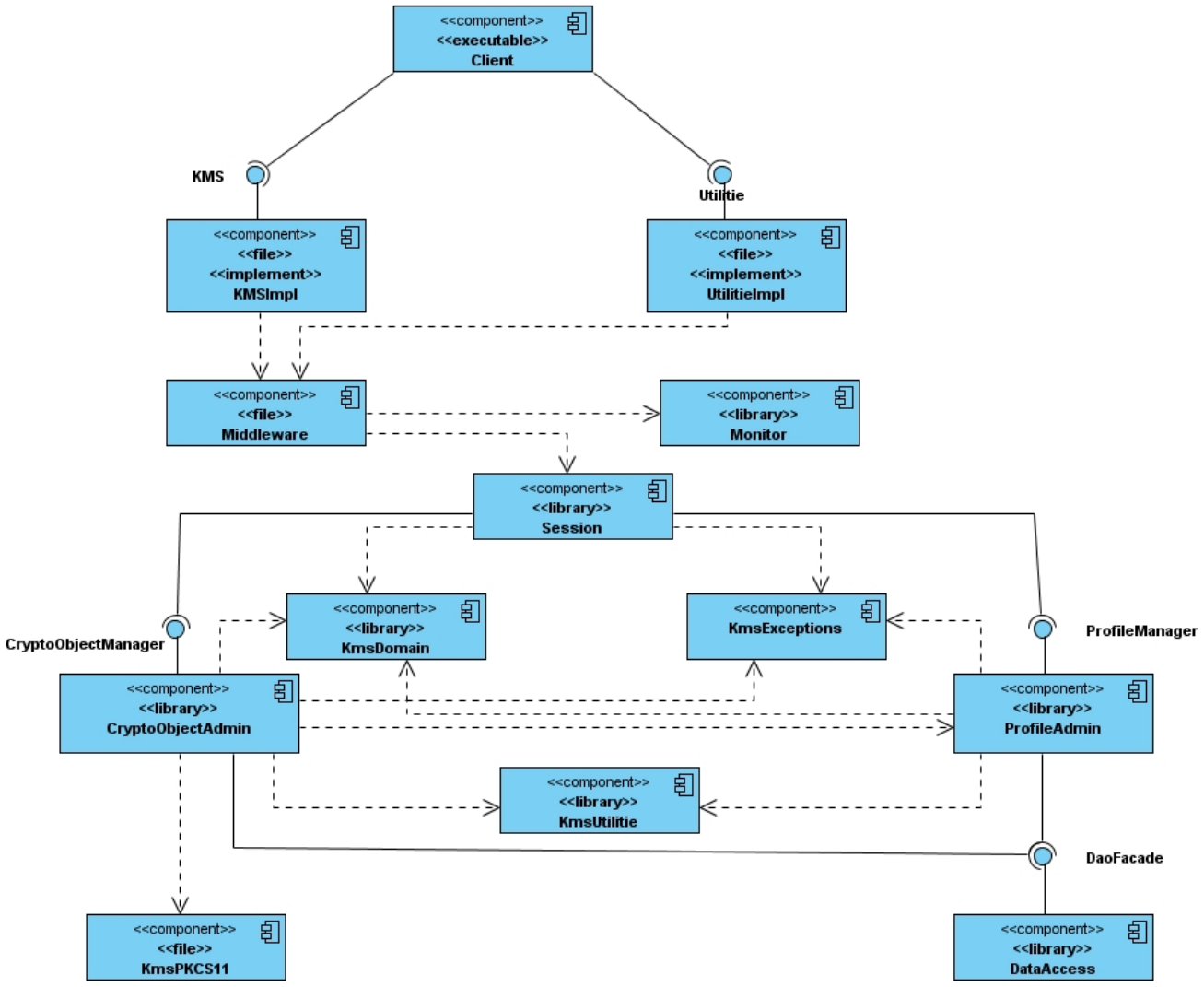


Figura 4.1 Diagrama de componentes.

Tabla 4.2 Descripción de componentes.

Componente	Propósito
Client.	Componente que contiene el ejecutable de la aplicación cliente.
KMSImpl.	Componente que contiene la implementación del servicio KMS.

UtilitieImpl.	Componente que contiene la implementación del servicio Utilitie.
Middleware.	Componente que intercepta las peticiones que se le hacen al servidor, prepara las respuestas que viajan por la red y notifica a los interesados el evento ocurrido.
Monitor.	Componente que se ocupa de cómo notificar los eventos que ocurren en el Middleware.
Session.	Componente que se ocupa del control de las sesiones y permisos asociados a cada sesión.
CryptoObjectAdmin.	Componente que contiene la implementación de las funcionalidades que se realizan sobre los objetos criptográficos.
ProfileAdmin.	Componente que contiene la implementación de las funcionalidades de gestión de los perfiles del sistema.
KmsDomain.	Componente que contiene los objetos de dominio del sistema.
KmsExceptions.	Componente que contiene las excepciones que son lanzadas por el sistema.
KmsUtilitie.	Componente que contiene la implementación de funcionalidades que son útiles en diversos niveles del sistema.
KmsPKCS11.	Componente que permite interactuar con el dispositivo criptográfico.
DataAccess.	Componente que contiene la implementación de la capa de acceso a datos.

4.4 Diagrama de despliegue

Un diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo similar. (27)

En el diagrama que se muestra en la **figura 4.2**, el nodo "PC Cliente de escritorio" representa la computadora en la cual estará desplegada la aplicación cliente (aplicación de interfaz gráfica). El nodo “PC cliente de personalización” representa la computadora donde estará desplegada la aplicación cliente de personalización de los documentos de identificación electrónica. Estas aplicaciones consumen los servicios que se exponen en el servidor (nodo "Servidor-KMS") utilizando el protocolo HTTPS. Este protocolo utiliza un cifrado basado en TLS[4] para crear un canal más apropiado para el tráfico de la información que se maneja. En el nodo "Servidor-KMS" se encuentran los componentes que implementan las funcionalidades del sistema. Para realizar dichas funcionalidades el servidor KMS debe comunicarse con un dispositivo criptográfico conectado a él por el puerto USB, PCI o a través de la red. Además para la persistencia de los datos, el servidor KMS se comunica con un servidor de base de datos utilizando la API JDBC que permite la ejecución de operaciones sobre base de datos desde el lenguaje de programación Java.

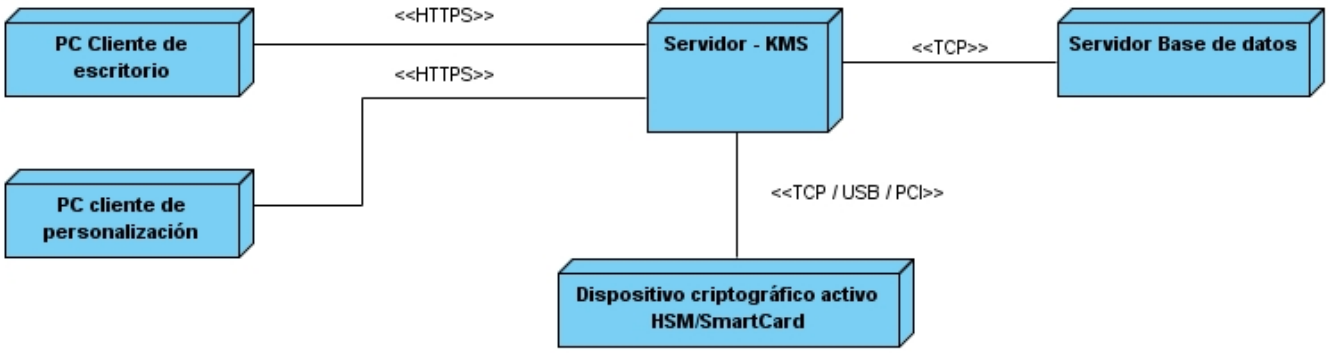


Figura 4.2 Diagrama de despliegue.

4.5 Prueba

Con el fin de detectar errores y verificar la calidad del software se realizaron las pruebas de unidad y las pruebas de caja negra. Las pruebas de unidad permitieron verificar el funcionamiento de cada módulo por separado y las pruebas de caja negra permitieron estudiar el software desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno.

[4] Transport Layer Security (TLS), sucesor del protocol criptográfico Secure Sockets Layer (SSL).

4.5.1 Prueba de unidad

Para la realización de las pruebas de unidad se utilizó la herramienta JUnit la cual facilita, dado una clase y la definición de los métodos a probar, la verificación del comportamiento de las funcionalidades. A continuación se describen los casos de prueba creados:

KmsImplTest.

La clase de prueba KmsImplTest cuenta con dos métodos:

- *testLogIn ()* que verifica el comportamiento de la funcionalidad "Autenticación en el KMS".
- *testLogInCryptoModule ()* que verifica el comportamiento de la funcionalidad "Autenticación en un HSM".

En la **figura 4.3** se muestra el resultado de la prueba, el ícono de color verde al lado del nombre de cada método significa la correcta realización de este, de no ser así se mostraría un ícono de color rojo. La barra de progreso en verde significa que la prueba se completó con éxito.

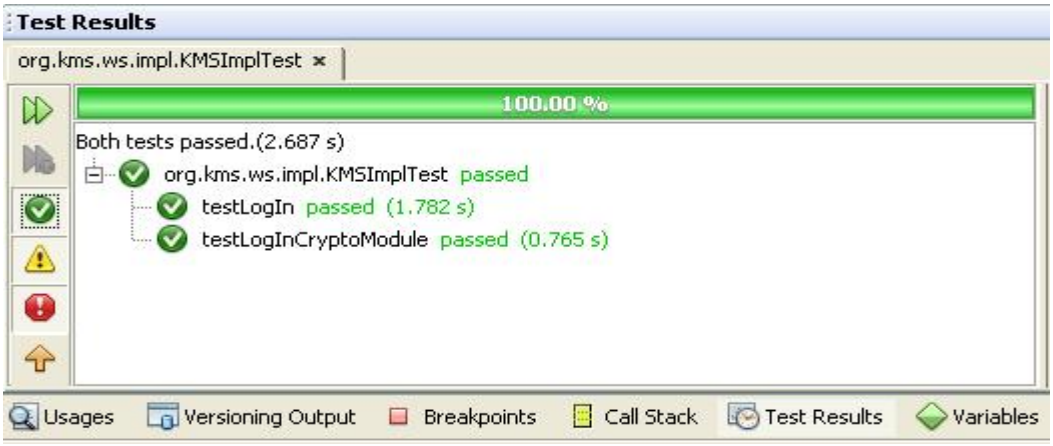


Figura 4.3 Realización del caso de prueba “KmsImplTest”.

CryptoObjectManagerTest.

La clase de prueba CryptoObjectManagerTest contiene tres métodos:

- *testSyncKsAndDB ()* que verifica el comportamiento de la funcionalidad "Sincronizar HSM con KMS".

