UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

MPC: Módulo parseador de certificados

Autor: Manuel Hernández Torregrosa

Tutor: Valentín Moreno Pelayo

Agradecimientos

En primer lugar quiero agradecer a mi tutor, Valentín Moreno el hecho de aceptar llevar este proyecto y ser paciente conmigo, sin lo cual hubiera sido imposible llegar a realizarlo.

En segundo lugar quiero dar las gracias a Javier Villarroel por ayudarme con las dudas que fueron surgiendo e intentar siempre que no hiciera las cosas de cualquier manera.

Gracias también a María Jesús por permitir que esto haya sido posible cuando lo fácil hubiera sido dejarlo pasar.

Quiero agradecer a mi padre, mi madre y mi hermano por apoyarme y darme ánimos en todo momento.

Por último pero más importante, quiero dar las gracias a Paloma, por ser mi compañera de viaje y estar siempre a mi lado compartiendo los buenos y malos momentos.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1 Introducción

1.1 Objetivo

En la administración pública existen multitud de aplicaciones, muchas para uso de ciudadanos y otras tantas de gestión para uso por parte del personal de la administración. Muchas de estas aplicaciones utilizan como método de autenticación el uso de certificados electrónicos mediante SSL.

De esta forma, no se necesita un usuario y contraseña para controlar el acceso a las mismas. Se puede tener por ejemplo una lista de DNIs autorizados almacenados en base de datos, y al acceder un usuario con un certificado digital comprobar que el DNI asociado a ese certificado es el mismo que se tiene en base de datos y por tanto permitir el acceso. Además, en muchos casos al exigir el uso de un certificado las aplicaciones pueden obtener información directamente del mismo en lugar de tener que solicitar al usuario que la escriba en un formulario. Esto permite que datos como nombre y apellidos, número de documento de identidad, dirección de correo electrónico, empresa a la que representa, etc. se obtengan directamente del propio certificado.

A raíz de todo esto, surge la necesidad de tener una herramienta capaz de extraer información de un certificado digital así como de filtrar qué certificados están permitidos para operar con las aplicaciones de la administración pública. El problema principal reside en la gran variedad de prestadores de certificados existentes. Aunque todos los prestadores utilizan un estándar común - el llamado formato X509 - cada uno tiene sus particularidades. Además un prestador puede emitir certificados de muchos tipos: persona jurídica, persona física, certificado de sello, certificado de sede, etc. y para diversos usos o propósitos: certificado de autenticación, certificado de firma, certificado exclusivo para uso de una determinada entidad, etc.

Es necesario por tanto crear un sistema centralizado que resuelva toda esta problemática de forma transparente a las aplicaciones. Las aplicaciones se conectarán a este sistema y mandarán el certificado de cliente obtenido mediante ssl y el sistema les devolverá los datos parseados del mismo así como una validación básica que consistirá en comprobar la caducidad, integridad y confianza del mismo.

1.2 Estructura y contenido del documento

En este apartado se describe brevemente la estructura y contenido del presente documento. La memoria está dividida en los siguientes apartados:

- **Estado del arte.** En este apartado se mostrará una explicación de cada una de las tecnologías, mecanismos y herramientas que intervendrán en la realización del proyecto. Se explicará qué es y para que se usan los certificados digitales, se hará una introducción a la infraestructura de clave pública, se explicará cómo se validan los certificados digitales, cuál es la labor de un prestador de certificados y se dará una introducción al estándar x509. También se verá la utilidad de los servicios web y el formato XML y como encajan perfectamente con el objetivo de este proyecto de intercambiar información entre aplicaciones. Por último se explicará brevemente la tecnología .NET de Microsoft utilizada para implementar el sistema, el complemento Microsoft Enterprise Library para crear el log y la capa de acceso a datos, y la librería BouncyCastle para el manejo de certificados digitales.
- **Análisis del sistema.** En este apartado se muestra el análisis del sistema, que es lo primero que se debe realizar para afrontar la creación del proyecto MPC. Dentro del análisis se especificarán los requisitos de la aplicación, los casos de uso del sistema y los diagramas de secuencia y de entidad-relación de la base de datos. También se realizará un análisis de las declaraciones de prácticas de certificación de cada prestador aceptado por el sistema, indicando qué certificados se soportan de cada tipo y que particularidades tienen estos. Finalmente en esta fase se definirán los datos de salida -explicando cada campo- que devolverán los 3 métodos del sistema MPC.
- **Diseño del sistema.** En esta fase del proyecto se realizará una definición formal de los datos de salida mediante la creación del esquema XSD de la aplicación, el cual deberán cumplir los ficheros xml devueltos por MPC. También se explicará el patrón Modelo-Vista-Controlador utilizado para estructurar y hacer más mantenible el código de la aplicación. Se mostrarán diagramas de componentes, de espacios de nombres y de clases del sistema MPC. Finalmente se explicarán los métodos más importantes mostrando parte de su código fuente.

- **Aplicación Resultante.** Aquí se podrá ver el resultado de la implementación del proyecto. Se mostrarán los archivos compilados y se describirá su función. También se mostrará como visualizar MPC mediante un navegador web y se darán ejemplos de salidas para cada método web eligiendo distintos certificados.
- **Planificación.** En este apartado se mostrará la planificación seguida para la realización del sistema MPC. Se explicarán que fases del proceso se estimó que llevarían más tiempo y cuales menos y se asignará un tiempo a cada tarea. Todo esto quedará especificado de forma formal en el diagrama Gantt del proyecto.
- **Presupuesto.** Se estimarán que recursos tanto humanos como de software y hardware son necesarios para completar el proyecto. Se obtendrá el coste total del proyecto teniendo en cuenta los recursos y la duración de cada tarea especificada en el diagrama de Gantt.
- **Futuras líneas de trabajo.** Se expondrán brevemente las evoluciones y posibles mejoras que puede incluir el sistema MPC en el futuro.
- **Conclusiones.** En este apartado se reflexionará sobre la experiencia, conocimientos y lecciones adquiridos tras realizar el sistema MPC.
- **Bibliografía.** Fuentes consultadas para la realización de este documento.

2 Definiciones, acrónimos y abreviaturas

Minetur	Ministerio de Industria, Energía y Turismo
MPC	Módulo parseador de certificados
SSL	Secure Sockets Layer (Capa de conexión segura)
PKI	Public Key Infrastructure (Infraestructura de clave pública)
Función Hash	Función Resumen o Función Digest
CRL	Certificate Revocation List (Lista de certificados revocados)
OCSP	Online Certificate Status Protocol (Protocolo online de estado de certificados)
DPC	Declaración de prácticas de certificación
x509	Estándar para certificados en PKI
SOAP	Simple Object Access Protocol (Protocolo simple de acceso a objetos)
WSDL	Web Services Description Language (Lenguaje para describir servicios web)
UDDI	Universal Description, Discovery and Integration (Descripción, Localización e Integración Universales)
XML	Extensible Markup Language (Lenguaje de marcas extensible)
DTD	Descripción de tipo de documento

XSD	Xml Schema Definition (Definición de esquema XML)
.NET	Framework de Microsoft para el desarrollo de aplicaciones
IIS	Internet Information Services (Servidor Web para entornos Windows)
FNMT	Fábrica Nacional de Moneda y Timbre
DNIE	Documento Nacional de Identidad Electrónico
CA / AC	Certification Authority (Autoridad de certificación)
DN	Distinguished name (Nombre distintivo)
OU	Organizational Unit (Unidad Organizativa)
CN	Common Name (Nombre común)
C	Country (País)
OID	Object Identifier (Identificador de objeto)
DLL	Dynamic Link Library (Librería de enlace dinámico)
MVC	Modelo vista controlador
NIF	Número de identificador fiscal
CIF	Código de identificador fiscal

Ilustración 1: Definiciones, acrónimos y abreviaturas

3 Estado del arte

3.1 Certificados digitales

A día de hoy el uso de certificados digitales en Internet se ha convertido en algo habitual e imprescindible. Los certificados digitales surgen de la necesidad de establecer comunicaciones seguras a través de la red, garantizando la autoría, integridad y privacidad de los mensajes intercambiados. Un certificado digital en esencia no es más que un archivo digital firmado en el que se asocia la identidad del titular (persona o entidad) a una clave pública. Esta asociación queda garantizada por medio de la entidad de certificación que emite el certificado, que es un tercer elemento de confianza. La entidad de certificación emisora garantiza que esa clave pública pertenece a ese individuo o entidad firmándolo digitalmente. Los certificados digitales son muy importantes ya que son utilizados por el protocolo SSL (Secure Sockets Layer) para establecer comunicaciones seguras entre cliente-servidor. En el proceso de negociación del protocolo SSL se deben intercambiar certificados digitales utilizando una infraestructura de clave pública.

Se puede decir que los certificados digitales son los “documentos de identidad” en Internet, de hecho, desde el año 2006 existe el DNI Electrónico, que no es más que dotar al documento nacional de identidad de validez en el ámbito de las comunicaciones electrónicas mediante la creación de un par de certificados digitales (uno de firma y otro de autenticación) asociados a cada DNI. La dirección general de la policía emite los certificados para el DNI electrónico, pero este solo es uno de los muchos prestadores de servicio de certificación existentes. Estos últimos son las entidades encargadas de expedir certificados y realizar tareas relacionadas con la firma electrónica.

Los certificados digitales tienen una importancia capital en Internet ya que constituyen con la ayuda de la criptografía asimétrica los pilares básicos de la infraestructura de clave pública o PKI. Gracias a esto, se pueden garantizar cuestiones tan importantes como las descritas a continuación:

- **Autenticación**. Mediante la autenticación se puede garantizar la identidad de una persona o entidad. De esta forma se puede por ejemplo garantizar la

identidad de un servidor frente a un usuario, la autoría de un software que vayamos a descargar o las identidades de los participantes en una comunicación electrónica.

- **Integridad.** La integridad nos permite garantizar que un elemento no ha sido alterado por un tercero sin que tengamos constancia de ello. Por ejemplo, a la hora de descargar un software estaremos seguros que este ha permanecido inmutable tal y como lo concibió su creador o que el contenido de un mensaje recibido es idéntico al contenido del mensaje en el momento del envío.
- **Privacidad.** Podremos garantizar que una información solo sea recibida y procesada por sus destinatarios o por elementos autorizados para tal efecto.
- **No repudio.** Mediante este mecanismo se garantiza que el emisor de un mensaje no pueda negar que lo ha emitido y a su vez cuando el remitente lo recibe este no puede negar que lo ha recibido. También es especialmente importante en las firmas de documentos impidiendo que se niegue la autoría de la firma.

El componente más importante de un certificado digital es su clave privada. La clave privada del certificado solo es conocida por el titular, y es esencial a la hora de firmar documentos o establecer un canal seguro de comunicación entre otras funcionalidades. El software que se utiliza actualmente en nuestros ordenadores está pensado para compartir solo la parte pública de nuestros certificados garantizando así que no se compromete la clave privada en ningún momento.

Gracias al uso de certificados digitales hoy en día los ciudadanos podemos llevar a cabo multitud de trámites de forma electrónica como la solicitud y aceptación de la declaración de la renta, consulta de vida laboral, pago de impuestos, banca electrónica segura etc.

Pero los certificados también cumplen una función importante en las redes internas de muchas empresas y organismos. En muchas administraciones públicas son muy utilizados para identificar y realizar tareas internas por parte del funcionariado y trabajadores públicos. En muchos ministerios por ejemplo, se utilizan tarjetas criptográficas inteligentes (smart cards) para identificar a los empleados públicos. Estas tarjetas tienen capacidad para almacenar de forma segura un certificado digital e

incluyen el software necesario para poder utilizar el certificado directamente desde la tarjeta sin que este tenga que ser instalado en un ordenador. De esta forma cada trabajador posee una tarjeta que lo identifica con un certificado digital válido y en regla para poder realizar actividades diarias relacionadas con su trabajo. Utilizando estos certificados digitales el personal de las administraciones puede acceder a aplicaciones software internas para realizar tareas como control horario, solicitud de vacaciones, intercambio de documentos entre departamentos, apuntes en registros electrónicos, peticiones de resolución de incidencias, solicitud de recursos, etc. Aplicaciones que facilitan en definitiva el control y la seguridad del flujo de trabajo diario dentro de las administraciones públicas.

3.2 Infraestructura de clave pública y criptografía asimétrica

La infraestructura de clave pública es un sistema que utilizando certificados digitales y criptografía asimétrica garantiza la seguridad de los servicios en Internet. Es por así decirlo el estándar aceptado de seguridad en Internet. Un sistema de infraestructura de clave pública se compone de varios elementos:

- **Autoridades de certificación.** Son las encargadas de emitir y revocar los certificados digitales.
- **Autoridades de registro**. Se encargan de registrar y gestionar las peticiones de emisión de certificados y también de comprobar la veracidad de sus datos.
- **Autoridades de validación**. Se encargan de validar lo certificados digitales.
- **Autoridades de sellado de tiempo**. Se encargan de dar fe de la fecha en que se emitió una firma.
- **Certificados digitales finales**. Es el documento digital que asocia al titular del mismo un par de claves para poder realizar operaciones mediante software que utiliza la infraestructura de clave pública.
- **Directorios de claves**. Son almacenes dónde se guardan las claves públicas de los certificados.

La infraestructura de clave pública se basa en la confianza, el remitente debe tener la certeza de que la clave pública que utilizará para verificar las comunicaciones pertenece realmente a quien dice ser el emisor. Por este motivo los certificados digitales están a su vez firmados por otros certificados digitales pertenecientes a entidades de certificación formando así lo que se conoce como una jerarquía de confianza. El nivel más alto de estas jerarquías son los certificados raíces que generalmente son autofirmados. Luego suelen estar en caso de que existan las denominadas entidades intermedias, que son las encargadas de emitir los certificados finales. No hay una regla a la hora de definir una jerarquía de certificación y esta puede tener varios niveles que separen el certificado final del certificado raíz.

La criptografía de clave pública utiliza un par de claves de cifrado diferentes. La clave privada se almacena en un lugar seguro, solo es conocida por el titular del

certificado y es utilizada principalmente para firmar las comunicaciones o mensajes. La clave pública se envía comúnmente al receptor y este la utiliza para verificar la firma de los mensajes del remitente. La criptografía asimétrica se basa por tanto en la utilización de claves distintas para cifrar y descifrar un mensaje.

Llegados a este punto es necesario introducir el concepto de Huella Digital o resumen de un documento y el de función hash. Una función hash transforma una cadena de entrada dando como resultado otra cadena distinta de valor finito. A este resultado le llamamos resumen o huella digital del documento. Idealmente una función hash devuelve para cadenas distintas resultados distintos. Se entiende que dos resultados idénticos al utilizar una misma función hash deben corresponder a dos cadenas de entrada idénticas. Así pues una mínima alteración en un documento producirá una gran alteración en su huella digital o resumen. Dada esta propiedad de las funciones hash cuando se quiere enviar un mensaje o firmar un documento lo que se hace es firmar un resumen de ese mensaje o documento en lugar de firmar el documento o mensaje completo ya que así resulta menos costoso y se ahorra tiempo, espacio y recursos.

A continuación vamos a ver un pequeño ejemplo de aplicación de la criptografía asimétrica. El emisor (A) quiere enviar un mensaje al receptor (B) de tal forma que este último no tenga duda de la autenticidad ni integridad del mismo. Para lograrlo se debería actuar de la siguiente manera:

El emisor utilizando una función hash crea un resumen del mensaje original:

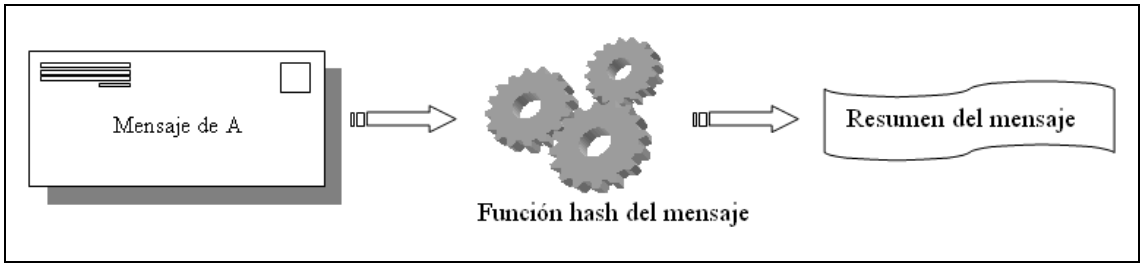


Ilustración 2: Cálculo del resumen del mensaje que se quiere enviar

El emisor cifra el resumen resultante utilizando su clave privada:

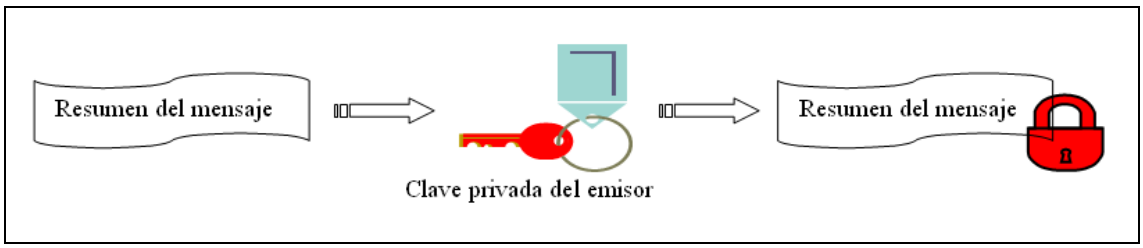


Ilustración 3: Cifrado de resumen con clave privada

Posteriormente el emisor envía el mensaje junto con el resumen cifrado del mismo a su destinatario.

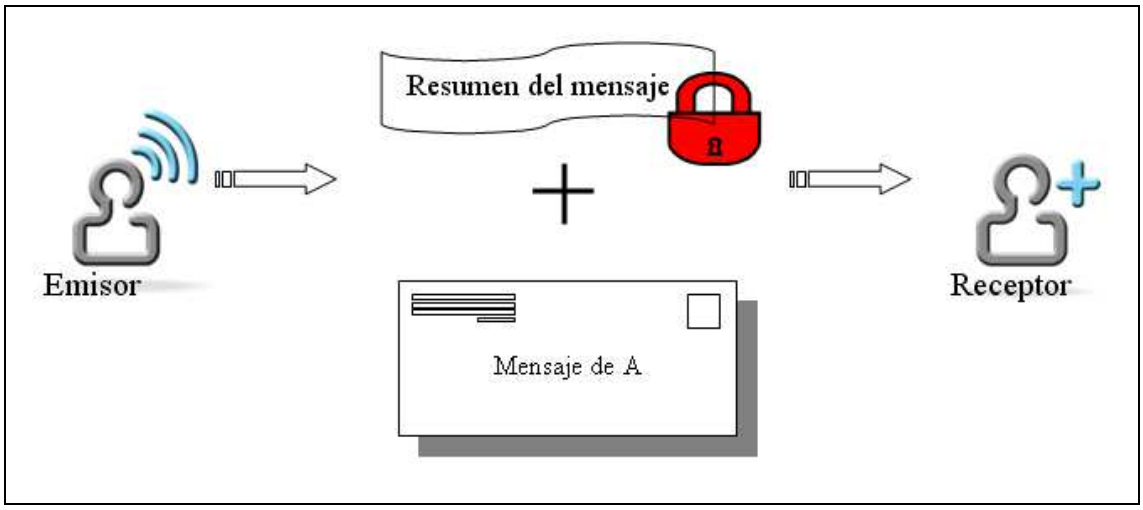


Ilustración 4: Envío de mensaje y resumen

El receptor del mensaje descifra el resumen recibido utilizando la clave pública del emisor obteniendo así el resumen original.

Ilustración 5: Obtención del resumen original

Acto seguido utilizando la misma función hash que usó el emisor - la cual es conocida por ambos – el receptor calcula el resumen del mensaje recibido.

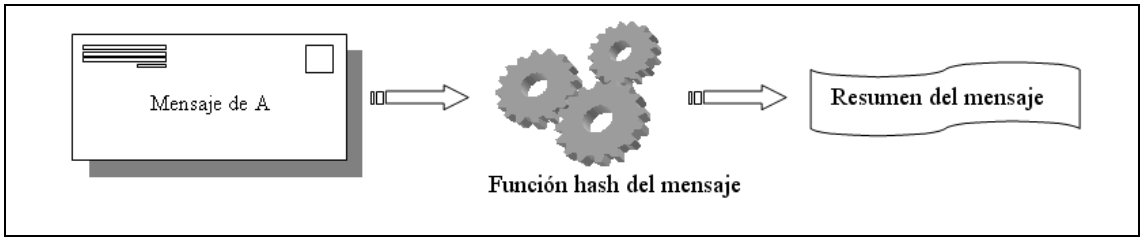


Ilustración 6: Cálculo del resumen a partir del mensaje recibido

Una vez obtenido el resumen del mensaje recibido se compara con el resumen descifrado o resumen original. Ambos resúmenes deben ser idénticos, si no lo fueran la integridad del mensaje estaría comprometida ya que para dos cadenas de entrada idénticas una función hash devuelve la misma cadena resumen y al no ser iguales significaría que el mensaje ha sido alterado.

Con este sistema, además de asegurar la integridad del mensaje se comprueba la autenticidad del mismo, es una manera de que el emisor se autentique ante otros usuarios o sistemas y de garantizar el no repudio (el emisor no podrá negar que es el emisor del mensaje), ya que en un sistema criptográfico asimétrico cada clave de cifrado va asociada a una única clave de descifrado. No es difícil darse cuenta que la seguridad real del sistema de infraestructura de clave pública reside en que la clave privada siga siendo en todo momento privada y solo conocida por el elemento que cifra. Así pues, hemos visto que la firma digital de un documento es un resumen del mismo obtenido mediante una función hash y cifrado posteriormente utilizando la clave privada del firmante.

3.3 Validación de certificados

La confianza es muy importante dentro de la infraestructura de clave pública. ¿Cómo podemos estar seguros que una clave pública pertenece a una persona o entidad? Ahí es donde entran en juego las entidades de certificación.

Las entidades o autoridades de certificación son entidades de confianza responsables de emitir y revocar certificados digitales. Es un tercer elemento de confianza dentro la infraestructura de clave pública. A ellos corresponde la labor de emitir los certificados digitales para el uso de terceros. Una autoridad de certificación puede revocar un certificado, indicando desde ese momento que el certificado debe ser considerado no válido. Esto puede ocurrir por varios motivos:

- La clave privada del certificado o de alguna de sus autoridades de certificación ha quedado comprometida.
- Que lo solicite expresamente el titular del certificado.
- Finalización del periodo de validez del certificado o extinción de la entidad a la que representa.
- Que una vez expedido el certificado se detecten anomalías o incorrecciones en los datos proporcionados por el solicitante.
- Fallecimiento o incapacidad del titular.
- Daño, destrucción o pérdida del soporte físico dónde se almacena el certificado.

Así pues es labor de las entidades de certificación indicar cuando el estado de un certificado ha cambiado. Puesto que esta información no viaja con el certificado era necesario crear algún tipo de mecanismo de consulta. Primeramente se usaban listas de certificados revocados, también conocidas por sus siglas en inglés, CRL (Certificate Revocation List). Las listas CRL no son más que listas de certificados digitales revocados, habitualmente sus números de serie. Puesto que para una misma autoridad de certificación el número de serie de un certificado debe ser único basta únicamente con este dato para consultar el estado del certificado en la lista. Las listas CRL aunque se siguen utilizando en la actualidad tienen importantes inconvenientes, entre ellos su frecuencia de actualización. Al no ser un método instantáneo puede darse el caso de que un certificado digital este revocado pero al consultar la lista CRL ésta no se haya actualizado todavía con el número de serie y el certificado se de por

bueno hasta la próxima vez que se consulte la lista. Además, con el paso del tiempo cada vez son más grandes los archivos y por tanto requieren un mayor tiempo de procesamiento.

Para hacer frente a todos estos inconvenientes y como forma mejorada y complementaria de consulta se creó el protocolo OCSP (Online Certificate Status Protocol). El protocolo OCSP permite consultar de manera inmediata el estado de un certificado. Además las aplicaciones no tienen que procesar las listas CRL ya que la consulta se realiza sobre un certificado concreto, esto también es una ventaja en cuanto a la privacidad y la protección de datos, ya que el estado de un certificado puede ser considerado como información confidencial.

3.4 Prestadores y políticas de certificación

Los entes con capacidad para expedir y gestionar certificados digitales son los llamados prestadores de servicio de certificación. Están obligados por ley a proporcionar un servicio actualizado y fiable de validación en el que pueda consultarse el estado de vigencia o suspensión de un certificado. Además, deben publicar un documento de políticas o prácticas de certificación (DPC). Son responsables de tutelar el buen uso de los certificados que emiten.

En España, es importante hacer mención a los denominados "certificados digitales reconocidos", que según establece la ley "*son los certificados electrónicos que se han expedido cumpliendo requisitos cualificados en lo que se refiere a su contenido, a los procedimientos de comprobación de la identidad del firmante y a la fiabilidad y garantías de la actividad de certificación electrónica."* Este tipo de certificados son imprescindibles para realizar firmas electrónicas reconocidas, por eso es muy importante conocer el emisor del certificado digital y que este sea un prestador confiable. La Unión Europea acordó que cada país miembro emitiría una lista de confianza con información sobre los prestadores de servicios de certificación que operen en cada país y que emitan certificados digitales reconocidos bajo la supervisión de cada estado miembro. En España esta lista la publica y gestiona el Ministerio de Industria, Energía y Turismo (Minetur).

Como se ha mencionado anteriormente los prestadores de servicio de certificación están obligados a publicar un DPC con información sobre su gestión de los certificados. La información más común e importante que uno debe encontrar en un documento DPC es la siguiente:

- Condiciones o requisitos que han de cumplirse para poder solicitar, usar o revocar un certificado digital.
- Definición del ciclo de vida del certificado.
- Información sobre el sistema de validación de certificados (Típicamente servidor OCSP o listas CRL) incluyendo garantías, condiciones de uso e información técnica de acceso al mismo.
- Información sobre procedimientos de seguridad implementados con el objetivo de garantizar la confianza en el sistema de infraestructura de clave pública.

Aquí se incluyen datos técnicos y no técnicos, desde medidas de seguridad software hasta información sobre planes de auditorías en las instalaciones dónde se almacenen los certificados raíces, etc.

- Información sobre la jerarquía o cadena de certificación que forman parte de la infraestructura de clave pública, niveles de confianza, números de serie de certificados raíces, entidades de certificación intermedia, etc.
- Información sobre los tipos de certificados emitidos: perfiles, formatos, extensiones críticas, extensiones privadas, usos admitidos, etc.
- Establecimiento de responsabilidades de las partes implicadas: prestador, suscriptor, entidad usuario, etc.

Este documento es de vital importancia de cara a realizar cualquier tipo de software que requiera interactuar con certificados digitales. Al poner en marcha un proyecto software de este tipo el documento DPC de cada prestador soportado por el sistema deberá ser estudiado en la fase de análisis, ya que en él se establece toda la información relevante a los certificados con los que se deberá trabajar.

3.5 El estándar x509

Un certificado digital es como dijimos en el punto 3.1 un archivo digital mediante el cual se asocia la identidad del titular (persona o entidad) a una clave pública. Esa es la esencia del certificado digital pero indudablemente un certificado debe contener mucha más información. Los campos más importantes en un certificado digital son los siguientes:

- Identidad del titular
- Clave pública del titular.
- Identidad de la entidad que firma y expide el certificado.
- Algoritmo utilizado para firmar el certificado.

Pero aunque esos serían los datos más importantes en la práctica son necesarios muchos más elementos para definir correctamente la estructura de un certificado. Así pues, con el fin de definir la composición que debería tener un certificado digital se creó el estándar x509, cuya primera versión se remonta a 1988. Existen varios tipos de formatos de certificado digital pero el más utilizado y en el que nos centraremos a lo largo de este documento es el que se describe en el estándar X509 v3 de la UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT). Actualmente es el estándar aceptado para la implementación de certificados digitales.

La estructura típica de un certificado x509 es la siguiente:

Versión
Número de serie
Identificador del algoritmo de la firma
Emisor
Periodo de validez

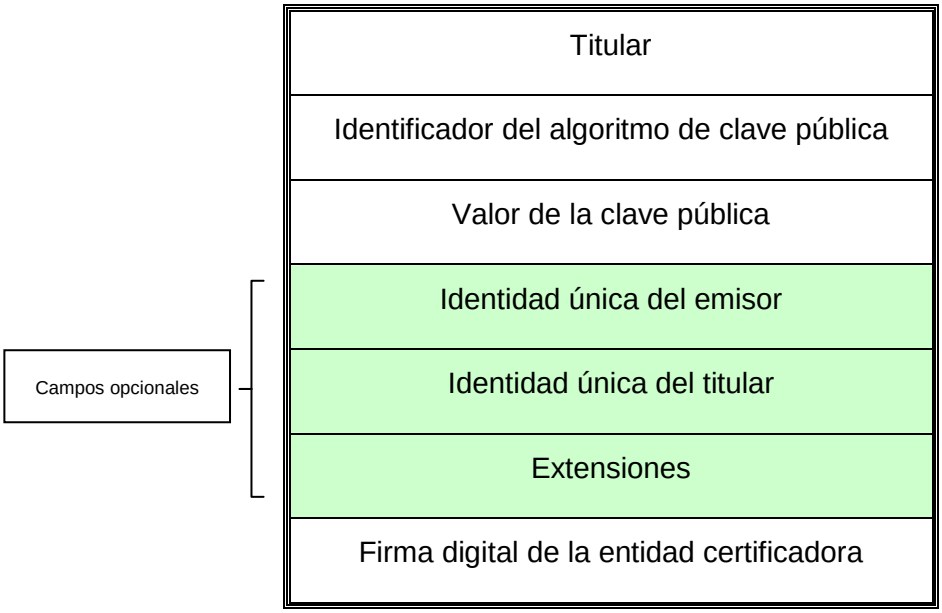


Ilustración 7: Estructura certificado x509 v3

- **Versión:** Indica la versión del estándar x509 que implementa el certificado.
- **Número de serie:** Número de serie del certificado.
- **Identificador del algoritmo de firma:** Identificador del algoritmo utilizado por la autoridad de certificación para firmar el certificado.
- **Emisor:** Información sobre la entidad que expidió el certificado.
- **Periodo de validez:** Rango de fechas entre las cuales el certificado puede ser válido.
- **Titular:** Información sobre el titular a quien le ha sido expedido el certificado digital y es por tanto dueño de la clave pública que se está firmando.
- **Identificador del algoritmo de clave pública:** Identificador del algoritmo con el cual se debe utilizar la clave pública.
- **Valor de la clave pública:** Valor explícito de la clave pública asociada al certificado.
- **Identidad única del emisor:** Campo opcional que identifica unívocamente al emisor del certificado. Introducido en la versión 2 del estándar x509.
- **Identidad única del titular:** Campo opcional que identifica unívocamente al titular del certificado y dueño de la clave pública que se firma. Introducido en la versión 2 del estándar x509.

- **Extensiones:** Permiten a cada prestador de servicios de certificación extender la información y los requisitos del contenido y formato de sus certificados digitales. Existen extensiones estándar, de Internet o privadas. Fueron la gran novedad de la versión 3 del estándar x509.

3.6 Servicios Web

Un servicio Web es un mecanismo mediante el cual se busca intercambiar información de forma sencilla y segura entre aplicaciones a través de la Web independientemente de su origen, propiedades o plataforma en que estén instaladas.

Podemos verlo de forma muy clara con un ejemplo. Hoy en día es muy común ver en aplicaciones de teléfonos móviles o en páginas representaciones de información meteorológica para una determinada ciudad. Sería realmente impensable que cada aplicación o Web que quisiera mostrar datos sobre el tiempo que va a hacer en una ciudad tuviera que tener su propia estación meteorológica en cada localidad española. Parece más lógico que sea el instituto nacional de meteorología o la empresa que ya tenga esa infraestructura de estaciones montada quien comparta esa información. Por su puesto eligiendo cobrar o no por el hecho de ponerla a disposición de terceros. Pero siguiendo con el ejemplo, esa información se podría compartir de muchas formas: entrando en una página Web, descargando un fichero de un servidor Ftp, solicitándola por correo electrónico, etc. Imaginemos que la información la requieren aplicaciones desarrolladas en diferentes lenguajes de programación, tendrá cada una de ellas que procesar por su cuenta el archivo, email o Web correspondiente creando la lógica de consulta de datos para localizar la ciudad en concreto de la que quiera saber sus datos meteorológicos. Partiendo de esta problemática está claro que surge la necesidad de crear algún tipo de mecanismo para compartir información de forma instantánea a través de la Web y que esa información no se comparta de cualquier manera sino mediante el uso de estándares que permitan a todas las aplicaciones tratar por igual la información recibida. Además, se podrían crear funciones de consulta con los datos para que las aplicaciones que consuman esos datos reciban una información más concreta. Por ejemplo una función que devuelva la temperatura a partir del nombre de la ciudad, pudiendo elegir si se quiere en grados Celsius o Fahrenheit, otra función que devuelva el listado de datos de todas las ciudades de una comunidad autónoma, etc. Para cubrir este tipo de necesidades e impulsar el intercambio de información entre plataformas surgieron los servicios Web.

Por tanto, dentro del mundo de la programación podemos decir que los servicios Web son clases que residen en un servidor Web y sirven peticiones. Los clientes llaman en definitiva a una función, pasan los parámetros en caso de que sean necesarios y les es devuelto el resultado con los datos solicitados.

Aclarado el concepto de servicio Web vamos a ver cómo se lleva esto a cabo de forma técnica. Los estándares más importantes que forman el universo de un servicio Web son los siguientes:

- **SOAP**: Es el protocolo sobre el cual se realiza el intercambio de datos entre la aplicación y el servicio Web. Está basado en XML para los datos y en http para el transporte. No es el único protocolo que existe, ya que el intercambio puede realizarse mediante XML-RPC, FTP u otros, pero SOAP es el más utilizado. Sus siglas significan Protocolo Simple de Acceso a Objetos (en inglés Simple Object Access Protocol).
- **XML**: Lenguaje de marcas extensible. Es el lenguaje estándar para el intercambio de datos entre aplicaciones. La información que proporciona un servicio Web viene representada en un archivo XML. Se explica más en profundidad en el punto 3.7.
- **WSDL:** Lenguaje de descripción de servicios Web. Está basado en XML y se utiliza para describir la interfaz pública de un servicio Web. Mediante el WSDL se establecen los formatos que han de tener los mensajes de las peticiones y respuestas en la comunicación entre aplicación y servicio Web. Es aquí donde se describen también las funciones o métodos que el servicio Web pone a disposición de las aplicaciones.
- **UDDI:** Sus siglas significan Universal Description, Discovery and Integration. Es un repositorio universal de servicios. Se utiliza para dar a conocer los servicios disponibles. Si una empresa ha creado un servicio Web y quiere darlo a conocer debe publicarlo en el UDDI bajo sus normas de implementación. El UDDI permite a las empresas localizar servicios Web a través de búsquedas por diversos parámetros. Su función es similar a la de las páginas amarillas de empresas, aunque su aceptación no ha sido la que esperaban los desarrolladores y no es muy utilizado.

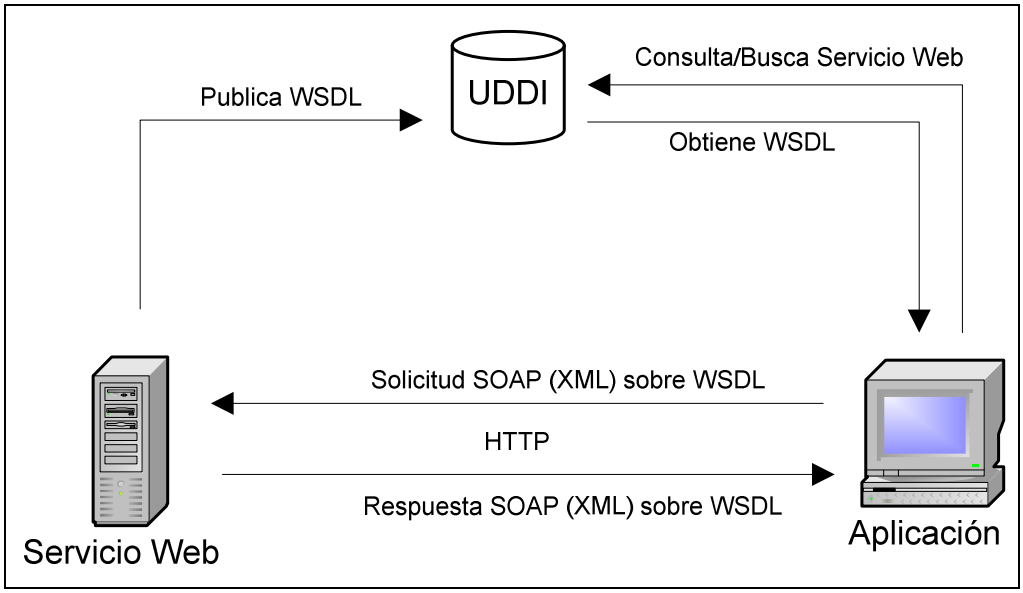


Ilustración 8: Funcionamiento de un servicio Web

En la imagen superior se muestra el funcionamiento general de un servicio Web. Lo primero que tendría que hacer la aplicación que quiera obtener información es localizar el servicio Web en el que está interesado, esto se haría consultando el repositorio UDDI que es dónde se encuentra el WSDL con la descripción de los formatos de solicitudes que deben utilizarse para conectar con el servicio Web. Aquí hay que aclarar que no hay ninguna obligatoriedad en el hecho de publicar un servicio Web en el repositorio UDDI, por tanto el WSDL puede estar publicado en una página Web tanto en intranet como en Internet conocida por la aplicación que quiera consumir el servicio. Acto seguido la aplicación al tener la definición del servicio mediante su WSDL, puede hacer peticiones SOAP respetando el formato especificado en su WSDL que enviará al servicio Web valiéndose del protocolo HTTP. El servicio Web si la petición recibida ha sido correcta enviará la respuesta con los datos solicitados en otro mensaje SOAP siguiendo el formato especificado en el WSDL.

3.7 XML

XML son las siglas de Lenguaje de marcado extensible (Extensible Markup Language) y es en esencia un formato basado en texto para representar información. Cuando hablamos de lenguaje de marcado estamos diciendo que es un lenguaje que puede definir sus propias etiquetas.