- *testDeleteCryptoObject ()* que verifica el comportamiento del método que se utiliza para la realización de la funcionalidad "Eliminar solicitud de certificado".
- *testDeleteByAlias ()* que verifica el comportamiento del método que se utiliza para la realización de la funcionalidad "Eliminar objetos criptográficos". (Ver **figura 4.4**)

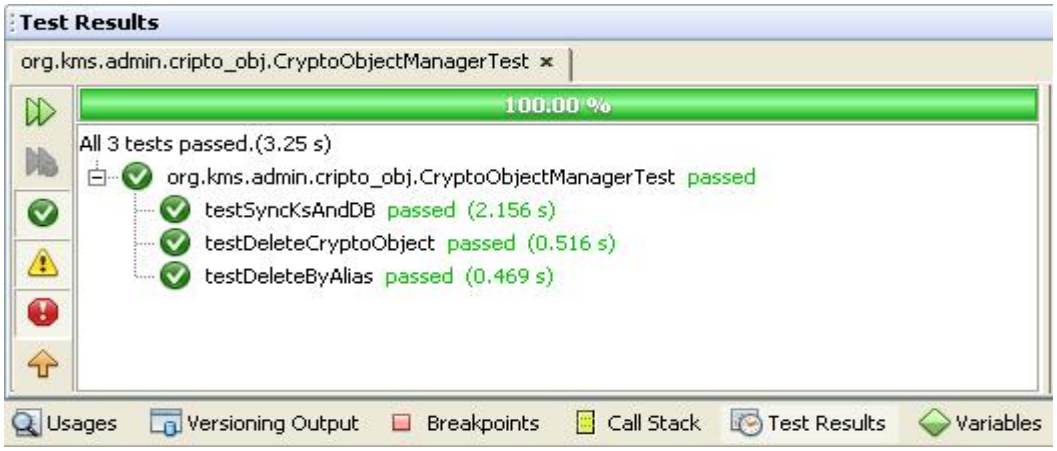


Figura 4.4 Realización del caso de prueba “CryptoObjectManagerTest”.

KeyManagerTest.

La clase de prueba KeyManagerTest cuenta con cinco métodos:

- *testGenSecretKey ()* que verifica el comportamiento de la funcionalidad "Generar llave simétrica".
- *testGenKeyPairAndCert ()* que verifica el comportamiento de la funcionalidad "Generar par de llaves y certificado auto-firmado".
- *testImportKey ()* que verifica el comportamiento de la funcionalidad "Importar llave".
- *testExportPublicKey ()* que verifica el comportamiento de la funcionalidad "Exportar llave pública".
- *testModifyActivatedField ()* que verifica el comportamiento del método que se utiliza para la realización de la funcionalidad "Activar llave de firma" y "Desactivar llave de firma". (Ver **figura 4.5**)

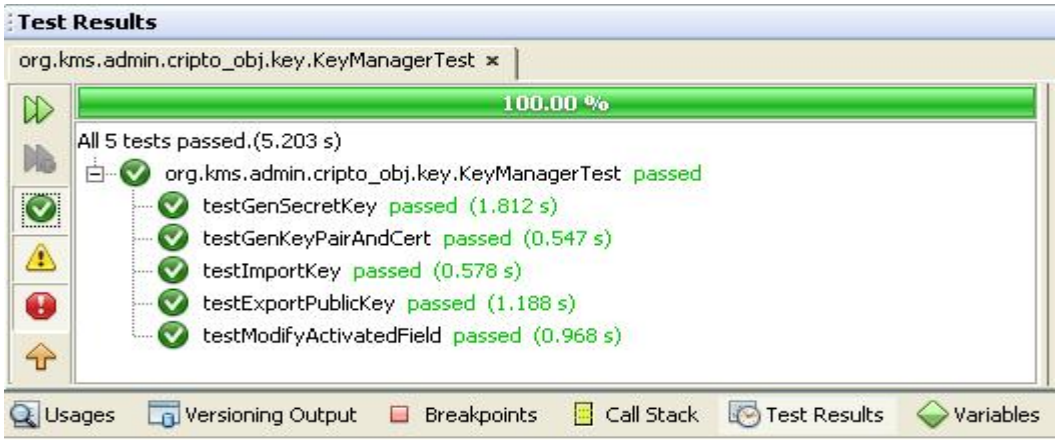


Figura 4.5 Realización del caso de prueba “KeyManagerTest”.

CertManagerTest.

La clase de prueba CertManagerTest cuenta con tres métodos:

- *testCreateCertificateRequest ()* que verifica el comportamiento del método que se utiliza para la realización de la funcionalidad "Exportar solicitud de certificado".
- *testExportCert ()* que verifica el comportamiento de la funcionalidad "Exportar certificado".
- *testRegisterCertificateRequest ()* que verifica el comportamiento de la funcionalidad "Generar solicitud de certificado". (Ver **figura 4.6**)

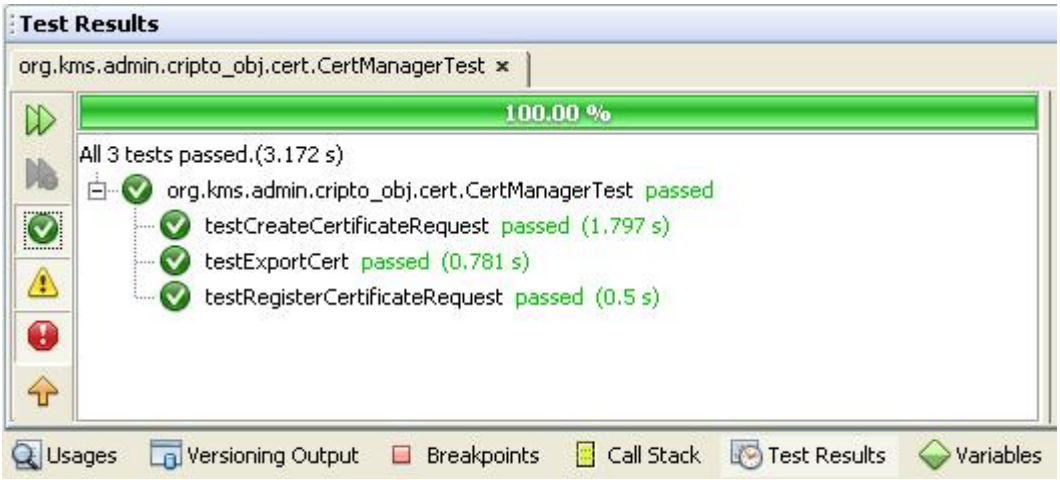


Figura 4.6 Realización del caso de prueba “CertManagerTest”.

4.5.2 Prueba de caja negra

Las pruebas de caja negra se realizaron a las funcionalidades que requieren que el usuario introduzca algún valor el cual puede estar o no erróneo. A continuación se muestra el registro del caso de prueba “Generar llave simétrica”.

Tabla 4.3 Descripción de las variables del caso de prueba “Generar llave simétrica”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Alias.	Campo de texto.	No.	Debe introducir un texto.
[2]	Algoritmo.	Campo de selección.	No.	Debe seleccionar un valor.
[3]	Tamaño de llave.	Campo de selección.	No.	Debe seleccionar un valor.
[4]	Contraseña.	Campo de texto.	Si.	Debe introducir un texto.
[5]	Re-contraseña.	Campo de texto.	Si.	Debe introducir un texto.
[6]	Cantidad de fragmento.	Campo de selección.	No.	Debe seleccionar un valor.
[7]	Uso.	Campo de selección.	No.	Debe seleccionar un valor.
[8]	Procedimiento de importación.	Campo de selección.	No.	Debe seleccionar un valor.

En la **tabla 4.4** se muestran las respuestas del sistema al probar la funcionalidad con las diversas combinaciones de valores posibles de las variables.

Tabla 4.4 Caso de prueba "Generar llave simétrica".

Flujo	Var 1. Alias	Var 2. Algoritmo	Var 3. Tamaño de llave	Var 4. Contraseña	Var 5. Re-contraseña	Var 6. Cantidad de fragmento	Var 7. Uso	Var 8. Procedimiento de importación	Respuesta del sistema	Resultado de la Prueba
Generar llave simétrica.	V[5]	V	V	V	V	V	V	V	El sistema crea la llave simétrica satisfactoriamente.	Satisfactorio.
	I[6]	V	V	V	V	V	V	V	El sistema muestra el mensaje: "Alias ya asignado, debe escoger otro alias".	Satisfactorio.
	V	V	V	NA[7]	NA	V	V	V	El sistema crea la llave simétrica satisfactoriamente.	Satisfactorio.
	I	V	V	NA	NA	V	V	V	El sistema muestra el mensaje: "Alias ya asignado, debe escoger otro alias".	Satisfactorio.

[5] **V** Significa que se introdujo un valor válido.
[6] **I** Significa que se introdujo un valor inválido.
[7] **NA (No Aplica)** Significa que para la prueba, a la variable no se le dio valor.

4.6 Métricas

En este apartado se muestran los indicadores utilizados para medir algunas de las características del sistema.

Tabla 4.5 Indicadores.

Dimensión	Indicador	Valor	Observaciones
Factibilidad.	Tiempo de desarrollo.	8 meses.	
	Cantidad de desarrolladores.	2 personas.	
	Cantidad de estándares utilizados.	4 estándares.	Se utilizó PKCS#11, PKCS#10 y PKCS#8[8]. Y además se utilizó el estándar X509.
Funcionalidad.	Funcionalidad.	Media.	La lista de funcionalidades se encuentra en la **tabla 2.1**.
Seguridad.	Controlabilidad de acceso.	Alto.	El sistema cuenta con dos niveles de seguridad: autenticación SSL y Servicio de autenticación y autorización de java (JAAS).
Rendimiento.	Cantidad de clientes soportados concurrentemente.	20 clientes.	Se puso a prueba el servidor KMS con 20 clientes y funcionó correctamente.

Analizando los valores de la **tabla 4.5** se puede deducir que es factible el desarrollo de un sistema para el manejo de llaves y certificados digitales cumpliendo con los estándares PKCS fundamentales en un tiempo relativamente corto. También se llega a la conclusión de que el sistema resultante de la

[8] PKCS#8. Estándar sobre la sintaxis de la información de la llave privada.

investigación es un software que contiene las funcionalidades básicas de un KMS y cuenta con un alto nivel de seguridad en cuanto al control de acceso a la aplicación.

4.7 Conclusiones

En este capítulo se definió el estilo de codificación utilizado en la implementación del sistema, lo que permitió mantener una uniformidad en la codificación y mayor claridad a la hora de leer o agregar código. Se realizó el diagrama de despliegue, que deja clara la distribución física del sistema entre los diferentes nodos de cómputo. Se confeccionó el diagrama de componente a través del cual se observó las dependencias entre los elementos físicos que componen al sistema. Y se ejecutaron las pruebas que permitieron verificar el buen funcionamiento del sistema.

CONCLUSIONES GENERALES

A partir de la necesidad del CISED de contar con un KMS que pudiera integrar a las soluciones de emisión de documentos de identificación electrónica desarrolladas allí, se diseñó e implementó un KMS que es capaz de gestionar de forma segura las llaves y los certificados digitales que se utilizan en soluciones para la emisión de documentos de identificación electrónicos y en otras soluciones que hagan uso de una infraestructura de llave pública. Para lo cual:

- Se realizó el estudio de 3 soluciones KMS usados a nivel mundial, lo que permitió identificar las principales características, funcionalidades y estándares presentes en estos tipos de software.
- Se analizaron y escogieron las herramientas y tecnologías a utilizar en el desarrollo del sistema de acuerdo a las características deseadas en este. Lo que permitió crear una solución multiplataforma y con el menor costo de construcción posible.
- Se aplicaron los estándares:
    - PKCS#11, lo que permite que el KMS se pueda comunicar con cualquier dispositivo criptográfico que implemente este estándar.
    - PKCS#10, lo que permite que las solicitudes de certificación generadas por el KMS sean entendidas por cualquier CA.
    - PKCS#8, lo que permite que las llaves privadas generadas en el KMS tengan la estructura correcta.
    - X509, lo que permite que los certificados generados en el KMS tengan la estructura correcta.
- Se realizaron pruebas unitarias y pruebas de caja negra que permitieron verificar el correcto funcionamiento del KMS.

RECOMENDACIONES

Con el fin de seguir mejorando el producto se recomienda:

- Crear un sistema que gestione los posibles perfiles de la aplicación KMS.
- Adicionarle a la funcionalidad "Generar par de llaves y certificado auto-firmado" la posibilidad de crear certificados con extensiones.
- Adicionarle a la funcionalidad "Generar solicitud de certificado (PKCS#10)" la posibilidad de crear una solicitud con atributos PKCS#9.

GLOSARIO DE TÉRMINOS

A

API: conjunto de métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Arquitectura: indica la estructura, funcionamiento e interacción entre las partes del software.

B

Backup: en este documento el término se utiliza para referirse al sistema que permite recuperarse de una catástrofe informática.

Base de datos: conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

C

CA: autoridad de certificación. Entidad responsable de emitir y revocar certificados digitales.

Cédula de Identidad: documento emitido por una autoridad administrativa competente para permitir la identificación personal de los ciudadanos.

Certificado digital: documento digital mediante el cual un tercero confiable garantiza la vinculación entre la identidad de un sujeto o entidad y una llave pública.

Clúster: conglomerado de sistemas.

Criteria: estructura usada por Hibernate para crear un árbol de criterios de búsquedas.

Criterion: tipo de dato que utiliza Hibernate para representar un criterio de búsqueda.

D

Dispositivo criptográfico: dispositivo en el que se ejecutan las funciones criptográficas.

F

Framework: módulos de software concretos con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

H

Hardware: parte tangible de un sistema informático.

K

KMS: software que interactúa con un dispositivo criptográfico con el fin de gestionar llaves y certificados digitales.

L

GLOSARIO DE TÉRMINOS

LDAP: protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

M

Métrica: medida o conjunto de medidas destinadas a conocer o estimar el tamaño u otra característica de un software.

P

PCI: puerto estándar para conectar dispositivos periféricos directamente a la placa madre de la computadora.

Plug-in: aplicación que se relaciona con otra para aportarle una funcionalidad nueva.

PKI (infraestructura de clave pública): es una combinación de hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital o el no repudio de transacciones electrónicas.

R

REST: es una técnica de arquitectura software para sistemas distribuidos.

S

SOAP: protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

SQL: lenguaje declarativo de acceso a bases de datos.

SSL: protocolo criptográfico que proporciona comunicaciones seguras por una red.

T

TLS: protocolo criptográfico sucesor de SSL.

Tupla: fila o registro de la base de datos.

W

WSDL: formato XML que se utiliza para describir servicios Web.

X

XML: es un metalenguaje extensible de etiquetas.

REFERENCIAS BIBLIOGRÁFICAS

1. **InformáticaHoy.** InformáticaHoy. [Online] 2011. [Cited: Septiembre 15, 2011.] http://www.informatica-hoy.com.ar/software-seguridad-virus-antivirus/Criptografia-Seguridad-informatica.php.
2. **Soluciones Web.** Soluciones Web. Soluciones Web. [Online] [Cited: Marzo 1, 2012.] http://solucionesweb.fullblog.com.ar/certificados-digitales.html.
3. **Real Academia Española.** Real Academia Española. Real Academia Española. [Online] [Cited: Octubre 2, 2011.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=criptograf%C3%ADa.
4. **López, Manuel José Lucena.** Criptografía y seguridad en computadores.1999.
5. **www.wikilearning.com.** www.wikilearning.com. [Online] [Cited: Mayo 9, 2012.] http://www.wikilearning.com/curso_gratis/curso_de_criptografia_basica_para_principian.
6. **BIFI.** BIFI: Instituto de Biocomputación y Física de Sistemas Complejos. BIFI: Instituto de Biocomputación y Física de Sistemas Complejos. [Online] [Cited: Mayo 7, 2012.] http://oldwww.bifi.es/research/complexm_fundphysics/ssue/ssue_es.php.
7. **www.galeon.com.** www.galeon.com. [Online] [Cited: Mayo 7, 2012.] http://www.galeon.com/analisisdealgoritmos/enlaces628082.html.
8. **Christian Cachin, Nishanth Chandran.** A Secure Cryptographic Token Interface: Complutense, 2011.
9. **idsegura.** www.idsegura.com. www.idsegura.com. [Online] [Cited: Mayo 7, 2012.] http://www.idsegura.com/seguridad_web/tokens.php.
10. **ARX.** ARX. ARX. [Online] [Cited: Octubre 2, 2011.] http://www.arx.com.
11. **THALES.** THALES-Esecurity. [Online] [Cited: Octubre 2, 2011.] http://www.thales-esecurity.com.
12. **SafeNet.** SafeNet. SafeNet. [Online] [Cited: Octubre 2, 2011.] http://www.safenet-inc.com.
13. **Futurex.** Futurex. Futurex. [Online] [Cited: Octubre 2, 2011.] http://www.futurex.com.
14. **Oracle.** Oracle. Oracle. [Online] [Cited: Octubre 2, 2011.] http://www.oracle.com.
15. **Biosca, Josep M. Miret.** Criptografía con curvas elípticas: RBA, 2011.
16. **Laboratorio RSA.** Laboratorio RSA. [Online] [Cited: Mayo 7, 2012.]. http://www.rsa.com.
17. **Laboratoires, RSA.** Certification Request Syntax Standard: Complutense, 2000.
18. **ITU.** ITU. ITU. [Online] [Cited: Marzo 2, 2012.] http://www.itu.int/rec/T-REC-X.509/en.

REFERENCIAS BIBLIOGRÁFICAS

19. **Larman, Craig.** UML y Patrones: Pearson, 2003.
20. **REGUEIRO.** Componente de acceso a datos para soluciones de emisión de documentos de identificación 2009: Seix-Barral,2009.
21. **Gavin King, Christian Bauer, Max Rydahl Andersen, Bernard, Steve Ebersole, Hardy Ferentschik.** Documentación de referencia de Hibernate: Club Universitario, 2011.
22. **www.hibernate.org.** www.hibernate.org. [Online] [Cited: Mayo 7, 2012.] http://www.hibernate.org/license.html.
23. **Apache.** Axis2. [Online] [Cited: Mayo 7, 2012.] http://axis.apache.org/axis2.
24. **Böck, Heiko.** The definitive guide to NetBeans Platform: Apress,2009.
25. **PostgreSQL.** PostgreSQL 9.1.2 Documentation. [Online] [Cited: Mayo 7, 2012.]. http://www.postgresql.org/.
26. **ingenieriasoftware2011.** ingenieriasoftware2011. [Online] [Cited: Mayo 24, 2012.] http://ingenieriasoftware2011.wordpress.com/2011/07/08/herramientas-case-ejemplos-par.
27. **www.taringa.net.** www.taringa.net. www.taringa.net. [Online] [Cited: Marzo 2, 2012.] http://www.taringa.net/posts/apuntes-y-monografias/10119154/Proceso-Unificado-de-Desarrollo-de-Software.html.
28. **www.aminegocio.com.** www.aminegocio.com. [Online] [Cited: Mayo 7, 2012.] http://www.aminegocio.com/amitiendavirtual/CAPACITACION/index.htm.
29. **blessedsoft.org.** blessedsoft.org. [Online] [Cited: Mayo 7, 2012.] http://blessedsoft.org/wiki/Wiki.jsp?page=XPOverview.
30. **www.step-10.com.** www.step-10.com. [Online] [Cited: Mayo 7, 2012.] http://www.step-10.com/SoftwareProcess/FeatureDrivenDevelopment/images/fdd.jpg.

BIBLIOGRAFÍA

- Qué es Java? [Online] [Cited: Mayo 7, 2012.]. http://www.iec.csic.es/criptonomicon/java/quesjava.html.
- **Beutelspacher, Albrecht.** Criptology. s.l. : The Mathematical Association of America, 1994.
- **Berner, Schmidt.** KMS-Key Management System User Manual: SM. 2009.
- **Departamento CERES.** CERES. [Online] 04 10, 2007. [Cited: Octubre 3, 2011.] http://www.cert.fnmt.es.
- Las características de java. Las características de java. [Online] [Cited: Mayo 7, 2012.]. http://zarza.usal.es.
- Lenguaje Unificado de Modelado Unified Modeling Language: Pearson, 2011.
- **Sun.** Oracle Key Manager Overwiev: LIST,2010.
- www.cren.net. [Online] [Cited: Noviembre 30, 2011.] http://www.cren.net/crenca/onepagers/hsm2.html.
- **Sun Microsystems.** Crypto Key Management Station: LIST, 2007.
- **Rousselz Blog.** Rousselz Blog. [Online] [Cited: Octubre 5, 2011.] http://rousselz.wordpress.com/2010/08/08/metodologia-fdd-feature-driven-development-desarrollo-basado-en-funciones/.

ANEXOS

Anexo 1: Análisis de metodologías de desarrollo de software

Tres alternativas metodológicas

En este apartado se realiza un análisis entre una metodología tradicional y dos ágiles, en el primer caso se tendrá en cuenta el Proceso Unificado de Rational, (RUP por sus siglas en inglés), en el segundo caso se analizarán Programación Extrema, *eXtreme Programming* (XP), y Desarrollo Guiado por la Funcionalidad, *Feature Driven Development* (FDD).

RUP

RUP (Rational Unified Process) es uno de los procesos más riguroso y completo de los existentes hoy día, pensado para cualquier proyecto, siempre y cuando se configure. Consta de 4 fases: Inicio, Elaboración, Construcción y Transición. Está distribuido por 9 flujos de trabajo, 6 de ellos considerados ingenieriles: Modelado de negocio, Requisitos, Análisis y Diseño, Implementación, Prueba y Despliegue y los otros tres son considerados flujos de apoyo: Gestión de proyecto, Gestión de Configuración y Cambio, Despliegue. Cada fase es realizada por iteraciones, y en cada iteración se desarrollan actividades de los flujos de trabajo, algunos más que otros, dependiendo del avance que tenga el software. (Ver **figura 5.1**)

ANEXOS

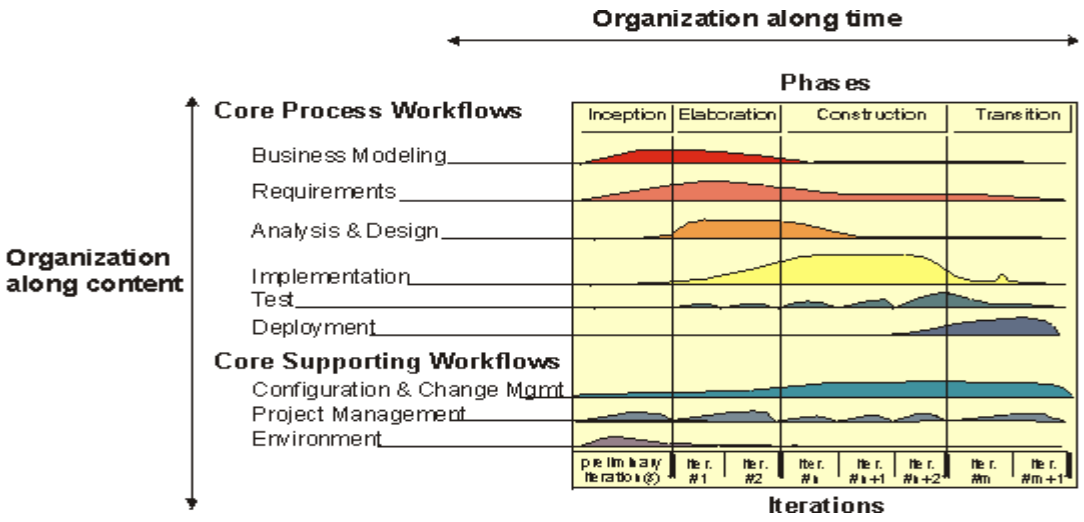


Figura 5.1 Vista de RUP. (28)

Es un proceso dirigido por casos de uso, ya que desde el modelado del negocio se captan en forma de casos de uso lo que ocurre en la organización, de cada uno se realizan diagramas de actividades para seleccionar las actividades a ser automatizadas, estas son la entrada para determinar las funcionalidades del sistema, las cuales se expresan mediante casos de uso del sistema, luego se realiza cada caso de uso mediante el análisis y diseño, para su posterior implementación y prueba.