XML fue elaborado por el Consorcio para la World Wide Web (W3C) para paliar las limitaciones de HTML (HiperText Markup Language). La primera definición de XML fue la de "Sistema para definir, validar y compartir formatos de documentos en la Web". Para diseñar XML se emplearon los mejores características tanto del lenguaje SGML (Standard Generalizad Markup Language) como del HTML. La diferencia principal entre HTML y XML es que el HTML estaba encaminado a la presentación de datos, mientras que XML está orientado a los datos en sí mismos, por lo que es un lenguaje realmente útil para el software informático y para los desarrollos Web ya que se da importancia tanto a los datos como a su tratamiento y no solo a la manera de mostrarlos. HTML se concibió como un lenguaje en el que la presentación era una de sus principales preocupaciones. Se creaban las etiquetas y sus atributos válidos y este conjunto tomaba un sentido visual para cada elemento del lenguaje. Sin embargo con XML no definimos las etiquetas, sino que existen unas reglas sintácticas para llevar a cabo los documentos. Por tanto, XML es lo que llamamos un metalenguaje, un lenguaje con el cual podemos definir otros lenguajes. La razón de ser de XML no es sustituir a HTML puesto que no desempeñan la misma función, HTML representa y da forma a la información y XML además de representar la información facilita el intercambio de los datos entre distintas plataformas de manera independiente a como estén representados. Es decir, se puede utilizar en bases de datos, hojas de cálculo, editores de texto, etc. y no sólo en la Web.

A la hora de crear el lenguaje XML la W3C se fijó los siguientes objetivos:

- XML debería ser claramente utilizable en Internet.
- XML debería soportar una amplia diversidad de aplicaciones.
- XML debería ser totalmente compatible con SGML.
- El número de características opcionales en XML debería ser mínimo, idealmente cero.

- Debería ser fácil escribir programas que procesaran documentos XML.
- Los documentos XML deberían ser fáciles de entender por las personas y suficientemente claros, simples pero siguiendo las normas.
- El diseño de un documento XML debería ser rápido.
- Los documentos XML se deberían crear de manera sencilla.
- La brevedad en las marcas XML será de nula importancia.

3.7.1 DTD y XML Schema

XML es un lenguaje que permite jerarquizar y organizar la información y representar los contenidos dentro del propio archivo, y también admite reutilizar elementos del documento. Todos los documentos XML tienen más o menos la misma estructura. Puesto que el objetivo principal del lenguaje XML es facilitar el intercambio de información de forma sencilla entre entidades de orígenes diversos surge la necesidad de definir el formato y sintaxis de un documento XML. Existen varias formas de definir la estructura de un documento XML pero vamos a comentar y comparar las dos principales: el DTD y el XML Schema.

Se podría pensar que puesto que el XML Schema surgió después que el DTD será por tanto mejor, pero como en todo, siempre que existen varias formas de llevar a cabo una tarea no suele haber una forma mejor que otra para todos los casos sino que más bien dependerá de cuantas necesidades necesitemos cubrir para realizar la tarea en cuestión. En el mundo del XML esto no es una excepción. A continuación vamos a enumerar los pros y contras del DTD y el XML Schema.

DTD

En un principio solo existían los archivos de definición de tipo de documento (DTD). Puesto que XML es un subconjunto de SGML y este último utilizaba documentos DTD es lógico que en un principio el lenguaje XML heredara este mismo tipo de documentos. La finalidad de estos documentos era la de describir la sintaxis y estructura de un documento XML. Cuando se creó XML uno de los conceptos más importantes a tener en cuenta era el de “validación”, de esta forma se puede comprobar que un archivo XML es correcto validándolo contra su documento DTD.

La principal ventaja de un DTD es que proporciona una guía de cómo validar el documento XML. En un DTD se detallan como unos elementos están relacionados con otros dentro del documento, qué elementos están anidados o contenidos dentro de otros, el número de apariencias que un elemento puede tener y una breve descripción del tipo de datos.

Como hemos comentado anteriormente debido a que los ficheros DTD fueron la primera forma de validar un archivo XML estos son considerados como el estándar de validación. De hecho la gran mayoría de los documentos XML son compatibles con la validación mediante DTD.

Aunque está demostrada la utilidad de los archivos DTD para validar documentos XML estos no fueron diseñados específicamente para este cometido puesto que son anteriores a la aparición del lenguaje extensible de marcado. El problema principal que presentan los DTD son los pocos recursos que ofrece para la definición de los tipos de datos. Además están escritos de forma un tanto arcaica y extraña y ofrecen carencias a la hora de definir la anidación de elementos dentro del documento.

XML Schema

Los esquemas nacen para superar las carencias de los archivos DTD y ser consecuentes con los objetivos marcados por la W3C a la hora de crear el lenguaje XML. Desde la aparición misma del lenguaje se han ido ampliando las reglas de los DTS hasta el punto de convertirse finalmente en lo que hoy conocemos como esquemas.

La principal fortaleza de los esquemas es su potencia a la hora de definir los tipos de datos, llegando incluso a poder definir tipos complejos. Además, el hecho de que estén así mismo expresados en el mismo lenguaje XML los hacen muchos más amigables y coherentes a la hora de tratar con ellos.

Conclusión

Realmente la estructura de un documento XML se puede definir mejor y de manera más precisa mediante el uso de un esquema (Por ejemplo, elementos como

minOccurs o maxOccurs para identificar el número de ocurrencias de un elemento concreto). Puesto que nuestra aplicación dará como resultado un archivo XML y este será procesado por múltiples aplicaciones es de vital importancia que la definición de la estructura del documente se lleve a cabo de la forma más completa posible. Por tanto para el desarrollo de nuestra aplicación no sería suficiente un DTD y utilizaremos XML Schemas para dicho cometido.

3.8 Expresiones regulares

Una expresión regular (también llamada patrón) es una expresión que describe un conjunto de caracteres sin enumerar sus elementos. Las expresiones regulares constituyen un mecanismo muy potente para poder hacer manipulaciones de cadenas de texto, definir patrones de búsqueda o representar conjuntos de caracteres. Son soportadas por multitud de lenguajes de programación y son muy utilizadas en validaciones de parámetros de entrada, editores de texto, correctores ortográficos y todo tipo de aplicaciones que trabajan con texto o necesitan de la definición de un lenguaje.

Las expresiones regulares se construyen con caracteres especiales o "reservados" mediante los cuales se define el patrón o estructura de texto correspondiente. A continuación se va a hacer una breve presentación de los principales elementos utilizados en la construcción de expresiones regulares, no se pretende enumerar todos pero si dar una pequeña introducción de los más importantes:

**** Barra invertida. La barra invertida por sí misma no tiene más utilidad que representarse a sí misma pero combinada con otros caracteres resulta de gran ayuda. Su función es la de que quitar o añadir funcionalidad especial a un carácter. Por ejemplo, el carácter especial "asterisco" sirve para indicar que algo aparece ninguna o muchas veces. ¿Pero qué ocurre si queremos buscar el propio carácter asterisco en una cadena? Al ponerle delante la barra invertida se le quitar el valor de carácter especial o reservado al mismo y se obliga a que se trate como un carácter más, es lo que se conoce como "escapar" el carácter. Así pues la expresión regular * encontrará las cadenas que contengan un carácter asterisco. Otra función de la contrabarra es justamente la contraria, dotar de un significado especial a caracteres que por si mismos no tienen funciones especiales, por ejemplo, combinada con el carácter "d" indica que el carácter debe ser un dígito: \d

. El punto. El punto se utiliza para representar 1 carácter de cualquier tipo. Así pues la expresión regular c.ma dará como coincidente con las cadenas "cama", "coma", "c1ma", "c@ma" pero no con "cma".

[ ] Corchetes. Su función es la de representar clases de caracteres. Por ejemplo, la expresión regular [md]orado encontrará la palabra “dorado” y “morado”. También se utilizan junto el carácter “-“ para representar rangos de caracteres. Por ejemplo, la expresión regular [A-Fa-f0-9] encontrará cualquier número hexadecimal.

| Barra vertical. Se utiliza como operador “or”. Indica las opciones excluyentes. Por ejemplo, la expresión regular izquierda|derecha encontrará tanto la palabra “izquierda” como “derecha”.

**$** Símbolo de dólar. Representa el final de la cadena o el final de línea. Así, la expresión regular pa$ encontrará las cadenas que terminen en “pa” como “carpa” pero no las que simplemente contengan “pa” como “mapache”.

^ Acento circunflejo. Se utiliza para representar el inicio de la cadena o el principio de línea. También se utiliza combinado con los corchetes como elemento de negación. Por ejemplo, la expresión [^a-z] encontrará todo menos los caracteres en minúsculas y la expresión ^A encontrará todas las cadenas que empiecen por A.

() Paréntesis. Los paréntesis se utilizan para agrupar caracteres al igual que los corchetes pero su significado es diferente. Su utilidad principal es la de designar subexpresiones regulares que puedan ser almacenadas en variables. También se utilizan combinadas con los paréntesis para crear expresiones opcionales. Por ejemplo, la expresión regular Vamos al (cine | teatro | parque) coincidirá con las frases “Vamos al cine”, “Vamos al teatro” y “Vamos al parque”.

? Signo de interrogación. El signo de interrogación se utiliza para designar que un carácter o conjunto de caracteres si se utiliza combinado con los paréntesis es opcional. Por ejemplo, la expresión regular abc(123)? Encontrará tanto la cadena “abc123” como la cadena “abc”. También se utiliza para nombrar o etiquetar subexpresiones, por ejemplo si queremos guardar la expresión regular para un año en una variable esto se haría de la siguiente forma: (?<Año> \d\d\d\d) La expresión \d indica cualquier dígito, es una forma abreviada para el rango [0-9].

{ } Llaves. Las llaves se utilizan para indicar el número máximo y mínimo de ocurrencias de una expresión regular. Van colocados a la derecha de una expresión regular y deben encerrar 1 o 2 números separados por una coma. La expresión regular a{2} encontrará todas las palabras que contengan exactamente 2 caracteres "a" consecutivos. La expresión regular b{1,5} dará por buenas las expresiones que contentan como mínimo un carácter "b" y como máximo cinco seguidos, como "b", "bb", "bbb", "abbbb", etc.

***** Asterisco. El carácter asterisco representa al cuantificador {0, }, esto quiere decir que indica que el carácter o expresión que le precede puede aparecer ninguna o infinitas veces. Por ejemplo, la expresión regular ab2*cd encontrará las cadenas "abcd", "ab2cd", "ab22222222cd", etc.

+ Símbolo de suma. El símbolo de suma indica que el carácter o expresión que le precede aparecerá una o infinitas veces. Representa al cuantificador {1, }. Por ejemplo, la siguiente expresión: $[a-z]^{+}1$ no dará como buena la cadena "123" pero sí la "a123" o la "t1ha".

Una vez vistos los principales elementos con los que se construyen expresiones regulares a continuación se va a estudiar un ejemplo más complejo para terminar de aclarar los conceptos vistos anteriormente.

La siguiente expresión regular permite validar una dirección de correo electrónico:

$$^{[\_a-z0-9-]^{+}}(\.[\_a-z0-9-]^{+})*@[a-z0-9-]^{+}(\.[a-z0-9-]^{+})*(\.[a-z]\{2,3\})\$$$

Esta expresión regular permite determinar si una dirección de email es válida o no, según algunos criterios concretos. Vamos a explicar como lo hace.

$^{[\_a-z0-9-]^{+}}$ Esta parte nos indica que la cadena debe empezar por cualquier letra, número, el símbolo de guión bajo "_" o el símbolo de guión medio "-" . De esta forma

direcciones del tipo “¿rew@prueba.com” o “{email}@prueba.es” no serán consideras válidas por comenzar por caracteres prohibidos.

(\.[_a-z0-9-]+)*@ Esta parte de la expresión regular nos indica lo que está permitido a continuación hasta el símbolo de arroba “@”. Vemos que el punto se ha escapado con la contrabarra para que sea considerado como un carácter más, es decir, estamos permitiendo que la dirección de correo contenga un punto a continuación. Nótese que puesto que la anterior parte de la expresión regular terminaba con un “+” esto quiere decir que se dará como mínimo 1 vez, por tanto la dirección de email no podrá comenzar nunca por un punto, este deberá aparecer como muy pronto detrás de uno de los caracteres permitidos en la primera parte de la expresión regular. A continuación vemos que vuelven a estar permitidos las letras, números y guiones altos y bajos. El hecho de que todo este seguido de un carácter “*” indica que todo el conjunto se podrá presentar ninguna o muchas veces. De esta forma conseguimos que direcciones como “antonio.lopez@prueba.com” o “_.prueba@prueba.com” serán consideradas válidas pero no así otras como “.antonio@prueba.com” por comenzar por “.” o “antonio.@prueba.com“ por contener un “.” justo antes de la “@”.

[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$ Esta sección indica lo que debe contener la dirección de email desde la “@” hasta el final, lo que se indica con el símbolo “$”. Vemos que la primera parte es igual a lo que está permitido antes del símbolo de “@”,[a-z0-9-]+(\.[a-z0-9-]+)* están permitidos las letras, números guiones bajos y guiones medios. El punto está permitido siempre que se encuentre entre dos caracteres de tipo letra, número, guión medio o guión bajo. A continuación nos encontramos con la última parte (\.[a-z]{2,3})$. Indica que la dirección de email debe terminar con un punto seguido de entre 2 o 3 letras, por ejemplo “.es”, “.com”, “.fr “. De esta forma se garantiza finalmente el correcto formato de la dirección de correo electrónico.

3.9 Microsoft .NET Framework

Microsoft .NET Framework es la plataforma de software de Microsoft para construir y ejecutar aplicaciones principalmente dentro de la familia de sistemas operativos Windows. Permite crear potentes aplicaciones con gran funcionalidad y una excelente experiencia de usuario. Incluye una gran librería de clases y proporciona interoperabilidad entre distintos lenguajes de programación. El Framework de .NET está compuesto principalmente por 3 elementos:

- Lenguajes de programación. .NET soporta multitud de lenguajes de programación como por ejemplo: C#, Visual Basic, Delphi, etc lo que facilita la integración entre equipos de desarrollo.
- Common Language Runtime. Es el corazón de .NET. Es el motor de ejecución que se encarga de garantizar la seguridad y el comportamiento de los programas escritos en .NET gestionando el área de memoria, la seguridad y el manejo de excepciones entre otras tareas.
- Biblioteca de Clases. Aquí reside realmente la potencia de .NET Framework. Se trata de una amplia colección orientada a objetos de clases que pueden ser utilizadas por todos los lenguajes soportados por .NET. Proporciona funcionalidades para desarrollar todo tipo de aplicaciones empresariales. Algunas de estas funcionalidades son la gestión de ficheros, manejo de hilos, acceso a base de datos, tratamiento de XML, serialización, criptografía, expresiones regulares, controles de usuario, etc.

Una de las mayores ventajas de .NET Framework con respecto a otras plataformas de desarrollo como JAVA es la potencia y rapidez de desarrollo que proporciona. .NET se caracteriza por facilitar mucho la vida al desarrollador gracias a la amplitud y diversidad de su biblioteca de clases. De hecho uno de los objetivos principales de Microsoft al comenzar el proyecto .NET fue el de simplificar la creación de software complejo. Además es el entorno ideal para poder integrarse con aplicaciones de Microsoft ampliamente utilizadas en el mundo empresarial como puede ser el paquete de ofimática Office. En España está muy extendido su uso para crear aplicaciones en las administraciones públicas.

3.10 Librería Bouncy Castle

La librería Bouncy Castle es una colección de APIs utilizadas en criptografía. Implementa algoritmos criptográficos así como utilidades relacionadas con la firma electrónica y los certificados digitales. Está publicada baja licencia de código libre del MIT y tiene implementaciones tanto para JAVA como para .NET.

En el aspecto criptográfico los frameworks tradicionales resultan insuficientes a la hora de llevar a cabo tareas más complejas. La librería Bouncy Castle aporta muchas funciones y herramientas que facilitan la tarea de creación de soluciones software que tengan que abordar aspectos criptográficos avanzados.

Bouncy Castle sufre actualizaciones y mejoras constantes gracias a su condición de código abierto y por tanto de proyecto colaborativo. La última versión publicada para .NET en lenguaje C# es de Abril de 2011.

3.11 Microsoft Enterprise Library

La librería Enterprise Library es una librería de Microsoft que proporciona herramientas para llevar a cabo funcionalidades típicas que pueden ser necesarias a la hora de crear un software empresarial. Típicamente se ha venido observando como en la mayor parte de aplicaciones se utilizan una serie de funcionalidades que se repiten en la mayoría de los proyectos software. Funcionalidades como la capa de negocio de acceso a datos, la gestión de excepciones centralizada, la escritura de un archivo Log, validaciones de parámetros. Microsoft creó Enterprise Library como un complemento al Framework .NET que facilita y encapsula la implementación de muchas de estas tareas, reduciendo así el tiempo de desarrollo de software empresarial.

Microsoft Enterprise Library se compone de varios bloques de aplicación, cada uno de los cuales provee funcionalidades distintas para su implementación en el software empresarial:

- **Caching Block**. Permite a los desarrolladores añadir una caché local a sus aplicaciones.
- **Cryptography Block**. Simplifica el desarrollo de funciones criptográficas-
- **Data Access Block**. Provee funcionalidad para acceso a base de datos.
- **Exception Handling Block**. Manejador de excepciones
- **Logging Block**. Funcionalidad para la gestión de Logs.
- **Policy Injection Block**. Funcionalidad para establecer política y reglas a determinadas clases
- **Security Block**. Ayuda a implementar funcionalidades relacionadas con la autorización, control de acceso y seguridad de la aplicación.
- **Validation Block**. Incluye una seria de clases con validadores.
- **Unity.** Contenedor de inyección de dependencia ligero y extensible.

4 Análisis del sistema

4.1 Descripción del catálogo de requisitos

A continuación se muestran los requisitos extraídos de las reuniones con el cliente. Primeramente se va a proceder a describir brevemente el contenido y la estructura del catálogo de requisitos:

- Requisitos de usuario. Son requisitos enfocados a lo que puede o no puede hacer un usuario en la aplicación. Se dividen en dos grupos:
  - Requisitos de capacidad. Descripción de las funcionalidades de la aplicación.
  - Requisitos de restricción. Descripción de las funcionalidades que el usuario no puede llevar a cabo en la aplicación.
- Requisitos de software. Son requisitos más técnicos enfocados al proceso de creación de la aplicación. Se dividen en dos grupos:
  - Requisitos funcionales. Descripción del comportamiento interno de la aplicación.
  - Requisitos no funcionales. Requisitos enfocados al diseño y la implementación de la aplicación. Se considerarán requisitos no funcionales los enumerados a continuación:
    - Requisitos de plataforma
    - Requisitos de interfaz
    - Requisitos de escalabilidad
    - Requisitos de rendimiento
    - Requisitos de disponibilidad
    - Requisitos de seguridad
    - Requisitos de extensibilidad

4.2 Requisitos de usuario

4.2.1 Requisitos de capacidad

Identificador	RU-C001		
Descripción	El sistema será capaz de validar y parsear los datos de certificados digitales en formato x509.		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C002		
Descripción	El sistema debe devolver los datos que se le soliciten siempre en formato XML		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C003
Descripción	El sistema constará de una base de datos en la que se insertará un registro para cada consulta realizada al mismo. Esto servirá para llevar

	un control del nivel de utilización del servicio.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C004		
Descripción	Las aplicaciones que utilicen el sistema MPC deberán poder hacer peticiones al sistema de forma sencilla.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C005		
Descripción	El sistema debe poder recibir y procesar peticiones de aplicaciones ubicadas tanto en la intranet como en la extranet del Ministerio.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	

Fuente	Cliente

Identificador	RU-C006		
Descripción	Para que una aplicación pueda utilizar el sistema el código de la misma debe estar dado de alta en la base de datos de la aplicación. Esto debe comprobarse para cada petición recibida.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C007		
Descripción	El sistema será capaz de procesar información de los 4 tipos de certificados utilizados por las aplicaciones internas del Ministerio: ✓ Fnmt Clase 2 CA ✓ Fnmt Ap ✓ Fnmt Ape ✓ DNIe		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C008		
Descripción	Existirán como mínimo dos formas de identificar el emisor y el tipo de certificado de la petición: ✓ Mediante Verificación: Se deben usar los certificados raíces para identificar el certificado emisor. ✓ Mediante Parseo: Se parseará el emisor del certificado y se comprobará que es uno de los posibles valores aceptados.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C009
Descripción	Los datos **comunes** a todos los tipos de certificados que el sistema debe parsear serán los siguientes: ✓ *Número de serie* ✓ *Emisor* ✓ *Asunto* ✓ *Periodo de validez* ✓ *Entidad certificadora* ✓ *Datos entidad certificadora* ✓ *Unidad entidad certificadora* ✓ *Tipo entidad certificadora* ✓ *Nombre completo*

	✓ *Primer apellido* ✓ *Segundo apellido* ✓ *Nombre* ✓ *Nif* ✓ *País* ✓ *Email*		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C010		
Descripción	El sistema deberá devolver a parte de los campos comunes establecidos en el requisito RU-C009, los siguientes campos derivados del análisis de las políticas de certificación de los certificados: ✓ *Tipo persona*. Solo podrá tener 2 valores posibles: Física o Jurídica. ✓ *Entidad jurídica*. Solo en caso de que Tipo persona sea Jurídica ✓ *Cif*. Solo en caso de que Tipo persona sea Jurídica ✓ *Propósito*. Solo en caso de que el certificado sea de tipo DNIe		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	

Fuente	Cliente

Identificador	RU-C011		
Descripción	El sistema deberá devolver a parte de los campos establecidos en los requisitos RU-C009 y RU-C010 , los siguientes campos para los certificados de empleado público: ✓ *Entidad suscriptora*. ✓ *Nif entidad suscriptora*. ✓ *Número de identificación personal*. ✓ *Unidad organizativa*. ✓ *Cargo*. ✓ *Situación laboral*.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C012
Descripción	En los casos en los que se solicite la validación del certificado el sistema deberá realizar una validación simple basada en los siguientes criterios: ✓ Caducidad ✓ Inicio periodo validez ✓ Confianza del emisor

	✓ Validez de la firma ✓ Extensiones válidas		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C013		
Descripción	El sistema contará con un sistema de log en un fichero de texto dónde se escribirán los errores producidos durante la ejecución.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RU-C014		
Descripción	El sistema deberá tener una clasificación de errores definidos de forma que se facilite la localización y solución de posibles incidencias producidas.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐

Estabilidad	Alta ☑	Baja ☐
Fuente	Cliente	

4.2.2 Requisitos de capacidad inversos

No aplican.

4.2.3 Requisitos de restricción

Identificador	RU-R001		
Descripción	El sistema no procesará certificados de sello o sede ni tampoco certificados raíces o intermedios. Solo serán válidos certificados vinculados a personas o entidades jurídicas. El sistema deberá por tanto ser capaz de identificar para los tipos soportados indicados anteriormente los perfiles de certificados admitidos.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.3 Modelo de casos de uso

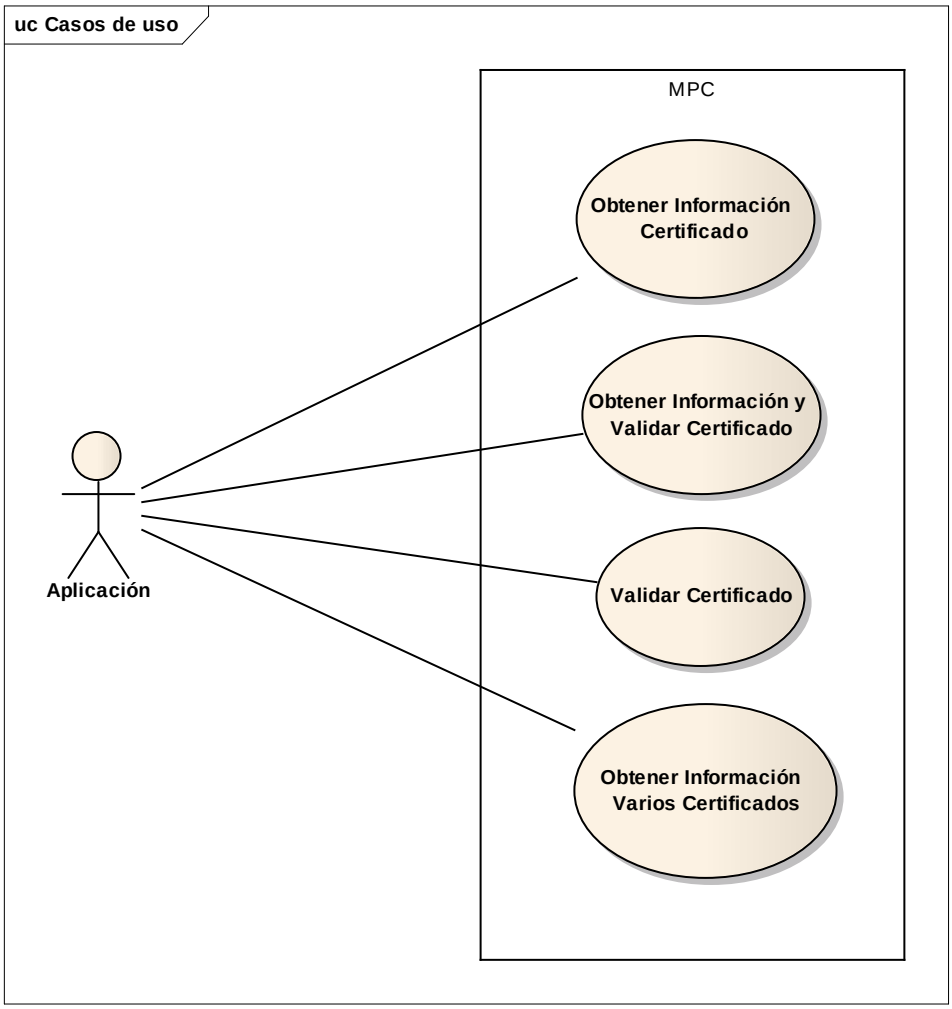


Ilustración 9 - Diagrama de casos de uso del sistema

4.3.1 Casos de uso

Nombre	Obtener información certificado
Actores	Aplicación externa que consume el servicio web MPC
Descripción	El servicio web MPC devolverá un XML con los datos parseados del certificado digital proporcionado por la aplicación que lo consume.
Precondiciones	o Se debe llamar al método ObtenerInformacion del servicio web MPC. o Debe pasarse como parámetro el identificador de la aplicación.

	o El identificador de la aplicación debe estar dado de alta en la base de datos de MPC. o Debe pasarse como parámetro un certificado digital. o El certificado debe ser de tipo X509 y estar codificado en base64. o El certificado debe ser de un tipo soportado por el sistema MPC. o Debe pasarse como parámetro el tipo de validación. o Para este caso de uso no se quiere validación por tanto el parámetro tipo de validación deberá ser 0.
Postcondiciones	Se devuelve un Xml a través del protocolo Soap con los datos parseados del certificado.
Escenario básico	o Servicio Web MPC iniciado. o Base de datos MPC accesible. o Aplicación externa con acceso al servicio Web.

Nombre	Obtener información y validar certificado
Actores	Aplicación externa que consume el servicio web MPC
Descripción	El servicio web MPC devolverá un XML con los datos parseados del certificado digital proporcionado por la aplicación que lo consume y con los datos de la validación del mismo.
Precondiciones	o Se debe llamar al método ObtenerInformacion del servicio web MPC. o Debe pasarse como parámetro el identificador de la aplicación. o El identificador de la aplicación debe estar dado de alta

	en la base de datos de MPC. o Debe pasarse como parámetro un certificado digital. o El certificado debe ser de tipo X509 y estar codificado en base64. o El certificado debe ser de un tipo soportado por el sistema MPC. o Debe pasarse como parámetro el tipo de validación. o Para este caso de uso se quiere validación por tanto el parámetro tipo de validación deberá ser 1.
Postcondiciones	Se devuelve un Xml a través del protocolo Soap con los datos parseados del certificado y el resultado de la validación.
Escenario básico	o Servicio Web MPC iniciado. o Base de datos MPC accesible. o Aplicación externa con acceso al servicio Web.

Nombre	Validar certificado
Actores	Aplicación externa que consume el servicio web MPC
Descripción	El servicio web MPC devolverá un XML con un 1 en caso de que el certificado pasado como parámetro por la aplicación que lo consume sea válido o con un 0 en caso de que no lo sea.
Precondiciones	o Se debe llamar al método EsValido del servicio web MPC. o Debe pasarse como parámetro el identificador de la aplicación. o El identificador de la aplicación debe estar dado de alta en la base de datos de MPC. o Debe pasarse como parámetro un certificado digital.

	o El certificado debe ser de tipo X509 y estar codificado en base64. o El certificado debe ser de un tipo soportado por el sistema MPC. o Debe pasarse como parámetro el tipo de validación. o Para este caso de uso se quiere validación por tanto el parámetro tipo de validación deberá ser 1.
Postcondiciones	Se devuelve un Xml a través del protocolo Soap con el valor booleano que representa el resultado de la validación del certificado.
Escenario básico	o Servicio Web MPC iniciado. o Base de datos MPC accesible. o Aplicación externa con acceso al servicio Web.

Nombre	Obtener información varios certificados
Actores	Aplicación externa que consume el servicio web MPC
Descripción	El servicio web MPC devolverá un XML con los datos parseados de los certificados digitales proporcionados por la aplicación que lo consume.
Precondiciones	o Se debe llamar al método ObtenerInformacionVariosCertificados del servicio web MPC. o Debe pasarse como parámetro el identificador de la aplicación. o El identificador de la aplicación debe estar dado de alta en la base de datos de MPC. o Debe pasarse como parámetro uno o varios certificados

	digitales. o Los certificados deben ser de tipo X509 y estar codificados en base64. o Los certificados deben ser de un tipo soportado por el sistema MPC.
Postcondiciones	Se devuelve un Xml a través del protocolo Soap con los datos parseados de los certificados.
Escenario básico	o Servicio Web MPC iniciado. o Base de datos MPC accesible. o Aplicación externa con acceso al servicio Web.

4.4 Requisitos de software

4.4.1 Requisitos Funcionales

Identificador	RS-F001		
Descripción	El sistema constará de 3 métodos: ✓ Validar Certificado ✓ Obtener Información ✓ Obtener Información de un grupo de certificados		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RS-F002		
Descripción	El sistema recibirá los siguientes parámetros: ✓ *Validar Certificado*: Recibirá un string con un certificado x509 codificado en base 64 para validar, un entero que indicará el modo de validación y otro entero con el identificador de la aplicación que usará el servicio. ✓ *Obtener Información*: Recibirá un string con un certificado x509 codificado en base 64 para parsear, un entero que indicará el modo de validación y otro entero con el identificador de la aplicación que usará el servicio. ✓ *Obtener información varios certificados*: Recibirá un array de n string con certificados X509 codificados en base 64 y un entero con el identificador de la aplicación que usará el servicio. La codificación de los certificados X509 en base64 surge de la necesidad de expresar los parámetros como tipos primitivos o arrays de tipos primitivos. Se elegirá esta codificación por ser una de las más utilizadas lo que simplificará el manejo de la información.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RS-F003
Descripción	El sistema deberá permitir cambiar mediante un archivo de configuración el método utilizado para identificar el emisor y el tipo de certificado procesado.

Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RS-C004		
Descripción	El sistema deberá devolver para el método ObtenerInformacion en caso de haber elegido validación el resultado de la misma en una región separada del resto de campos.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RS-C005		
Descripción	El nivel de detalle del error tanto en el log como en la salida debe ser parametrizable y configurable desde un archivo de configuración.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	

Fuente	Cliente

Identificador	RS-F003		
Descripción	El xml de salida debe estar bien formado y ser válido acorde con su esquema XSD.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.4.2 Requisitos No Funcionales

4.4.2.1 Plataforma

Identificador	RS-NF-P001		
Descripción	El sistema deberá ejecutarse en la intranet del Ministerio, bajo un servidor Internet Information Server (IIS 6 o superior).		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.4.2.2 Interfaz

Identificador	RS-NF-I001		
Descripción	Al tratarse de un servicio web no se requiere de ninguna interfaz específica. Aun así sería interesante disponer de una interfaz web desde la que poder probar los métodos directamente sin tener que utilizar una aplicación.		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☑
Prioridad	Alta ☐	Media ☑	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.4.2.3 Escalabilidad

Identificador	RS-NF-ES001		
Descripción	El sistema debe estar diseñado de forma modular para poder ser fácilmente mantenible y escalable. Cada uno de los 4 tipos principales de certificados debe tener una librería independiente para facilitar posibles modificaciones futuras en el parseo o la validación relacionadas con cambios en las políticas de certificación de los certificados.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RS-NF-ES002		
Descripción	Se quiere que el sistema implemente el patrón MVC (Modelo Vista Controlador) para separar interfaz, lógica de negocio y datos.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.4.2.4 Rendimiento

Identificador	RS-NF-R001		
Descripción	Se intentará dentro de los medios disponibles que el servicio web responda lo más rápidamente a las peticiones solicitadas por las aplicaciones que lo consuman.		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.4.2.5 Disponibilidad

Identificador	RS-NF-D001
Descripción	El sistema deberá tener una disponibilidad alta ya que será utilizado por numerosas aplicaciones. Se utilizará un balanceador de carga y

	dos nodos principales para maximizar la disponibilidad.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.4.2.6 Seguridad

Identificador	RS-NF-S001		
Descripción	El sistema MPC estará ubicado en la intranet, siendo imposible su acceso a través de Internet.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

Identificador	RS-NF-S002		
Descripción	El sistema MPC solo podrá ser utilizado por aplicaciones autorizadas. Para que una aplicación pueda utilizar el sistema MPC tendrá que tener dada de alta en su base de datos el identificador de la aplicación. En caso contrario se le mostrará un mensaje de error en la respuesta.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐

Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.4.2.7 Extensibilidad

Identificador	RS-NF-EX001		
Descripción	El sistema debe ser fácilmente extensible ya que es probable que en el futuro se añadan nuevos tipos de certificados y de validaciones (Ejemplo: Validación por OCSP).		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Alta ☑	Baja ☐	
Fuente	Cliente		

4.5 Diagramas de secuencia

A continuación se muestran los diagramas de secuencia correspondientes a los 4 casos de uso de la aplicación. Puesto que existen clases distintas para cada tipo de certificado y por tanto muchos caminos diferentes a tomar, se va a optar por elegir como ejemplo para todos los casos de uso la consulta de un certificado de tipo FNMT de tipo clase 2 CA de persona física. Así pues el flujo de ejecución en cada diagrama de secuencia asumirá que se ha recibido como parámetro este tipo de certificado y que todos los demás parámetros (modo validación e identificador de la aplicación) son correctos, quedando excluidos del diagrama los casos en que se produzcan excepciones. Para todos los casos de uso salvo para el último se dará por hecho que en el archivo de configuración del sistema está configura la identificación de certificados mediante verificación.

También hay que mencionar que el caso de uso para el que se mostrarán los diagramas completos del mismo será el más complejo: Obtener Información y Validar Certificado. Para los demás casos de uso puesto que son subconjuntos o variantes de este solo se mostrará lo que les caracterice en exclusiva con el fin de evitar la redundancia de información en los diagramas representados.