Es centrado en la arquitectura, la cual captura los elementos más importantes del sistema, y los refleja mediante vistas arquitectónicas.

Es iterativo e incremental, ya que en cada iteración se desarrollan actividades de todos los flujos de trabajo, y cada resultado es un incremento para el software.

XP

XP (eXtreme Programming) es una metodología que intenta reducir la complejidad del software por medio de un trabajo orientado al objetivo, basado en las relaciones interpersonales y la velocidad con la que trabajan. Intenta minimizar los fallos en el proceso, a través de la colaboración permanente de algún representante del cliente, el cual debe tener conocimientos profundos, para responder de una forma correcta y en un tiempo breve cualquier pregunta realizada por algún miembro del equipo de desarrollo.

Define historias de usuario para describir las funciones del sistema, las cuales son escritas por el cliente. Se crea un plan de entrega entre clientes y equipo de desarrollo, tomando como base las historias de usuario y la arquitectura. En cada entrega se discuten entre las partes los objetivos y se definen las iteraciones necesarias para cumplir los objetivos. Cada iteración tiene como resultado un programa que se entrega al cliente para que lo analice, en base a su respuesta se planifican las próximas iteraciones, y si no está de acuerdo se ajusta el plan de entrega hasta que se satisfagan sus necesidades.

Los escenarios de prueba, describen la forma de comprobar la realización de las historias de usuario, y tanto uno como el otro son la base para el trabajo del desarrollador. (Ver **figura 5.2**)

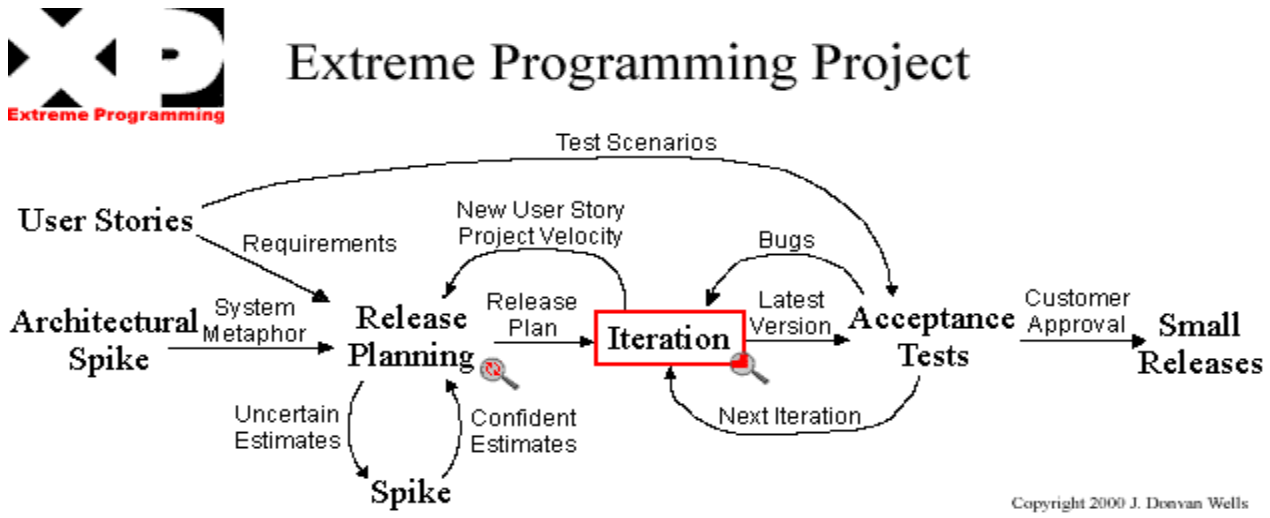


Figura 5.2 Vista general de XP. (29)

La codificación en XP es desarrollada por parejas (dos programadores en una sola PC), por lo que se espera un software con rigurosa calidad. El código de cada pareja es socializado entre los miembros del equipo, de forma tal que puede ser modificado por cualquier integrante, y las parejas son rotativas, de forma tal que todos tengan conocimiento del software.

FDD

FDD (Feature Driven Development) es un proceso iterativo, con iteraciones cortas (2 semanas), las cuales producen un software funcional que los clientes pueden ver y monitorizar. Ha sido pensado

para proyectos con un tiempo de desarrollo menor de dos años. Las iteraciones se deciden teniendo en cuenta las funcionalidades, que representan fragmentos del software con significado para el cliente.

Consta de 5 fases: Desarrollo de un modelo global, Construcción de la lista de funcionalidades, Planificación por funcionalidad, Diseño por funcionalidad e Implementación por funcionalidad. (Ver **figura 5.3**)

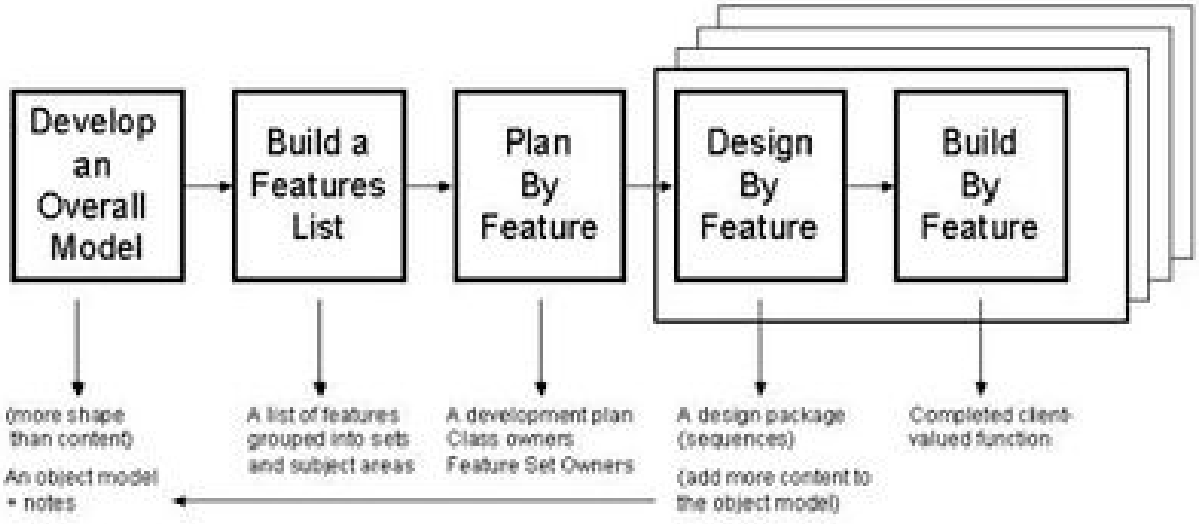


Figura 5.3 Vista general de FDD. (30)

El trabajo tanto de modelado como de desarrollo se realiza en grupo. Las funcionalidades a implementar en una entrega son repartidas entre los distintos subgrupos del equipo, las clases de software tienen propietario, es decir, sólo el creador puede modificarlas. Es por ello que el equipo que implementa cierta funcionalidad tiene que tener todos los propietarios de las clases implicadas. Implementar una funcionalidad lleva implícito la preparación y ejecución de pruebas, así como revisión del código e integración de las partes que componen el software.

Selección de la metodología de desarrollo de software

Teniendo en cuenta el tiempo de desarrollo del sistema, así como los recursos, y los cambios constantes en las funcionalidades, se descarta la idea de usar RUP como metodología pues esta,

aunque es configurable, no es factible para dicho desarrollo, ya que no es el control del proceso el que se persigue, ni la planificación exhaustiva.

Por tanto, entre XP y FDD ambos procesos regidos por el manifiesto ágil, se decide usar FDD ya que, genera modelos y documentación aceptable, no siendo así en XP, donde la documentación es pobre. Además FDD es a medio camino entre el desarrollo y la organización, no siendo así en XP donde lo importante es el desarrollo. FDD no define explícitamente la obtención de requisitos, sino el proceder a partir de que se han capturado dichos requisitos de la forma que el equipo de desarrollo haya estimado y propone implementar en grupo y no en pareja con un solo ordenador como propone XP. Todos estos aspectos hicieron decidirse por FDD, ya que permitirá desarrollar y documentar lo necesario en tiempo relativamente corto.

Anexo 2: Modelo de datos

Tabla 5.1 Descripción de la tabla “certifícate_profile”.

Nombre	certificate_profile.	
Descripción	En esta tabla persisten los perfiles de certificado aceptados por el sistema.	
Atributos	**Tipo**	**Descripción**
+id.	numeric (19,0).	Identificador del perfil.
version.	int4.	Versión del estándar X509 que debe seguir el certificado.
#ncl_signature_algorithmid.	int4.	Identificador del algoritmo de firma del certificado.
#ncl_useid.	int4.	Identificador del uso que se le dará al certificado.

Tabla 5.2 Descripción de la tabla “key_profile”.

Nombre	key_profile.	
Descripción	En esta tabla persisten los perfiles de llave aceptados por el sistema.	
Atributos	**Tipo**	**Descripción**
+id.	numeric (19,0).	Identificador del perfil.

count_fragment.	int4.	Cantidad de fragmento en que puede ser dividida la llave.
version.	int4.	Versión del estándar PKCS#8 que debe seguir la llave.
assambler_class_name.	varchar (255).	Nombre de la clase encargada de ensamblar la llave.
#import_procedureid.	int4.	Identificador del procedimiento de importación.
#ncl_useid.	int4.	Identificador del uso que se le dará a la llave.
#ncl_key_typeid.	int4.	Identificador del tipo de llave.
#ncl_key_formatid.	int4.	Identificador del formato de la llave.
#ncl_key_algorithmid.	int4.	Identificador del algoritmo de generación de llave.

Tabla 5.3 Descripción de la tabla “application_profile”.

Nombre	application_profile.	
Descripción	En esta tabla persiste el perfil de aplicación aceptado.	
Atributos	**Tipo**	**Descripción**
+id.	int4.	Identificador del perfil.
name.	varchar (255).	Nombre del perfil.
simultaneuous_connections_allowed.	int4	Cantidad de conexiones a la vez permitidas.
allow_rest.	bool.	Si es true, se permite utilizar la técnica de arquitectura software REST (Transferencia de Estado Representacional).
#login_configurationid.	int4.	Identificador del tipo de configuración

		permitido en la aplicación.

Tabla 5.4 Descripción de la tabla “device”.

Nombre	device.	
Descripción	En esta tabla persistirán datos de los dispositivos criptográficos con los que el KMS ha interactuado.	
Atributos	**Tipo**	**Descripción**
+id.	int4.	Identificador del dispositivo en el KMS.
update.	date.	Última fecha de sincronización.
description.	varchar (255).	Descripción del dispositivo.
enable.	bool.	Si es true, indica que este es el dispositivo que se está utilizando actualmente.

Tabla 5.5 Descripción de la tabla “fragment”.

Nombre	fragment.	
Descripción	En esta tabla persisten los fragmentos de las llaves que aún no se han ensamblado y almacenado en el HSM.	
Atributos	**Tipo**	**Descripción**
+#crypto_objectid.	varchar (255).	Identificador de la llave.
+order.	int4.	Orden de importación.
encoded.	varchar (255).	Fragmento de codificación de la llave.

Anexo 3: Clase y archivo de mapeo de Hibernate

Hibernate mapea de una base de datos relacional a un modelo de objeto a través de archivos XML. Cada tabla de la base de datos (excepto las que surgen de la relación mucho – mucho entre dos

tablas) tiene vinculado un archivo XML que permite transformar cada tupla de esta a una instancia de la clase correspondiente. En esta sección se muestra un ejemplo de archivo de mapeo XML, tabla de la base de datos y su clase correspondiente.

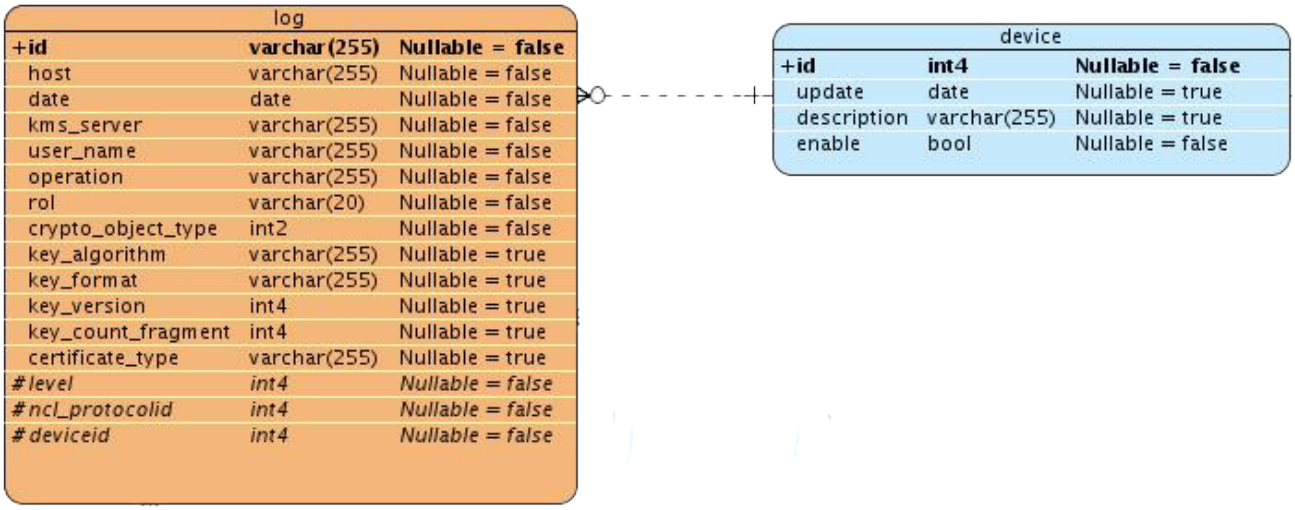


Figura 5.4 Modelo de dato de la tabla “device”.

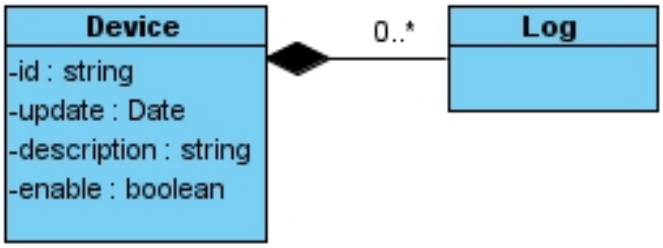


Figura 5.5 Diagrama de clase “device”.