4.5.1 Diagrama de Obtener Información y Validar certificado

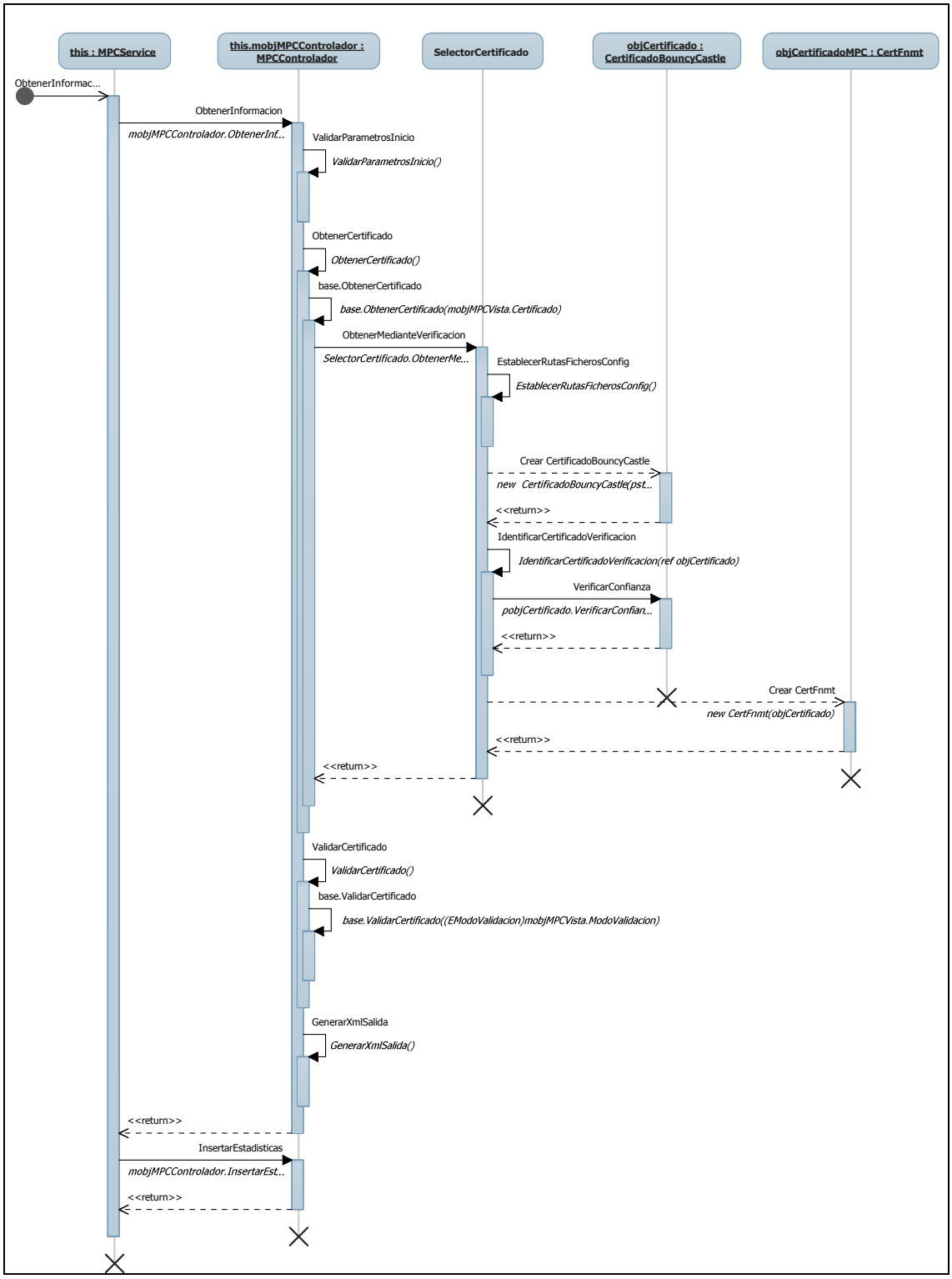


Ilustración 10: Diagrama de secuencia de Obtener Información y Validar certificado I

Debido a su tamaño se ha separado el diagrama en varios. El diagrama siguiente muestra la parte concerniente al constructor de la clase **CertFnmt** (Crear CertFnmt), que es dónde realmente se realiza el parseo del certificado digital:

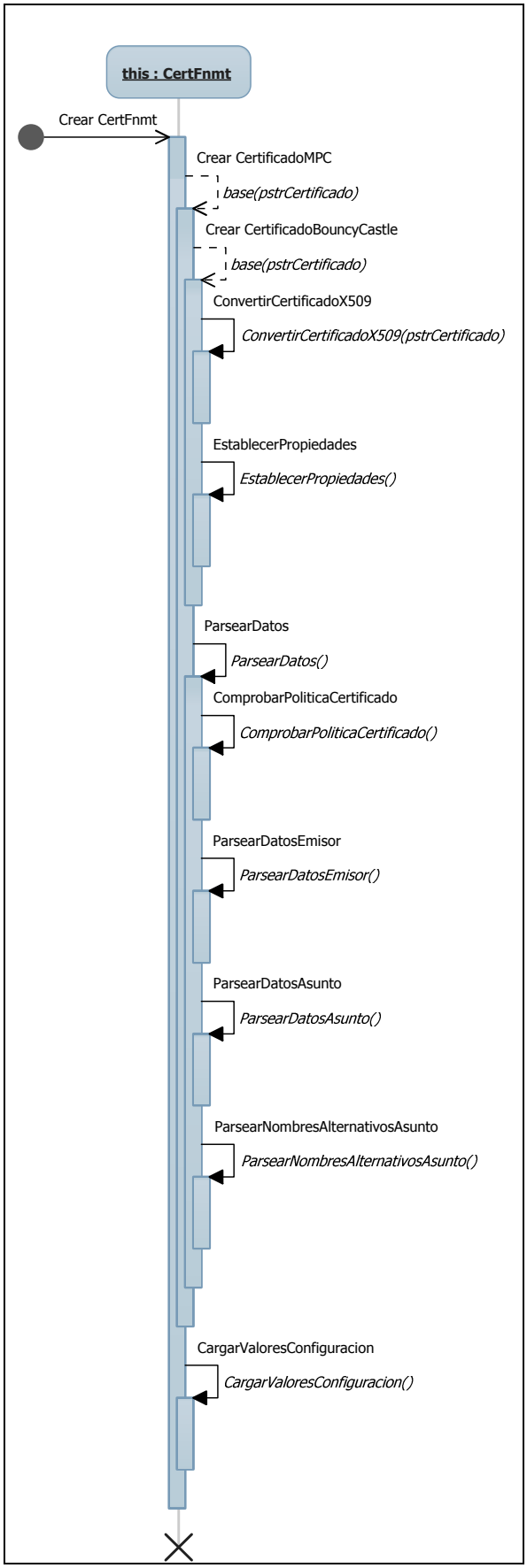


Ilustración 11: Diagrama de secuencia de Obtener Información y Validar certificado II

El método **ParsearDatos()** como puede comprobarse realiza llamadas a otros 4 métodos. A continuación se muestran por separado debido a su relevancia dentro de la aplicación los diagramas de cada uno de esos métodos

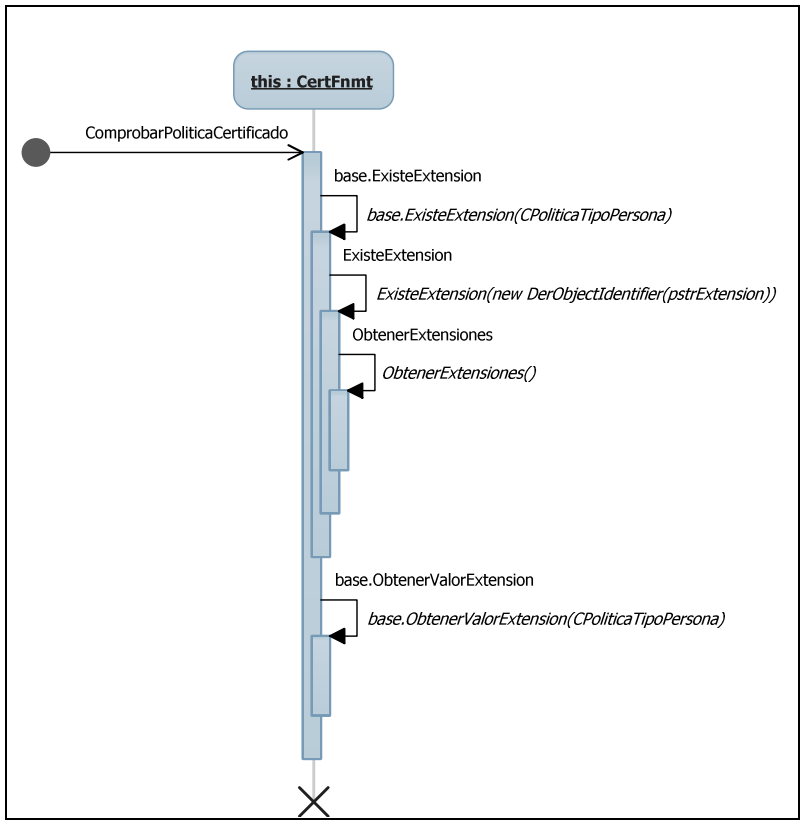


Ilustración 12: Diagrama de secuencia del método ComprobarPoliticaCertificado()

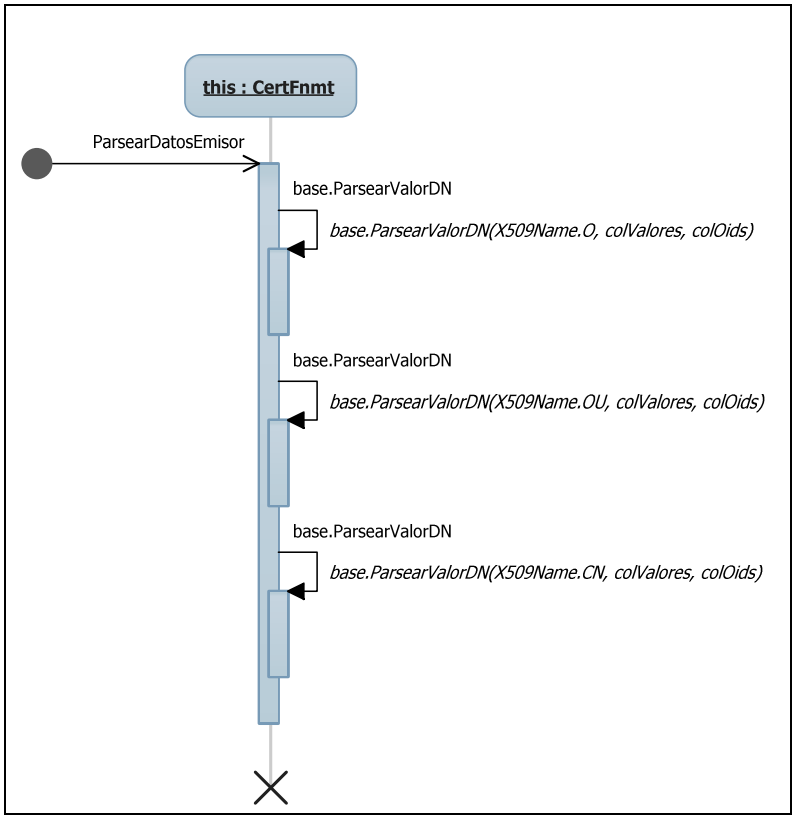


Ilustración 13: Diagrama de secuencia del método ParsearDatosEmisor()

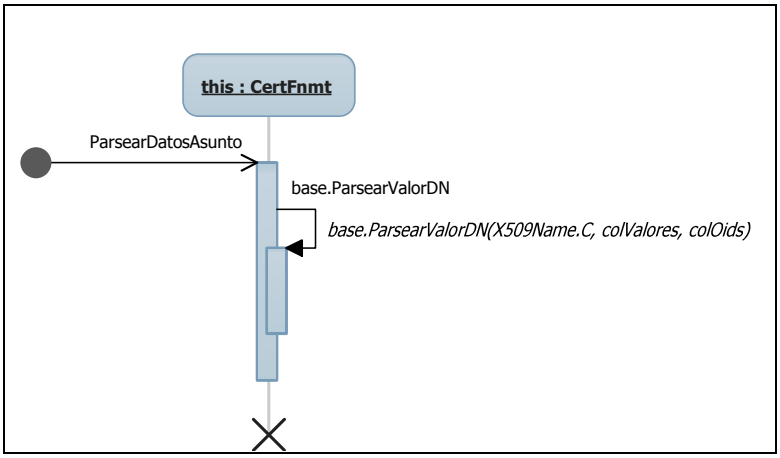


Ilustración 14: Diagrama de secuencia del método ParsearDatosAsunto()

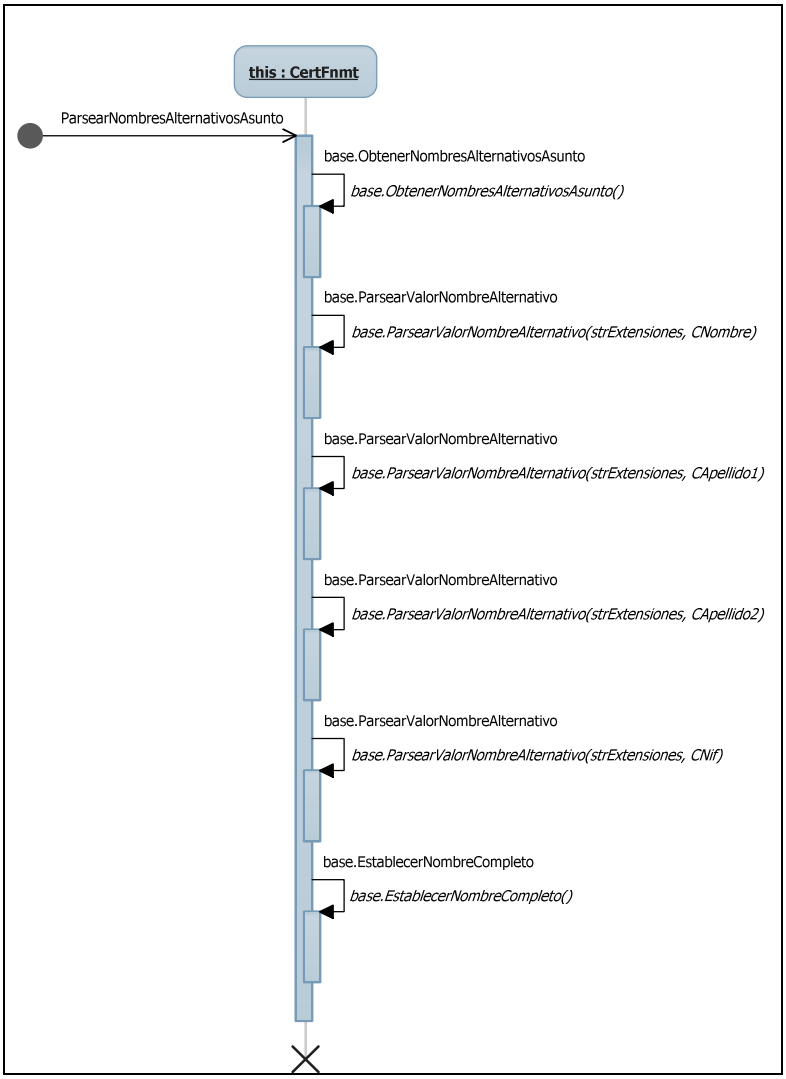


Ilustración 15: Diagrama de secuencia del método ParsearNombresAlternativosAsunto()

A continuación se muestra el diagrama concerniente al método **ValidarCertificado()** del controlador que es el encargado de comprobar la validez del certificado:

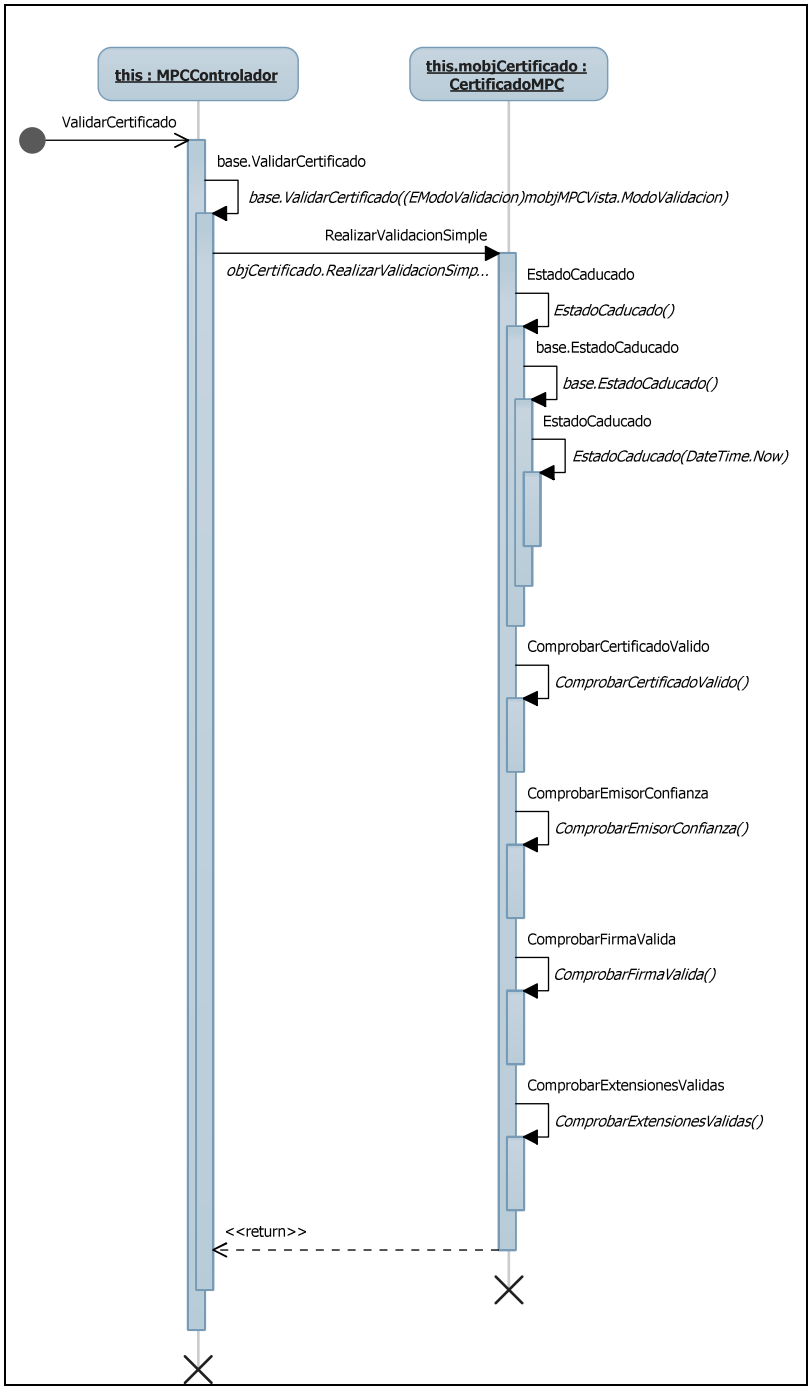


Ilustración 16: Diagrama de secuencia del método ValidarCertificado()

A continuación se muestra el diagrama referente al método **GeneralXmlSalida()**, que es el encargado de dar forma al xml que devolverá el sistema MPC como salida:

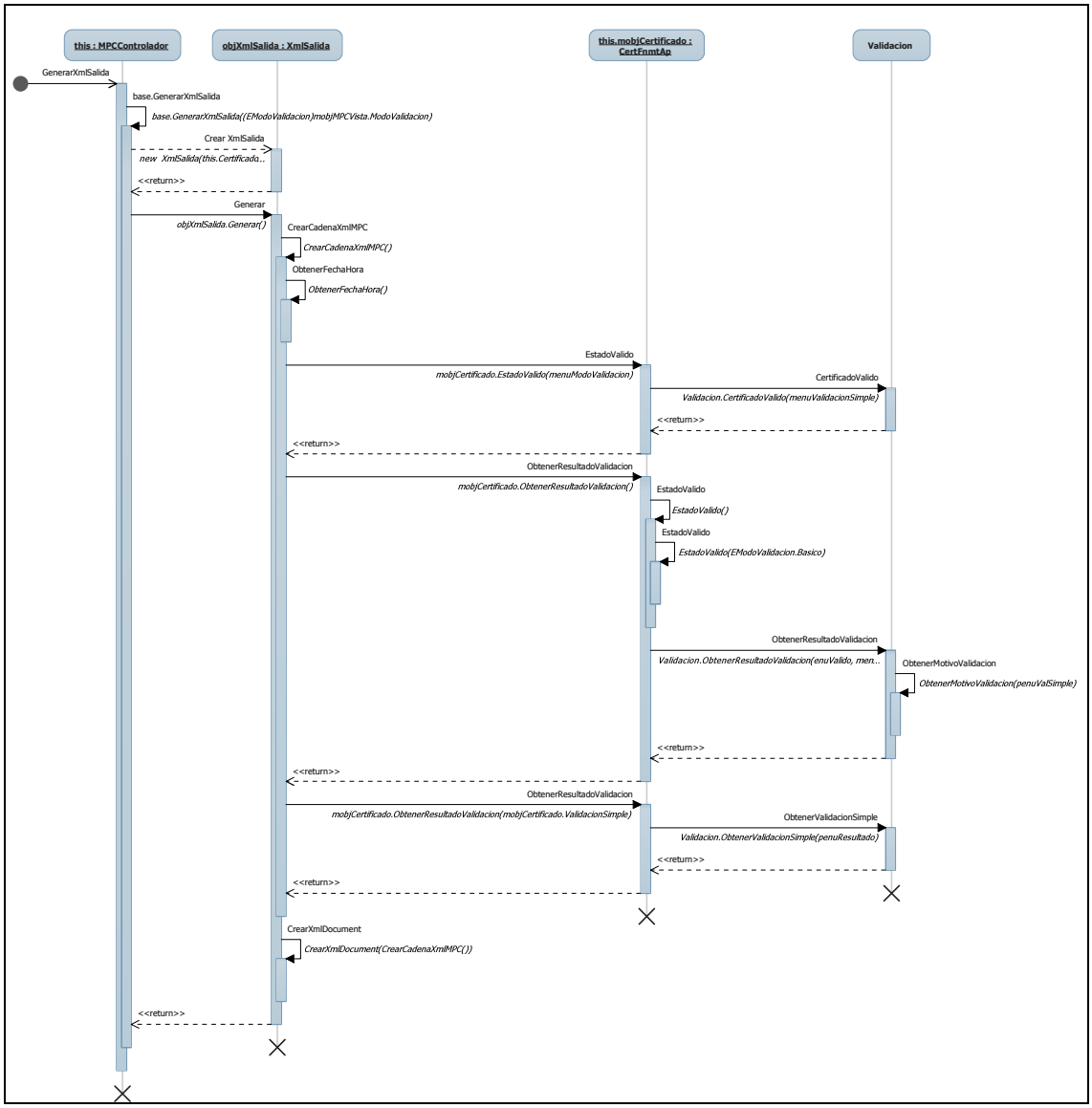


Ilustración 17: Diagrama de secuencia del método GenerarXmlSalida()

A continuación y para finalizar el diagrama de secuencia del caso de uso Obtener Información y Validación Certificado se muestra el diagrama correspondiente al método **InsertarEstadisticas()**, que es el encargado de grabar los datos de la consulta realizada en base de datos:

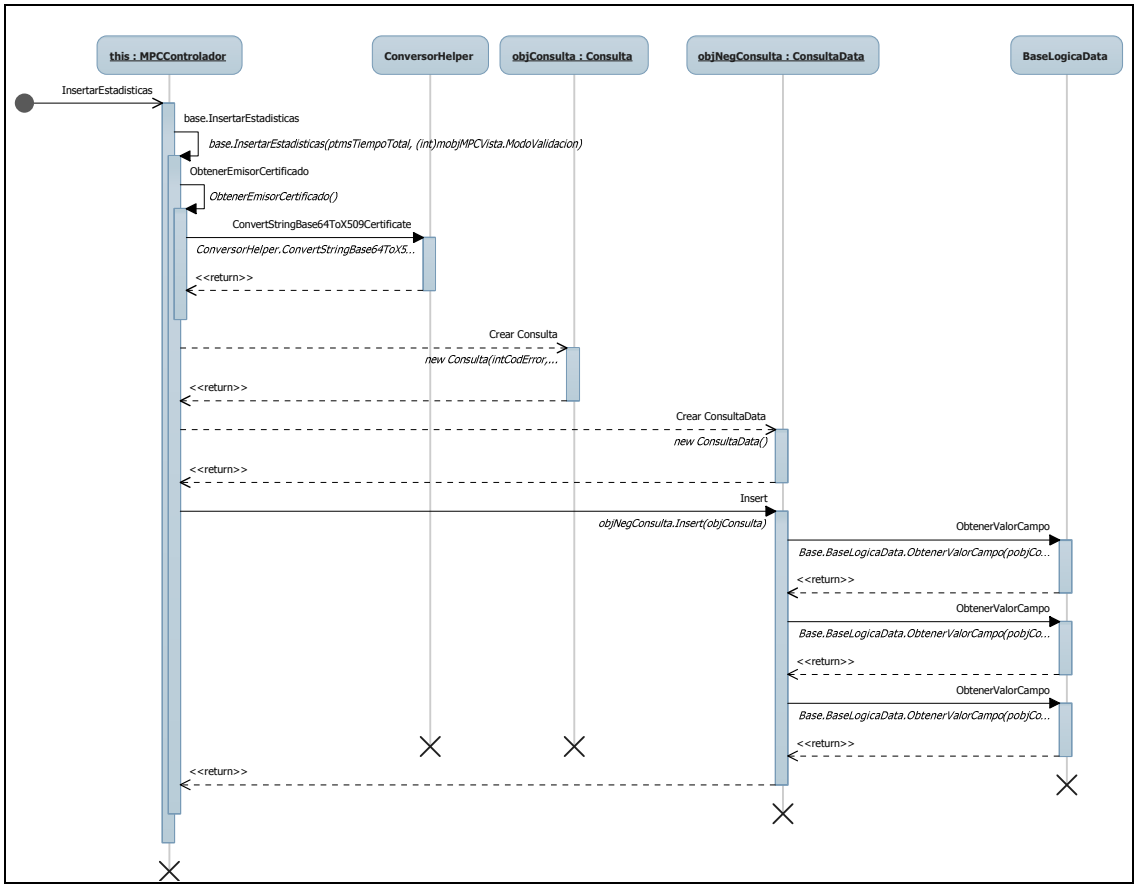


Ilustración 18: Diagrama de secuencia del método InsertarEstadisticas()

4.5.2 Diagrama de Obtener Información Certificado

Como es lógico suponer, el diagrama de secuencia de Obtener Información Certificado está contenido dentro del diagrama anteriormente mostrado en el punto 4.5.1 puesto que es un subconjunto de aquel. El flujo de ejecución es el mismo con la salvedad de que no se ejecuta el método ValidarCertificado() ni se tiene en cuenta dentro GenerarlXmlSalida(), por tanto solo se va a mostrar el diagrama principal para este caso de uso sin entrar en más detalle:

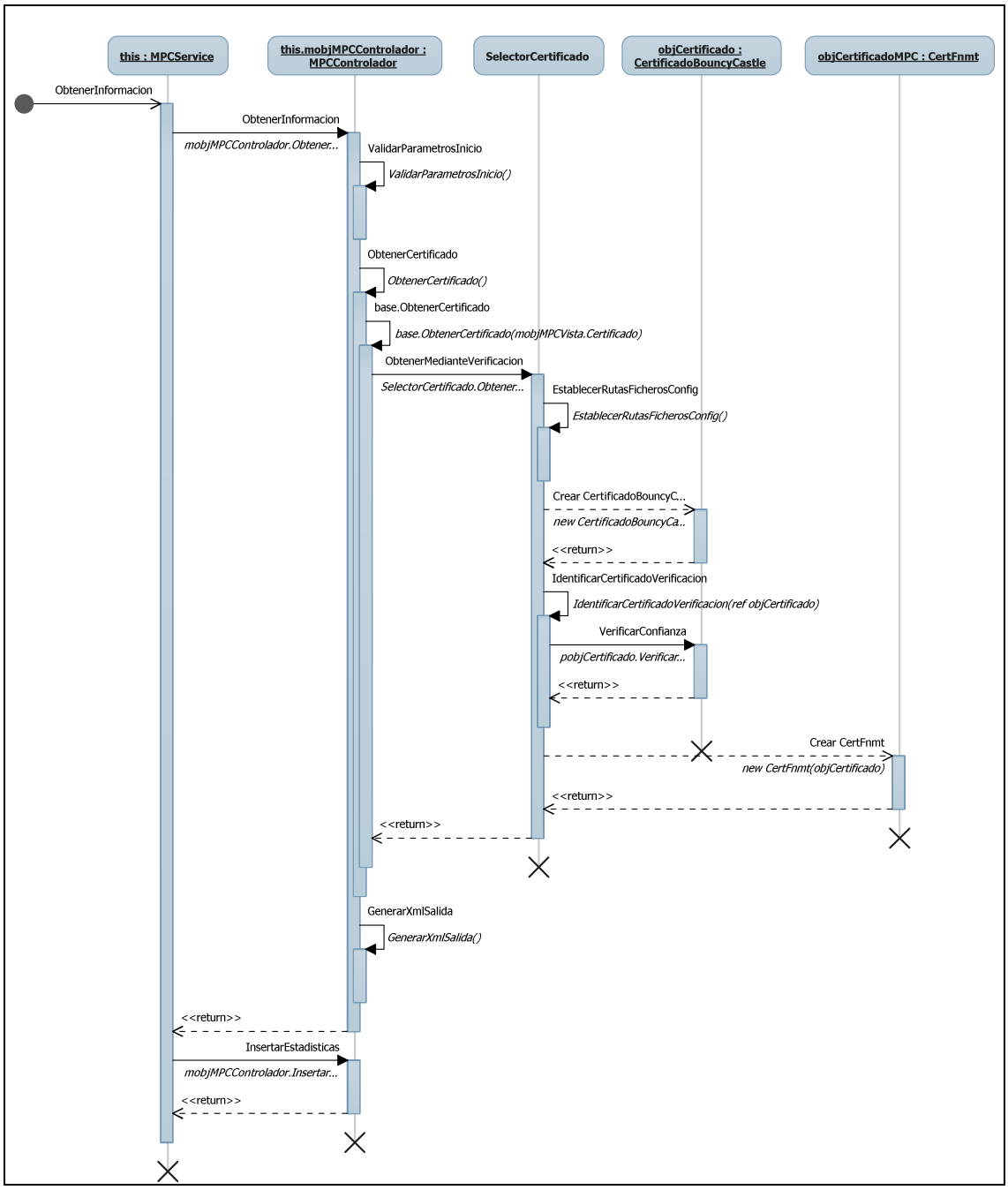


Ilustración 19: Diagrama de secuencia de Obtener Información Certificado

4.5.3 Diagrama de Validar Certificado

El diagrama de secuencia para el caso de uso Validar Certificado es muy parecido al visto en el punto 4.5.1. Para este caso de uso también se parsean los datos del certificado puesto que son necesarios para validar el mismo. La diferencia reside en el xml de salida, que simplemente será un 1 o un 0 según el resultado de la validación del mismo. A continuación se muestra el diagrama general para el caso de uso Validar Certificado recogido en el método **EsValido()** de la aplicación:

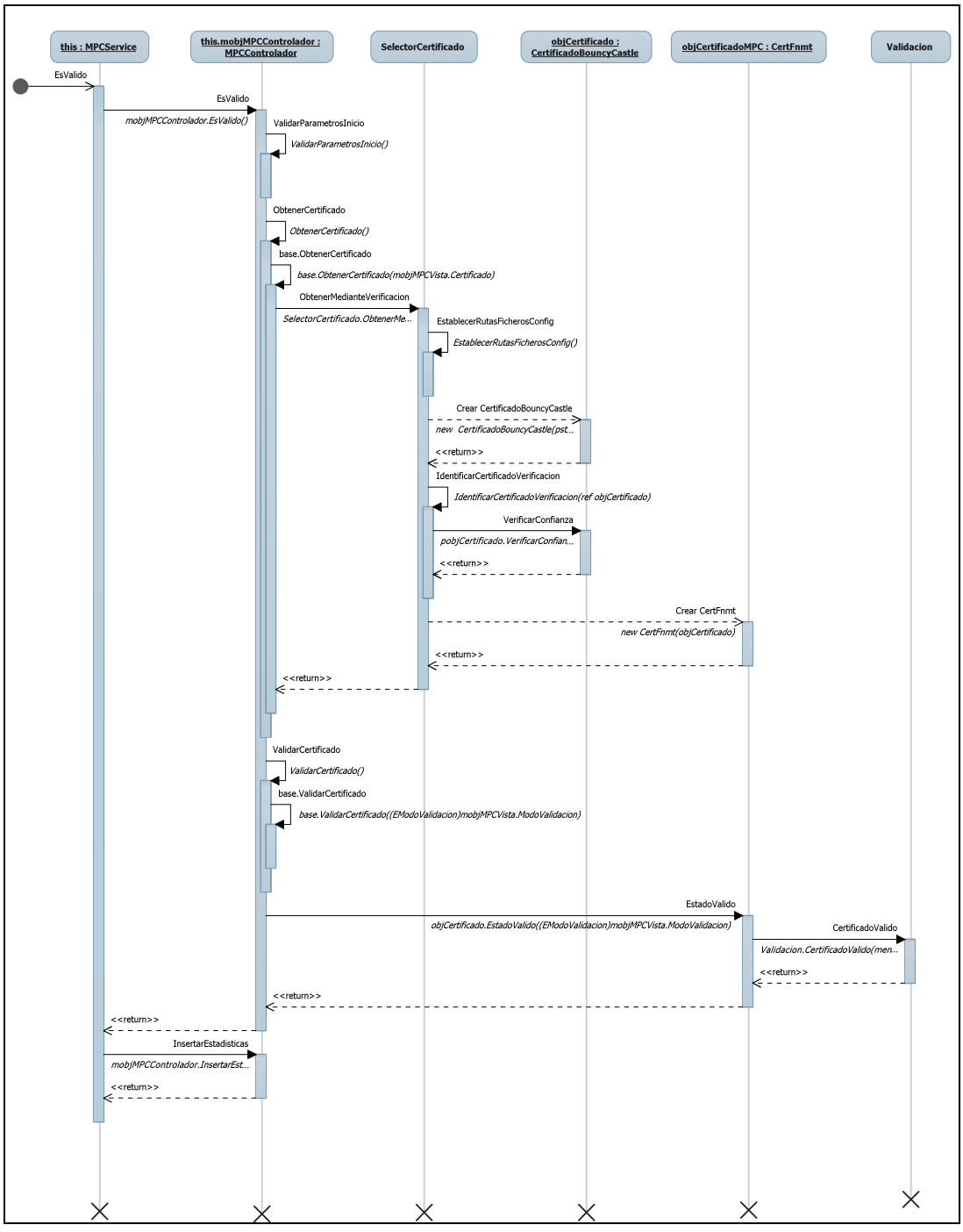


Ilustración 20: Diagrama de secuencia de Validar Certificado

4.5.4 Diagrama de Obtener Información Varios Certificados

Este último diagrama representa el caso de uso de Obtención de información para varios certificados. En este caso de uso no se validan los certificados, solo se parsean. Para este diagrama se va a suponer el parseo de un certificado de tipo FNMT y otro de tipo FNMTAp. La sección marcada como bucle se ejecutará 2 veces. En la primera iteración se creará un objeto de tipo CertFnmt y en la segunda uno de tipo CertFnmtAp. Como novedad en este último diagrama se realiza la identificación de los certificados mediante parseo en lugar de por verificación como se ha mostrado en los casos anteriores:

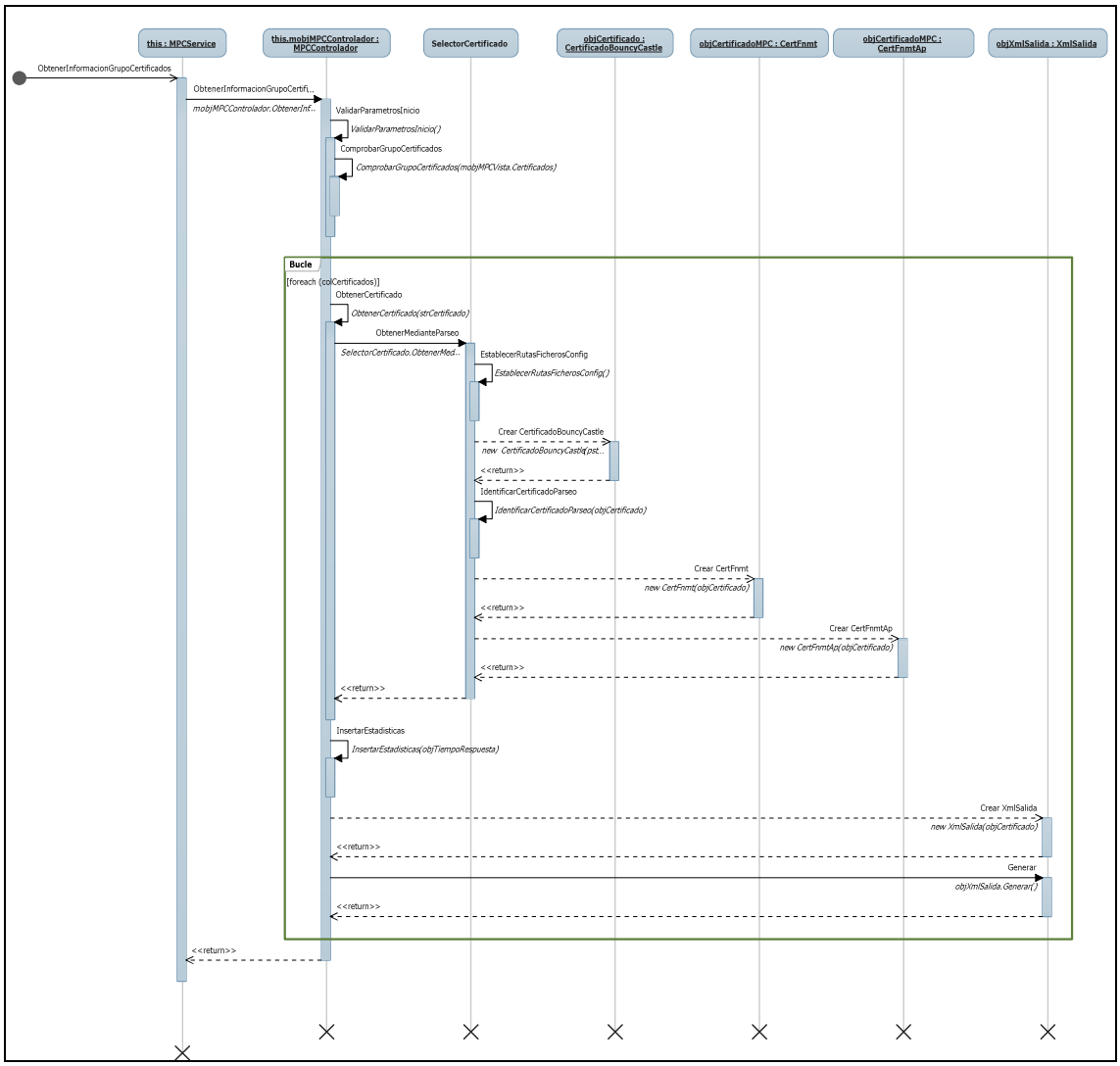


Ilustración 21: Diagrama de secuencia de Obtener Información de varios certificados

4.5.5 Arquitectura física del sistema

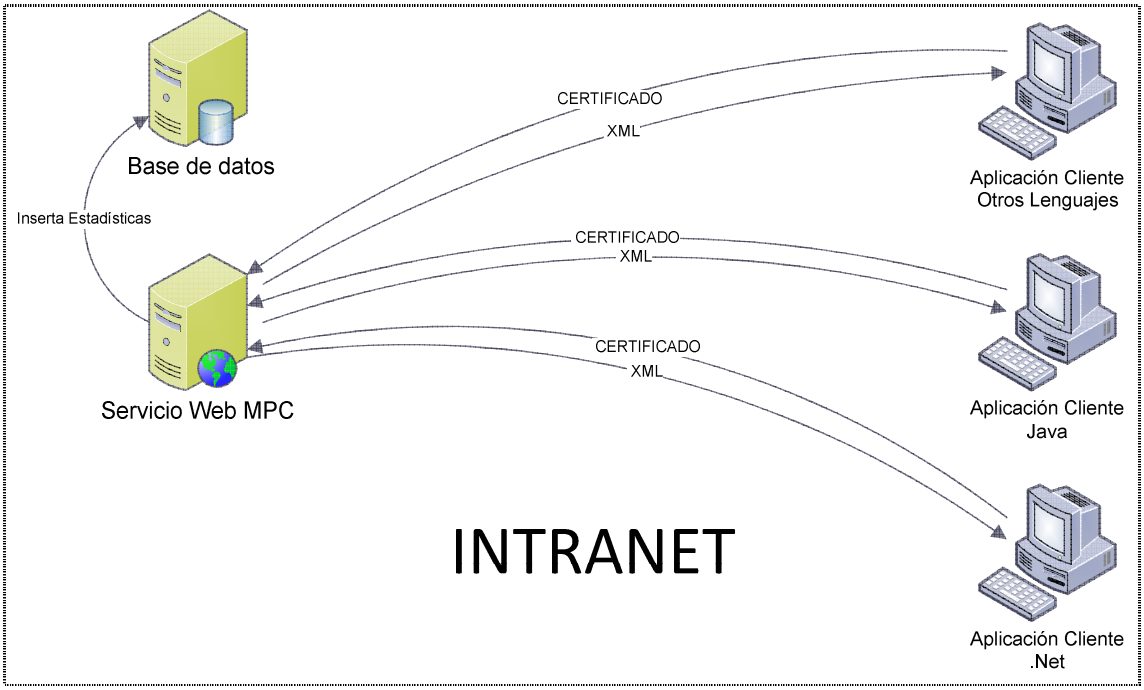


Ilustración 22: Arquitectura del sistema MPC

El sistema MPC estará ubicado dentro de un directorio virtual en un servidor de tipo Microsoft IIS 6 o superior en una máquina del entorno intranet. La idea es que esta máquina sea accesible solo para aplicaciones internas del Ministerio, puesto que se trata de un servicio horizontal.