ANEXOS

Device.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Oct 24, 2011 9:28:20 AM by Hibernate Tools 3.2.1.GA -->
<hibernate-mapping>
  <class lazy="true" name="org.kms.data_access.entity.Device" schema="public" table="device">
    <id name="id" type="string" unsaved-value="0">
      <column name="id"/>
      <generator class="assigned"/>
    </id>
    <property name="update" type="date">
      <column name="update"/>
    </property>
    <property name="description" type="string">
      <column name="description"/>
    </property>
    <property name="enable" type="boolean">
      <column name="enable" not-null="true"/>
    </property>
    <set cascade="none" inverse="true" lazy="true" name="logs">
      <key>
        <column name="deviceid" not-null="true"/>
      </key>
      <one-to-many class="org.kms.data_access.entity.Log"/>
    </set>
  </class>
</hibernate-mapping>
```

Figura 5.6 Archivo de mapeo de la clase “device”.

Anexo 4: Descripción de las clases de la capa de lógica de negocio

Tabla 5.6 Descripción de clase “KmsPKCS11”.

Nombre: KmsPKCS11.	
Tipo de clase: controladora. Contiene todas las operaciones que necesitan de la comunicación con el dispositivo (HSM).	
Atributo	**Tipo**
WRAPPING_KEY_ALIAS.	String.
keyGens.	Map<String, KeyGenerator>.

createdSignatures.	Map<String, Signature>.
ciphers.	Map<String, Cipher>.
p11Prov.	SunPKCS11.
keyStoreBuilder.	Builder.
hsmCBH.	HSMCallbackHandler.
p11KS.	KeyStore.
caKS.	KeyStore.
transportKS.	KeyStore.
sessionId.	String.
loged.	Boolean.
p11ChanelsBySession.	Map<String, KmsPKCS11>.

Para cada responsabilidad:

Nombre:	**KmsPKCS11(PKCS11_CFG cfg, String sessionId)**
Descripción:	Crea una instancia de la clase.
Nombre:	**threadChanel()**
Descripción:	Devuelve la instancia de KmsPKCS11 para la sesión actual.
Nombre:	**threadChanel(String sessionId)**
Descripción:	Devuelve la instancia de KmsPKCS11 para la sesión que tiene como identificador el valor que se pasa por parámetro.

Nombre:	**getTokenInfo()**
Descripción:	Obtiene la información del dispositivo activo.
Nombre:	**initToken()**
Descripción:	Inicializa el dispositivo criptográfico, teniendo en cuenta el archivo de configuración y luego lo adiciona a la lista de dispositivos del sistema.
Nombre:	**loadKeyStore(String ksType, InputStream is, String passwrd)**
Descripción:	Carga un fichero keyStore.
Nombre:	**loadCacertsKeyStore()**
Descripción:	Carga el keyStore del KMS que guarda los certificados de confianza.
Nombre:	**login(Subject sbjct, CallbackHandler ch)**
Descripción:	Permite autenticarse en el dispositivo criptográfico.
Nombre:	**logOut()**
Descripción:	Cierra la conexión con el dispositivo criptográfico.
Nombre:	**listAliases()**
Descripción:	Devuelve una lista con los alias de los objetos criptográficos que se encuentran en el dispositivo.
Nombre:	**recoverEntry(String alias, String entryPasswrd)**
Descripción:	Devuelve la entrada del keyStore correspondiente al alias pasado por parámetro.
Nombre:	**recoverKeyHSM(String alias, String entryPasswrd)**
Descripción:	Devuelve la llave privada que se encuentra en el keyStore correspondiente al alias

	pasado por parámetro.
Nombre:	**forceRecoveryKey(String alias, String passwrd)**
Descripción:	Devuelve la llave privada que se encuentra en el keyStore correspondiente al alias pasado por parámetro.
Nombre:	**getKeyStoreCertsBySubject(Map<Principal, List<Certificate>> ks, KeyStore ks)**
Descripción:	A partir del keyStore pasado por parámetro crea una tabla de hash donde la clave es el sujeto del certificado y el valor es el certificado.
Nombre:	**deleteObject(String alias)**
Descripción:	Elimina la entrada del keyStore que corresponde al alias pasado por parámetro.
Nombre:	**genKeyPair(String alias, String keyAlgName, String sigAlgName, HSMCallbackHandler cbhEntryPasswrd, int keysize, X500Name x500Name, Date notBefore, Date notAfter)**
Descripción:	Crea un par de llaves y el certificado en el keyStore.
Nombre:	**genKeyPair(String alias, String keyAlgName, HSMCallbackHandler cbhEntryPasswrd, int keysize)**
Descripción:	Crea un par de llaves asimétricas en el keyStore.
Nombre:	**importKey(Key key)**
Descripción:	Importa una llave al keyStore del dispositivo criptográfico activo.
Nombre:	**sign(String alg, PrivateKey privKey, byte[] data)**
Descripción:	Firma un conjunto de byte.

Nombre:	**sign(String alg, PrivateKey privKey, InputStream is, OutputStream os)**
Descripción:	Firma un flujo de datos.
Nombre:	**verifySign(java.security.cert.Certificate cert, byte[] data)**
Descripción:	Verifica la firma de un certificado.
Nombre:	**encrypt(InputStream is, String alg, SecretKey secKey, OutputStream os)**
Descripción:	Cifra un flujo de datos.
Nombre:	**createEmptyKS(String type, String passwrd, OutputStream os)**
Descripción:	Crea un fichero de tipo keyStore.
Nombre:	**exportKS(String alias, String entryPasswrd, String type, String ksPasswrd, String ksEntryPass, OutputStream os)**
Descripción:	Exporta un keyStore.

Tabla 5.7 Descripción de clase “DataStore”.

Nombre: DataStore.	
Tipo de clase: controladora. Es la encargada de instanciar y decidir qué DTO se va a utilizar.	
Atributo	**Tipo**
dtos.	Map<Class, DBAssembler>.
instance.	DataStore.
Para cada responsabilidad:	
Nombre:	**DataStore()**

Descripción:	Crea una instancia de la clase.
Nombre:	**getInstance()**
Descripción:	Devuelve la instancia de la clase.
Nombre:	**getDTO(Class<? extends BussinessObject> clazz)**
Descripción:	Devuelve el DTO correspondiente con la clase del negocio que se pasa por parámetro.
Nombre:	**countRowInResult(BussinessObject object)**
Descripción:	Devuelve la cantidad de resultados al hacer una búsqueda por ejemplo con el objeto que se pasa por parámetro.
Nombre:	**accessAll(BussinessObject object, int numberPage, int pageSize)**
Descripción:	Devuelve los objetos de la base de datos que se obtienen al buscar por ejemplo el objeto que se pasa por parámetro. Los valores enteros se utilizan para devolver el resultado paginado.
Nombre:	**access(BussinessObject object)**
Descripción:	Obtiene el objeto en la base de datos con el identificador del objeto que se pasa por parámetro.
Nombre:	**mutate(BussinessObject object)**
Descripción:	Modifica en la base de datos el objeto que se pasa por parámetro.
Nombre:	**delete(BussinessObject object)**
Descripción:	Eliminar en la base de datos el objeto que se pasa por parámetro.
Nombre:	**deleteByExample(BussinessObject object)**

Descripción:	Elimina en la base de datos todos los objetos que se obtendrían al hacer una búsqueda por ejemplo con el objeto que se pasa por parámetro.
Nombre:	**store(BussinessObject object)**
Descripción:	Almacena en la base de datos el objeto que se pasa por parámetro.
Nombre:	**close()**
Descripción:	Cierra la conexión con la base de datos.

Tabla 5.8 Descripción de clase "CryptoObjectManager".

Nombre: CryptoObjectManager.	
Tipo de clase: controladora. Contiene las operaciones que pueden ser realizadas sobre cualquier objeto criptográfico.	
Atributo	**Tipo**
instance.	CryptoObjectManager.
Para cada responsabilidad:	
Nombre:	**CryptoObjectManager()**
Descripción:	Crea una instancia de la clase.
Nombre:	**getInstance()**
Descripción:	Devuelve la instancia de la clase.
Nombre:	**getCerManager()**
Descripción:	Devuelve la instancia de la clase CertManager.
Nombre:	**getKeyManager()**

Descripción:	Devuelve la instancia de la clase KeyManager.
Nombre:	**listAllCryptoObjects()**
Descripción:	Devuelve una lista con todos los objetos criptográficos en la base de datos.
Nombre:	**listAllKsCryptoObjects()**
Descripción:	Devuelve una lista con todos los objetos criptográficos en el keyStore.
Nombre:	**syncKsAndDB()**
Descripción:	Sincroniza la base de datos con el keyStore.
Nombre:	**storeCryptoObject(KeyStoreEvent evt)**
Descripción:	Almacena un objeto criptográfico en el keyStore y en la base de datos.
Nombre:	**deleteCryptoObject(CryptoObject cryptoObject)**
Descripción:	Elimina un objeto criptográfico en el keyStore y en la base de datos.
Nombre:	**deleteEntry(KeyStoreEvent evt)**
Descripción:	Elimina la entrada en el keyStore y todos los objetos en la base de datos correspondientes a esta entrada.
Nombre:	**setKeyStoreEntry(KeyStoreEvent evt)**
Descripción:	Modifica una entrada en el keyStore.
Nombre:	**loadKeyStore(KeyStoreEvent evt)**
Descripción:	Sincroniza la base de datos con el keyStore en caso de no haber realizado esta operación anteriormente.
Nombre:	**exportKS (String alias, String entryPasswrd, KeyStoreType type, String**

	ksPasswrd,String ksEntryPass, OutputStream os)
Descripción:	Exporta un keyStore.
Nombre:	**createEmptyKS(KeyStoreType type, String passwrd, OutputStream os)**
Descripción:	Crea un keyStore vacío.
Nombre:	**listCryptoObjectByExample(CryptoObject example, Integer numberPage, Integer pageSize)**
Descripción:	Devuelve una lista con el resultado de buscar por ejemplo el objeto criptográfico.
Nombre:	**countCryptoObjectByExample(CryptoObject example)**
Descripción:	Devuelve la cantidad de objetos que se devolverían al hacer una búsqueda por ejemplo con el objeto que se pasa por parámetro.
Nombre:	**deleteByAlias(String alias)**
Descripción:	Elimina del keyStore y de la base de datos todos los objetos con el alias que se pasa por parámetro.

Tabla 5.9 Descripción de clase “CertManager”.

Nombre: CertManager.	
Tipo de clase: controladora. Contiene las operaciones que se realizan sobre los certificados.	
Atributo	**Tipo**
instance.	CertManager.
Para cada responsabilidad:	

Nombre:	**CertManager()**
Descripción:	Crea una instancia de la clase.
Nombre:	**getInstance()**
Descripción:	Devuelve la instancia de la clase.
Nombre:	**recoverCert(String alias)**
Descripción:	Devuelve el certificado correspondiente al alias que se pasa por parámetro.
Nombre:	**listAllKsCerts()**
Descripción:	Devuelve una lista con los certificados que se encuentran en el keyStore.
Nombre:	**listAllDbCerts()**
Descripción:	Devuelve una lista con los certificados que se encuentran en la base de datos.
Nombre:	**listCertsByUsage(CryptoObjectUsage certUsage, int numberPage, int pageSize)**
Descripción:	Devuelve una lista con los certificados que tienen el uso que se pasa por parámetro.
Nombre:	**exportCert(string alias, PrintStream out)**
Descripción:	Exporta el certificado con el alias que se pasa por parámetro.
Nombre:	**isSelfSignedCert(X509Certificate cert)**
Descripción:	Retorna true si el certificado es auto-firmado.
Nombre:	**isTrustedCert(Certificate cert)**
Descripción:	Retorna true si el certificado es de confianza para el KMS.

Nombre:	**createCertificate(InputStream is)**
Descripción:	Crea un certificado a partir de un fichero con su codificación.
Nombre:	**createCertificate(InputStream is)**
Descripción:	Crea una cadena de certificado a partir de un fichero con su codificación.
Nombre:	**createCertificate(byte[] certData)**
Descripción:	Crea un certificado a partir de su codificación.
Nombre:	**createCertificateRequest(CertificateRequest certRequest, String password, PrintStream out)**
Descripción:	Exporta y almacena en la base de datos la solicitud de certificado que se pasa por parámetro.
Nombre:	**registerCertificateRequest(CertificateRequest certRequest)**
Descripción:	Almacena en la base de datos la solicitud de certificado que se pasa por parámetro.
Nombre:	**importCert(String alias, String password, InputStream certIs)**
Descripción:	Importa un certificado al HSM y lo almacena en la base de datos.
Nombre:	**buildCertChain(X509Certificate certToVerify, List< Certificate> chain, Map<Principal, List< Certificate>> certs)**
Descripción:	Construye la cadena de certificado.
Nombre:	**establishCertChain(X509Certificate certToVerify)**
Descripción:	Verifica que el certificado sea de confianza y en caso de serlo devuelve la cadena de certificado.

Tabla 5.10 Descripción de clase “KeyManager”.

<table>
<tr><td colspan="3">Nombre: KeyManager.</td></tr>
<tr><td colspan="3">Tipo de clase: controladora.

Contiene las operaciones que se realizan sobre las llaves.</td></tr>
<tr><td colspan="2">Atributo</td><td>Tipo</td></tr>
<tr><td colspan="2">instance.</td><td>KeyManager.</td></tr>
<tr><td colspan="3">Para cada responsabilidad:</td></tr>
<tr><td>Nombre:</td><td colspan="2">KeyManager()</td></tr>
<tr><td>Descripción:</td><td colspan="2">Crea una instancia de la clase.</td></tr>
<tr><td>Nombre:</td><td colspan="2">getInstance()</td></tr>
<tr><td>Descripción:</td><td colspan="2">Devuelve la instancia de la clase.</td></tr>
<tr><td>Nombre:</td><td colspan="2">genSecretKey(String alias, String password, KeyProfile keyProfile, int length)</td></tr>
<tr><td>Descripción:</td><td colspan="2">Genera una llave simétrica y la almacena en la base de datos y en el keyStore.</td></tr>
<tr><td>Nombre:</td><td colspan="2">genKeyPairAndCert (String alias, String passwrd, Integer keyLength, KeyProfile keyProf, Certificate cert)</td></tr>
<tr><td>Descripción:</td><td colspan="2">Genera un par de llaves y su certificado y lo almacena en la base de datos y en el keyStore.</td></tr>
<tr><td>Nombre:</td><td colspan="2">genKeyPair(String alias, String passwrd, KeyProfile keyProf, Integer keyLength, User u)</td></tr>