Existirá también un servidor de base datos, dónde correrá un SQL SERVER 2005 o superior con la base de datos del sistema. La base de datos estará en una máquina ubicada también en intranet y a la cual solo podrá conectarse el servicio web MPC mediante autentificación con usuario y contraseña para poder insertar estadísticas de cada consulta realizada y comprobar el identificador de cada aplicación cliente.

Las aplicaciones que utilicen los métodos del sistema MPC deberán tener sus números de identificación de aplicación dados de alta en la tabla Aplicaciones de la base de datos del sistema MPC.

4.6 Modelo de datos

A continuación se muestra el modelo de datos de la aplicación MPC. Se trata de una base de datos sencilla puesto que su finalidad es la de proporcionar información sobre el uso de la aplicación que pueda explotarse como estadísticas en el futuro.

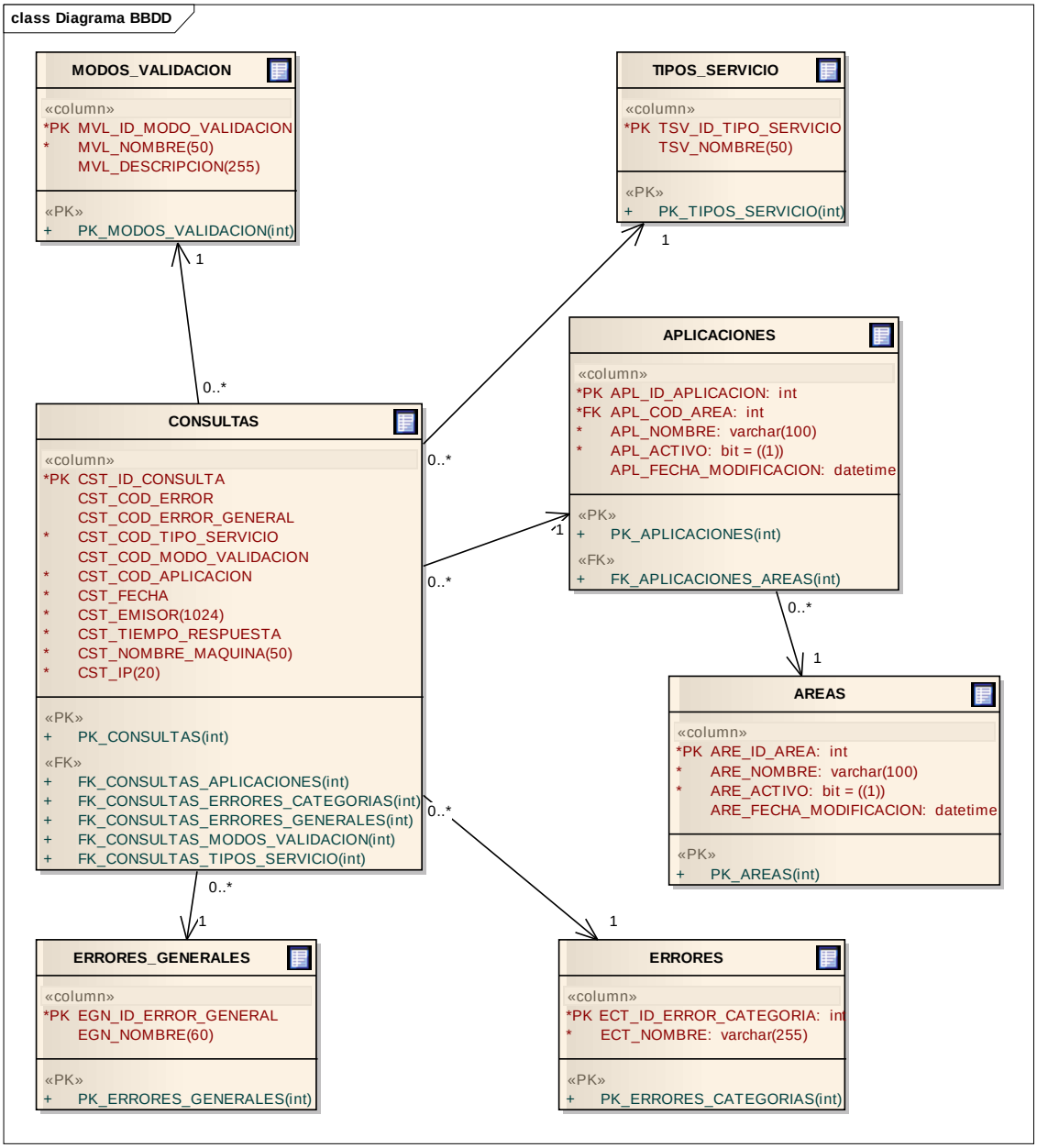


Ilustración 23: Diagrama de base de datos del sistema MPC

4.6.1 Diccionario de datos

A continuación se van a describir las tablas que forman parte del modelo de datos del sistema. Para su representación se va a optar por tablas con cinco columnas. El significado de cada columna es el siguiente:

- Atributo. Indica el nombre del campo tal y como aparece en el modelo de base de datos.
- Tipo. Indica el tipo de dato del atributo.
- Nulo. Indica si el atributo permite o no valores nulos.
- Valor defecto. Indica el valor por defecto que tomará el atributo en caso de nuevo registro.
- Descripción. Breve descripción del significado y utilidad del atributo.

4.6.1.1 Consultas

Es la tabla principal del modelo de datos. Tiene un registro por cada certificado que se le pasa a cada método del servicio web.

ATRIBUTO	TIPO	NULO	VALOR DEFECTO	DESCRIPCIÓN
CST_ID_CONSULTA	Int	No	Autonumérico	Identificador que sirve de clave primaria para la tabla consultas.
CST_COD_ERROR	Int	Si	Null	Clave ajena del error producido en la consulta.
CST_COD_ERROR_GENERAL	Int	Si	Null	Clave ajena del error general producido en la consulta.
CST_COD_TIPO_SERVI	Int	No		Clave ajena del tipo de servicio

CIO				de la consulta.
CST_COD_MODO_VALIDACION	Int	Si	Null	Clave ajena del modo de validación de la consulta.
CST_COD_APLICACION	Int	No		Clave ajena de la aplicación que realizó la consulta.
CST_FECHA	Datetime	No		Fecha en la que se realizó la consulta.
CST_EMISOR	Varchar(1024)	No		Campo emisor del certificado sobre el que se realiza la consulta.
CST_TIEMPO_RESPUESTA	Int	No		Tiempo total en milisegundos de la consulta.
CST_NOMBRE_MAQUINA	Varchar(50)	No		Nombre de la máquina desde la que se realiza la consulta.
CST_IP	Nchar(20)	No		Dirección Ip de la máquina desde la que se realiza la consulta.

Ilustración 24: Tabla Consultas

4.6.1.2 Errores Generales

Se trata de una tabla maestra que almacena los tipos de errores que pueden ocurrir a lo largo de la ejecución de una consulta. No se detalla el error sino que se da una visión general del origen del mismo.

ATRIBUTO	TIPO	NULO	VALOR DEFECTO	DESCRIPCIÓN
EGN_ID_ERROR_GENERAL	Int	No	Autonumérico	Identificador que sirve de clave primaria para la tabla Errores Generales.
EGN_NOMBRE	Varchar(60)	No		Nombre del error general producido.

Ilustración 25: Tabla Errores generales

A continuación se muestran los valores de la tabla maestra Errores Generales:

EGN_ID_ERROR_GENERAL	EGN_NOMBRE
1	Formato de entrada no válido.
2	Certificado no soportado por el sistema.
3	Error al obtener los datos del certificado.
4	Error al validar el certificado.
5	Error al obtener los datos y validar el certificado.
6	Error de conexión con base de datos.
7	Error desconocido.

Ilustración 26: Valores de la tabla Errores generales

4.6.1.3 Errores

La tabla maestra Errores amplia la información de los errores producidos durante la consulta de un certificado. Da una visión más detallada que la tabla Errores Generales.

ATRIBUTO	TIPO	NULO	VALOR DEFECTO	DESCRIPCIÓN
ECT_ID_ERROR	Int	No	Autonumérico	Identificador que sirve como clave primaria para la tabla Errores.
ECT_NOMBRE	Varchar(255)	No		Nombre del error producido durante la consulta.

Ilustración 27: Tabla Errores

A continuación se muestran los valores de la tabla maestra Errores:

ECT_ID_ERROR	ECT_NOMBRE
1	Parámetro de entrada no válido. El formato del certificado no es correcto.
2	Parámetro de entrada nulo. El certificado no puede ser nulo.
3	Parámetro de entrada no válido. El identificador de la aplicación. no es correcto, debe ser un número mayor de 0
4	Parámetro de entrada no válido. El modo de validación debe ser uno de los valores {0,1}.
5	Error al intentar convertir el certificado a un formato válido.
6	Error al leer el fichero de configuración.
7	No se ha podido identificar el prestador al que pertenece el certificado.

8	Certificado no soportado por el sistema.
9	Error al comprobar la política del certificado.
10	Error al parsear los datos del emisor del certificado.
11	Error al parsear los datos del asunto del certificado.
12	Error al parsear los nombres alternativos del certificado.
13	Error al validar el estado de caducidad del certificado.
14	Error al validar que la fecha actual no sea anterior a la fecha de comienzo del periodo de validez del certificado.
15	Error al validar la confianza del emisor del certificado.
16	Error al validar la firma del certificado.
17	Error al validar las extensiones del certificado.
18	Error al insertar estadísticas en la base de datos.
19	Error al buscar el Identificador de aplicación en la base de datos.
20	Error desconocido del MPC.

Ilustración 28: Valores de la tabla Errores

4.6.1.4 Modos Validación

En esta tabla maestra se reflejan los posibles modos de validación que acepta la aplicación MPC. En esta primera fase solo se contempla el modo simple de validación pero se quiere tener en una tabla separada para futuras ampliaciones del proyecto.

ATRIBUTO	TIPO	NULO	VALOR DEFECTO	DESCRIPCIÓN
MVL_ID_MODO_VALIDACION	Int	No	Autonumérico	Identificador que sirve de clave primaria para la tabla Modos

				Validación.
MVL_NOMBRE	Varchar(50)	No		Nombre del modo de validación aplicado en la consulta.
MVL_DESCRIPCION	Varchar(255)	Si	Null	Descripción del modo de validación.

Ilustración 29: Tabla Modos validación

A continuación se muestran los posibles valores de la tabla maestra Modos Validación:

MVL_ID_MODO_VALIDACION	MVL_NOMBRE	MVL_DESCRIPCION
0	Sin validación	No se valida el certificado.
1	Básica.	Se valida la caducidad, integridad y confianza del certificado.

Ilustración 30: Valores de la tabla Modos validación

4.6.1.5 Tipos Servicio

La tabla Tipos Servicio determina el tipo de actuación que debe llevar a cabo el programa según la opción elegida. El sistema puede obtener información de un certificado, validarlo o ambas opciones a la vez. De esta forma se almacena que tipo de actuación se ha llevado a cabo en cada consulta. Es importante comentar que para el método ObtenerInformacionVariosCertificados el tipo servicio será siempre ObtenerInformacion ya que en realidad lo que se lleva a cabo es la obtención de información de varios certificados y existirá una consulta en base de datos por cada certificado de la lista pasada como parámetro.

ATRIBUTO	TIPO	NULO	VALOR DEFECTO	DESCRIPCIÓN
TSV_ID_TIPO_SERVICIO	Int	No	Autonumérico	Identificador que sirve de

				clave primaria para la tabla Tipos Servicio.
TSV_NOMBRE	Varchar(50)	No		Nombre del tipo de servicio que se presta.

Ilustración 31: Tabla Tipos servicio

A continuación se muestran los valores de la tabla maestra Tipos Servicio:

TSV_ID_TIPO_SERVICIO	TSV_NOMBRE
1	Obtención de información.
2	Validación de datos.
3	Obtención de información y Validación de datos.

Ilustración 32: Valores de la tabla Tipos Servicio

4.6.1.6 Áreas

Esta tabla guarda información sobre el área dentro del Ministerio al que pertenece la aplicación que está realizando la consulta al sistema MPC.

ATRIBUTO	TIPO	NULO	VALOR DEFECTO	DESCRIPCIÓN
ARE_ID_AREA	Int	No	Autonumérico	Identificador que sirve como clave primaria para la tabla Áreas.
ARE_NOMBRE	Varchar(100)	No		Nombre del área al que pertenece una aplicación.
ARE_ACTIVO	Bit	No	1	Valor booleano que indica si el

				área se mantiene en activo o ha dejado de existir.
ARE_FECHA_MODIFICIACION	Datetime	Si	Null	Fecha en la que se modifica el registro.

Ilustración 33: Tabla Áreas

A continuación se muestran los valores de la tabla maestra Áreas:

ARE_ID_AREA	ARE_NOMBRE	ARE_ACTIVO	ARE_FECHA_MODIFICIACION
0	Área Desconocida	1	Null
1	Servicios Horizontales	1	Null
2	Registro	1	Null
3	Servicios Verticales	1	Null
4	Desarrollo	1	Null

Ilustración 34: Valores de la tabla Áreas

4.6.1.7 Aplicaciones

En esta tabla se almacenan la información sobre las aplicaciones que están autorizadas para utilizar el servicio MPC. Siempre que una nueva aplicación quiera usar el servicio MPC se debe dar de alta en esta tabla.

ATRIBUTO	TIPO	NULO	VALOR DEFECTO	DESCRIPCIÓN
APL_ID_APLICACION	Int	No		Identificador único de una aplicación dentro del

				Ministerio que sirve de clave primaria para la tabla Aplicaciones.
APL_COD_AREA	Int	No		Clave ajena del área al que pertenece la aplicación.
APL_NOMBRE	Varchar(100)	No		Nombre de la aplicación.
APL_ACTIVO	Bit	No		Valor booleano que indica si la aplicación se mantiene en activo o ha dejado de existir.
APL_FECHA_MODIFICACION	Datetime	No		Fecha en la que se modifica el registro.

Ilustración 35: Tabla Aplicaciones

A continuación se muestran algunos ejemplos de registros de la tabla maestra Aplicaciones:

APL_ID_APLICACION	APL_COD_AREA	APL_NOMBRE	APL_ACTIVO	APL_FECHA_MODIFICACION
0	0	Aplicación Desconocida	1	Null
25	1	Aplicación 01	1	Null

46	4	Aplicación 02	0	Null
67	1	Aplicación 03	1	Null
177	2	Aplicación 04	1	Null

Ilustración 36: Ejemplo de valores para la tabla Aplicaciones

4.7 Análisis de las declaraciones de prácticas de certificación

El sistema MPC extrae información de 4 tipos distintos de certificados. Todos los certificados que reconoce el sistema MPC son considerados certificados reconocidos según establece la legislación española, Para el correcto funcionamiento de la aplicación se hace imprescindible realizar un análisis de los documentos DPC de cada certificado admitido. En estos documentos como se ha indicado anteriormente es dónde los prestadores ofrecen toda la información detallada sobre los certificados que emiten. El hecho de analizar estos documentos nos permitirá tener información de primera mano de la estructura del certificado, jerarquía de las entidades de certificación, estructura de campos, extensiones críticas y todos los detalles imprescindibles que necesitamos conocer para realizar la aplicación de manera óptima.

A continuación se analizarán detalles de cada tipo de certificado extraídos de sus correspondientes documentos DPC.

4.7.1 Fnmt

La Fábrica Nacional de Moneda y Timbre es desde 1996 el prestador de servicios de certificación más importante de nuestro país. Es el encargado de proveer los servicios técnicos y administrativos necesarios para garantizar la seguridad, validez y eficacia de la emisión y recepción de comunicaciones y documentos a través de técnicas y medios electrónicos, informáticos y telemáticos (EIT) en las relaciones que se produzcan entre los órganos de la administración general del estado y entre las personas físicas o jurídicas que forman parte de dicha administración. El Documento de Prácticas de Certificación de la FNMT se puede consultar en la siguiente URL:

http://www.cert.fnmt.es

Para el presente proyecto nos interesan los certificados de tipo FNMT Clase 2 CA, FNMT APE y FNMT AP. Dentro de estas tres ramas de la FNMT, existen varios perfiles de certificados. En el catálogo de requisitos se especifica que solo se tendrán en cuenta los certificados vinculados a personas físicas o jurídicas quedando fuera del ámbito de este proyecto el tratamiento de certificados de sede o sello (emitidos por la FNMT para las administraciones públicas) así como los propios certificados raíces o intermedios.

4.7.1.1 Fnmt Clase 2 CA

Los certificados del tipo FNMT Clase 2 CA son los certificados de la Fábrica Nacional de la Moneda y Timbre que se ponen a disposición de cualquier ciudadano particular o empresa que lo solicite y cumpla los requisitos para su obtención.

La cadena de certificación de los certificados de tipo FNMT Clase 2 CA es la siguiente

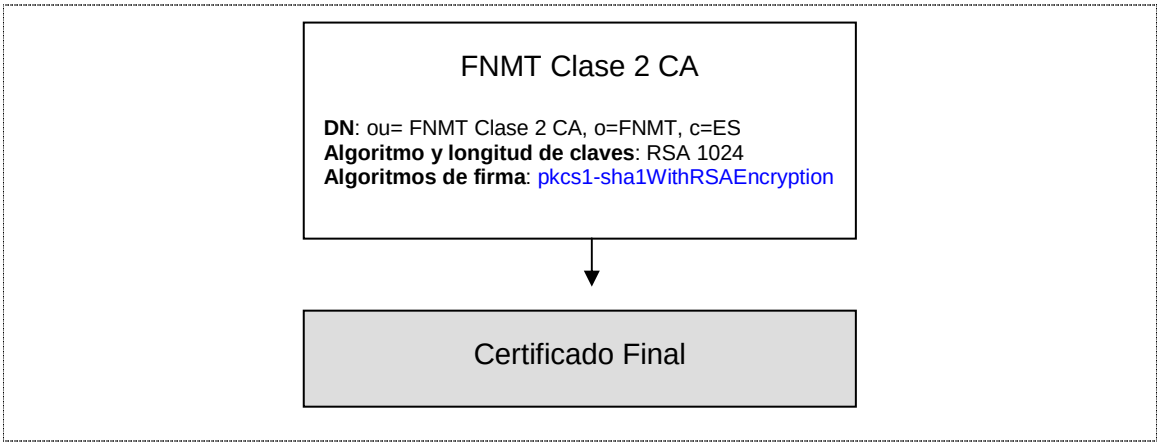


Ilustración 37: Cadena de certificación FNMT Clase 2 CA

Como se observa solo existe un certificado raíz que es directamente el responsable de firmar todos los certificados finales de tipo FNMT Clase 2 CA. Necesitaremos tener cargada la parte pública de este certificado en nuestro sistema para poder verificar la firma de los certificados finales.

Los certificados de tipo FNMT Clase 2 CA tienen no solo uno sino varios perfiles de certificados que entran dentro del alcance de nuestro proyecto. Por tanto, se deben analizar todos para ver las diferencias de cara al parseo de datos. Los perfiles de los certificados que nos interesa analizar se muestran a continuación tal y como aparecen en su documento de prácticas de certificación:

Perfil del Certificado de identificación de persona física:

Lo primero que se nos indica en el documento DPC es la composición del nombre distintivo (DN) del suscriptor del certificado. En el caso de los certificados de identificación de persona física el DN está compuesto por los siguientes elementos:

DN=CN, OU, OU, OU, O, C

Los atributos OU, OU, OU, O, C se utilizan para representar la rama del directorio en la que se encuentra ubicada la entrada del suscriptor en cuestión. Se utilizan para conseguir que el nombre distintivo sea único dentro de la infraestructura de clave pública.

El atributo DN contiene los datos del suscriptor, y para este tipo de certificado sigue el siguiente esquema:

CN = NOMBRE a1 a2 n – NIF 12345678A

Dónde NOMBRE y NIF son etiquetas que sirven para facilitar el parseo de los datos y a1 a2 y n son respectivamente el primer apellido, el segundo apellido y el nombre del suscriptor. 12345678A es un ejemplo de NIF.

Ejemplo de DN para un certificado de identificación de persona física de tipo FNMT Clase 2 CA:

CN = NOMBRE ESPAÑOL ESPAÑOL JUAN - NIF 12345678A
OU = 500650017
OU = fnmt clase 2 ca
O = FNMT
C = es

Ilustración 38: Ejemplo DN para certificado persona física FNMT Clase 2 CA

Para facilitar la obtención de estos datos también se incluye la identidad alternativa del suscriptor, según permite el estándar x509 en su versión v3. De esta forma se tienen los mismos datos que se proporcionan en el DN separados en atributos, de forma que sea mucho más sencilla su obtención. Dentro de la extensión subjectAltName se utiliza el campo DirectoryName para incluir los siguientes atributos:

Información	Atributo FNMT	OID (*)
Nombre	fnmtNombre	fnmtoid.1.1
Primer apellido	fnmtApellido1	fnmtoid.1.2
Segundo apellido	fnmtApellido2	fnmtoid.1.3
NIF	fnmtNif	fnmtoid.1.4

(*) fnmtoid = 1.3.6.1.4.1.5734.

Ilustración 39: Atributos del campo DirectoryName para certificado persona física FNMT Clase 2 CA

El perfil completo del certificado de identificación de persona física de tipo FNMT Clase 2 CA tal y como aparece en su documento de política de prácticas de certificación es el siguiente:

Campo	O.I.D	Valor
Campos Básicos		
Version		2 (X.509 v3)
SerialNumber		Número de serie del *Certificado.* [1]
Issuer		C=ES,O=FNMT,OU=FNMT Clase 2 CA
Validity		[2]
Subject		**Nombre distintivo del *Suscriptor.*** [3]
SubjectPublicKeyInfo		RsaEncryption, *Clave Pública.* [4]
SignatureAlgIdentifier	1.2.840.113549.1.1.5	Identificador de Algoritmo de Firma electrónica utilizado. [5]
Extensiones Estándar		
KeyUsage	2.5.29.15	[6]
PrivateKeyUsageperiod	2.5.29.16	El mismo que Validity
SubjectAltName	2.5.29.17	[7]
CertificatePolicies	2.5.29.32	*Política de Certificación* [8]
CRLDistributionPoints	2.5.29.31	Cn=CRLnnn,c=ES, o=FNMT,OU=FNMT Clase 2 CA [9]
AuthorityKeyIdentifier	2.5.29.35	Identificador de *Clave* del *Prestador de Servicios de Certificación*
SubjectKeyIdentifier	2.5.29.14	Identificador de *Clave* del *Suscriptor*
BasicConstraints	2.5.29.19	Restricciones básicas. Entidad Final
Extensiones Privadas		
NetscapeCertType	2.16.840.1.113730.1	[10]
QCStatement	1.3.6.1.5.5.7.1.3	[11]
fnmtTipoCertificado	1.3.6.1.4.1.5734.1.33	[12]

Ilustración 40: Perfil Certificado de persona física FNMT Clase 2 CA

De esto con lo que nos tenemos que quedar es con qué extensiones son críticas. Según el documento DPC para este certificado ninguna de sus extensiones han sido marcadas como críticas. También es importante que nos quedemos con la extensión privada fnmtTipoCertificado, la cual nos servirá para identificar el tipo de certificado dentro de la familia de certificados de la FNMT. Para este tipo de certificado el valor que tomará la extensión privada fnmtTipoCertificado es el siguiente:

"PERSONA FISICA"

Perfil del Certificado de persona jurídica para el ámbito tributario.

Para este tipo de certificados el nombre distintivo (DN) tiene la misma estructura que para los certificados de tipo persona física:

DN=CN, OU, OU, OU, O, C

La diferencia en este caso está en el CN, el cual para este tipo de certificados de persona jurídica para el ámbito tributario sigue el siguiente esquema:

CN =ENTIDAD e – CIF 12345678B - NOMBRE a1 a2 n – NIF 12345678A

Dónde ENTIDAD, CIF, NOMBRE y NIF son etiquetas que sirven para facilitar el parseo de los datos, e es la denominación o razón social de la persona jurídica suscriptora del certificado y 12345678B es el CIF de la persona jurídica.

Ejemplo de DN para un certificado de identificación de persona jurídica para el ámbito tributario de tipo FNMT Clase 2 CA:

CN = ENTIDAD FUNDACION PRUEBA - CIF 12345678B - NOMBRE ESPAÑOL ESPAÑOL JUAN - NIF 12345678A

OU = 500650017

OU = FNMTClase 2 ca

O = FNMT

C = es

Ilustración 41: Ejemplo de DN para persona jurídica ámbito tributario FNMT Clase 2 CA

En este caso también se incluye la identidad alternativa del suscriptor dentro de la extensión subjectAltName. Como se observa en la siguiente tabla se añaden dos nuevos atributos a los mencionados anteriormente. Por tanto en el campo DirectoryName se incluyen los siguientes atributos para este tipo de certificado:

Tipo Certificado	*Información*	*Atributo FNMT*	*OID (*)*
Persona Jurídica Ámbito Tributario	Entidad	fnmtRepEntidad	fnmtoid.1.6
	CIF Entidad	fnmtRepCif	fnmtoid.1.7
	Nombre del *Solicitante*	fnmtNombre	fnmtoid.1.1
	Apellido 1 *Solicitante*	fnmtApellido1	fnmtoid.1.2
	Apellido 2 *Solicitante*	fnmtApellido2	fnmtoid.1.3
	NIF *Solicitante*	fnmtNIF	fnmtoif.1.4

(*) OID = 1.3.6.1.4.1.5734.

Ilustración 42: Atributos del campo DirectoryName para certificado persona jurídica FNMT Clase 2 CA

El perfil completo del certificado de persona jurídica para el ámbito tributario de tipo FNMT Clase 2 CA tal y como aparece en su documento de política de prácticas de certificación es el siguiente:

Campo	O.I.D	Valor
Campos Básicos		
Version		2 (X.509 v3)
SerialNumber		Número de serie del *Certificado.* [1]
Issuer		C=ES,O=FNMT,OU=FNMT Clase 2 CA
Validity		[2]
Subject		**Nombre distintivo del *Suscriptor*.** [3]
SubjectPublicKeyInfo		RsaEncryption, *Clave Pública.* [4]
SignatureAlgIdentifier	1.2.840.113549.1.1.5	Identificador del Algoritmo de Firma electrónica utilizado. [5]
Extensiones Estándar		
KeyUsage	2.5.29.15	[6]
PrivateKeyUsageperiod	2.5.29.16	El mismo que Validity
SubjectAltName	2.5.29.17	[7]
CRLDistributionPoints	2.5.29.31	Cn=CRLnnn, c=ES, o=FNMT,OU=FNMT Clase 2 CA [8]
AuthorityKeyIdentifier	2.5.29.35	Identificador de *Clave* del *PSC*
SubjectKeyIdentifier	2.5.29.14	Identificador de *Clave* del *Suscriptor*
BasicConstraints	2.5.29.19	Restricciones básicas. Entidad Final
Extensiones Privadas		
NetscapeCertType	2.16.840.1.113730.1	[9]
fnmtTipoCertificado	1.3.6.1.4.1.5734.1.33	[10]

Ilustración 43: Perfil Certificado de persona jurídica FNMT Clase 2 CA

Según el documento DPC para este certificado ninguna de sus extensiones han sido marcadas como críticas. También es importante que nos quedemos con la extensión privada fnmtTipoCertificado, la cual nos servirá para identificar el tipo de certificado dentro de la familia de certificados de la FNMT. Para este tipo de certificado el valor que tomará la extensión privada fnmtTipoCertificado es el siguiente:

"CERTIFICADO EXCLUSIVO PARA EL AMBITO TRIBUTARIO"

Perfil del Certificado de persona jurídica para el ámbito público y otros ámbitos.

Para este tipo de certificados el nombre distintivo (DN) tiene la misma estructura exacta que para los certificados de persona jurídica para el ámbito tributario

La identidad alternativa del suscriptor está ubicada en el mismo lugar que se ha descrito en los anteriores tipos de certificado de tipo FNMT Clase 2 CA. Para este tipo concreto de certificados se incluyen los siguientes atributos:

Tipo Certificado	*Información*	*Atributo FNMT*	*OID (*)*
Persona Jurídica Ámbito Público/Otros Ámbitos	Entidad	fnmtRepEntidad	fnmtoid.1.6
	CIF Entidad	fnmtRepCif	fnmtoid.1.7
	Nombre del *Solicitante*	fnmtNombre	fnmtoid.1.1
	Apellido 1 *Solicitante*	fnmtApellido1	fnmtoid.1.2
	Apellido 2 *Solicitante*	fnmtApellido2	fnmtoid.1.3
	NIF *Solicitante*	fnmtNIF	fnmtoif.1.4

(*) OID = 1.3.6.1.4.1.5734.

Ilustración 44: Atributos del campo DirectoryName para certificado PJ ámbito público FNMT Clase 2 CA

El perfil del certificado de persona jurídica para el ámbito público y otros ámbitos es exactamente igual al del certificado de persona jurídica para el ámbito tributario. No existen extensiones críticas según el DPC. La única diferencia es el valor de la extensión privada fnmtTipoCertificado, que en este caso toma el siguiente valor:

"CERTIFICADO CPJ FNMT/ÁMBITO PÚBLICO/OTROS ÁMBITOS AUTORIZADOS-DESTINATARIO"

Perfil del Certificado de entidad sin personalidad jurídica para el ámbito tributario.

Para este tipo de certificados el nombre distintivo (DN) tiene la misma estructura exacta que para los certificados de persona jurídica para el ámbito tributario y persona jurídica para el ámbito público y otros ámbitos.

La identidad alternativa del suscriptor está ubicada en el mismo lugar que se ha descrito en los anteriores tipos de certificado de tipo FNMT Clase 2 CA. Para este tipo concreto de certificados se incluyen los siguientes atributos:

Tipo Certificado	*Información*	*Atributo FNMT*	*OID (*)*
Entidad Sin personalidad Jurídica Ámbito Tributario	Entidad	fnmtRepEntidad	fnmtoid.1.6
	CIF Entidad	fnmtRepCif	fnmtoid.1.7
	Nombre del *Solicitante*	fnmtNombre	fnmtoid.1.1
	Apellido 1 *Solicitante*	fnmtApellido1	fnmtoid.1.2
	Apellido 2 *Solicitante*	fnmtApellido2	fnmtoid.1.3
	NIF *Solicitante*	fnmtNIF	fnmtoif.1.4

(*) OID = 1.3.6.1.4.1.5734.

Ilustración 45: Atributos del campo DirectoryName para certificado de entidad sin personalidad jurídica

El perfil completo del certificado de entidad sin personalidad jurídica para el ámbito tributario de tipo FNMT Clase 2 CA tal y como aparece en su documento de política de prácticas de certificación es el siguiente:

Campo	O.I.D	Valor
Campos Básicos		
Version		2 (X.509 v3)
SerialNumber		Número de serie del *Certificado.* [1]
Issuer		C=ES,O=FNMT,OU=FNMT Clase 2 CA
Validity		[2]
Subject		**Nombre distintivo del *Suscriptor.*** [3]
SubjectPublicKeyInfo		RsaEncryption, *Clave Pública.* [4]
SignatureAlgIdentifier	1.2.840.113549.1.1.5	Identificador del Algoritmo de Firma electrónica utilizado. [5]
Extensiones Estándar		
KeyUsage	2.5.29.15	[6]
PrivateKeyUsageperiod	2.5.29.16	El mismo que Validity
SubjectAltName	2.5.29.17	[7]

CRLDistributionPoints	2.5.29.31	Cn=CRLnnn, c=ES, o=FNMT,OU=FNMT Clase 2 CA [8]
AuthorityKeyIdentifier	2.5.29.35	Identificador de *Clave* del *PSC*
SubjectKeyIdentifier	2.5.29.14	Identificador de *Clave* del *Suscriptor*
BasicConstraints	2.5.29.19	Restricciones básicas. Entidad Final
Extensiones Privadas		
NetscapeCertType	2.16.840.1.113730.1	[9]
fnmtTipoCertificado	1.3.6.1.4.1.5734.1.33	[10]
fnmtTipoEntidadSinPJ	1.3.6.1.4.1.5734.1.36	[11]

Ilustración 46: Perfil certificado de entidad sin personalidad jurídica FNMT Clase 2 CA

Según el documento DPC para este certificado ninguna de sus extensiones han sido marcadas como críticas. También es importante que nos quedemos con la extensión privada fnmtTipoCertificado, para este tipo de certificado el valor que tomará dicha extensión privada es el siguiente:

"CERTIFICADO DE ENTIDAD SIN PERSONALIDAD JURÍDICA EXCLUSIVO PARA EL ÁMBITO TRIBUTARIO"

Valor que tomará el campo TipoPersona

Es importante tener claro llegados a este punto que el sistema MPC solo quiere recoger si un certificado tiene tratamiento de persona física o de persona jurídica de cara a que este dato sea utilizado por las aplicaciones internas del ministerio, por tanto una vez vistos todos los perfiles de certificados FNMT Clase 2 CA se muestra a continuación la tabla de equivalencia en el tratamiento del campo TipoPersona:

Tipo Certificado	Valor campo TipoPersona
PERSONA FISICA	FÍSICA
CERTIFICADO EXCLUSIVO PARA EL AMBITO TRIBUTARIO	JURÍDICA
CERTIFICADO CPJ FNMT/ÁMBITO PÚBLICO/OTROS ÁMBITOS AUTORIZADOS-DESTINATARIO	JURÍDICA
CERTIFICADO DE ENTIDAD SIN PERSONALIDAD JURÍDICA EXCLUSIVO PARA EL ÁMBITO TRIBUTARIO	JURÍDICA

Ilustración 47: Relación TipoCertificado - valor campo TipoPersona

4.7.1.2 Fnmt Ape

Los certificados de la Fábrica Nacional de Moneda y Timbre de tipo FNMT APE son certificados emitidos especialmente para el entorno de las administraciones públicas. La función que lleva a cabo la FNMT en este caso es la de emitir certificados para la identificación de sedes electrónicas, sistemas de firma electrónica para la actuación administrativa automatizada y certificados emitidos para el personal al servicio de la Administración Pública. Actualmente están siendo sustituidos por los certificados FNMT AP, que cumplen mejor las necesidades y requisitos de la administración. Aun así todavía son utilizados y por tanto se deben tener en cuenta de cara a que la aplicación sea capaz de interpretar sus datos correctamente.

La cadena de certificación de los certificados FNMT APE se muestra a continuación:

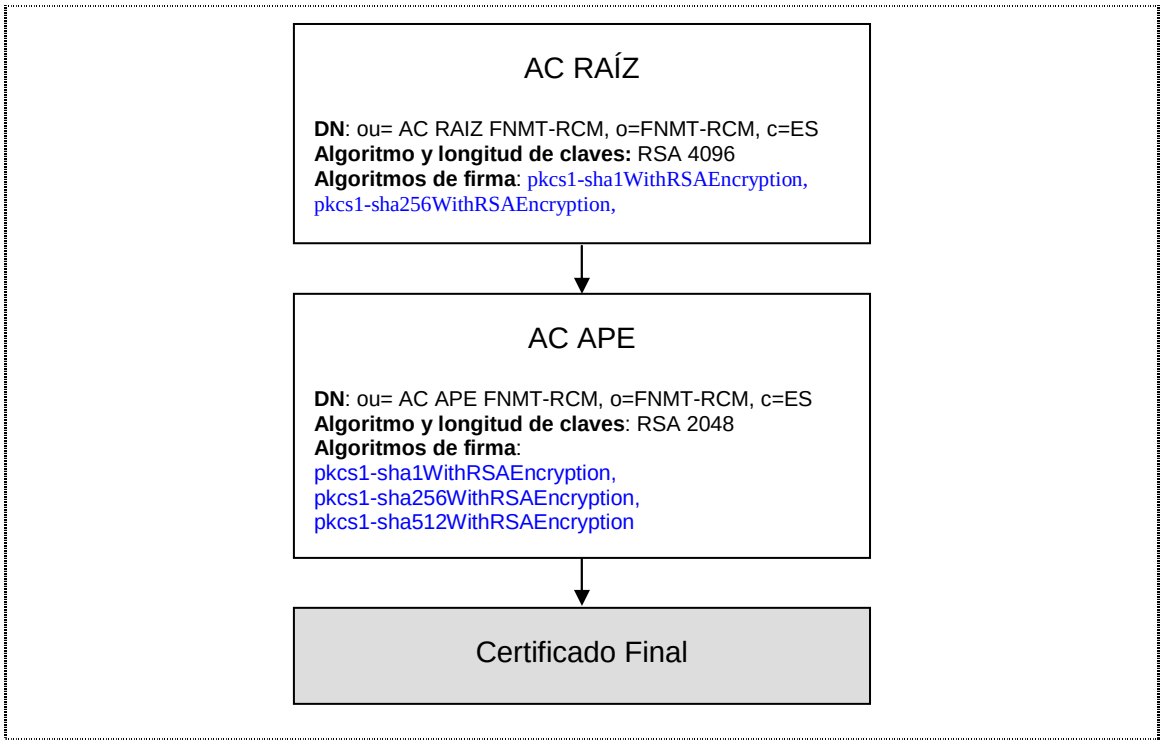


Ilustración 48: Cadena de certificación FNMT APE

Como puede observarse, en este caso la cadena de certificación está formada por 2 elementos, el certificado raíz (nótese que es diferente al certificado raíz de los certificados de tipo FNMT Clase 2 CA) y por una entidad intermedia. Este certificado intermedio AC APE es el encargado de firmar los certificados finales de tipo FNMT APE, por tanto será imprescindible poseer la parte pública del mismo para poder verificar la firma de los certificados FNMT APE.

Como se ha comentado anteriormente solo nos interesa analizar los perfiles de los certificados emitidos para el personal al servicio de la administración pública, quedan exentos de este análisis los certificados para la identificación de sedes electrónicas (sede) y los certificados emitidos para la actuación administrativa automatizada (sello).

El perfil del certificado del personal de la administración pública tal y como aparece en el documento de prácticas de certificación es el siguiente:

Perfil del Certificado: Personal APE emitido por la Autoridad de Certificación "AC APE" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.14		
CAMPO	**CONTENIDO**	**ESPECIFICACIONES**
1. Versión	V3	
2. Serial Number	Secuencial	*[RFC3280]: the serial number MUST be a positive integer, not longer than 20 octets (1 < SN < 2^{159}). Processing components MUST be able to interpret such long numbers.*
3. Signature Algorithm	Sha1withRsaEncryption Sha256withRsaEncryption	OID: 1.2.840.113549.1.1.5 OID: 1.2.840.113549.1.1.11 Norma PKCS#1 v2.1 y RFC 3447.
4. Issuer Distinguished Name	OU= AC APE O=FNMT-RCM C=ES	Todos los *DirectoryString* codificados en UTF8. El atributo "C" (*countryName*) se codificará de acuerdo a *"ISO 3166-1-alpha-2 code elements"*, en *PrintableString*. [ETSI-CPN]: *the issuer name MUST contain the countryName and the organizationName attributes.*
5. Validez	48 meses	48 meses a partir del momento de la emisión.
6. Subject	CN= "Identidad del firmante" OU= "Organizacion_Entidad" OU=AC APE O=FNMT-RCM C=ES	Todos los *DirectoryString* codificados en UTF8. El atributo "C" (*countryName*) se codificará de acuerdo a *"ISO 3166-1-alpha-2 code elements"*, en *PrintableString*. El atributo "SN" (*serialNumber*) se codificará en *PrintableString*. La Ley de Firma Electrónica establece para los certificados reconocidos: "*La identificación del firmante, en el supuesto de personas físicas, debe realizar por su* ***nombre y apellidos y su número de documento nacional de identidad o a través de un seudónimo*** *que conste como tal de manera inequívoca y, en el supuesto de personas jurídicas, por su denominación o razón social y su código de identificación fiscal*". El número de serie convierte en único el nombre del sujeto. Todos los *DirectoryString* codificados en UTF8. La identidad del firmante (CN) seguirá la siguiente sintaxis: CN= NOMBRE a1 a2 n – NIF 12345678A Donde: [1] NOMBRE y NIF son etiquetas [2] n, a1 y a2 son los nombres, primer y segundo apellido del personal al servicio de la Administración Pública respectivamente [3] 12345678A es su correspondiente NIF. [1] Las etiquetas siempre van en mayúsculas y se separan del valor por un espacio en blanco. Las duplas <etiqueta, valor> se separan entre ellas con un espacio en blanco, un guión y otro espacio en blanco (" – ") [2] Con todos sus caracteres en mayúsculas, excepto la letra eñe, que irá siempre en minúscula. No se incluirán símbolos (comas, guiones, etc.) ni caracteres acentuados. [3] NIF del personal al servicio de la Administración Pública = 8 cifras + 1 letra mayúscula, sin ningún tipo de separación entre ellas. En el caso que un NIF ocupe menos de 8 cifras, se incluirán ceros al comienzo del número hasta completar las 8 cifras. Nota: La identidad del Titular del certificado vendrá reflejada (de forma obligatoria) en la extensión subjectAltName [ETSI-CPN]: *The subject field of EE certificates for natural persons SHALL include at least the commonName or the givenNamen and surname attribute.*
7. Subject Public Key Info	Algoritmo: RSA Encryption Longitud: 2048 bits	
8. issuerUniqueIdentifier	No se utilizará	
9. subjectUniqueIdentifier	No se utilizará	
10. Subject Key Identifier	Función hash SHA-1 sobre la clave pública del sujeto.	RFC 3280: hash SHA-1 de 20 bytes calculado sobre el valor BIT STRING del campo *subjectPublicKey* del sujeto (excluyendo etiqueta, longitud y número de bits no usados).

Perfil del Certificado: Personal APE emitido por la Autoridad de Certificación "AC APE" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.14			
CAMPO		**CONTENIDO**	**ESPECIFICACIONES**
11. Authority Key Identifier		Función de hash SHA-1 sobre la clave pública de la AC emisora (AC subordinada). NO SE incluye la identificación del certificado de la AC emisora (en este caso, DN de la AC Raíz, número de serie de la AC Subordinada).	RFC 3280: hash SHA-1 de 20 bytes calculado sobre el valor BIT STRING del campo *subjectPublicKey* de la AC emisora (excluyendo etiqueta, longitud y número de bits no usados). Coincide con el campo *Subject Key Identifier* de la AC emisora (AC subordinada).
12. KeyUsage			La Norma ETSI TS 102 280 V1.1.1 (2004-03) – "*X.509 V.3 Certificate Profile for Certificates Issued to Natural Persons*" recomienda el uso exclusivo del bit de no repudio para certificados destinados a validar el compromiso con el contenido firmado, tales como firmas electrónicas sobre transacciones o acuerdos. En el *Technical Corrigendum 3* de la ITU-T X509 de 04/2004, el bit de no-repudio es renombrado a **contentCommitment**. En este documento se recoge lo siguiente: "*Note that it is not incorrect to refer to this keyUsage bit using the identifier nonRepudiation. However, the use of this identifier has been deprecated. Regardless of the identifier used, the semantics of this bit are as specified in this Directory Specification.*"
	Digital Signature	1	
	Non Repudiation	1	
	Key Encipherment	1	
	Data Encipherment	1	
	Key Agreement	0	
	Key Certificate Signature	0	
	CRL Signature	0	
13.extKeyUsage		Autenticación de cliente: 1.3.6.1.5.5.7.3.2 Protección de correo electrónico: 1.3.6.1.5.5.7.3.4 Inicio de sesión de tarjeta inteligente: 1.3.6.1.4.1.311.20.2.2	*[RFC3280]: Certificate using applications MAY require that a particular purpose be indicated in order for the certificate to be acceptable to that application. If a CA includes extended key usages to satisfy such applications, but does not wish to restrict usages of the key, the CA can include the special keyPurposeID anyExtendedKeyUsage. If the anyExtendedKeyUsage key purpose is present, the extension SHOULD NOT be critical.*
14. privateKeyUsagePeriod		No se utilizará	
15. Certificate Policies			RFC 3739 obliga la existencia de al menos un valor. La Ley de Firma Electrónica dice para los certificados reconocidos: "*La identificación del prestador de servicios de certificación que expide el certificado y su domicilio*". Se incluirá en la DPC. *In an end entity certificate, these policy information terms indicate the policy under which the certificate has been issued and the purposes for which the certificate may be used.*
	Policy Identifier	OID asociado a la DPC o PC de certificado de firma de usuario final. (fnmtPolCertPersonalAdministracion) 1.3.6.1.4.1.5734.3.14	
	URL CPS	http://www.cert.fnmt.es/dpcs/	
	Notice Reference	Sujeto a las condiciones de uso expuestas en la Declaración de Prácticas de Certificación de la FNMT-RCM (C/Jorge Juan 106-28009-Madrid-España)	
16. qcStatements		id-etsi-qcs-QcCompliance id-etsi-qcs-QcSSCD id-etsi-qcs-QcLimitValue : 0€ Id-etsi-qcs-QcRetentionPeriod: 15 años	ETSI TS 101 862 define la inclusión de las siguientes declaraciones para certificados cualificados: 1.- **id-etsi-qcs-QcCompliance** – Indica que el certificado es qualificado. Solo si no está explicito en las políticas indicadas en la extensión correspondiente. 2.- **id-etsi-qcs-QcLimitValue** – Indica el valor límite para transacciones. (No Aplicable). 3.- **id-etsi-qcs-QcRetentionPeriod** –Indica el número de años a partir de la caducidad del certificado que se dispone de los datos de registro y otro información relevante. En este caso la Ley obliga: "***Conservar registrada*** *por cualquier medio seguro toda la información y documentación relativa a un certificado reconocido y las declaraciones de prácticas de certificación vigentes en cada momento, al menos* ***durante 15 años contados desde el momento de su expedición,*** *de manera que puedan verificarse las firmas efectuadas con el mismo.*". Dado que la FNMT va a conservar la información de manera indefinida, y que la norma no establece un valor para indicar un período indefinido, no se incluirá este QC y se indicará en la DPC. 4.- **id-etsi-qcs-QcSSCD** – Indica que la clave privada correspondiente al certificado está almacenada en un dispositivo seguro de creación de firma de acuerdo al Anexo III de la Directiva del Parlamento Europeo 1999/93/EC, sobre un framework comunitario para firmas electrónicas.

<table>
<tr><th colspan="3">Perfil del Certificado: Personal APE emitido por la Autoridad de Certificación "AC APE" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.14</th></tr>
<tr><th>CAMPO</th><th>CONTENIDO</th><th>ESPECIFICACIONES</th></tr>
<tr><td>17. Subject Alternate Names</td><td>Nombre RFC822= email
Dirección del directorio:
OID.1.3.6.1.4.1.5734.1.1=Nom
OID.1.3.6.1.4.1.5734.1.2=Ape1
OID.1.3.6.1.4.1.5734.1.3=Ape2
OID.1.3.6.1.4.1.5734.1.4=NIF
OID.1.3.6.1.4.1.5734.1.38=fnmtAPECargo
OID.1.3.6.1.4.1.5734.1.39=fnmtAPEEntidad
OID.1.3.6.1.4.1.5734.1.40=fnmtAPESituacion
OID.1.3.6.1.4.1.5734.1.44=fnmtAPENIP
OID.1.3.6.1.4.1.5734.1.45=fnmtUnidadOrganizativa
OID.1.3.6.1.4.1.5734.1.15=fnmtPropCIF

UPN:
OID.1.3.6.1.4.1.311.20.2.3=email (Cadena UTF8 codificada en ASN1)</td><td>Por interoperatividad con las aplicaciones ya desplegadas que hacen uso del formato de identificación de los certificados FNMT Clase 2 CA, se mantiene el contenido de SubjectAltName.

<table><tr><th>Info</th><th>Atributo FNMT</th><th>OID</th></tr><tr><td>Nombre</td><td>fnmtNombre</td><td>fnmtoid.1.1</td></tr><tr><td>Primer apellido</td><td>fnmtApellido1</td><td>fnmtoid.1.2</td></tr><tr><td>Segundo apellido</td><td>fnmtApellido2</td><td>fnmtoid.1.3</td></tr><tr><td>NIF</td><td>fnmtNif</td><td>fnmtoid.1.4</td></tr><tr><td>Cargo</td><td>fnmtAPECargo</td><td>fnmtoid.1.38</td></tr><tr><td>Entidad en la que presta servicio</td><td>fnmtAPEEntidad</td><td>fnmtoid.1.39</td></tr><tr><td>Situación laboral</td><td>fnmtAPEsituacion</td><td>fnmtoid.1.40</td></tr><tr><td>Número identificación</td><td>fnmtAPENIP</td><td>fnmtoid.1.44</td></tr><tr><td>Unidad Organizativa</td><td>fnmtAPEUnidad Organizativa</td><td>fnmtoid.1.45</td></tr></table>
Además del subcampo directoryName de la extensión subjectAltName, en el caso de que se haya aportado una dirección de correo electrónico por el personal al servicio de la Administración Pública durante el proceso de solicitud de emisión del Certificado, ésta estará incluida en el subcampo rfc822Name.

fnmtoid: 1.3.6.1.4.1.5734: Espacio de numeración asignado a la Fabrica Nacional de Moneda y Timbre – Real Casa de la Moneda por el IANA.</td></tr>
<tr><td>18. Issuer Alternate Names</td><td>No se utilizará</td><td></td></tr>
<tr><td>19. Basic Constraints</td><td></td><td rowspan="3">[RFC 3280] This extension MAY appear as a critical or non-critical extension in end entity certificates.</td></tr>
<tr><td>Subject Type</td><td>Entidad Final</td></tr>
<tr><td>Path Length Constraint</td><td>No utilizado</td></tr>
<tr><td>20. Policy Constraints</td><td>No utilizado</td><td></td></tr>
<tr><td>21. CRLDistributionPoints</td><td>LDAP:
ldap://ldapape.cert.fnmt.es:puerto/CN=CRLnnn,OU=AC APE, O=FNMT, C=ES?certificateRevocationList;binary?base?objectclass=cRLDistributionPoint
HTTP:
http://www.cert.fnmt.es/crlsape/CRLnnn.crl</td><td></td></tr>
<tr><td>22. Auth. Information Access</td><td>OCSP:
http://ocspape.cert.fnmt.es/ocspape/OcspResponder
CA:
http://www.cert.fnmt.es/certs/ACAPE.crt</td><td>Contendrá información de localización del OCSP Responder. URL resuelta a diferentes servidores. Ley de Firma Electrónica: "Garantizar la disponibilidad de un servicio de consulta sobre la vigencia de los certificados rápido y seguro".</td></tr>
<tr><td>23. netscapeCertType</td><td>sSLCLIENT,sMIME</td><td>Tipo de certificado según Netscape</td></tr>
<tr><td>24. fnmtTipoCertificado</td><td>1.3.6.1.4.1.5734.1.33: "Personal adscrito a la Administración"</td><td>Tipo de certificado para Personal adscrito a la Administración</td></tr>
</table>

Ilustración 49: Perfil certificado del personal de la administración pública FNMT APE

De este perfil sacamos mucha información. Es importante ver que se han añadido los datos referentes a la identidad alternativa del suscriptor, lo que facilita mucho la obtención de datos como el nombre, los apellidos, el DNI y todos los datos recogidos en el punto 17. Subject Alternative Names. Este certificado, al ser específico para los empleados de las administraciones públicas tiene una serie de campos destinados a guardar información relacionada con la condición de empleado público del suscriptor. Estos campos específicos son los siguientes:

- Cargo
- Entidad en la que presta servicio

- Situación laboral (solo aparece en este tipo de certificados)
- Número de identificación
- Unidad organizativa

Según nos indica el documento de prácticas de certificación de este certificado las extensiones críticas de este certificado son **KeyUsage** y **BasicConstraints**. Se debe comprobar como parte de la validación de este tipo de certificados que ambas extensiones están presentes en el certificado.

Por último para poder identificar correctamente los certificados de la FNMT de tipo APE tenemos que mirar el campo fnmtTipoCertificado que debe tener el siguiente valor:

"PERSONAL ADSCRITO A LA ADMINISTRACION"

4.7.1.3 Fnmt Ap

Los certificados de la Fábrica Nacional de Moneda y Timbre de tipo FNMT AP son los nuevos certificados destinados al uso en la administración general del estado. Los propósitos de su emisión son los mismos que los certificados de tipo FNMT APE, pero presentan diferencias con respecto a estos enfocadas a satisfacer mejor las necesidades y requisitos de las administraciones públicas.

La cadena de certificación de los certificados FNMT AP es la siguiente:

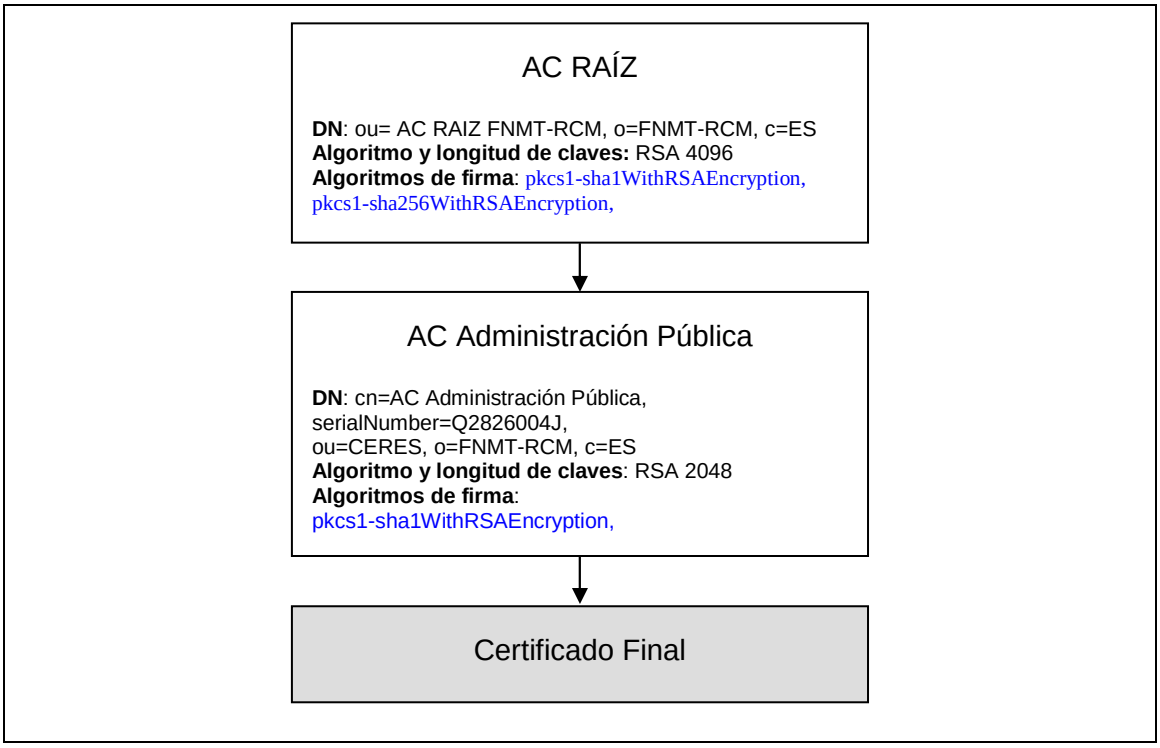


Ilustración 50: Cadena certificación FTMT AP

Como vemos la cadena de certificación vuelve a estar compuesta por dos certificados. El certificado raíz que en este caso coincide con el certificado raíz de los certificados de tipo FNMT APE y un certificado intermedio AC Administración Pública, este sí, único y responsable de firmar los certificados finales de tipo FNMT AP. Por tanto, nuevamente necesitaremos tener acceso a la parte pública de este certificado intermedio para poder comprobar la autenticidad de la firma de los certificados de tipo FNMT AP.

El perfil del certificado para el uso del personal de la administración pública tal y como aparece en el documento de prácticas de certificación es el siguiente:

Personal APE emitido por la Autoridad de Certificación "AC Administración Pública" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.3.4.4.1				
Campo		**Contenido**	**Obligatoriedad**	**Especificaciones**
1. Version		2	Si	Integer:=2 ([RFC5280] describe la versión del certificado. El valor 2 equivale a decir que el certificados es versión 3 (X509v3)
2. Serial Number		Número identificativo único del certificado.	Si	Integer. SerialNumber = ej: 111222. Establecido automáticamente por la Entidad de Certificación. [RFC5280]. Será un "integer" positivo, no mayor 20 octetos ($1-2^{159}$). El número de serie se asignará de forma creciente.
3. Signature Algorithm		Sha1withRsaEncryption	Si	String UTF8 (40). Identificando el tipo de algoritmo (OID 1.3.14.3.2.26)
4. Issuer Distinguish Name		Entidad emisora del certificado	Si	
	4.1. Country	C=ES	Si	Se codificará de acuerdo a "ISO 3166-1-alpha-2 code elements". PrintableString, tamaño 2 (rfc5280)
	4.2. Organization	Denominación (nombre "oficial" de la organización) del prestador de servicios de certificación (emisor del certificado). o=FNMT-RCM.	Si	UTF8 String, tamaño máximo 128 (rfc5280)
	4.3. Organizational Unit	Unidad organizativa dentro del prestador de servicios, responsable de la emisión del certificado. ou=CERES	Si	UTF8 String, tamaño máximo 128 (rfc5280)
	4.4. Serial Number	Número único de identificación de la entidad, aplicable de acuerdo con el país. En España, NIF de la entidad suscriptora. serialNumber=Q2826004J	Si	PrintableString, tamaño 64 (X520). En nuestro caso, el tamaño es 9
	4.5. Common Name	cn=AC Administración Pública	Si	UTF8 String, tamaño máximo 128 (rfc5280)
5. Validity		3 años	Si	Validez máxima limitada por "Esquema de Identificación y Firma. Perfiles de Certificados"
6. Subject		Identificación/descripción del c ustodio/responsable de las claves certificadas	Si	
	6.1. Country	Estado cuya ley rige el nombre, que será "España" por tratarse de entidades públicas. C=ES	Si	Se codificará de acuerdo a "ISO 3166-1-alpha-2 code elements". PrintableString, tamaño 2 (rfc5280)

Personal APE emitido por la Autoridad de Certificación "AC Administración Pública" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.3.4.4.1				
Campo		Contenido	Obligatoriedad	Especificaciones
	6.2. Organization	Denominación (nombre "oficial" de la organización) titular de los servicios de certificación	Sí	UTF8 String, tamaño máximo 128 (rfc5280). Por ejemplo: o=MINISTERIO DE ECONOMÍA
	6.3. Organizational Unit	Descripción del tipo de certificado. En este caso: ou=certificado electrónico de empleado público	Sí	
	6.4. Organizational Unit	Unidad, dentro de la Administración, en la que desempeña su labor el suscriptor del certificado.	Opcional	UTF8 String, tamaño máximo 128 (rfc5280). Por ejemplo: ou=SUBDIRECCIÓN DE SISTEMAS DE INFORMACIÓN Se establecerá el valor de la unidad organizativa si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.
	6.5. Organizational Unit	Número de identificación del suscriptor del certificado (supuestamente unívoco). Identificador del empleado público.	Opcional	UTF8 String, tamaño máximo 128 (rfc5280). Por ejemplo: ou=ADM5689 Se establecerá el valor del identificador de funcionario/empleado público si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.
	6.6. Serial Number	DNI/NIE del empleado público.	Sí	Por ejemplo: serialNumber=99999999R PrintableString, tamaño 64 (X520). En nuestro caso, el tamaño es 9
	6.7. Surname	Apellidos de acuerdo con documento de identificación	Sí	UTF8String (rfc5280). Por ejemplo: sn=ESPAÑOL ESPAÑOL
	6.8. Given Name	Nombre de pila, de acuerdo con documento de identidad (DNI/Pasaporte)	Sí	UTF8String (rfc5280). Por ejemplo: gn=JUAN
	6.9. Common Name	Nombre y apellidos de acuerdo con documento de identidad y número del DNI	Sí	UTF8String (rfc5280). Por ejemplo: cn=JUAN ESPAÑOL ESPAÑOL – DNI 99999999R
7. Authority Key Identifier		Identificador de la clave pública de la CA para la Administración Pública. Medio para identificar la clave pública correspondiente a la clave privada utilizada por la CA para firmar un certificado.	Sí	RFC 5280: hash SHA-1 de 20 bytes calculado sobre el valor BIT STRING del campo subjectPublicKey del emisor del certificado (excluyendo etiqueta, longitud y número de bits no usados). Coincide con el campo Subject Key Identifier de la AC emisora.
8. Subject Public Key Info		Clave pública asociada al empleado público, codificada de acuerdo con el algoritmo criptográfico. En este caso RSA Encryption.	Sí	Campo para transportar la clave pública y para identificar el algoritmo con el cual se utiliza la clave. La longitud de la clave será 2048
9. Subject Key Identifier		Identificador de la clave pública del suscriptor o poseedor de claves. Medio para identificar certificados que contienen una clave pública particular y facilita la construcción de rutas de certificación.	Sí	RFC 5280: hash SHA-1 de 20 bytes calculado sobre el valor BIT STRING del campo subjectPublicKey del sujeto (excluyendo etiqueta, longitud y número de bits no usados).

Personal APE emitido por la Autoridad de Certificación "AC Administración Pública" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.3.4.4.1				
Campo		Contenido	Obligatoriedad	Especificaciones
10. Key Usage				
	10.1. Digital Signature	1		Permite realizar la operación de firma electrónica
	10.2. Content Commitment	1		Se refiere a la cualidad de un tipo de certificado que indica al software en el que se usa que debe permitir que el usuario conozca lo que firma (firma de contratos, acuses de recibo, resguardos, etc.)
	10.3. Key Encipherment	1		Se utiliza para gestión y transporte de claves para establecimiento de sesiones seguras
	10.4. Data Encipherment	1		Se utiliza para cifrar datos que no sean claves criptográficas.
	10.5. Key Agreement	0		Para uso en el proceso de acuerdo de claves
	10.6. Key Certificate Signature	0		Se permite usar para firmar certificados. Se utiliza en los certificados de autoridades de certificación.
	10.7. CRL Signature	0		Se permite usar para firmar listas de revocación de certificados.
11. Extended Key Usage		Uso mejorado o extendido de las claves	Si	Esta extensión indica que uno o más propósitos para los cuales el certificado de clave pública se puede utilizar, además de o en lugar de los usos básicos que se indican en la extensión KeyUsage.
	11.1. Email protection	1.3.6.1.5.5.7.3.4	Si	Protección de correo electrónico
	11.2. Client Authentication	1.3.6.1.5.5.7.3.2	Si	Autenticación de cliente
	11.3. Any Extended Key Usage	Otros propósitos (ver comentario de columna "Especificaciones") 2.5.29.37.0	Si	*[RFC5280]: Certificate using applications MAY require that a particular purpose be indicated in order for the certificate to be acceptable to that application. If a CA includes extended key usages to satisfy such applications, but does not wish to restrict usages of the key, the CA can include the special keyPurposeID* ***anyExtendedKeyUsage****. If the anyExtendedKeyUsage key purpose is present, the extension SHOULD NOT be critical.*
	11.4. Microsoft Smart Card Logon	1.3.6.1.4.1.311.20.2.2	Si	*Necesaria para realizar logon en Windows con tarjeta/token*
12. Qualified Certificate Statements				

Personal APE emitido por la Autoridad de Certificación "AC Administración Pública" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.3.4.4.1					
Campo			Contenido	Obligatoriedad	Especificaciones
	12.1. QcCompliance		Certificado es qualificado. (OID: 0.4.0.1862.1.1)	Si	Indica que el certificado es cualificado. Solo si no está explícito en las políticas indicadas en la extensión correspondiente.
	12.2. QcEuRetentionPeriod		15 años (OID: 0.4.0.1862.1.3)	Si	Número de años a partir de la caducidad del certificado que se dispone de los datos de registro y otro información relevante. En este caso la Ley obliga: "Conservar registrada por cualquier medio seguro toda la información y documentación relativa a un certificado reconocido y las declaraciones de prácticas de certificación vigentes en cada momento, al menos durante 15 años contados desde el momento de su expedición, de manera que puedan verificarse las firmas efectuadas con el mismo.".
	12.3. QcLimitValue		0 € (OID: 0.4.0.1862.1.2)	Si	Límite de responsabilidad
	12.4. QcSSCD		Claves generadas en un DSCF (OID: 0.4.0.1862.1.4)	No	Indica que la clave privada correspondiente al certificado está almacenada en un dispositivo seguro de creación de firma de acuerdo al Anexo III de la Directiva del Parlamento Europeo 1999/93/EC, sobre un framework comunitario para firmas electrónicas.
13. Certificate Policies			Política de certificación	Si	
	13.1. Policy Identifier		Identificador unívoco de la política de certificación asociada a los certificados de tipo "empleado público". En este caso: 1.3.6.1.4.1.5734.3.3.4.4.1	Si	Identificador de la política de certificado para Empleado público-Nivel medio (tarjeta)
	13.2. Policy Qualifier Id				
		13.2.1 CPS Pointer	http://www.cert.fnmt.es/dpcs/	Si	IA5String String. URL de las condiciones de uso.
		13.2.2 User Notice	Certificado reconocido de empleado público. Sujeto a las condiciones de uso expuestas en la Declaración de Prácticas de Certificación de la FNMT-RCM (C/Jorge Juan 106-28009-Madrid-España)	Si	UTF8 String. Longitud máxima 200 caracteres.
14. Subject Alternative Names			Identificación/ descripción de Identidad Administrativa	Si	
	14.1. rfc822 Name		Correo electrónico del empleado público	Opcional	Por ejemplo: rfc822Name=jespanol@meh.es Se establecerá el valor del e-email contacto si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.

Personal APE emitido por la Autoridad de Certificación "AC Administración Pública" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.3.4.4.1					
Campo			Contenido	Obligatoriedad	Especificaciones
	14.2. UPN		UPN (nombre de login de red) para smartcard logon.	Opcional	Campo destinado a incluir el smart card logon de Windows para el responsable del certificado. Se establecerá el valor del UPN si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.
	14.3. Directory Name		Identidad Administrativa	Sí	Campos específicos definidos por la Administración para los certificados LAECSP.
		14.3.1 Tipo certificado	Naturaleza del certificado / tipo de certificado. ID Campo/Valor: 2.16.724.1.3.5.3.2.1 =certificado electrónico de empleado público	Sí	UTF8 String.
		14.3.2 Entidad suscriptora	Nombre de la entidad propietaria del certificado. Id Campo/Valor: 2.16.724.1.3.5.3.2.2=<Entidad Suscriptora>	Sí	UTF8 String. Por ejemplo: 2.16.724.1.3.5.3.2.2=MINISTERIO DE ECONOMÍA Y HACIENDA
		14.3.3 NIF Entidad	Número único de identificación de la entidad (NIF) Id Campo/Valor: 2.16.724.1.3.5.3.2.3 =<NIF>	Sí	UTF8 String, tamaño 9. Por ejemplo: 2.16.724.1.3.5.3.2.3=Q2826004J
		14.3.4 DNI del empleado	Identificador de identidad del suscriptor-custodio de las claves. (NIF). Id Campo/Valor: 2.16.724.1.3.5.3.2.4 =<NIF>	Sí	UTF8 String, tamaño 9. Por ejemplo: 2.16.724.1.3.5.3.2.4=99999999R
		14.3.5 Número de identificación de personal	Número de identificación del suscriptor del certificado (supuestamente unívoco). Número de identificación de funcionario/empleado público Id Campo/Valor: 2.16.724.1.3.5.3.2.5 =<NRP>	Opcional	UTF8 String. Por ejemplo: 2.16.724.1.3.5.3.2.5=ADM12347 Se establecerá el valor del identificador de funcionario/empleado público si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.
		14.3.6 Nombre	Nombre de pila del suscriptor del certificado Id Campo/Valor: 2.16.724.1.3.5.3.2.6 =<Nombre de pila>	Sí	UTF8 String. Por ejemplo: 2.16.724.1.3.5.3.2.6=JUAN
		14.3.7 Apellido 1	Primer apellido del suscriptor del certificado Id Campo/Valor: 2.16.724.1.3.5.3.2.7 =<Apellido 1>	Sí	UTF8 String. Por ejemplo: 2.16.724.1.3.5.3.2.7=ESPAÑOL

Personal APE emitido por la Autoridad de Certificación "AC Administración Pública" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.3.4.4.1					
Campo			Contenido	Obligatoriedad	Especificaciones
		14.3.8 Apellido 2	Segundo apellido del suscriptor del certificado Id Campo/Valor: 2.16.724.1.3.5.3.2.8 =<Apellido 2>	Sí	UTF8 String. Por ejemplo: 2.16.724.1.3.5.3.2.8=ESPAÑOL
		14.3.9 Correo electrónico	Correo electrónico de la persona responsable del certificado. Id Campo/Valor: 2.16.724.1.3.5.3.2.9 =<email de contacto>	Opcional	UTF8 String, tamaño 9. Por ejemplo: 2.16.724.1.3.5.3.2.9=jespanol@meh.es Se establecerá el valor del e-email contacto si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.
		14.3.10 Unidad organizativa	Unidad, dentro de la Administración, en la que desempeña su labor el suscriptor del certificado. Id Campo/Valor: 2.16.724.1.3.5.3.2.10 =<Unidad Organizativa>	Opcional	UTF8 String. Por ejemplo: 2.16.724.1.3.5.3.2.10=SUBDIRECCIÓN DE SISTEMAS DE INFORMACIÓN Se establecerá el valor de la unidad organizativa si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.
		14.3.11 Puesto / cargo	Puesto desempeñado por el suscriptor del certificado dentro de la administración. Id Campo/Valor: 2.16.724.1.3.5.3.2.11 =<Puesto/Cargo>	Opcional	UTF8 String. Por ejemplo: 2.16.724.1.3.5.3.2.11=ANALISTA DE INFORMATICA Se establecerá el valor del puesto de trabajo o cargo si se aporta en la solicitud de certificado. En caso contrario este campo no estará presente en el certificado.
15. CRL Distribution Point			Punto de distribución (localizador) de la CRL	Sí	
	15.1. Distribution Point 1		Punto de publicación de la CRL1 http://www.cert.fnmt.es/crls_acape/CRL<xxx*>.crl *xxx: número entero identificador de la CRL (CRL particionadas)	Sí	UTF8String Ruta donde reside la CRL (punto de distribución 1).
	15.2. Distribution Point 2		Punto de publicación de la CRL2. ldap://ldapape.cert.fnmt.es/CN=CRL<xxx*>,cn=AC%20Administraci%F3n%20P%Fablica,ou=CERES,o=FNMT-RCM,C=ES?certificateRevocationList;binary?base?objectclass=cRLDistributionPoint *xxx: número entero identificador de la CRL (CRL particionadas)	Sí	UTF8String. Ruta del servicio LDAP donde reside la CRL (punto de distribución 2).
16. Authority Info Access				Sí	
	16.1. Access Method 1		Identificador de método de acceso a la información de revocación: 1.3.6.1.5.5.7.48.1 (ocsp)	Sí	Acceso al servicio OCSP

Personal APE emitido por la Autoridad de Certificación "AC Administración Pública" en soporte Tarjeta Criptográfica y bajo el OID 1.3.6.1.4.1.5734.3.3.4.4.1				
Campo		Contenido	Obligatoriedad	Especificaciones
	16.2. Acces Location 1	http://ocspap.cert.fnmt.es/ocspap/OcspResponder	Si	URL para acceder al servicio OCSP. No es necesario firmar las peticiones OCSP
	16.3. Access Method 2	Identificador de método de acceso a la información de certificados adicionales necesarios para la validación: 1.3.6.1.5.5.7.48.2 (ca cert)	Si	Emisor de la entidad emisora de certificados (CA Raíz) De la rfc 5280: *"the id-ad-caIssuers OID is used when the additional information lists certificates that were issued to the CA that issued the certificate containing this extension. The referenced CA issuers description is intended to aid certificate users in the selection of a certification path that terminates at a point trusted by the certificate user."*
	16.4. Acces Location 2	http://www.cert.fnmt.es/certs/ACAP.crt	Si	Ruta para descarga de certificados adicionales para la validación de la cadena de certificación. En este caso la ruta del certificado raíz de la FNMT-RCM.
17. Basic Contraints		Esta extensión sirve para identificar si el sujeto de certificación es una CA así como el máximo nivel de "profundidad" permitido para las cadenas de certificación". También sirve para distinguir una CA de las entidades finales	Si	De la rf5280: *" This extension MAY appear as a critical or non-critical extension in end entity certificates.*
	17.1. Subject Type	Entidad final (valor FALSE)		Con este certificado no se pueden emitir otros

Ilustración 51: Perfil certificado del personal de la administración pública FNMT AP

Como podemos comprobar viendo el perfil, los certificados FNMT AP también poseen información referente a la identidad alternativa del suscriptor, lo que nos permite obtener de forma sencilla datos como el nombre, los apellidos, el DNI y todos los datos recogidos en el punto 14. Subject Alternative Names. Observando el perfil del certificado FNMT AP se ven diferencias con respecto a los certificados APE. Este certificado, que también es específico para los empleados de las administraciones públicas tiene una serie de campos destinados a guardar información relacionada con la condición de empleado público del suscriptor. Estos campos específicos son los siguientes:

- Entidad suscriptora
- Nif entidad suscriptora
- Número de identificación personal
- Unidad organizativa
- Puesto / Cargo

Según nos indica el documento de prácticas de certificación de este certificado las extensiones críticas de este perfil son **KeyUsage** y **BasicConstraints**. Se debe comprobar como parte de la validación de este tipo de certificados que ambas extensiones están presentes en el certificado.

Por último para poder identificar correctamente los certificados de la FNMT de tipo AP tenemos que mirar el campo 14.3.1 TipoCertificado que debe tener el siguiente valor:

“certificado electrónico de empleado público”

4.7.2 DNIe

El DNIe o Documento Nacional de Identidad electrónico es emitido por la dirección general de la policía. Permite al ciudadano establecer sus relaciones de confianza con terceros a través de las nuevas tecnologías, tal y como lo lleva haciendo durante más de 50 años con el DNI tradicional. La idea era crear toda una infraestructura de clave pública asociada esta vez al Documento Nacional de Identidad de cara a facilitar y normalizar el uso de certificados digitales por parte de la ciudadanía de forma fácil e intuitiva. Así pues, como se ha podido comprobar con el paso de los años se han ido sustituyendo los antiguos documentos de identidad por nuevos DNIe en smart cards con capacidad para almacenar y utilizar certificados digitales.