<tr><td>Descripción:</td><td colspan="2">Genera un par de llaves y lo almacena en la base de datos y en el keyStore.</td></tr>
</table>

Nombre:	**recoverKey(String alias, String entryPasswrd)**
Descripción:	Obtiene la llave correspondiente al alias que pasan por parámetro.
Nombre:	**exportPrivateKey(String alias, String passwrd, PrintStream out)**
Descripción:	Exporta la llave privada correspondiente al alias que pasan por parámetro.
Nombre:	**exportPublicKey(String alias, PrintStream out)**
Descripción:	Exporta la llave pública correspondiente al alias que pasan por parámetro.
Nombre:	**listAllKsKeys()**
Descripción:	Devuelve una lista con las llaves del keyStore.
Nombre:	**listAllDbKeys()**
Descripción:	Devuelve una lista con las llaves en la base de datos.
Nombre:	**listKeysByUsage(CryptoObjUsage keyUsage, Integer numberPage, Integer pageSize)**
Descripción:	Devuelve una lista con el resultado de buscar por ejemplo las llaves que tienen el uso que se pasa por parámetro.
Nombre:	**modifyActivatedField(Key key, boolean active)**
Descripción:	Activa o desactiva la llave que se pasa por parámetro.

Tabla 5.11 Descripción de clase “UserManager”.

Nombre: UserManager.
Tipo de clase: controladora. Contiene las operaciones necesarias para gestionar usuarios, roles y operaciones.

Atributo	**Tipo**
No tiene.	-
Para cada responsabilidad:	
Nombre:	**registerUser(User user)**
Descripción:	Almacena un usuario en la base de datos.
Nombre:	**updateUser(User user)**
Descripción:	Modifica un usuario en la base de datos.
Nombre:	**getUserByExample(User u)**
Descripción:	Devuelve una lista de usuarios que contiene el resultado de buscar por ejemplo.
Nombre:	**removeUser(User u)**
Descripción:	Elimina el usuario que se pasa por parámetro de la base de datos.
Nombre:	**addRol(Rol rol)**
Descripción:	Registra un rol en la base de datos.
Nombre:	**getRols()**
Descripción:	Devuelve una lista con los roles que existen en la base de datos.
Nombre:	**getRolsByExample(Rol example)**
Descripción:	Devuelve una lista con el resultado de buscar rol por ejemplo.
Nombre:	**updateRol(Rol rol)**
Descripción:	Modifica un rol de la base de datos.

Nombre:	**removeRol(Rol rol)**
Descripción:	Elimina el rol que se pasa por parámetro de la base de datos.
Nombre:	**registerOperation(Operation op)**
Descripción:	Registra una operación en la base de datos.
Nombre:	**getOperationsByExample(Operation op)**
Descripción:	Devuelve una lista con el resultado de buscar operaciones por ejemplo.
Nombre:	**updateOperation(Operation op)**
Descripción:	Modifica una operación en la base de datos.
Nombre:	**removeOperation(Operation op)**
Descripción:	Elimina la operación de la base de datos.

Tabla 5.12 Descripción de clase “PKCS11_CFG”.

Nombre: PKCS11_CFG.	
Tipo de clase: entidad. Modela el archivo de configuración del dispositivo criptográfico.	
Atributo	**Tipo**
library.	String.
name.	String.
description.	String.
slot.	int.

slotListIndex.	int.
enabledMechanisms.	List<PKCS11Mechanism>.
disabledMechanisms.	List<PKCS11Mechanism>.
attributes.	List<PKCS11Attribute>.
Para cada responsabilidad:	
Nombre:	**PKCS11_CFG**
Descripción:	Crea una instancia de la clase.
Nombre:	**(String library, String name, String description, Integer slot, Integer slotListIndex, List<PKCS11Mechanism> enabledMechanisms, List<PKCS11Mechanism> disabledMechanisms, List<PKCS11Attribute> attributes)**
Descripción:	Crea una instancia de la clase.
Nombre:	**inputStream()**
Descripción:	Exporta el archivo de configuración.
Nombre:	**toByteArray()**
Descripción:	Devuelve un arreglo de bytes con el contenido del archivo de configuración.

Anexo 5: Descripción de las clases de la capa sesión

Tabla 5.13 Descripción de clase "KmsAccessControler".

Nombre: KmsAccessControler.
Tipo de clase: controladora. Clase encargada de controlar toda la autenticación y autorización para la realización de cualquier

acción en el sistema.	
Atributo	**Tipo**
sessionFactory.	SessionFactory.
Para cada responsabilidad:	
Nombre:	**exec(CheckedAction<?> action)**
Descripción:	Ejecuta una acción de forma controlada, chequeando que al usuario de la sesión actual le hayan sido concedidos todos los permisos requeridos para ejecutar dicha acción.
Nombre:	**logIn(String userName, String passwrd)**
Descripción:	Autentica un usuario en el sistema, creando una nueva sesión para dicho usuario.
Nombre:	**logInHSM(String passwrd)**
Descripción:	Establece una sesión con el dispositivo criptográfico.
Nombre:	**logOutHSM()**
Descripción:	Cierra la sesión con el dispositivo criptográfico.
Nombre:	**logOut()**
Descripción:	Cierra la sesión actual.
Nombre:	**checkPermission(String actions)**
Descripción:	Verifica que al usuario asociado a la sesión actual le haya sido concedido el permiso especificado.
Nombre:	**getCurrentSession()**

Descripción:	Obtiene la sesión actual.
Nombre:	**activeSessions()**
Descripción:	Obtiene un listado de todas las sesiones activas en el sistema.
Nombre:	**setSessionPool(SessionPool sp)**
Descripción:	Establece un nuevo almacén de sesiones.

Tabla 5.14 Descripción de clase “CheckedAction”.

Nombre: CheckedAction<T>.	
Tipo de clase: interfaz. Clase que define el comportamiento de una acción chequeada.	
Atributo	**Tipo**
No tiene.	-
Para cada responsabilidad:	
Nombre:	**perform()**
Descripción:	Acción en concreto que se quiere ejecutar.
Nombre:	**String[] requiredPermisions()**
Descripción:	Permisos requeridos para la ejecución completa de la acción que se desea ejecutar.

Tabla 5.15 Descripción de clase “SessionPool”.

Nombre: SessionPool.
Tipo de clase: controladora.

Clase que almacena todas las sesiones creadas en el sistema.	
Atributo	**Tipo**
sessions.	Map<String, Session>.
timer.	Timer.
tasks.	Map<String, TimerTask>.
Para cada responsabilidad:	
Nombre:	**SessionPool()**
Descripción:	Crea una nueva instancia de la clase.
Nombre:	**put(Session s)**
Descripción:	Almacena una nueva sesión.
Nombre:	**get(String id)**
Descripción:	Obtiene una sesión según su id o ticket.
Nombre:	**get(User u)**
Descripción:	Obtiene una sesión según el usuario con el cual está asociada.
Nombre:	**size()**
Descripción:	Obtiene la cantidad de sesiones almacenadas en el SessionPool.
Nombre:	**cancelDestroy(String user)**
Descripción:	Cancela la acción de cerrar la sesión asociada al usuario con nombre de usuario dado.

Nombre:	**destroySession(final String user, long timeOut)**
Descripción:	Planifica el cierre de la sesión asociada a un usuario para luego de transcurrido un tiempo.
Nombre:	**destroySession(String userName)**
Descripción:	Cierra inmediatamente la sesión asociada con el usuario dado.
Nombre:	**lockSession(String user)**
Descripción:	Bloquea la sesión con la cual está asociado el usuario dado.
Nombre:	**destroyAllSessions()**
Descripción:	Cierra todas las sesiones.
Nombre:	**getActivesSessions()**
Descripción:	Obtiene todas las sesiones activas en el sistema.

Tabla 5.16 Descripción de clase “SessionFactory”.

Nombre: SessionFactory.	
Tipo de clase: interfaz. Define el comportamiento de una fábrica de sesiones.	
Atributo	**Tipo**
No tiene.	-
Para cada responsabilidad:	
Nombre:	**createSession(User user)**
Descripción:	Crea una nueva sesión asociada con el usuario dado.

Nombre:	**init()**
Descripción:	Inicializa la fábrica de sesiones.
Nombre:	**getLonginContext(CallbackHandler callbackHandler)**
Descripción:	Obtiene el contexto de autenticación configurado en el sistema, dado el manipulador de datos necesarios para la autenticación.

Tabla 5.17 Descripción de clase “Session”.

Nombre: Session.	
Tipo de clase: entidad. Clase que almacena la información de una sesión.	
Atributo	**Tipo**
sessionId.	String.
user.	User.
lock.	Boolean.
loginCtx.	LoginContext.
property.	Map<String, Object>.
Para cada responsabilidad:	
Nombre:	**Session(User user, LoginContext loginCtx, String sessionId)**
Descripción:	Crea una nueva sesión.
Nombre:	**setProperty(String key, Object value)**
Descripción:	Almacena un objeto en la sesión.

Nombre:	**getProperty(String key)**
Descripción:	Obtiene un objeto que ha sido almacenado previamente en la sesión.
Nombre:	**isLocked()**
Descripción:	Pregunta si la sesión está bloqueada o no.
Nombre:	**lock()**
Descripción:	Bloquea la sesión.
Nombre:	**unlock(String passwrd)**
Descripción:	Desbloquea la sesión.
Nombre:	**logout()**
Descripción:	Cierra la sesión.
Nombre:	**login()**
Descripción:	Inicia la sesión.
Nombre:	**getSubject()**
Descripción:	Obtiene el sujeto asociado con la sesión así como sus credenciales e información de identificación.
Nombre:	**checkPermission(String actions)**
Descripción:	Verifica que dentro de la sesión se pueda ejecutar la acción dada.
Nombre:	**getActivesSessions()**
Descripción:	Obtiene todas las sesiones activas en el sistema.

Las clases Policy, DbPolicy, OperationExcecutionPermission, DbLoginModule y DbCallbackHandler, forman parte del mecanismo JAAS (Servicio de autenticación y autorización de java, por sus siglas en inglés), el cual está especificado en JavaTM 2 SDK, Standard Edition (J2SDK) v1.3 e integrado al J2SDK 1.4. Estas clases se integran con JAAS para utilizar las potencialidades de su autenticación confiable y segura y su control de acceso basado en permisos a nivel de la máquina virtual.

Anexo 6: Descripción de las clases de la capa middleware

Tabla 5.18 Descripción de clase "MiddlewareEventSource".

<table>
<tr><td colspan="3">Nombre: MiddlewareEventSource.</td></tr>
<tr><td colspan="3">Tipo de clase: controladora.

Clase que forma parte del patrón Observer. Contiene un listado de interesados (Listeners) en un evento específico.</td></tr>
<tr><td colspan="2">Atributo</td><td>Tipo</td></tr>
<tr><td colspan="2">listeners.</td><td>List<MiddlewareEventListener>.</td></tr>
<tr><td colspan="2">id.</td><td>long.</td></tr>
<tr><td colspan="3">Para cada responsabilidad:</td></tr>
<tr><td>Nombre:</td><td colspan="2">MiddlewareEventSource()</td></tr>
<tr><td>Descripción:</td><td colspan="2">Crea una instancia de la clase.</td></tr>
<tr><td>Nombre:</td><td colspan="2">removeListener(int index)</td></tr>
<tr><td>Descripción:</td><td colspan="2">Elimina de la lista de listeners el que se encuentra en la posición especificada.</td></tr>
<tr><td>Nombre:</td><td colspan="2">removeListener(Object o)</td></tr>
<tr><td>Descripción:</td><td colspan="2">Elimina de la lista de listeners el primero que sea igual según el método equals al objeto pasado por parámetro.</td></tr>
</table>

Nombre:	**addListener(MiddlewareEventListener e)**
Descripción:	Adiciona a la lista de listeners uno nuevo.
Nombre:	**fireRequestEvent(String fromAddress, String whoUserName, String whoRolName, String serviceName, String action, String protocol, String msg)**
Descripción:	Dispara un evento de tipo **Request**, notificando a cada uno de los listeners suscritos. Espera por parámetro información de un **Request**: desde dónde se realizó la petición, quién la realizó, qué rol tiene, a qué servicio le hizo la petición, qué operación del servicio llamó, por qué protocolo se realizó la petición y alguna información de la petición.
Nombre:	**fireResponseEvent(String fromAddress, String whoUserName, String whoRolName, String serviceName, String action, String protocol, String msg, MiddlewareEvent parent)**
Descripción:	Dispara un evento de tipo **Response**, notificando a cada uno de los listeners suscritos. Espera por parámetro información de un **Response**: hacia dónde se envía la respuesta, a qué usuario se le envía, qué rol tiene, qué servicio responde, qué operación del servicio, por qué protocolo se responde, alguna información de la respuesta y el evento de tipo **Request** que precedió a esta respuesta.
Nombre:	**fireErrorEvent(String fromAddress, String whoUserName, String whoRolName, String serviceName, String action, String protocol, KMSException kmsEx, String msg, MiddlewareEvent parent)**
Descripción:	Dispara un evento de tipo **Error**, notificando a cada uno de los listeners suscritos. Espera por parámetro información de un **Error**: hacia dónde se envía el error, a qué usuario se le envía, qué rol tiene, en qué servicio ocurrió el error, en qué operación del servicio, por qué protocolo se envía el error, el objeto **KMSException** que representa el error, alguna información adicional del error y el evento de tipo

	Request que precedió a este error.

Tabla 5.19 Descripción de clase “MiddlewareEventListener”.

Nombre: MiddlewareEventListener.	
Tipo de clase: interfaz. Clase que define qué operaciones debe implementar un listener del middleware. Forma parte del patrón Observer.	
Atributo	**Tipo**
No tiene.	-
Para cada responsabilidad:	
Nombre:	**onRequest(MiddlewareEvent evt)**
Descripción:	Operación que se ejecutará si ocurre un evento de tipo **Request**.
Nombre:	**onResponse(MiddlewareEvent evt);**
Descripción:	Operación que se ejecutará si ocurre un evento de tipo **Response**.
Nombre:	**onError(MiddlewareEvent evt)**
Descripción:	Operación que se ejecutará si ocurre un evento de tipo **Error**.

Tabla 5.20 Descripción de clase “MiddlewareEvent”.

Nombre: MiddlewareEvent.
Tipo de clase: entidad. Contiene todos los posibles datos que trae un evento del middleware. Forma parte del patrón Observer.

Atributo	**Tipo**