El documento de prácticas de certificación del DNIe se puede consultar en la siguiente URL:

http://www.dnielectronico.es/PDFs/politicas_de_certificacion.pdf

Según este documento, la cadena de certificación de los certificados DNI electrónicos es la siguiente:

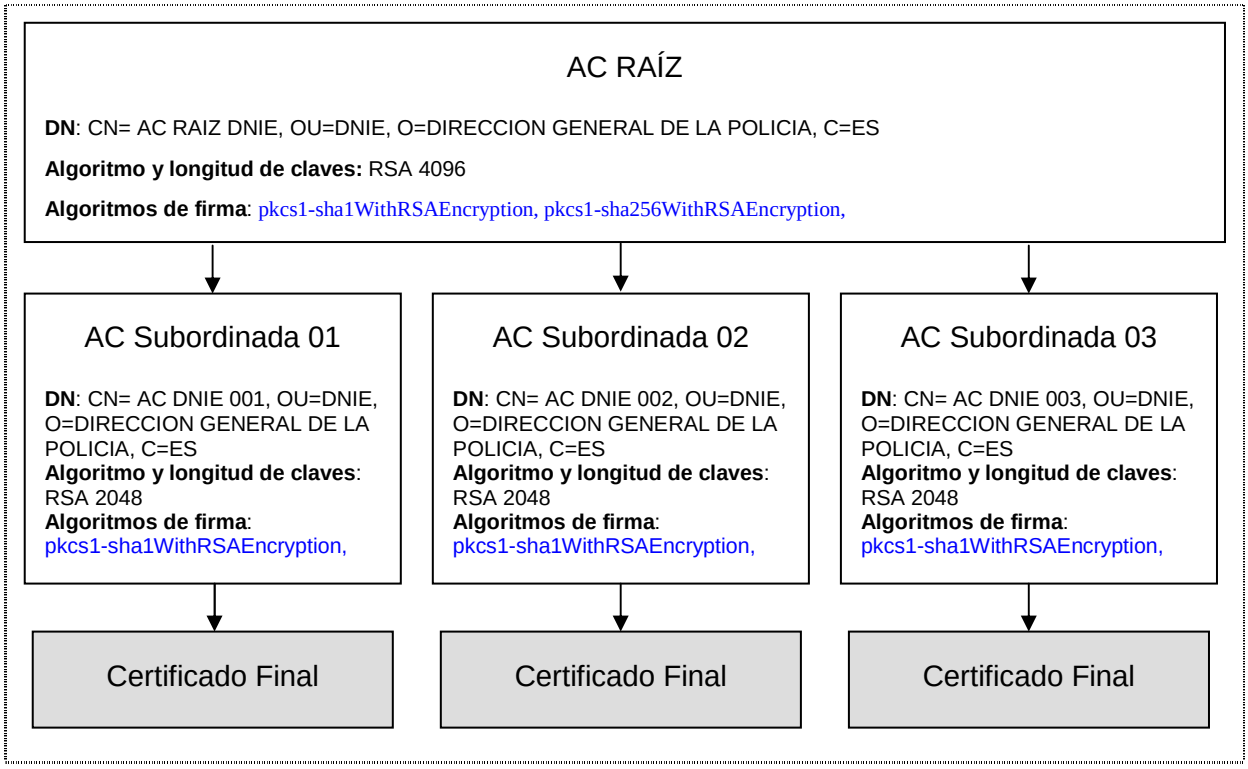


Ilustración 52: Cadena de certificación DNIe

Como vemos en el caso del DNIe tenemos una entidad raíz que es la encargada de firmar las 3 entidades intermedias o subordinadas. Cada una de estas 3 entidades

intermedias tiene capacidad para firmar certificados de tipo DNIe, por tanto será necesario incorporar al proyecto MPC las partes públicas de estas entidades subordinadas de cara a poder verificar la firma de los certificados finales tipo DNIe.

Consultando el documento de prácticas de certificación vemos que los certificados digitales DNIe siempre están asociados a una persona física, por tanto en este caso no sería posible que nuestra aplicación devolviera “Jurídica” en el campo TipoPersona.

Existen para cada documento nacional de identidad electrónico dos certificados asociados a cada titular, uno para la firma de documentos y otro para la autenticación. Ambos entran dentro del ámbito de este proyecto. A continuación se muestra el perfil de los certificados DNIe según aparecen en el documento DPC de la dirección general de la policía:

Certificado de Firma de Ciudadano		
CAMPO	**CONTENIDO**	**CRÍTICA para extensiones**
Campos de X509v1		
1. Versión	V3	
2. Serial Number	No secuencial	
3. Signature Algorithm	SHA256withRSAEncryption SHA1withRSAEncryption	
4. Issuer Distinguished Name	CN=AC DNIE XXX [4] OU=DNIE O=DIRECCION GENERAL DE LA POLICIA C=ES	
5. Validez	30 meses	
6. Subject	CN=APELLIDO1 APELLIDO2, NOMBRE (FIRMA) G=NOMBRE SN=APELLIDO1 NÚMERO DE SERIE=DNI (con letra) C=ES	
7. Subject Public Key Info	Algoritmo: RSA Encryption Longitud clave: 2048 bits	
Campos de X509v2		
1. issuerUniqueIdentifier	No se utilizará	
2. subjectUniqueIdentifier	No se utilizará	
Extensiones de X509v3		
1. Subject Key Identifier	Derivada de utilizar la función de hash SHA-1 sobre la clave pública del sujeto.	NO
2. Authority Key Identifier	Derivada de utilizar la función de hash SHA-1 sobre la clave pública de la AC emisora.	NO

Certificado de Firma de Ciudadano		
CAMPO	**CONTENIDO**	**CRÍTICA para extensiones**
3. KeyUsage		SI
Digital Signature	0	
ContentCommitment	1	
Key Encipherment	0	
Data Encipherment	0	
Key Agreement	0	
Key Certificate Signature	0	
CRL Signature	0	
4.extKeyUsage	No se utilizará	
5. privateKeyUsagePeriod	No se utilizará	
6. Certificate Policies		NO
Policy Identifier	2.16.724.1.2.2.2.3	
URL DPC	http://www.dnie.es/dpc	
Notice Reference		
7.Policy Mappings		
8. Subject Alternate Names	No se utilizará	NO
9. Issuer Alternate Names	No se utilizará	
10. Subject Directory Attributes	dateOfBirth	
11. Basic Constraints		SI
Subject Type	Entidad Final	
Path Length Constraint	No se utilizará	
12. Policy Constraints	No se utilizará	
13. CRLDistributionPoints	No se utilizará	NO
14. Auth. Information Access	OCSP http://ocsp.dnie.es CA http://www.dnie.es/certs/ACraiz.crt	NO
15.netscapeCertType	No se utilizará	
16. netscapeRevocationURL	No procede	
17. netscapeCAPolicyURL	No procede	
18. netscapeComment	No procede	
19. Biometricinfo	Hash de los datos biométricos SHA256/SHA1	NO
20. personalDataInfo (2.16.724.1.2.2.3.1)	Hash de los datos biográficos (datos impresos en el DNIe) SHA1/SHA256	
21. qcstatements	id-etsi-qcs-QcCompliance id-etsi-qcs-QcSSCD	

Ilustración 53: Perfil certificado de firma DNIe

Certificado de Autenticación de Ciudadano		
CAMPO	**CONTENIDO**	**CRÍTICA para extensiones**
Campos de X509v1		
1. Versión	V3	
2. Serial Number	No secuencial	
3. Signature Algorithm	SHA256withRSAEncryption SHA1withRSAEncryption	
4. Issuer Distinguished Name	CN=AC DNIE XXX OU=DNIE O=DIRECCION GENERAL DE LA POLICIA C=ES	
5. Validez	30 meses	
6. Subject	CN=APELLIDO1 APELLIDO2, NOMBRE (AUTENTICACIÓN) G=NOMBRE SN=APELLIDO1 NÚMERO DE SERIE=DNI (con letra) C=ES	
7. Subject Public Key Info	Algoritmo: RSA Encryption Longitud clave: 2048 bits	
Campos de X509v2		
1. issuerUniqueIdentifier	No se utilizará	
2. subjectUniqueIdentifier	No se utilizará	
Extensiones de X509v3		
1. Subject Key Identifier	Derivada de utilizar la función de hash SHA-1 sobre la clave pública del sujeto.	NO
2. Authority Key Identifier	Derivada de utilizar la función de hash SHA-1 sobre la clave pública de la AC emisora.	NO
3. KeyUsage		SI
Digital Signature	1	
ContentCommitment	0	
Key Encipherment	0	
Data Encipherment	0	
Key Agreement	0	
Key Certificate Signature	0	
CRL Signature	0	
4.extKeyUsage	No se utilizará	
5. privateKeyUsagePeriod	No se utilizará	
6. Certificate Policies		NO
Policy Identifier	2.16.724.1.2.2.2.4	
URL DPC	http://www.dnie.es/dpc	

Certificado de Autenticación de Ciudadano		
CAMPO	**CONTENIDO**	**CRÍTICA para extensiones**
Notice Reference		
7.Policy Mappings		
8. Subject Alternate Names	No se utilizará	NO
9. Issuer Alternate Names	No se utilizará	
10. Subject Directory Attributes	dateOfBirth	
11. Basic Constraints		SI
Subject Type	Entidad Final	
Path Length Constraint	No se utilizará	
12. Policy Constraints	No se utilizará	
13. CRLDistributionPoints	No se utilizará	NO
14. Auth. Information Access	OCSP http://ocsp.dnie.es CA http://www.dnie.es/certs/ACraiz.crt	NO
15.netscapeCertType	No se utilizará	
16. netscapeRevocationURL	No procede	
17. netscapeCAPolicyURL	No procede	
18. netscapeComment	No procede	
19. Biometricinfo	Hash de los datos biométricos SHA256/SHA1	NO
20. personalDataInfo (2.16.724.1.2.2.3.1)	Hash de los datos biográficos (datos impresos en el DNI-e) SHA1/SHA256	
21. qcstatements	id-etsi-qcs-QcCompliance id-etsi-qcs-QcSSCD	

Ilustración 54: Perfil certificado de autenticación DNIe

Los certificados de tipo DNIe no poseen campos especialmente relevantes o diferentes a los de la FNMT. De hecho poseen mensos campos para parsear ya aquí no están presentes los específicos para las administraciones públicas. La particularidad de los certificados de DNIe reside en que al existir un certificado para firma y otro para autenticación se debe hacer constar esto a la hora de parsear sus datos. Para determinar si es o no un certificado de firma o auntenticación debemos mirar el campo Policy Identifier dentro de Certificate Policies. Si nos encontramos con 2.16.724.1.2.2.2.4 será un certificado de autenticación y si en cambio tenemos 2.16.724.1.2.2.2.3 será un certificado de firma. Otra forma de averiguar el tipo de certificado es mirando el campo del valor Digital Signature dentro de Key Usage. Contrariamente a lo que nos diría la intuición y según el documento de DPC, si el

campo Digital Signature contiene un 1 (verdadero) el certificado será de autenticación y si contiene un 0 (falso) será de firma.

Es importante hacer constar que en este caso los perfiles de certificados no incluyen la identidad alternativa del suscriptor, por tanto para extraer el nombre, apellidos y nif se tendrán que utilizar expresiones regulares.

Según nos indica el documento de prácticas de certificación de este certificado las extensiones críticas de este certificado son **KeyUsage** y **BasicConstraints**. Se debe comprobar como parte de la validación de este tipo de certificados que ambas extensiones están presentes en el certificado.

4.8 Definición de los datos de salida

De todos los campos que proporciona el estándar x509 no todos son importantes para nuestra aplicación. Después de las reuniones con el cliente se especificaron los campos fundamentales que se quieren recoger en el parseo y que son los que al consultar a las áreas implicadas más se utilizan en sus correspondientes aplicaciones. Se van a parsear cuatro tipos de certificados diferentes, según su entidad certificadora que son los siguientes:

- Fnmt
- Dnie
- FnmtApe
- FnmtAp

Muchas aplicaciones que utilizarán MPC, lo harán para permitir acceder al usuario a sus sistemas y rellenar formularios con los campos recogidos en el mismo. También se establecieron en la reuniones los criterios de validación que se deben seguir, se optó por una validación simple (caducidad, integridad y confianza) ya que la importancia de la aplicación radica en el parseo de datos y no así en la validación de los mismos mediante OCSP, la cual sería añadida más adelante en futuras mejoras.

4.8.1 Datos de salida Generales

Como salida común a todos los métodos que implementará el sistema se quieren obtener los siguientes campos:

- **Fecha consulta:** Indicará la hora exacta en que se realizó la consulta al servicio MPC. La fecha incluirá horas, minutos y segundos.
- **Versión Respuesta:** Indicará la versión del formato de salida utilizada. De esta forma, si en el futuro se van añadiendo nuevos campos a la salida del sistema este valor se irá incrementando y quedará documentado en un histórico de versiones indicando las novedades de cada nueva versión.

4.8.2 Datos de salida para la Obtención de información

A continuación se enumeran y describen los campos que se quieren obtener parseados cuando se escoja la obtención de datos de un certificado. Estos campos son fruto del análisis del las políticas de certificación de los 4 tipos de certificados que se parsearán así como del propio estándar X509. Los campos que se quieren obtener son los siguientes:

- **Número de serie:** Número de serie del certificado digital. Las entidades de certificación deben garantizar que sea un entero positivo. También debe ser único para cada certificado expedido por una misma entidad de certificación.
- **Emisor (Issuer):** Este campo identifica la entidad que ha firmado y emitido el certificado digital. Debe contener un Nombre Distintivo (DN Distinguished Name) no vacío definido como del tipo Name del estándar X.509, compuesto de atributos y valores que identifican al emisor.
- **Asunto (Subject):** Este campo identifica la entidad asociada a la clave pública almacenada en el certificado. Representa la persona o entidad titular del certificado y su formato es similar al emisor. Debe contener un Nombre Distintivo (DN Distinguished Name) definido como del tipo Name del estándar X.509, compuesto de atributos y valores que identifican al titular.
- **Periodo de Validez:** Es el intervalo de fechas entre las cuales la entidad certificadora emisora garantiza tener información sobre el estado de validez del certificado. Se compone de dos fechas, la fecha de comienzo del periodo de validez, antes de la cual el certificado no puede considerarse válido y la fecha de finalización del periodo de validez, después de la cual un certificado se considerará caducado. Ambas fechas deben expresarse en UTCTime (finalización antes de 2049) o en GeneralizedTime (finalización en 2050 o posterior). Incluye horas, minutos y segundos.
- **Entidad Certificadora:** Se quiere obtener la entidad certificado parseada del campo Emisor. Más concretamente se quiere el valor del atributo organizationName (O) del campo emisor, también conocido como Nombre de la organización emisora o razón social.
- **Unidad Entidad Certificadora:** Se quiere obtener la Unidad correspondiente a la entidad certificadora del campo Emisor. Más concretamente se quiere el

valor del atributo organizationalUnitName (OU) del campo emisor, también conocido como Unidad organizativa dentro de la organización emisora.

- **Datos Adicionales Entidad Certificadora:** Siempre que esté presente se quiere obtener parseado el valor del atributo commonName (CN) del campo emisor.
- **Tipo Entidad Certificadora:** Será un número entero que identificará el tipo de certificado soportado por el sistema según su entidad emisora. Los posibles valores serán:
    - *0 Fnmt.* Certificados de tipo Clase 2 CA de la Fábrica nacional de moneda y timbre.
    - *1 Dnie:* Certificados de Dnie electrónico emitidos por la dirección general de la policía.
    - *2: FnmtApe:* Certificados de empleado público emitidos por la Fábrica nacional de moneda y timbre (OU = AC APE)
    - *3 FnmtAp:* Certificados de empleado público emitidos por la Fábrica nacional de moneda y timbre (CN = AC Administración Pública)

- **Tipo Persona:** Será un número entero que identificará si el titular del certificado es una persona física o jurídica. Aunque existen certificados que no son ni una cosa ni la otra (Certificados de Sello por ejemplo) no se tendrán en cuenta en la clasificación ya que solo se parsearán certificados de persona física o jurídica. Los posibles valores serán:
    - *0 Física:* Certificados cuyo titular sea una persona física según su política de certificación.
    - *1 Jurídica:* Certificados cuyo titular sea una persona jurídica (entidad) según lo indique su política de certificación.
- **Entidad Jurídica:** En caso de que el titular del certificado sea una entidad jurídica se quiere obtener el nombre de la misma.
- **CIF:** En caso de que el titular del certificado sea una entidad jurídica se quiere obtener el CIF (Código de Identificación Fiscal) de la misma.
- **Propósito:** Solo aparecerá cuando el certificado sea un DNI. El propósito indica si el certificado es para Autenticación o para Firma electrónica. Un

ciudadano al obtener el DNIe recibe un certificado de cada tipo. Los valores posibles serán:

  - *0 Certificado de autenticación.*
  - *1 Certificado de firma*

- **Nombre Completo:** Nombre completo del titular del certificado, sin separar los apellidos del nombre.
- **Primer Apellido:** Primer apellido parseado del titular del certificado.
- **Segundo Apellido:** Segundo apellido parseado del titular del certificado.
- **Nombre:** Nombre parseado del titular del certificado.
- **Nif:** Número de documento nacional de identidad del titular del certificado.
- **País:** Estado cuyas leyes rigen la expedición del certificado digital.
- **Email:** Dirección de correo electrónico del titular del certificado.
- **Entidad Suscriptora:** Nombre de la entidad propietaria del certificado. No confundir con entidad certificadora ni con entidad jurídica. Aparece solo en certificados de la administración pública. Ejemplo: *Ministerio de Industria, Energía y Turismo.*
- **Nif Entidad Suscriptora:** Número de identificación fiscal correspondiente a la entidad suscriptora del certificado. Solo aparece en certificados de la administración pública.
- **Número de identificación personal:** Número de identificación único del suscriptor del certificado. Identifica a un funcionario/empleado público dentro del organismo al que pertenece. Solo aparece en certificados de la administración pública.
- **Unidad Organizativa:** Unidad dentro de la entidad suscriptora en la que desempeña su labor el titular del certificado. Solo aparecerá en certificados de la administración pública. Ejemplo: *Subdirección de sistemas de información*.
- **Cargo:** Puesto desempeñado por el titular del certificado dentro de la entidad suscriptora. Solo aparece en certificados de la administración pública. Ejemplo: *Analista informático*.

- **Situación laboral:** Estado laboral del titular del certificado dentro de la Entidad suscriptora. Este campo aparece exclusivamente en certificados de empleado público de tipo FnmtApe. Ejemplo: *Personal Laboral.*

4.8.3 Datos de salida para la Validación del certificado

Como se ha indicado antes, el objetivo principal de MPC es facilitar información parseada de un certificado proporcionado por una aplicación. Así pues en lo que respecta a validación, no se ha querido implementar los mecanismos para validar completamente el certificado (No se consulta a OCSP o CRL para saber el estado de revocación del mismo). Aún siendo esto así si que se ha querido implementar una validación más de tipo local atendiendo a los criterios de caducidad, integridad y confianza del certificado. Para que el sistema MPC de un certificado por válido se debe comprobar las siguientes características:

- ✓ Caducidad
- ✓ Inicio periodo validez
- ✓ Confianza del emisor
- ✓ Validez de la firma
- ✓ Extensiones

Para representar esto se quiere que la salida correspondiente a la validación del certificado incluya los siguientes campos:

- **Valido:** Código numérico que indicará de forma resumida si el certificado es o no válido. Los valores posibles serán
  - 0 *No válido*. El certificado se considera no válido.
  - 1 *Válido.* El certificado se considera válido.
- **Resultado Validación:** Resumen en texto del campo Válido. Los posibles valores serán:
  - *Certificado OK.*
  - *El certificado no pasó la validación,*

- **Código Validación Simple:** Entero que representa numéricamente el resultado de la validación. Aparecerá solo cuando se haya elegido validación. Los valores posibles serán:
    - 0 *Validación satisfactoria*. El certificado es válido, ha pasado todas las validaciones de caducidad, integridad y confianza de forma satisfactoria.
    - 1 *Certificado caducado.* El certificado está caducado en la fecha de consulta. La fecha de consulta es posterior a la fecha de fin del periodo de validez del certificado.
    - 2 Certificado aún no válido. El certificado aún no es válido. La fecha de consulta es anterior a la fecha de inicio del periodo de validez del certificado.
    - 3 *Firma no válida.* La firma del certificado no es válida. Los certificados digitales vienen firmados con la clave privada de su certificado emisor correspondiente a su CA. En esta validación se comprueba que la firma del certificado haya sido realizada por su certificado emisor. Para esto se utiliza la clave pública del emisor.
    - 4 *Emisor no es de confianza*. El certificado ha sido emitido por un emisor que no es de confianza. El certificado emisor debe ser uno de los almacenados en el sistema.
    - 5 *Extensiones no válidas*. El certificado no posee extensiones marcadas como críticas en su política de certificación. Si una extensión es crítica es obligatorio que este presente en el certificado para que este se considere válido.
- **Descripción validación simple:** Texto con una breve descripción del significado del código de validación obtenido. Los posibles valores serán:
    - *Validación satisfactoria.*
    - *Certificado caducado.*
    - *Certificado aún no válido.*
    - *Firma no válida.*
    - *El emisor del certificado no es de confianza o no se encuentra registrado.*
    - *El certificado posee extensiones que no son válidas.*

4.8.4 Datos de salida para Errores

En los casos en los que se produzca algún error en la ejecución del servicio web debe mostrarse de forma clara para que se pueda determinar el origen y posible solución del mismo. Así pues deben aparecer los siguientes campos:

- **Origen error:** Número entero que representa el origen o parte del programa dónde se ha producido el error. Los posibles valores son los siguientes:
    - 1 *Obtener información del certificado*. El error se ha producido al obtener información del certificado
    - 2 *Validar el certificado.* El error se ha producido al obtener la validación del certificado
    - 3 *Obtener información y validar certificado*. El error se ha producido al obtener información y validar el certificado.
    - -1 Desconocido. El origen del error no se ha podido determinar (Valor por defecto)
- **Código error:** Número entero que identifica un tipo de error producido de forma general. Es una clasificación general del tipo de error producido, sin especificar los detalles. Los posibles valores son:
    - 1 *Formato de entrada no válido.* Error producido al comprobar los parámetros de entrada del servicio web. Puede producirse porque se ha dejado un campo en blanco, el tipo del parámetro no coincide con el esperado, la cadena del certificado no está bien formada, etc.
    - 2 *Certificado no soportado por el sistema.* Error producido al pasarle al sistema un certificado que no está incluido en la lista de los admitidos por el sistema. Ejemplo, un certificado no emitido por la Fábrica nacional de moneda y timbre o por la dirección general de la policía o un certificado de sede o sello.
    - 3 *Error al obtener los datos parseados del sistema.* Error producido al intentar obtener algún dato del certificado.
    - 4 *Error al validar el certificado.* Error producido al intentar realizar la validación del certificado.
    - 5 *Error al obtener los datos parseados del sistema y al validar el certificado. Se produce cuando se producen a la vez errores de tipo 3 y 4.*

- *6 Error de conexión con base de datos.* Error producido al intentar conectar, leer o insertar en la base de datos de estadísticas.
- *7 Error desconocido.* Error general de tipo indeterminado.

- **Descripción error:** descripción en texto del código de error.
- **Detalle error:** En caso de estar activado en el archivo de configuración de la aplicación el sistema devolverá el detalle del error cuando este se produzca. El campo detalle error es un campo de texto que proporciona información adicional (más específica y concreta) de la excepción producida. Los posibles valores serán:
  - *MPC01: Parámetro de entrada no válido. El formato del certificado no es correcto.* Mensaje mostrado en caso de pasarle como parámetro "certificado" algo que no sea una cadena de texto en base 64.
  - *MPC02: Parámetro de entrada nulo. El certificado no puede ser nulo.* Excepción producida en caso de dejar vacío el campo certificado.
  - *MPC03: Parámetro de entrada no válido. El identificador de la aplicación no es correcto, debe ser un valor mayor que 0.* Excepción producida cuando se introduce un valor negativo como identificador de aplicación y también cuando este no está dado de alta en la base de datos.
  - *MPC04: Parámetro de entrada no válido. El modo de validación debe ser uno de los valores: 0 (validación simple) o -1 (sin validación).*
  - *MPC05: Error al intentar convertir el certificado a un formato válido.* El formato del certificado es incorrecto, es una cadena en base64 pero se produce una excepción al convertirlo a x509.
  - *MPC06: Error al leer el archivo de configuración.*
  - *MPC07: No se ha podido identificar el prestador al que pertenece el certificado.* Excepción desconocida al identificar el prestador.
  - *MPC08: Certificado no soportado por el sistema.* El certificado pasado como parámetro no se encuentra entre los que el sistema es capaz de parsear y validar.
  - *MPC09: Error al comprobar la política del certificado.* Error al determinar el tipo persona o si el certificado es de firma o autenticación en caso de ser DNIe.
  - *MPC10: Error al parsear los datos del emisor del certificado.*
  - *MPC11: Error al parsear los datos del asunto del certificado.*

- *MPC12: Error al parsear los nombres alternativos del certificado.*
- *MPC13: Error al validar el estado de caducidad del certificado.*
- *MPC14: Error al validar que la fecha actual no sea anterior a la fecha de inicio del periodo de validez del certificado.*
- *MPC15: Error al validar la confianza del emisor del certificado.* Excepción producida al buscar en la lista de emisores de confianza de la aplicación el emisor del certificado pasado como parámetro.
- *MPC16: Error al validar la firma del certificado.* Excepción producida al comprobar con la clave pública de su emisor que la firma del certificado pasado como parámetro es correcta.
- *MPC17: Error al validar las extensiones del certificado.* Excepción producida al comprobar la existencia de las extensiones críticas del certificado según establece su DPC.
- *MPC18: Error al insertar estadísticas en la base de datos.*
- *MPC19: Error al buscar el identificador de la aplicación en la base de datos.*
- *MPC20: Error desconocido del MPC.* Error de tipo general, ocurre cuando se produce una excepción inesperada y por tanto no clasificada entre las anteriores.

5 Diseño del sistema

5.1 Definición del XML de salida

Basándonos en los datos recogidos en el análisis y en los requisitos se ha creado la estructura que tendrá el xml de salida del MPC. Para ello se ha definido el esquema XSD. Primeramente se va a mostrar de forma esquemática para una mejor comprensión del mismo.

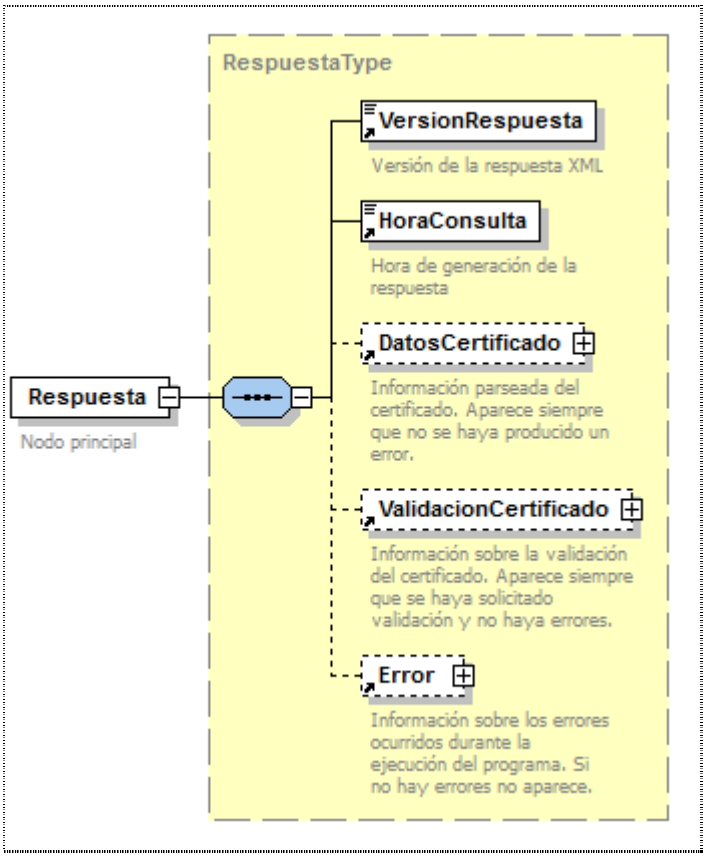
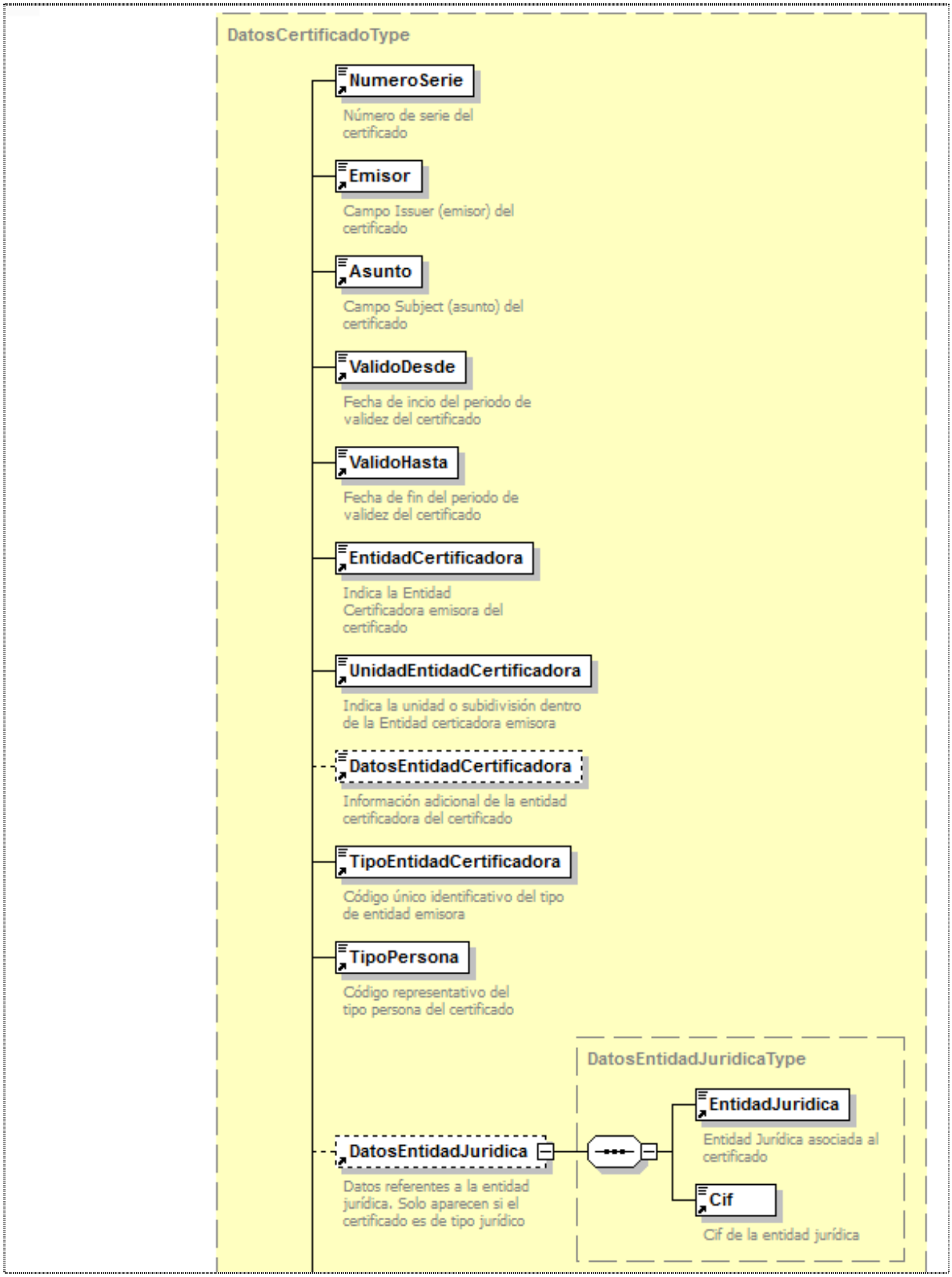


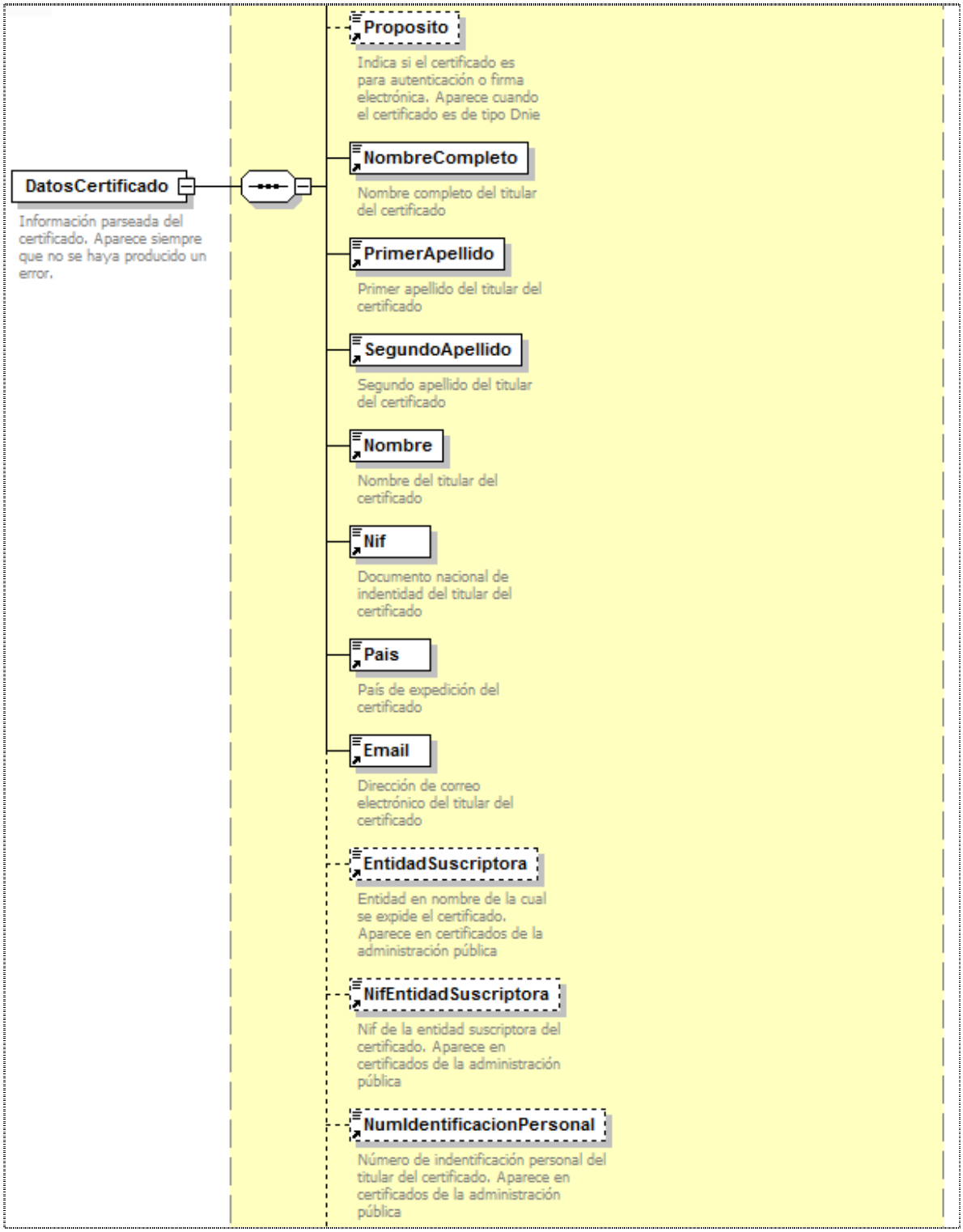
Ilustración 55: Diagrama Xsd resumido

Existe un nodo **Respuesta** principal en el cual estará agrupada toda la información de salida del XML. Este nodo siempre aparecerá. Los nodos marcados con linea continua aparecerán siempre. Los marcados con línea discontinua solo aparecerán en algunos casos. Dentro del nodo respuesta aparecerán siempre el nodo **VersionRespuesta** y el nodo **HoraConsulta.** Los nodos **DatosCertificado**, **ValidacionCertificado** y **Error** aparecerán o no según las opciones elegidas y los errores producidos.

A continuación se van a expandir los nodos DatosCertificado, ValidacionCertificado y Error para mostrar su contenido.