id.	long.
when.	long.
address.	String.
whoUserName.	String.
whoRolName.	String.
serviceName.	String.
action.	String.
protocol.	String.
kmsEx.	KMSException.
msg.	String.
parent.	MiddlewareEvent.
Para cada responsabilidad:	
Nombre:	**MiddlewareEvent(Object source, long id, long when, String address, String whoUserName, String whoRolName, String serviceName, String action, String protocol, KMSException kmsEx, String msg, MiddlewareEvent parent)**
Descripción:	Crea una nueva instancia de un evento del middleware.

Tabla 5.21 Descripción de clase “MWEvtListenerAdapter”.

Nombre: MWEvtListenerAdapter.
Tipo de clase: controladora.

Clase que ofrece una implementación por defecto a las operaciones de un listener del middleware. Forma parte del patrón Adapter.	
Atributo	**Tipo**
No tiene.	-
Para cada responsabilidad:	
Nombre:	**onRequest(MiddlewareEvent evt)**
Descripción:	Operación que se ejecutará si ocurre un evento de tipo **Request**.
Nombre:	**onResponse(MiddlewareEvent evt);**
Descripción:	Operación que se ejecutará si ocurre un evento de tipo **Response**.
Nombre:	**onError(MiddlewareEvent evt)**
Descripción:	Operación que se ejecutará si ocurre un evento de tipo **Error**.

Tabla 5.22 Descripción de clase "MiddlewareFlowInterceptor".

Nombre: MiddlewareFlowInterceptor.	
Tipo de clase: controladora. Clase encargada de interceptar y manipular todo el flujo petición-respuesta-error entre el cliente y el servidor.	
Atributo	**Tipo**
mw.	MiddleWare.
Para cada responsabilidad:	
Nombre:	**MiddlewareFlowInterceptor()**

Descripción:	Crea una instancia de la clase.
Nombre:	**invoke(MessageContext msgContext)**
Descripción:	Recibe el contexto del mensaje de cualquier flujo (entrada, salida, error) entre el cliente y el servidor. Obtiene información necesaria, depura datos entrantes y prepara los salientes.
Nombre:	**publishError(MessageContext msgCtx,String fromAddress, String serviceName, String action, String protocol, KMSException kmsEx, String msg)**
Descripción:	Publica una notificación de error.
Nombre:	**publishRequest(MessageContext msgCtx,String fromAddress, String serviceName, String action, String protocol, String msg)**
Descripción:	Publica una notificación de petición.
Nombre:	**publishResponse(MessageContext msgCtx,String fromAddress, String serviceName, String action, String protocol, String msg)**
Descripción:	Publica una notificación de respuesta.

Tabla 5.23 Descripción de clase “StdoutNotifier”.

Nombre: StdoutNotifier.	
Tipo de clase: controladora. Clase encargada de realizar la salida estándar de los eventos del middleware.	
Atributo	**Tipo**
mw.	MiddleWare.
Para cada responsabilidad:	

Nombre:	**MiddlewareFlowInterceptor()**
Descripción:	Crea una instancia de la clase.
Nombre:	**invoke(MessageContext msgContext)**
Descripción:	Recibe el contexto del mensaje de cualquier flujo (entrada, salida, error) entre el cliente y el servidor. Obtiene información necesaria, depura datos entrantes y prepara los salientes.
Nombre:	**publishError(MessageContext msgCtx,String fromAddress, String serviceName, String action, String protocol, KMSException kmsEx, String msg)**
Descripción:	Publica una notificación de error.
Nombre:	**publishRequest(MessageContext msgCtx,String fromAddress, String serviceName, String action, String protocol, String msg)**
Descripción:	Publica una notificación de petición.
Nombre:	**publishResponse(MessageContext msgCtx,String fromAddress, String serviceName, String action, String protocol, String msg)**
Descripción:	Publica una notificación de respuesta.

Anexo 7: Descripción de las clases de la capa presentación

Tabla 5.24 Descripción de clase “KMS”.

Nombre: KMS.	
Tipo de clase: controladora. Clase que exporta las funcionalidades del sistema como servicios web.	
Atributo	**Tipo**

No tiene.	-
Para cada responsabilidad:	
Nombre:	**registerUser(User user)**
Descripción:	Registra un usuario en el sistema.
Nombre:	**updateUser(User user)**
Descripción:	Actualiza los datos del usuario que se pasa por parámetro.
Nombre:	**getUsersByExample(User u)**
Descripción:	Obtiene todos los usuarios registrados en el sistema a partir de datos conocidos.
Nombre:	**removeUser(User u)**
Descripción:	Elimina un usuario del sistema.
Nombre:	**logIn(String user, String passwrd)**
Descripción:	Autentica un usuario en el sistema.
Nombre:	**lockUser(String user, long timeOut)**
Descripción:	Bloquea la sesión de un usuario autenticado en el sistema.
Nombre:	**logInCryptoModule(String passwrd)**
Descripción:	Autentica al usuario de la sesión actual en el dispositivo criptográfico.
Nombre:	**logOut(String user)**
Descripción:	Cierra la sesión de un usuario.
Nombre:	**logOutCryptoModule()**
Descripción:	Cierra la sesión del usuario actual en el dispositivo criptográfico.
Nombre:	**setCryptoModulePasswrd(String passwrd)**
Descripción:	Modifica el pin del dispositivo criptográfico.

Nombre:	**registerRol(Rol rol)**
Descripción:	Crea un nuevo rol en el sistema.
Nombre:	**getRolsByExample(Rol example)**
Descripción:	Obtiene todos los roles registrados en el sistema a partir de datos conocidos.
Nombre:	**updateRol(Rol rol)**
Descripción:	Actualiza los datos de un rol registrado.
Nombre:	**removeRol(Rol rol)**
Descripción:	Elimina el rol del sistema.
Nombre:	**getOperationsByExample(Operation op)**
Descripción:	Obtiene todas las operaciones registradas en el sistema a partir de datos conocidos.
Nombre:	**getKeyProfiles()**
Descripción:	Obtiene todos los perfiles de llaves permitidos en el sistema.
Nombre:	**getApplicationProfile()**
Descripción:	Obtiene el perfil general del sistema.
Nombre:	**listImportProceduresByExample(ImportProcedure impProc)**
Descripción:	Obtiene los procedimientos de importación registrados en el sistema a partir de datos conocidos.
Nombre:	**getInstalledKeyAssemblersClassName()**
Descripción:	Obtiene los ensambladores de llaves instalados en el sistema.
Nombre:	**getActiveCryptoModule()**
Descripción:	Obtiene el dispositivo criptográfico activo en el sistema.
Nombre:	**listSupportedDevices()**

Descripción:	Obtiene los dispositivos criptográficos soportados por el sistema.
Nombre:	**setActiveCryptoModule(Device cryptoModule)**
Descripción:	Establece un dispositivo criptográfico como activo.
Nombre:	**addCryptoModuleSupport(Device module, AttachmentInfo pkcs11Cfg, AttachmentInfo pkcs11Lib)**
Descripción:	Añade soporte al sistema para un dispositivo criptográfico.
Nombre:	**removeCryptoModuleSupport(Device module)**
Descripción:	Elimina el soporte para un dispositivo.
Nombre:	**forceSyncKsAndDbAsync(DefaultAxisCallback cb)**
Descripción:	Fuerza la sincronización del sistema con el dispositivo criptográfico activo.
Nombre:	**genKey(String alias, KeyProfile keyProfile, Integer keyLength, String passwrd)**
Descripción:	Genera una llave simétrica.
Nombre:	**genKeyPair(String alias, String passString, Integer keyLength, KeyProfile keyProfile, Certificate cert)**
Descripción:	Genera un par de llaves asimétricas con su certificado correspondiente.
Nombre:	**recoverPublicKey(String alias)**
Descripción:	Obtiene una llave pública dado su alias.
Nombre:	**createSelfSignCertV3(String alias, User u, SignatureAlgorithm signAlg)**
Descripción:	Crea un certificado auto-firmado.
Nombre:	**addTrustedCert(String alias, String pass, AttachmentInfo dataAttachInfo)**
Descripción:	Importa un certificado validado al sistema.
Nombre:	**addCACert(String alias, AttachmentInfo dataAttachInfo)**
Descripción:	Importa un certificado raíz de una autoridad certificadora de confianza.

Nombre:	**exportCert(String alias, AttachmentInfo out)**
Descripción:	Exporta un certificado dado su alias.
Nombre:	**exportPublicKey(String alias, AttachmentInfo out)**
Descripción:	Exporta una llave pública dado su alias.
Nombre:	**importKey(String alias, String pass, String wrappingKeyAlias, String wrappingKeyPass, byte[] rawKey, CryptoObjUsage coUsage, KeyAlgorithm keyAlg, KeyType keyType)**
Descripción:	Importa una llave al sistema.
Nombre:	**exportPrivateKey(String alias, String password, AttachmentInfo out)**
Descripción:	Exporta una llave privada si es exportable.
Nombre:	**recoverCert(String alias)**
Descripción:	Obtiene un certificado según su alias.
Nombre:	**signData(AttachmentInfo dataAttachInfo)**
Descripción:	Firma datos con los objetos de firma activos.
Nombre:	**sign(SignatureAlgorithm alg, PrivateKey privKey, byte[] data)**
Descripción:	Firma datos con una llave especificada.
Nombre:	**verifySign(Certificate cert, AttachmentInfo dataAttachInfo)**
Descripción:	Verifica la firma.
Nombre:	**listCryptoObjects()**
Descripción:	Obtiene todos los objetos criptográficos.
Nombre:	**listPublicKeys(Integer numberPage, Integer pageSize)**
Descripción:	Obtiene las llaves públicas.
Nombre:	**listPrivateKeys(Integer numberPage, Integer pageSize)**

Descripción:	Obtiene las llaves privadas.
Nombre:	**listKeysByUsage(CryptoObjUsage keyUsage, Integer numberPage, Integer pageSize)**
Descripción:	Obtiene todas las llaves cuyo uso sea el especificado.
Nombre:	**listCertsByUsage(CryptoObjUsage certUsage, Integer numberPage, Integer pageSize)**
Descripción:	Obtiene todos los certificados cuyo uso sea el especificado.
Nombre:	**deleteCryptoObject(String alias)**
Descripción:	Elimina un objeto criptográfico.
Nombre:	**deleteCertificateRequest(CertificateRequest req)**
Descripción:	Elimina una solicitud de certificación.
Nombre:	**createCertificateRequest(CertificateRequest certRequest, String password, AttachmentInfo out)**
Descripción:	Crea y exporta una solicitud de certificación.
Nombre:	**registerCertificateRequest(CertificateRequest certRequest)**
Descripción:	Registra una solicitud de certificación.
Nombre:	**listCertificateRequest(Integer numberPage, Integer pageSize)**
Descripción:	Obtiene todas las solicitudes de certificación que existen en el sistema.
Nombre:	**totalObjectByUse(CryptoObjUsage use)**
Descripción:	Obtiene el total de objetos criptográficos existentes en el sistema cuyo uso es el especificado.
Nombre:	**totalPrivateKey()**
Descripción:	Obtiene el total de llaves privadas existentes en el sistema.
Nombre:	**totalPublicKey()**

Descripción:	Obtiene el total de llaves públicas existentes en el sistema útil para la búsqueda por paginado.
Nombre:	**totalCertificate()**
Descripción:	Obtiene el total de certificados existentes en el sistema.
Nombre:	**totalCertificateRequest()**
Descripción:	Obtiene el total de solicitudes de certificación existentes en el sistema.
Nombre:	**exportKeyStore(String alias, String entryPasswrd, KeyStoreType type, String ksPasswrd, String ksEntryPass, AttachmentInfo outKS)**
Descripción:	Exporta un almacén de llaves.
Nombre:	**modifyActivatedField(Key key, boolean active)**
Descripción:	Activa y desactiva los objetos de firma en el sistema.
Nombre:	**encrypt(String alias, String keyPass, CipherType cipherType, AttachmentInfo data, AttachmentInfo out, AxisCallback cb)**
Descripción:	Cifra datos con el certificado o la llave secreta asociada al alias especificado.
Nombre:	**decrypt(String alias, String keyPass, CipherType cipherType, AttachmentInfo data, AttachmentInfo out, AxisCallback cb)**
Descripción:	Descifra datos con la llave privada o secreta asociada al alias especificado.

Tabla 5.25 Descripción de clase “Utilitie”.

Nombre: Utilitie.	
Tipo de clase: controladora. Clase que exporta las funcionalidades utilitarias del sistema.	
Atributo	**Tipo**
No tiene.	-
Para cada responsabilidad:	

Nombre:	**getServerOperatingSystem()**
Descripción:	Obtiene el sistema operativo sobre el cual está corriendo el servidor.
Nombre:	**getServerDate()**
Descripción:	Obtiene la fecha del servidor.
Nombre:	**getKMSServerVersion()**
Descripción:	Obtiene la versión del sistema que está siendo ejecutado.
Nombre:	**getKMSServerStatus()**
Descripción:	Obtiene el estado del servidor.
Nombre:	**getAvailableLangsInServer()**
Descripción:	Obtiene los lenguajes disponibles en el servidor.
Nombre:	**getNclProtocols()**
Descripción:	Obtiene los protocolos permitidos para la comunicación con el sistema.
Nombre:	**getNclLogLevel()**
Descripción:	Obtiene los niveles de registro existentes en el sistema.
Nombre:	**getNclControlFlag()**
Descripción:	Obtiene las banderas de control de los módulos de autenticación del sistema.
Nombre:	**getNclKeyUsages()**
Descripción:	Obtiene los usos de llaves admitidos por el sistema.
Nombre:	**getNclCryptoObjUsages()**

Descripción:	Obtiene los posibles usos de los objetos criptográficos dentro del proceso en el cual ha sido desplegado el sistema.
Nombre:	**getNclSignatureAlgorithms()**
Descripción:	Obtiene los algoritmos de firma permitidos por el sistema.
Nombre:	**getNclKeyTypes()**
Descripción:	Obtiene los tipos de llaves permitidos en el sistema.
Nombre:	**getNclCertTypes()**
Descripción:	Obtiene los tipos de certificados permitidos por el sistema.
Nombre:	**getNclAsimetricKeyAlgorithms()**
Descripción:	Obtiene los algoritmos de llaves públicas permitidos en el sistema.
Nombre:	**getNclSimetricKeyAlgorithms()**
Descripción:	Obtiene los tipos de algoritmos simétricos permitidos en el sistema.
Nombre:	**getNclKeyFormats()**
Descripción:	Obtiene los formatos de llaves permitidos en el sistema.