Nodo Datos Certificado:




DatosCertificado
Información parseada del certificado. Aparece siempre que no se haya producido un error.
Proposito
Indica si el certificado es para autenticación o firma electrónica. Aparece cuando el certificado es de tipo Dnie
NombreCompleto
Nombre completo del titular del certificado
PrimerApellido
Primer apellido del titular del certificado
SegundoApellido
Segundo apellido del titular del certificado
Nombre
Nombre del titular del certificado
Nif
Documento nacional de indentidad del titular del certificado
Pais
País de expedición del certificado
Email
Dirección de correo electrónico del titular del certificado
EntidadSuscriptora
Entidad en nombre de la cual se expide el certificado. Aparece en certificados de la administración pública
NifEntidadSuscriptora
Nif de la entidad suscriptora del certificado. Aparece en certificados de la administración pública
NumIdentificacionPersonal
Número de indentificación personal del titular del certificado. Aparece en certificados de la administración pública

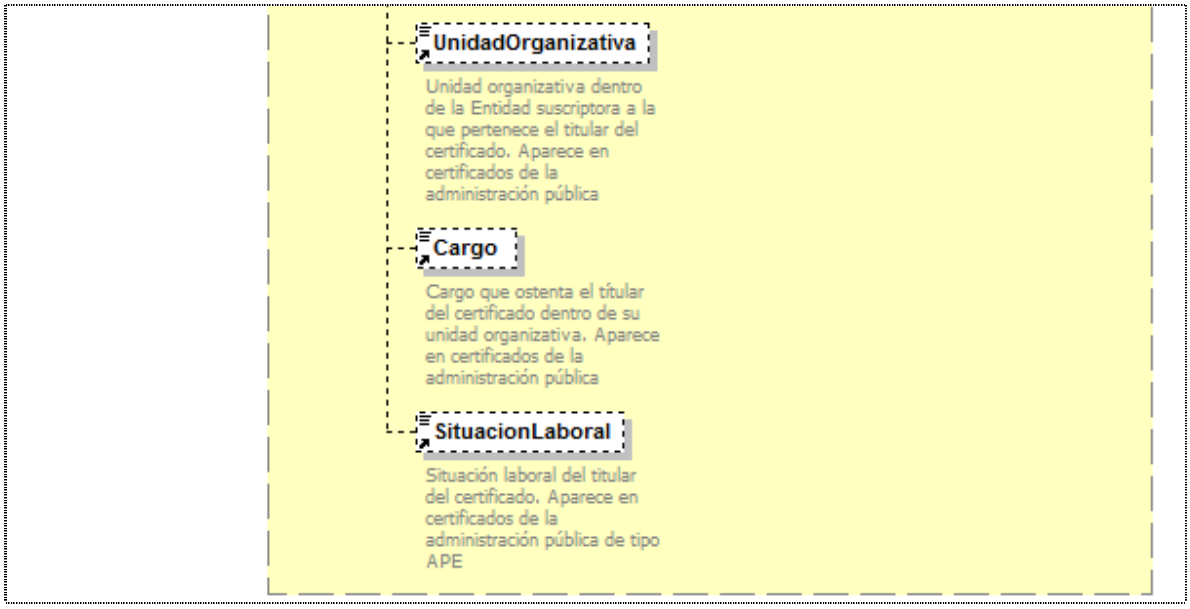


Ilustración 56: Diagrama Xsd Nodo Datos Certificado

Nodo Validación Certificado:

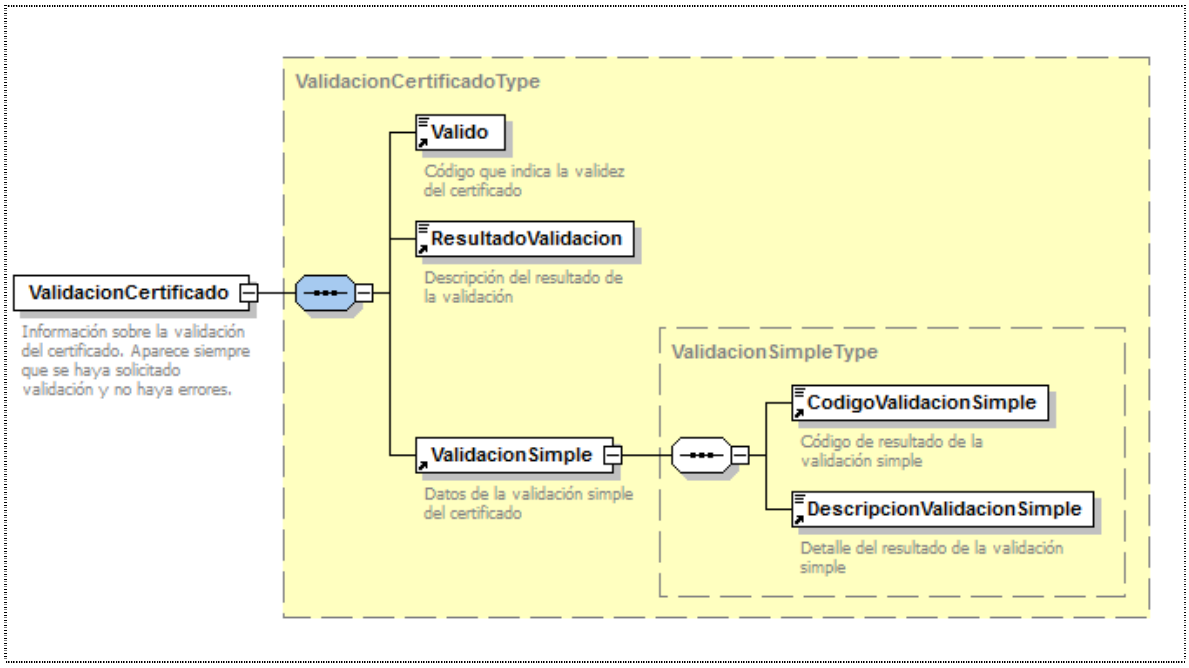


Ilustración 57: Diagrama Xsd Nodo Validación Certificado

Nodo Error:

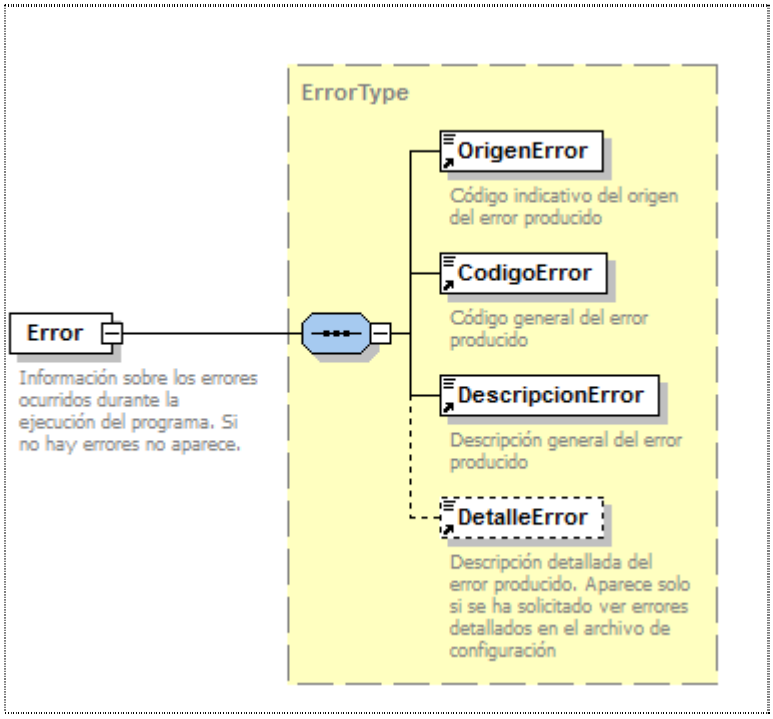


Ilustración 58: Diagrama Xsd Nodo Error

A continuación se muestra la definición formal del esquema XSD para el método ObtenerInformacion():

Primeramente están definidos los elementos:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.w3schools.com"
targetNamespace="http://www.w3schools.com">
<!-- Elementos -->
	<xs:element name="Respuesta" type="RespuestaType">
		<xs:annotation>
			<xs:documentation>Nodo principal</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:element name="DatosCertificado" type="DatosCertificadoType">
		<xs:annotation>
			<xs:documentation>Información parseada del certificado. Aparece siempre
que no se haya producido un error.</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:element name="ValidacionCertificado" type="ValidacionCertificadoType">
		<xs:annotation>
			<xs:documentation>Información sobre la validación del certificado. Aparece
siempre que se haya solicitado validación y no haya errores.</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:element name="Error" type="ErrorType">
		<xs:annotation>
			<xs:documentation>Información sobre los errores ocurridos durante la
ejecución del programa. Si no hay errores no aparece.</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:element name="VersionRespuesta" type="VersionRespuestaType">
		<xs:annotation>
```

```
                        <xs:documentation>Versión de la respuesta XML</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="HoraConsulta" type="HoraConsultaType">
                <xs:annotation>
                        <xs:documentation>Hora de generación de la respuesta</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="NumeroSerie" type="NumeroSerieType">
                <xs:annotation>
                        <xs:documentation>Número de serie del certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Emisor" type="EmisorType">
                <xs:annotation>
                        <xs:documentation>Campo Issuer (emisor) del certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Asunto" type="AsuntoType">
                <xs:annotation>
                        <xs:documentation>Campo Subject (asunto) del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="ValidoDesde" type="ValidoDesdeType">
                <xs:annotation>
                        <xs:documentation>Fecha de incio del periodo de validez del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="ValidoHasta" type="ValidoHastaType">
                <xs:annotation>
                        <xs:documentation>Fecha de fin del periodo de validez del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="EntidadCertificadora" type="EntidadCertificadoraType">
                <xs:annotation>
                        <xs:documentation>Indica la Entidad Certificadora emisora del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="UnidadEntidadCertificadora" type="UnidadEntidadCertificadoraType">
                <xs:annotation>
                        <xs:documentation>Indica la unidad o subidivisión dentro de la Entidad
certicadora emisora</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="DatosEntidadCertificadora" type="DatosEntidadCertificadoraType">
                <xs:annotation>
                        <xs:documentation>Información adicional de la entidad certificadora del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="TipoEntidadCertificadora" type="TipoEntidadCertificadoraType">
                <xs:annotation>
                        <xs:documentation>Código único identificativo del tipo de entidad
emisora</xs:documentation>
                        <xs:documentation>Los posibles valores son:
                                0 Fnmt
                                1 Dnie
                                2 FnmtApe
                                3 FnmtAp
                                4 Otro
                           -1 Desconocido (No se ha podido determinar)
                </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="TipoPersona" type="TipoPersonaType">
                <xs:annotation>
                        <xs:documentation>Código representativo del tipo persona del
certificado</xs:documentation>
                        <xs:documentation>Los posibles valores son:
                                0 Persona Física
                                1 Persona Jurídica
                           -1 Desconocido (No se ha podido determinar)
                        </xs:documentation>
```

```
                </xs:annotation>
        </xs:element>
        <xs:element name="DatosEntidadJuridica" type="DatosEntidadJuridicaType">
                <xs:annotation>
                        <xs:documentation>Datos referentes a la entidad jurídica. Solo aparecen si
el certificado es de tipo jurídico</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="EntidadJuridica" type="EntidadJuridicaType">
                <xs:annotation>
                        <xs:documentation>Entidad Jurídica asociada al
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Cif" type="CifType">
                <xs:annotation>
                        <xs:documentation>Cif de la entidad jurídica</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Proposito" type="PropositoType">
                <xs:annotation>
                        <xs:documentation>Indica si el certificado es para autenticación o firma
electrónica. Aparece cuando el certificado es de tipo Dnie</xs:documentation>
                        <xs:documentation>Los valores posibles son:
                                0 Certificado de autenticación
                                1 Certificado para firma electrónica
                           -1 Desconocido (No se ha podido determinar)
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="NombreCompleto" type="NombreCompletoType">
                <xs:annotation>
                        <xs:documentation>Nombre completo del titular del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="PrimerApellido" type="PrimerApellidoType">
                <xs:annotation>
                        <xs:documentation>Primer apellido del titular del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="SegundoApellido" type="SegundoApellidoType">
                <xs:annotation>
                        <xs:documentation>Segundo apellido del titular del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Nombre" type="NombreType">
                <xs:annotation>
                        <xs:documentation>Nombre del titular del certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Nif" type="NifType">
                <xs:annotation>
                        <xs:documentation>Documento nacional de indentidad del titular del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Pais" type="PaisType">
                <xs:annotation>
                        <xs:documentation>País de expedición del certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Email" type="EmailType">
                <xs:annotation>
                        <xs:documentation>Dirección de correo electrónico del titular del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="EntidadSuscriptora" type="EntidadSuscriptoraType">
                <xs:annotation>
                        <xs:documentation>Entidad en nombre de la cual se expide el certificado.
Aparece en certificados de la administración pública</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="NifEntidadSuscriptora" type="NifEntidadSuscriptoraType">
```

```
                <xs:annotation>
                        <xs:documentation>Nif de la entidad suscriptora del certificado. Aparece
en certificados de la administración pública</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="NumIdentificacionPersonal" type="NumIdentificacionPersonalType">
                <xs:annotation>
                        <xs:documentation>Número de indentificación personal del titular del
certificado. Aparece en certificados de la administración pública</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="UnidadOrganizativa" type="UnidadOrganizativaType">
                <xs:annotation>
                        <xs:documentation>Unidad organizativa dentro de la Entidad suscriptora a
la que pertenece el titular del certificado. Aparece en certificados de la administración
pública</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Cargo" type="CargoType">
                <xs:annotation>
                        <xs:documentation>Cargo que ostenta el títular del certificado dentro de
su unidad organizativa. Aparece en certificados de la administración pública</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="SituacionLaboral" type="SituacionLaboralType">
                <xs:annotation>
                        <xs:documentation>Situación laboral del titular del certificado. Aparece
en certificados de la administración pública de tipo APE</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="Valido" type="ValidoType">
                <xs:annotation>
                        <xs:documentation>Código que indica la validez del
certificado</xs:documentation>
                        <xs:documentation>Los valores posibles son:
                                0 No válido
                                1 Válido
                           -1 Desconocido (No se ha podido determinar)
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="ResultadoValidacion" type="ResultadoValidacionType">
                <xs:annotation>
                        <xs:documentation>Descripción del resultado de la
validación</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="ValidacionSimple" type="ValidacionSimpleType">
                <xs:annotation>
                        <xs:documentation>Datos de la validación simple del
certificado</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="CodigoValidacionSimple" type="CodigoValidacionSimpleType">
                <xs:annotation>
                        <xs:documentation>Código de resultado de la validación
simple</xs:documentation>
                        <xs:documentation>Los valores posibles son:
                                0 Validación satisfactoria
                                1 El certificado está caducado
                                2 El certificado aún no válido
                                3 La firma del certificado no es válida
                                4 El emisor del certificado no es de confianza o no se encuentra
registrado
                                5 El certificado posee extensiones que no son válidas según su
política de certificación
                           -1 Desconocido (No se ha podido determinar)
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="DescripcionValidacionSimple" type="DescripcionValidacionSimpleType">
                <xs:annotation>
                        <xs:documentation>Detalle del resultado de la validación
simple</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="OrigenError" type="OrigenErrorType">
```

```
                <xs:annotation>
                        <xs:documentation>Código indicativo del origen del error
producido</xs:documentation>
                        <xs:documentation>Los valores posibles son:
                                1 Obtener información del certificado
                                2 Validar el certificado
                                3 Obtener información y validar el certificado
                           -1 Desconocido (No se ha podido determinar)
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="CodigoError" type="CodigoErrorType">
                <xs:annotation>
                        <xs:documentation>Código general del error producido</xs:documentation>
                        <xs:documentation>Los valores posibles son:
                                1 Formato de entrada no válido
                                2 Certificado no soportado por el sistema
                                3 Error al obtener los datos parseados del certificado
                                4 Error al validar el certificado
                                5 Error al obtener los datos parseados del certificados y también
en la validación del mismo
                                6 Error al conectar con la base de datos
                                7 Error desconocido
                           -1 No se ha producido ningún error
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="DescripcionError" type="DescripcionErrorType">
                <xs:annotation>
                        <xs:documentation>Descripción general del error
producido</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="DetalleError" type="DetalleErrorType">
                <xs:annotation>
                        <xs:documentation>Descripción detallada del error producido. Aparece solo
si se ha solicitado ver errores detallados en el archivo de configuración</xs:documentation>
                </xs:annotation>

        </xs:element>
```

Ilustración 59: Esquema XSD Elementos

A continuación se definen los tipos complejos:

```
        <!-- Tipos complejos definidos -->
        <xs:complexType name="RespuestaType">
                <xs:sequence>
                        <xs:element ref="VersionRespuesta"/>
                        <xs:element ref="HoraConsulta"/>
                        <xs:element ref="DatosCertificado" minOccurs="0"/>
                        <xs:element ref="ValidacionCertificado" minOccurs="0"/>
                        <xs:element ref="Error" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="DatosCertificadoType">
                <xs:sequence>
                        <xs:element ref="NumeroSerie"/>
                        <xs:element ref="Emisor"/>
                        <xs:element ref="Asunto"/>
                        <xs:element ref="ValidoDesde"/>
                        <xs:element ref="ValidoHasta"/>
                        <xs:element ref="EntidadCertificadora"/>
                        <xs:element ref="UnidadEntidadCertificadora"/>
                        <xs:element ref="DatosEntidadCertificadora" minOccurs="0"/>
                        <xs:element ref="TipoEntidadCertificadora"/>
                        <xs:element ref="TipoPersona"/>
                        <xs:element ref="DatosEntidadJuridica" minOccurs="0"/>
                        <xs:element ref="Proposito" minOccurs="0"/>
                        <xs:element ref="NombreCompleto"/>
                        <xs:element ref="PrimerApellido"/>
                        <xs:element ref="SegundoApellido"/>
```

```
                        <xs:element ref="Nombre"/>
                        <xs:element ref="Nif"/>
                        <xs:element ref="Pais"/>
                        <xs:element ref="Email"/>
                        <xs:element ref="EntidadSuscriptora" minOccurs="0"/>
                        <xs:element ref="NifEntidadSuscriptora" minOccurs="0"/>
                        <xs:element ref="NumIdentificacionPersonal" minOccurs="0"/>
                        <xs:element ref="UnidadOrganizativa" minOccurs="0"/>
                        <xs:element ref="Cargo" minOccurs="0"/>
                        <xs:element ref="SituacionLaboral" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ValidacionCertificadoType">
                <xs:sequence>
                        <xs:element ref="Valido"/>
                        <xs:element ref="ResultadoValidacion"/>
                        <xs:element ref="ValidacionSimple"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ErrorType">
                <xs:sequence>
                        <xs:element ref="OrigenError"/>
                        <xs:element ref="CodigoError"/>
                        <xs:element ref="DescripcionError"/>
                        <xs:element ref="DetalleError" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="DatosEntidadJuridicaType">
                <xs:sequence>
                        <xs:element ref="EntidadJuridica"/>
                        <xs:element ref="Cif"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ValidacionSimpleType">
                <xs:sequence>
                        <xs:element ref="CodigoValidacionSimple"/>
                        <xs:element ref="DescripcionValidacionSimple"/>
                </xs:sequence>

        </xs:complexType>
```

Ilustración 60: Esquema XSD Tipo complejos

Finalmente se definen los tipos simples:

```
<!-- Tipos simples definidos -->
        <xs:simpleType name="VersionRespuestaType">
                <xs:restriction base="xs:decimal"/>
        </xs:simpleType>
        <xs:simpleType name="HoraConsultaType">
                <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
        <xs:simpleType name="NumeroSerieType">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="EmisorType">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="AsuntoType">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="ValidoDesdeType">
                <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
        <xs:simpleType name="ValidoHastaType">
                <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
        <xs:simpleType name="EntidadCertificadoraType">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="UnidadEntidadCertificadoraType">
                <xs:restriction base="xs:string"/>
```

```
	</xs:simpleType>
	<xs:simpleType name="DatosEntidadCertificadoraType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="TipoEntidadCertificadoraType">
		<xs:restriction base="xs:byte">
			<xs:enumeration value="0"/>
			<xs:enumeration value="1"/>
			<xs:enumeration value="2"/>
			<xs:enumeration value="3"/>
			<xs:enumeration value="4"/>
			<xs:enumeration value="-1"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="TipoPersonaType">
		<xs:restriction base="xs:byte">
			<xs:enumeration value="0"/>
			<xs:enumeration value="1"/>
			<xs:enumeration value="-1"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="EntidadJuridicaType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="CifType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="PropositoType">
		<xs:restriction base="xs:byte">
			<xs:enumeration value="0"/>
			<xs:enumeration value="1"/>
			<xs:enumeration value="-1"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="NombreCompletoType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="PrimerApellidoType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="SegundoApellidoType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="NombreType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="NifType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="PaisType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="EmailType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="EntidadSuscriptoraType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="NifEntidadSuscriptoraType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="NumIdentificacionPersonalType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="UnidadOrganizativaType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="CargoType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="SituacionLaboralType">
		<xs:restriction base="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="ValidoType">
		<xs:restriction base="xs:byte">
			<xs:enumeration value="0"/>
			<xs:enumeration value="1"/>
```

```
                        <xs:enumeration value="-1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ResultadoValidacionType">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="CodigoValidacionSimpleType">
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="0"/>
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                        <xs:enumeration value="3"/>
                        <xs:enumeration value="4"/>
                        <xs:enumeration value="5"/>
                        <xs:enumeration value="-1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DescripcionValidacionSimpleType">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="OrigenErrorType">
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                        <xs:enumeration value="3"/>
                        <xs:enumeration value="-1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="CodigoErrorType">
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                        <xs:enumeration value="3"/>
                        <xs:enumeration value="4"/>
                        <xs:enumeration value="5"/>
                        <xs:enumeration value="6"/>
                        <xs:enumeration value="7"/>
                        <xs:enumeration value="-1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DescripcionErrorType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Formato de entrada no válido"/>
                        <xs:enumeration value="Certificado no soportado por el sistema"/>
                        <xs:enumeration value="Error al obtener los datos parseados del
certificado"/>
                        <xs:enumeration value="Error al validar el certificado"/>
                        <xs:enumeration value="Error al obtener los datos parseados del
certificado y también en la validación del mismo"/>
                        <xs:enumeration value="Error al conectar con la base de datos"/>
                        <xs:enumeration value="Error desconocido"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DetalleErrorType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="MPCG01: Parámetro de entrada no válido. El formato
del certificado no es correcto"/>
                        <xs:enumeration value="MPCG02: Parámetro de entrada nulo. El certificado
no puede ser nulo"/>
                        <xs:enumeration value="MPCG03: Parámetro de entrada no válido. El
identificador de la aplicación no es correcto, debe ser un número mayor de 0"/>
                        <xs:enumeration value="MPCG04: Parámetro de entrada no válido. El modo de
validación debe ser uno de los valores {0,1,2,-1}"/>
                        <xs:enumeration value="MPCG05: Error al intentar convertir el certificado
a un formato válido"/>
                        <xs:enumeration value="MPCG06: Error al leer el fichero de
configuración"/>
                        <xs:enumeration value="MPCG07: No se ha podido identificar el prestador al
que pertenece el certificado"/>
                        <xs:enumeration value="MPCG08: Certificado no soportado por el sistema"/>
                        <xs:enumeration value="MPCG09: Error al comprobar la política del
certificado"/>
                        <xs:enumeration value="MPCG10: Error al parsear los datos del emisor del
certificado"/>
                        <xs:enumeration value="MPCG11: Error al parsear los datos del asunto del
certificado"/>
                        <xs:enumeration value="MPCG12: Error al parsear los nombres alternativos
```

```
del certificado"/>
                        <xs:enumeration value="MPCG13: Error al validar el estado de caducidad del
certificado"/>
                        <xs:enumeration value="MPCG14: Error al validar que la fecha actual no sea
anterior a la fecha de comienzo del periodo de validez del certificado"/>
                        <xs:enumeration value="MPCG15: Error al validar la confianza del emisor
del certificado"/>
                        <xs:enumeration value="MPCG16: Error al validar la firma del
certificado"/>
                        <xs:enumeration value="MPCG17: Error al validar las extensiones del
certificado"/>
                        <xs:enumeration value="MPCG18: Error al insertar estadísticcas en la base
de datos"/>
                        <xs:enumeration value="MPCG19: Error al buscar el identificador de la
aplicación en la base de datos"/>
                        <xs:enumeration value="MPCG20: Error desconocido del MPC"/>
                </xs:restriction>
        </xs:simpleType>

</xs:schema>
```

Ilustración 61: Esquema XSD Tipos simples

Datos de salida del resto de métodos del sistema MPC

El método EsValido() no devuelve un xml, sino un valor entero, como se ha explicado anteriormente.

El método ObtenerInformacionGrupoCertificados() devuelve tantos nodos “Respuesta” como certificados se le pasen agrupados dentro del nodo raíz “Certificados”, con la particularidad de que no existe validación, y por tanto nunca aparecerá el nodo ValidacionCertificado. Además y como ayuda para el parseo se añade el atributo “Total” al nodo “Certificados” el cual indica el número de respuestas (certificados parseados) incluidos en la respuesta:

```
<xs:element name="Certificados">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="Respuesta" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="total" use="required">
                        <xs:simpleType>
                                <xs:restriction base="xs:byte">
                                        <xs:enumeration value="2"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
        </xs:complexType>

</xs:element>
```

Ilustración 62: Esquema XSD ObtenerInformacionGrupoCertificados()

5.2 Modelo vista controlador. (Arquitectura distribuida)

El Modelo vista controlador es un patrón de diseño utilizado en arquitectura de software que separa la parte gráfica de entrada y salida de datos de la aplicación (interfaz de usuario) de las operaciones que manipulan dichos datos (capa de negocio). Según la teoría, en el patrón MVC se tiene:

- **Vista**: Está formada por los propios controles gráficos y por la lógica necesaria para mostrar y obtener datos de ella.
- **Modelo:** Es el conjunto de clases y métodos para manejar la capa de acceso a datos de la aplicación.
- **Controlador:** Actúa de puente entre la vista y el Modelo. El Controlador actualiza la vista utilizando el modelo y viceversa: accede a los datos de la vista para actualizar con ellos el modelo. En el controlador es dónde se deben implementar todos los métodos y operaciones de manipulación para dejar la vista y el modelo únicamente para los propósitos anteriormente mencionados.

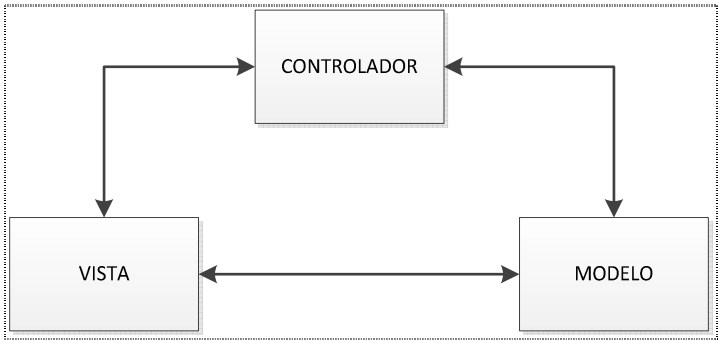


Ilustración 63: Modelo Vista Controlador

Aplicación del patrón Modelo Vista Controlador al sistema MPC

Como se puede intuir al ser nuestro sistema MPC un servicio Web, en él no existe técnicamente una interfaz de usuario como tal, por lo que podría parecer ineficiente el uso del patrón Modelo Vista Controlador. Sin embargo, el patrón sigue siendo útil, ya que organiza el código y facilita la reutilización, escalabilidad y flexibilidad del mismo. Además, podemos considerar los Métodos Web del sistema como la interfaz (es lo que ven las aplicaciones externas que usan el servicio) y los parámetros de entrada como los controles de usuario.

De esta forma, se tiene una vista por cada servicio web, la cual será de tipo “Interface”, con propiedades para cada uno de los parámetros recibidos por los métodos web. Estas propiedades deberán ser implementadas en la propia clase principal del servicio web. Luego tendremos un controlador, el cual será llamado desde la clase principal y contendrá los métodos que llevarán a cabo las acciones solicitadas. Aquí, en este nivel es dónde se encontrarán todos los métodos importantes y dónde se realizarán todas las comprobaciones y validaciones. El controlador accederá a los parámetros recibidos por los métodos del servicio web mediante la vista. Finalmente se tiene el modelo, el cual será utilizado por el controlador para insertar las estadísticas en la base de datos del sistema.

Finalmente comentar, que mediante la aplicación del patrón MVC, se podrían tener varios Servicios Web con varios métodos cada uno, cada uno de ellos tendría su propia vista y su propio controlador pero ambos podrían compartir métodos mediante un controlador base y una vista base comunes.

5.3 Diagrama de componentes del sistema

A continuación se van a explicar los distintos componentes que forman el sistema MPC. La aplicación MPC es un servicio web pero se ha querido abstraer toda la funcionalidad posible en componentes de cara a mejorar la reutilización, escalabilidad y mantenimiento del código. Al encontrarnos en un entorno .NET se ha optado por separar en librerías de enlace dinámico (DLL) los distintos módulos que utiliza la aplicación. Primeramente se va a mostrar un diagrama de dependencias entre las distintas librerías para mostrar una visión general del sistema:

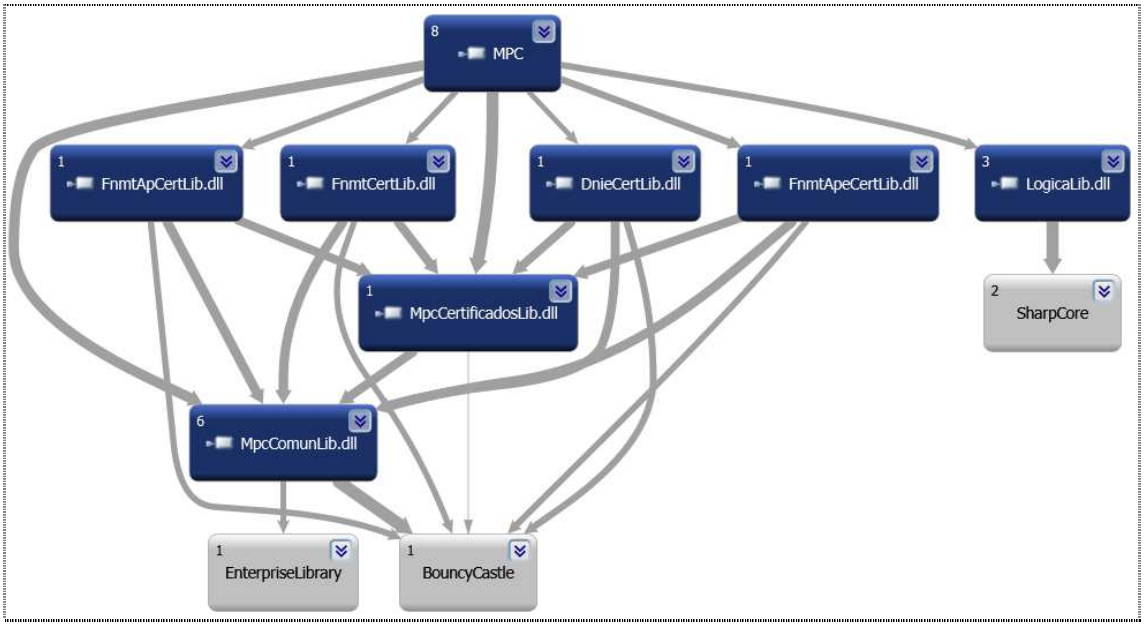


Ilustración 64: Diagrama de componentes del sistema

El sistema MPC utiliza 7 componentes propios y 3 componentes externos principales (en realidad se utilizan muchas librerías del Framework pero se han eliminado del diagrama ya que se consideran parte del propio entorno .NET). A continuación se describen brevemente cada una de las librerías:

- **LogicaLib.dll:** Es una librería que engloba toda la capa de acceso a datos de la aplicación. Tiene las clases y métodos para poder leer y escribir a la base de datos de la aplicación. Utiliza el componente SharpCore que proporciona la herramienta Data Tier Generator.

- **MpcComunLib.dll**: En esta librería se han intentado aislar funcionalidades y conceptos que pueden ser reutilizables en otros proyectos. Las funcionalidades más importantes que tiene esta DLL son las siguientes:

  - *Certificados.* Clases y métodos para el manejo básico de certificados X509 utilizando BouncyCastle.

- *Configuración.* Clases y métodos para leer e interpretar los archivos de configuración de la aplicación.

- *Log.* Clases y métodos para escribir el log de la aplicación utilizando Microsoft Enterprise Library.

- *Útiles.* Clases y métodos heterogéneos que cubren diversas funcionalidades: conversiones de datos, validación de cadenas, manejo de enumeraciones etc.

- **MpcCertificados.dll**: Esta librería tiene la finalidad de aunar todos los conceptos comunes a la aplicación MPC, centrándose principalmente en el parseo de certificados. Dentro de esta librería y apoyándonos en las clases y métodos definidos en la librería **MpcComunLib.dll** se proporcionan los métodos y las clases para parsear y validar de forma genérica un certificado X509. Aquí están definidas también todas las excepciones personalizadas que puede lanzar la aplicación así como enumeraciones e interfaces.

Para que la aplicación sea lo más modular posible y de cara a la posible adición futura de nuevos tipos de certificados se decidió crear una librería para cada tipo específico de certificado que tendría que ser procesado por la aplicación. Cada librería extendería y concretaría los métodos y clases definidos en las librerías **MpcCertificados.dll** y **MpcComunLib.dll** y supondrían el nivel más detallado de abstracción. Estas 4 librerías son las siguientes:

- **FnmtCertLib.dll**: Conjunto de métodos y clases para realizar el parseo y la validación de los distintos certificados de tipo Fnmt Clase 2 CA.

- **FnmtApCertLib.dll**: Conjunto de métodos y clases para realizar el parseo y la validación de los distintos certificados de tipo Fnmt AP.

- **FnmtApeCertLib.dll**: Conjunto de métodos y clases para realizar el parseo y la validación de los distintos certificados de tipo Fnmt APE.

- **DniCertLib.dll**: Conjunto de métodos y clases para realizar el parseo y la validación de los distintos certificados de tipo DNIe.

Finalmente se tienen los 3 componentes externos principales que se utilizan en la aplicación:

- **BouncyCastle**: Librería de código abierto que proporciona clases y métodos para el manejo de certificados X509. Es usada por la mayoría de componentes de la aplicación.

- **EnterpriseLibrary**: Librería de Microsoft que proporciona soluciones a funcionalidades típicas en el software empresarial. En concreto se utiliza la funcionalidad para gestión de Logs que se encuentra implementada en la librería **MpcComunLib.dll.**

- **SharpCore**: Librería utilizada por el software Data Tier Generator que sirve de base para la creación de la capa de acceso a datos de la aplicación, implementada en la librería **LogicaLib.dll.**

5.4 Espacios de nombres del sistema

Mediante los espacios de nombre de una aplicación se pueden organizar de forma lógica los identificadores de métodos y clases. A continuación se muestra un diagrama con los espacios de nombres de los componentes del sistema MPC:

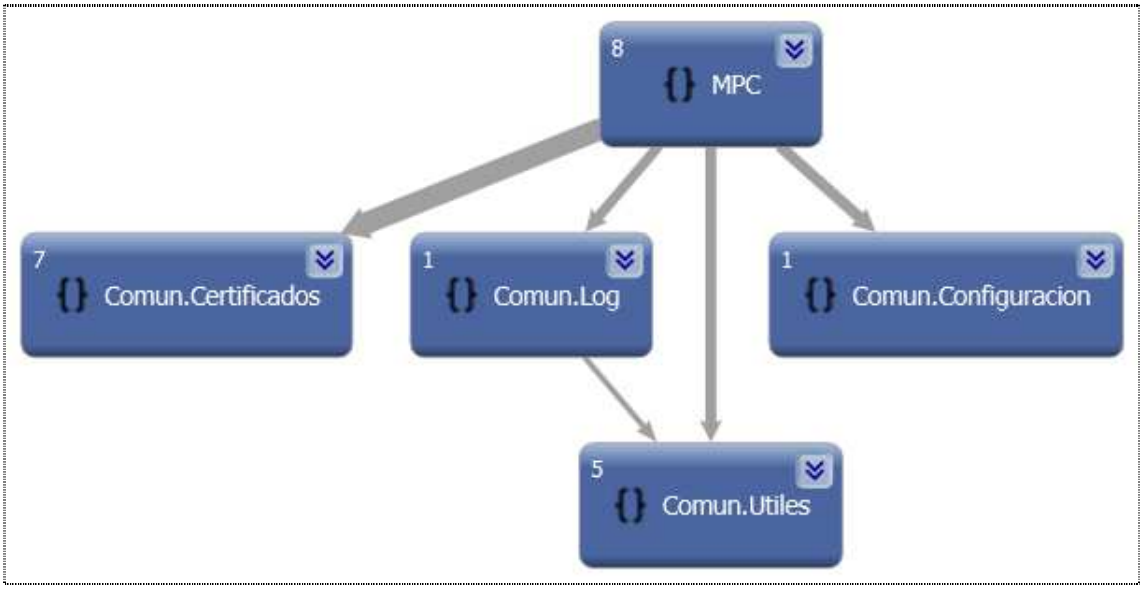


Ilustración 65: Espacio de nombres del sistema

Dónde si expandimos el espacio de nombres MPC tenemos lo siguiente:

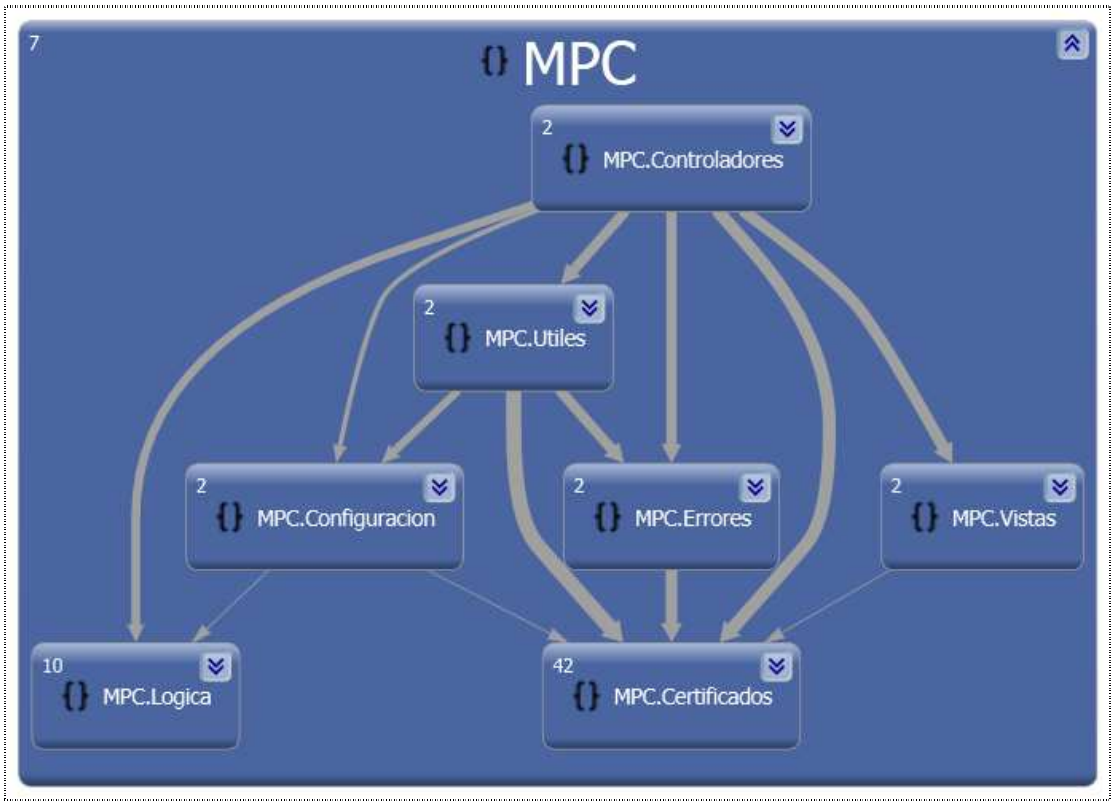


Ilustración 66: Detalle espacio de nombre MPC

5.5 Diagrama de clases del sistema

A continuación se van a mostrar los distintos diagramas de clases que forman la aplicación. Se mostrarán por partes para una mejor comprensión, ya que un solo diagrama sería demasiado extenso y confuso.

Empezamos por el diagrama de clases de la librería de **LogicaLib.dll**, dónde se encuentran los métodos y las clases necesarias para acceder a la base de datos de la aplicación. Primeramente dentro del espacio de nombres Lógica tenemos:

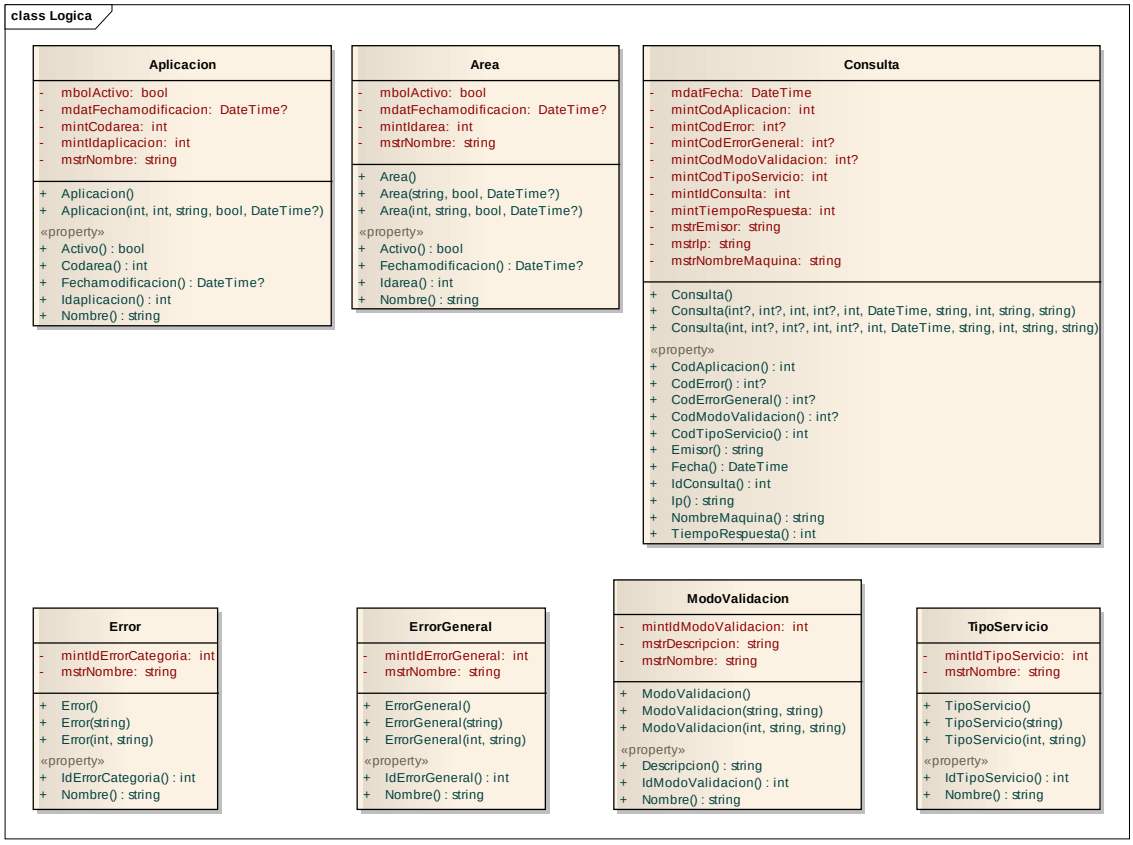


Ilustración 67: Diagrama de clases de la librería LogicaLib

Existe una clase por cada tabla de la base de datos, lo que se considera una entidad, de tal forma que se puedan crear objetos con los campos necesarios para insertar un registro en una tabla de la base de datos. La parte de lógica se completa con las siguientes clases dentro del espacio de nombres Data:

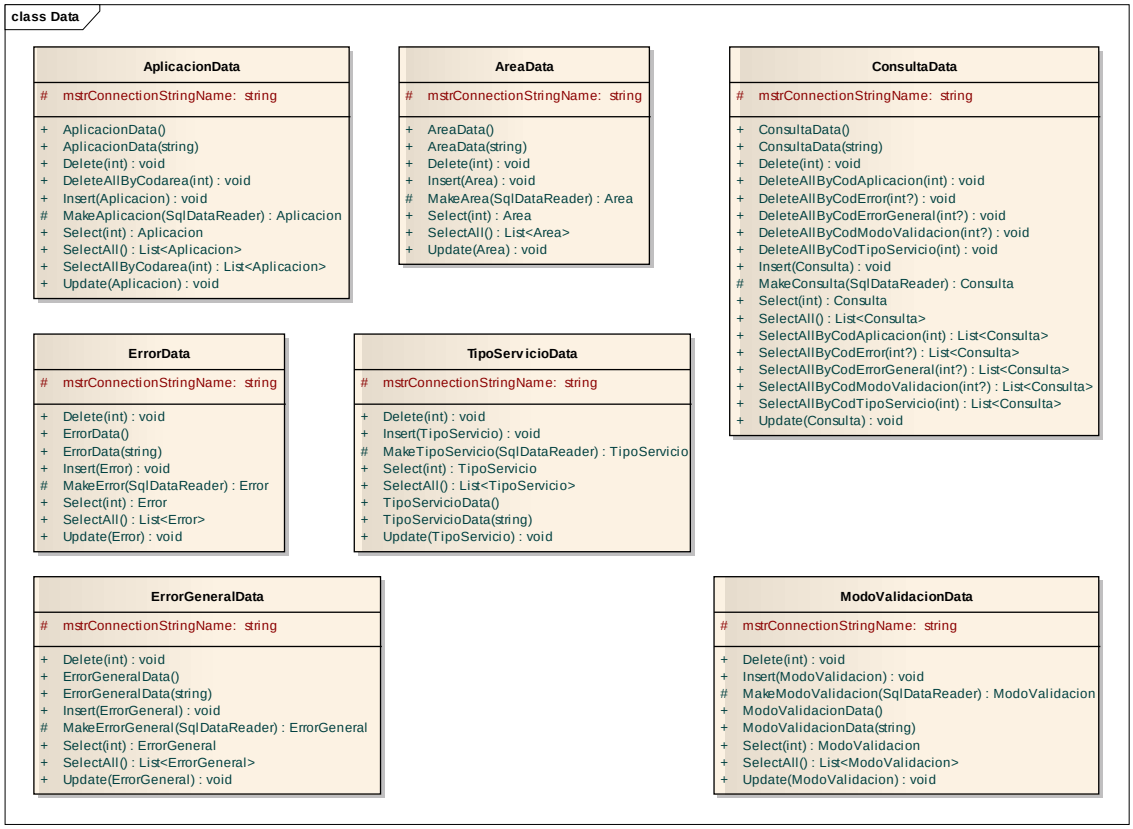


Ilustración 68: Clases del espacio de nombres Data

Existe una clase de negocio con el sufijo “Data” por cada entidad de la base de datos. Estas clases son las que tienen los métodos para insertar, acceder o eliminar registros de la base de datos de la aplicación. Por ejemplo mediante la clase ConsultaData se puede insertar un registro de acceso en base datos mediante el método Insert(Consulta) que recibe como parámetro un objeto de tipo Consulta cuya estructura coincide con la de la tabla Consultas de la base de datos.

A continuación se muestran los diagramas de clase que componen la librería **MpcComunLib.dll**. Las clases más importantes de estas librerías y las que sirven de base para el funcionamiento de la aplicación son las que se agrupan bajo el espacio de nombres certificados:

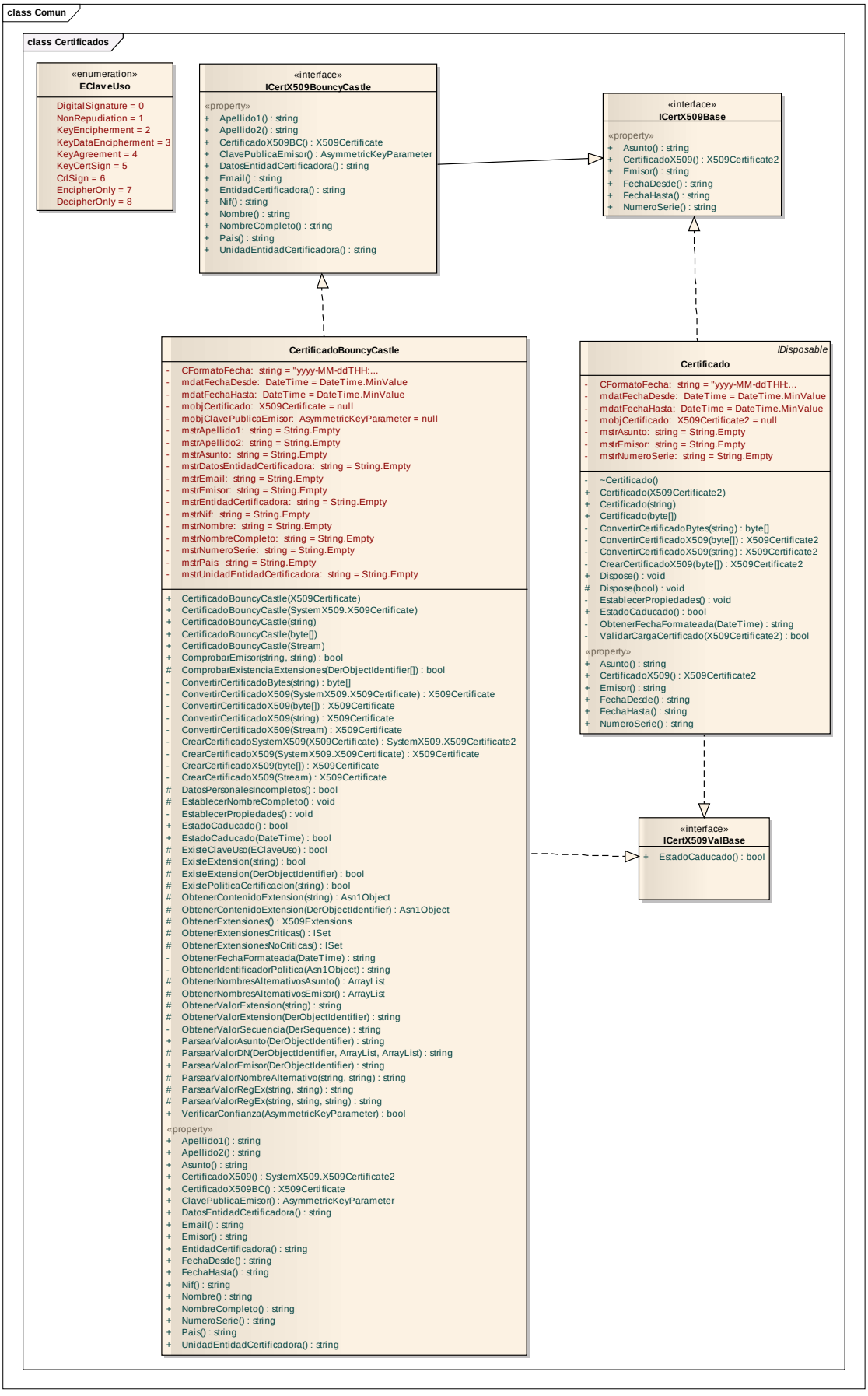


Ilustración 69: Clases de la librería MpcComunLib. Espacio de nombres Certificados

Como se comentó en la descripción de la librerías, la finalidad de la librería **MpcComunLib** es la de recoger toda la funcionalidad que aun siendo necesaria en la aplicación puede ser reutilizable 100% por otras aplicaciones. Por tanto en lo referente a los certificados esto quiere decir que aquí se tienen clases con métodos y propiedades comunes a todos los certificados digitales de tipo X509. La clase *Certificado* implementa la interfaz *ICertX509base* que tiene lo básico: asunto, emisor, número de serie, fechas del periodo de validez y el propio certificado en el formato x509 de .NET. La clase *Certificado* también implemente la interfaz *ICertX509ValBase* que tiene la validación más básica y la cual con las propiedades que tiene la clase *Certificado* ya puede implementarse: la validación de la caducidad del certificado. Esta clase en realidad no es utilizada directamente por la aplicación por ser demasiado sencilla para nuestro propósito de parseo y validación. Para ampliar la funcionalidad se ha creado la clase *CertificadoBouncyCastle*, que aprovecha toda la potencia de la librería BouncyCastle para el manejo de certificados digitales de tipo X509. Esta clase implementa la interfaz *ICertX509BouncyCastle,* la cual amplía las propiedades de *ICertX509base.* De esta forma se tienen todos los campos comunes (Nif, email, clave pública, etc) a todos los certificados que pueden ser obtenidos mediante BouncyCastle. La clase *CertificadoBouncyCastle* posee multitud de constructores para poder crear una instancia a partir de un certificado x509 en distintos formatos (array de bytes, x509 en formato .NET, string en base64 etc.) También se tienen métodos que apoyados en BouncyCastle proporcionarán información útil de cara al parseo y validación del certificado, muchos de estos métodos son protegidos para poder usarse en las clases específicas de cada aplicación, ya que por sí mismos proporcionan información demasiado abstracta.

Siguiendo con la librería **MpcComunLib** pero dentro de espacios de nombre distintos tenemos clases con funcionalidades referentes al log, la configuración o usos varios (útiles):

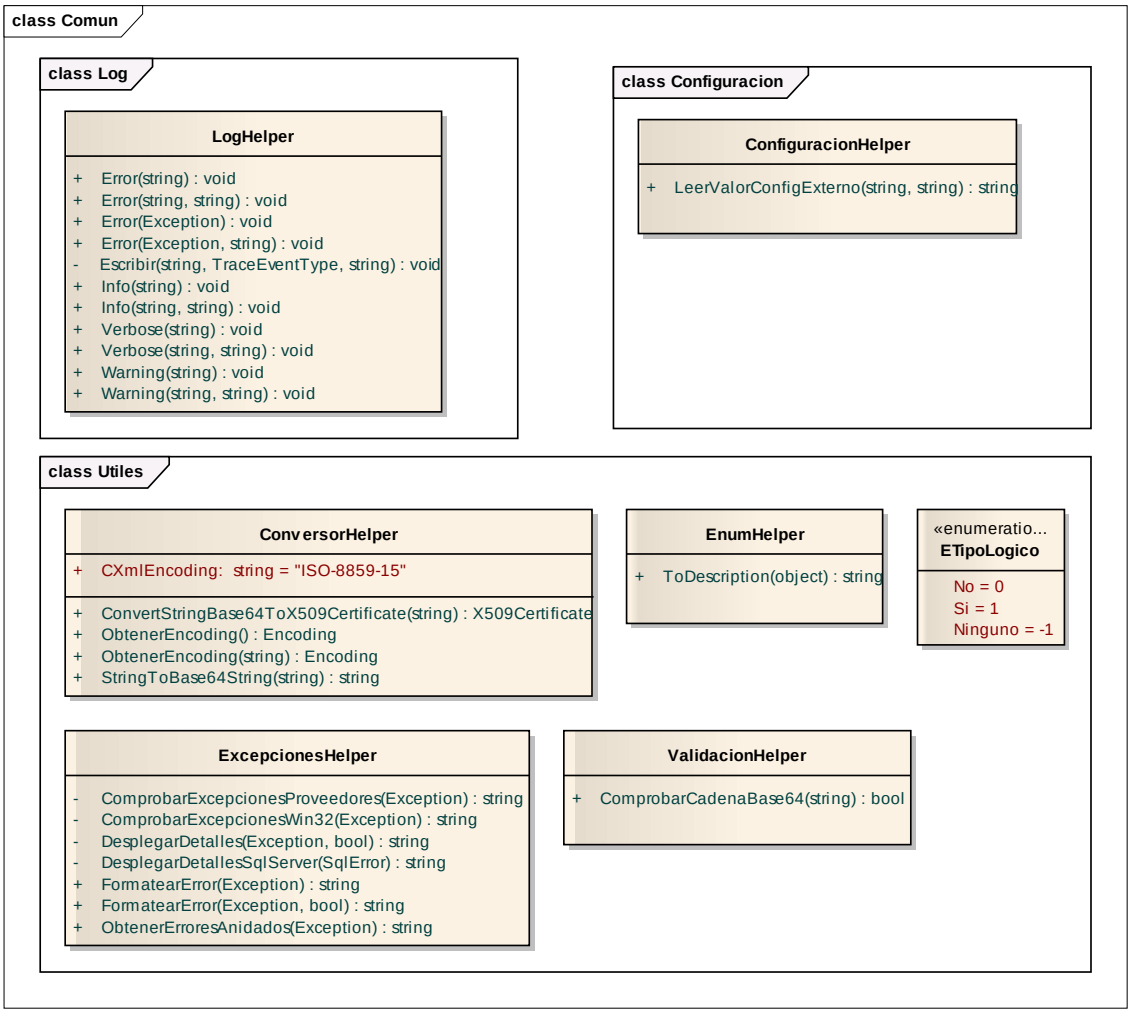


Ilustración 70: Resto de clases de la librería MpcComunLib

A continuación pasamos a descomponer la librería **MpcCertificadosLib** que como se ha comentado anteriormente aglutina la funcionalidad común al parseo y validación de certificados de la aplicación MPC, siendo ya algo específico de ésta aplicación y no reutilizable por otras pero cumpliendo una importante función de cara al mantenimiento y evolución del sistema MPC. Dentro del espacio de nombres Certificados tenemos:

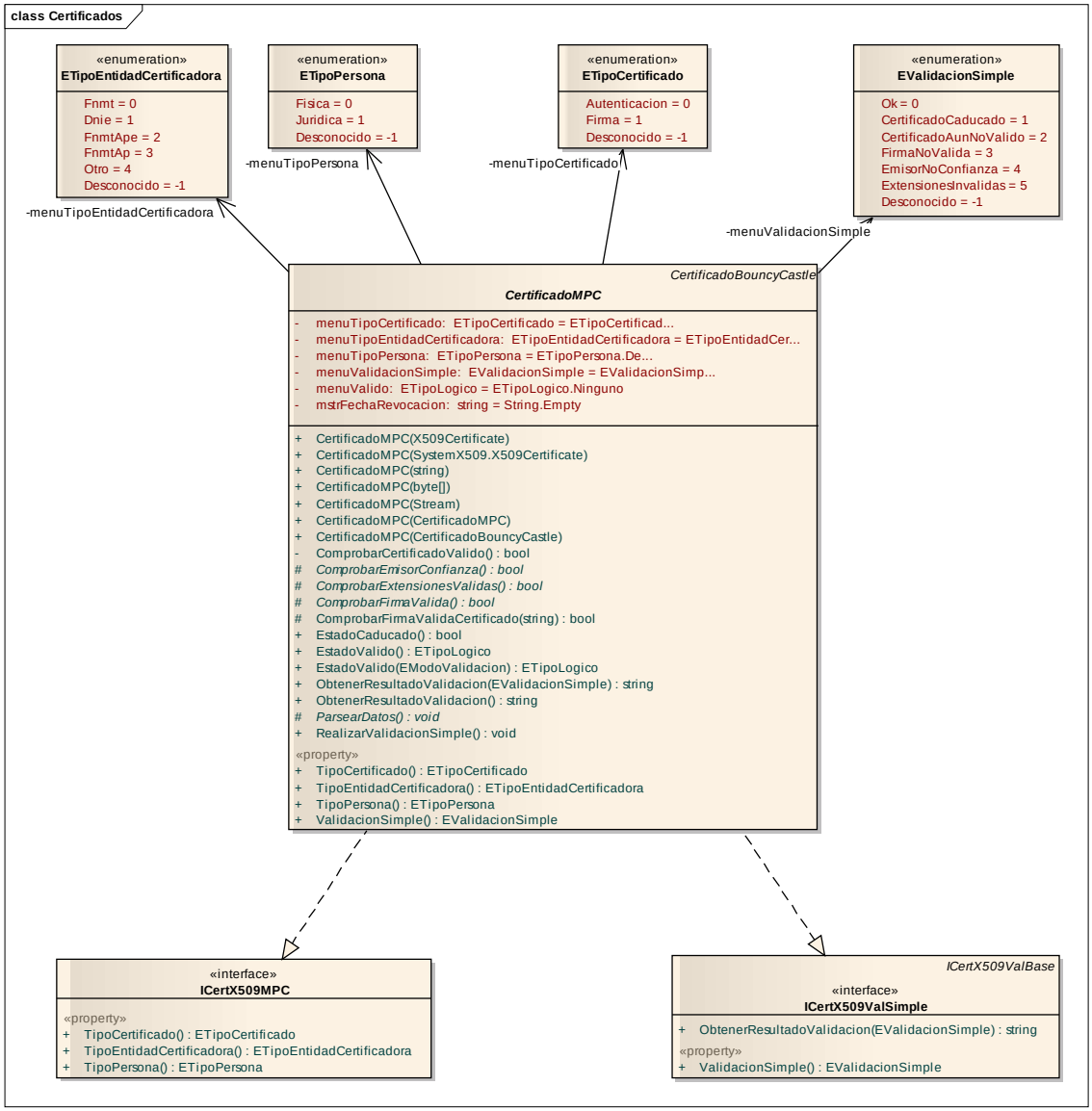


Ilustración 71: Diagrama de clases de la librería MpcCertificadosLib.dll. Espacio de nombres Certificados

La clase *CertificadoMPC* es una de las más importantes de la aplicación. Esta clase expande la clase *CertificadoBouncyCastle* de la librería **MpcComunLib,** implementa la interface *ICertX509MPC* que tiene 3 propiedades comunes a todos los tipos de certificados que soporta la aplicación MPC (TipoCertificado, TipoEntidadCertificadora y TipoPersona) así como la interface *ICertX590ValSimple* para el método que obtiene la validación simple del certificado. Esta validación se lleva a cabo en el método RealizarValidacionSimple():

```
/// <summary>
/// Realiza una validación simple del certificado (caducidad, integridad y confianza)
/// </summary>
public void RealizarValidacionSimple()
{
    EValidacionSimple enuValidacion = EValidacionSimple.Desconocido;
    try
    {
        // Comprueba que no esté caducado
        if (!EstadoCaducado())
        {
            // Comprobar si aún no es válido
            if (ComprobarCertificadoValido())
            {
                // Comprueba si el emisor es de confianza y soportado
                if (ComprobarEmisorConfianza())
                {
                    // Comprueba si la firma es válida (firmado con la clave pública del emisor)
                    if (ComprobarFirmaValida())
                    {
                        // Comprueba si el certificado posee extensiones que no son válidas
                        if (ComprobarExtensionesValidas())
                        {
                            enuValidacion = EValidacionSimple.Ok;
                        }
                        else
                        {
                            enuValidacion = EValidacionSimple.ExtensionesInvalidas;
                        }
                    }
                    else
                    {
                        enuValidacion = EValidacionSimple.FirmaNoValida;
                    }
                }
                else
                {
                    enuValidacion = EValidacionSimple.EmisorNoConfianza;
                }
            }
            else
            {
                enuValidacion = EValidacionSimple.CertificadoAunNoValido;
            }
        }
        else
        {
            enuValidacion = EValidacionSimple.CertificadoCaducado;
        }

        menuValidacionSimple = enuValidacion;
    }
    catch (CaducidadCertificadoException)
    {
        throw;
    }
    catch (ValidezCertificadoException)
    {
        throw;
    }
    catch (EmisorConfianzaCertificadoException)
    {
        throw;
    }
    catch (ValidezFirmaCertificadoException)
    {
        throw;
    }
    catch (ValidezExtensionesCertificadoException)
    {
        throw;
    }
    catch (Exception)
    {
        throw;
    }
}
```

Ilustración 72: Método RealizarValidacionSimple()

Este método como se puede observar utiliza a su vez cinco métodos que serán los encargados de comprobar diversas características del certificado:

- **EstadoCaducado().** Método que comprueba si el certificado está caducado en la fecha actual. Este método es público ya que la caducidad se comprueba de la misma forma en cualquier tipo de certificado.

- **ComprobarCertificadoValido().** Método que comprueba si el certificado ha entrado en el periodo de validez comparándolo con la fecha actual. Este método está implementado en esta clase ya que se puede comprobar si se ha entrado en el periodo de validez de la misma forma para cualquier tipo de certificado.

- **ComprobarEmisorConfianza().** Método que comprueba si la entidad certificadora que ha emitido el certificado es de confianza y por tanto la que el sistema espera que sea para ese certificado en concreto. Este método es abstracto, esto quiere decir que se implementa en clases heredadas de esta ya que para comprobar si el emisor es de confianza se necesita saber de qué tipo concreto es el certificado y por tanto este método será diferente para cada tipo de certificado.

- **ComprobarFirmaValida().** Método que verifica la firma del certificado. En este método se verifica la firma del certificado usando para ello la clave pública del certificado emisor. Este método por tanto también es abstracto, ya que se necesita saber qué tipo de certificado es para poder comprobar con la clave pública de su emisor la firma del mismo. La implementación de este método estará en clases heredadas y será distinta para cada tipo de certificado.

- **ComprobarExtensionesValidas()** Método que comprueba que el certificado contenga las extensiones críticas admitidas en su política de certificación. Este método también es abstracto y su implementación será diferente para cada tipo de certificado concreto ya que cada certificado tiene unas extensiones críticas diferentes y en algunos casos no posee ninguna.

En la clase *CertificadoMPC* también se encuentra el método abstracto ParsearDatos(). Este método será implementado por cada clase particular de cada tipo de certificado y será el encargado de extraer y parsear los datos del certificado que deben ser devueltos por la aplicación y que no han podido ser parseados en las clases bases o clases padres.

Siguiendo en el espacio de nombres Certificados dentro de la librería **MpcCertificadosLib** también están definidas las siguientes clases:

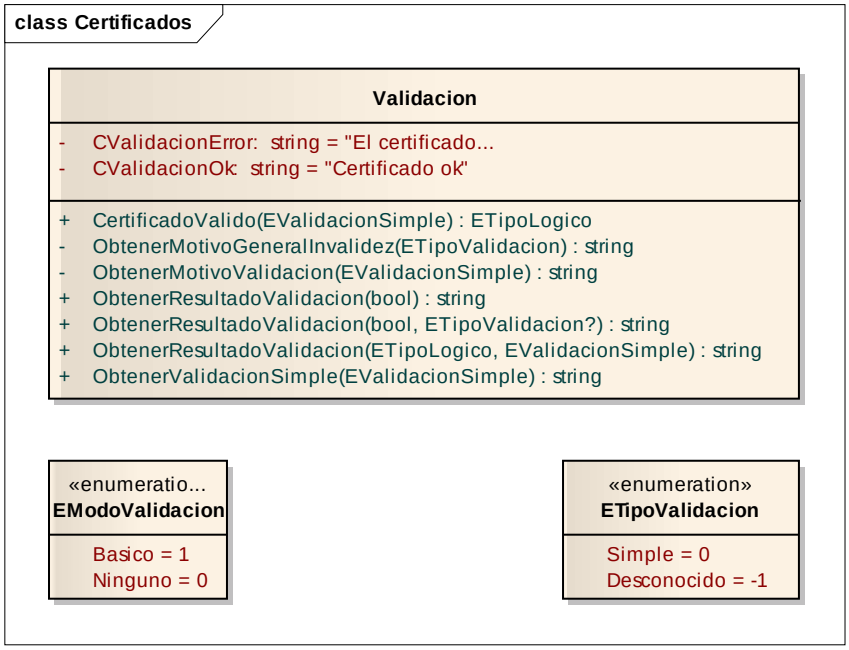


Ilustración 73: Clases de la librería MpcCertificadosLib. Espacio de nombres Certificados

La clase estática *Validacion* tiene métodos para obtener los textos de los posibles motivos de invalidez del certificado a partir de las enumeraciones que se utilizan en distintas partes del código. Se utiliza en la parte de construcción del xml de salida. También se encuentran definidas varias enumeraciones que se utilizan en la aplicación (Errores, TipoServicio, ModoValidacion, etc.)

Dentro del espacio de nombres *Certificados* y siguiendo dentro de la librería **MpcCertificadosLib** tenemos las excepciones personalizadas que podrá lanzar la aplicación:

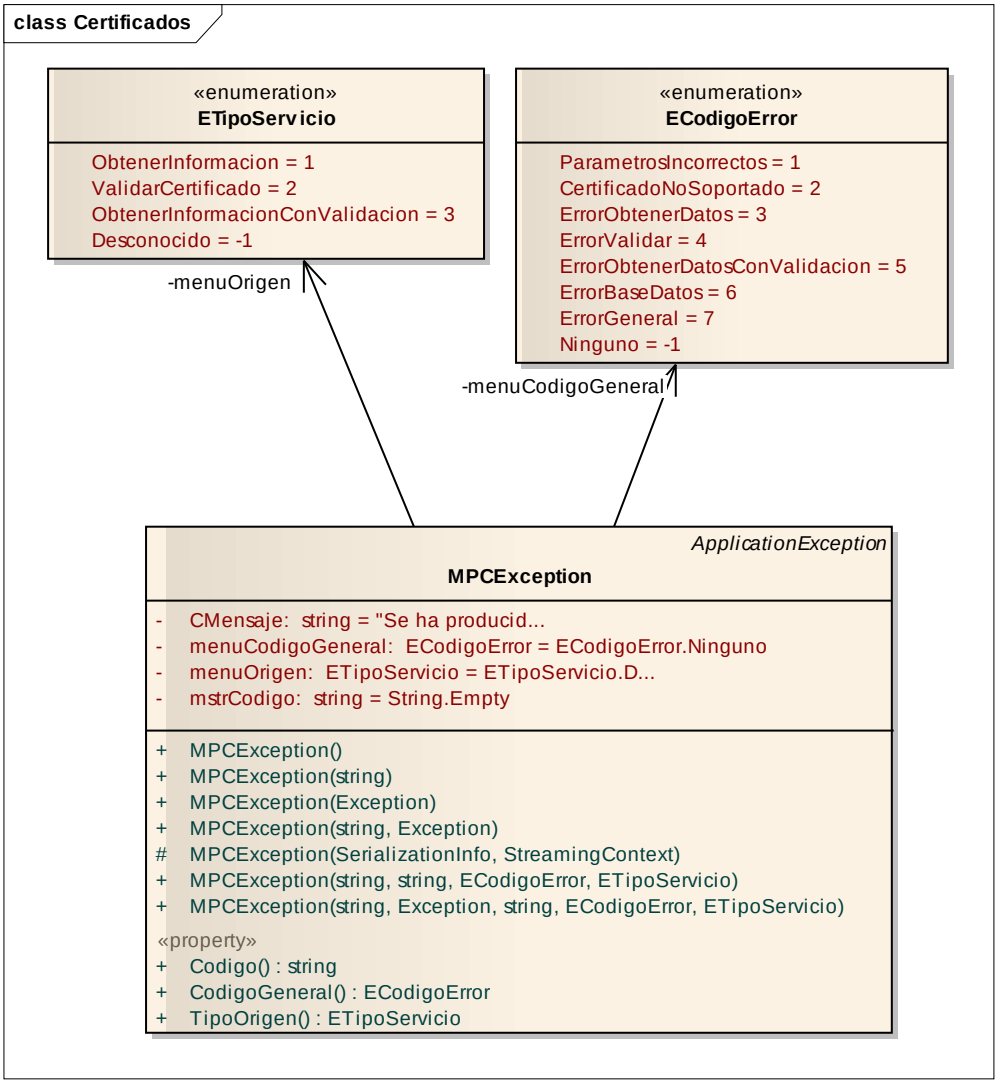


Ilustración 74: Clases de la librería MpcCertificadosLib. Espacio de nombres Certificados

La clase *MpcException* expande la clase de .NET *AplicationException* la cual representa una excepción de la aplicación. Se han respetado los constructores de tal forma que llamen a los constructores base y se ha creado uno propio que asigne las propiedades que utiliza el sistema MPC: el código de error general de la aplicación, el código de error específico y el origen del error. Tanto el origen como el código general son enumerados definidos en el espacio de nombres *Certificados* en cambio el código específico de error será un texto que en esta clase será genérico ("Se ha producido un error con el Servicio MPC") y que en cada excepción derivada de esta será más específico.

Excepciones de Formato de Entrada:

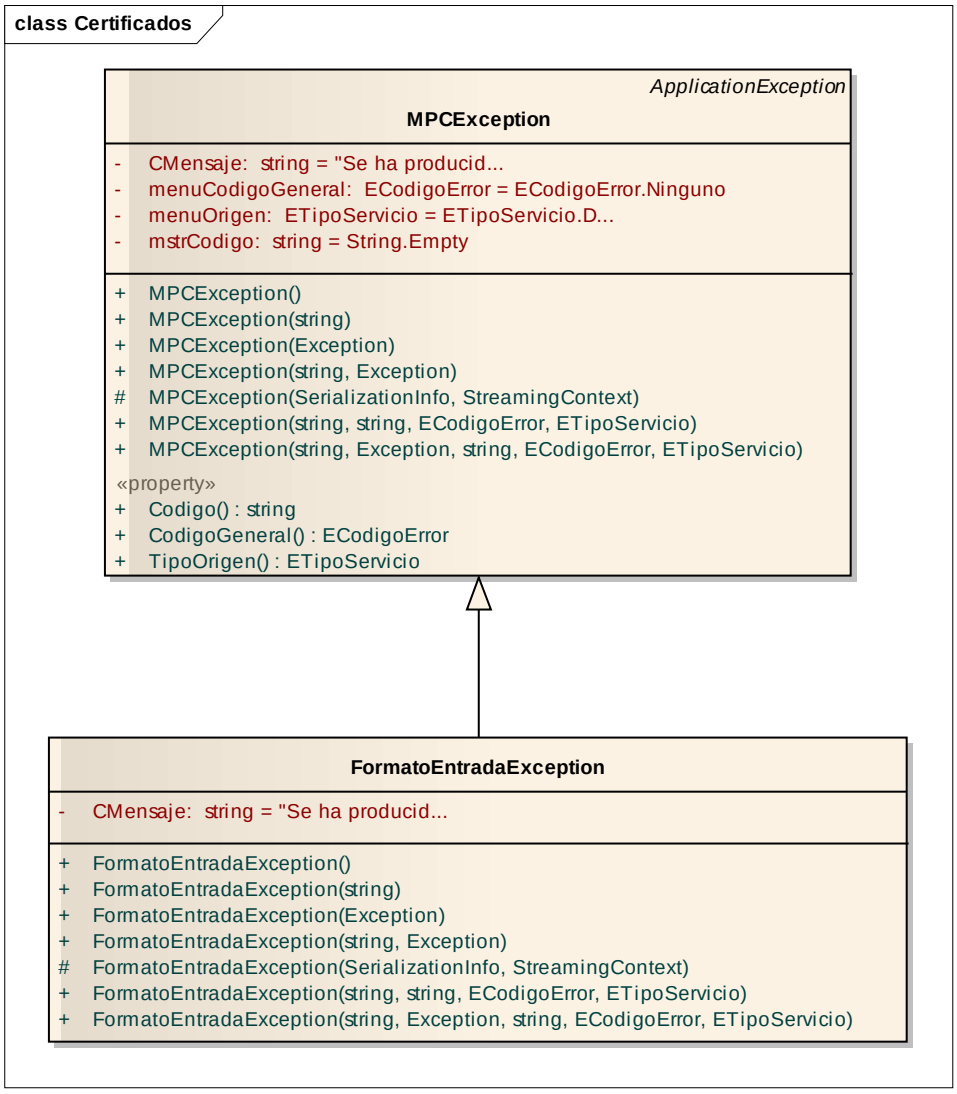


Ilustración 75: Clases para Excepciones de formato de entrada

La clase FormatoEntradaException hereda de la clase MpcException y establece el siguiente mensaje de error específico: "Se ha producido un error con el formato de entrada". Se lanza cuando hay algún problema con los parámetros de entrada de la aplicación.

Excepciones de Base de datos:

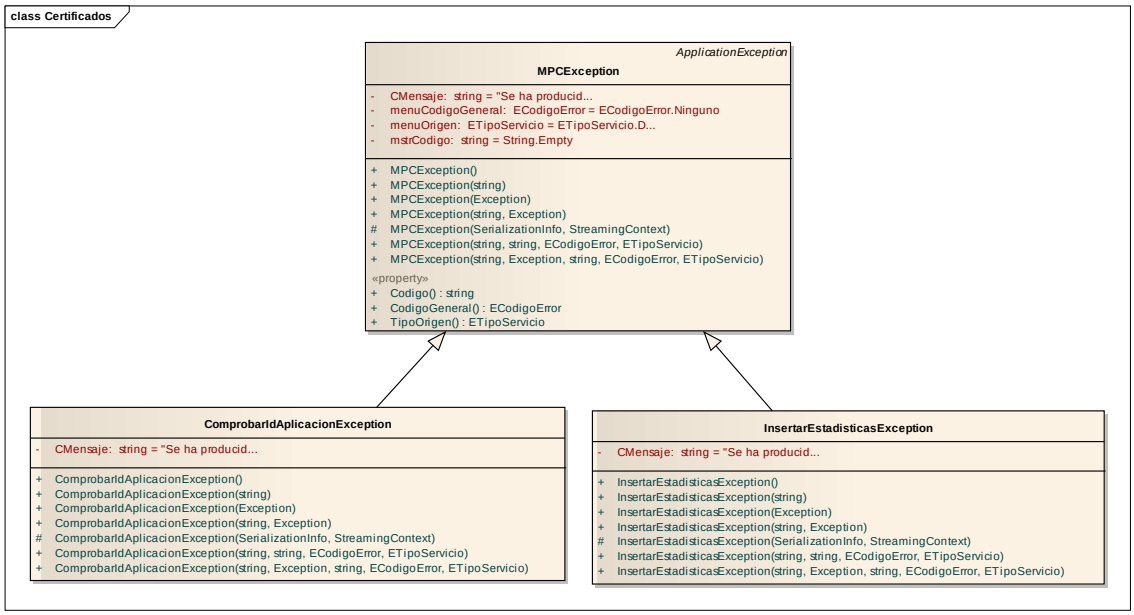


Ilustración 76: Clases para Excepciones de base de datos

Las clases *ComprobarIdAplicacionException* e *InsertarEstadisticasException* heredan también de *MpcException* y son excepciones que se lanzan cuando existe algún problema de conexión con la base de datos de la aplicación. Las clases establecen los siguientes mensajes de error:

- *ComprobarIdAplicacionException:* "Se ha producido un error al comprobar que el Identificador de Aplicación está en Base de datos."
- *InsertarEstadisticasException:* "Se ha producido un error al insertar estadísticas en Base de datos."

Excepciones de Obtención de obtención de datos

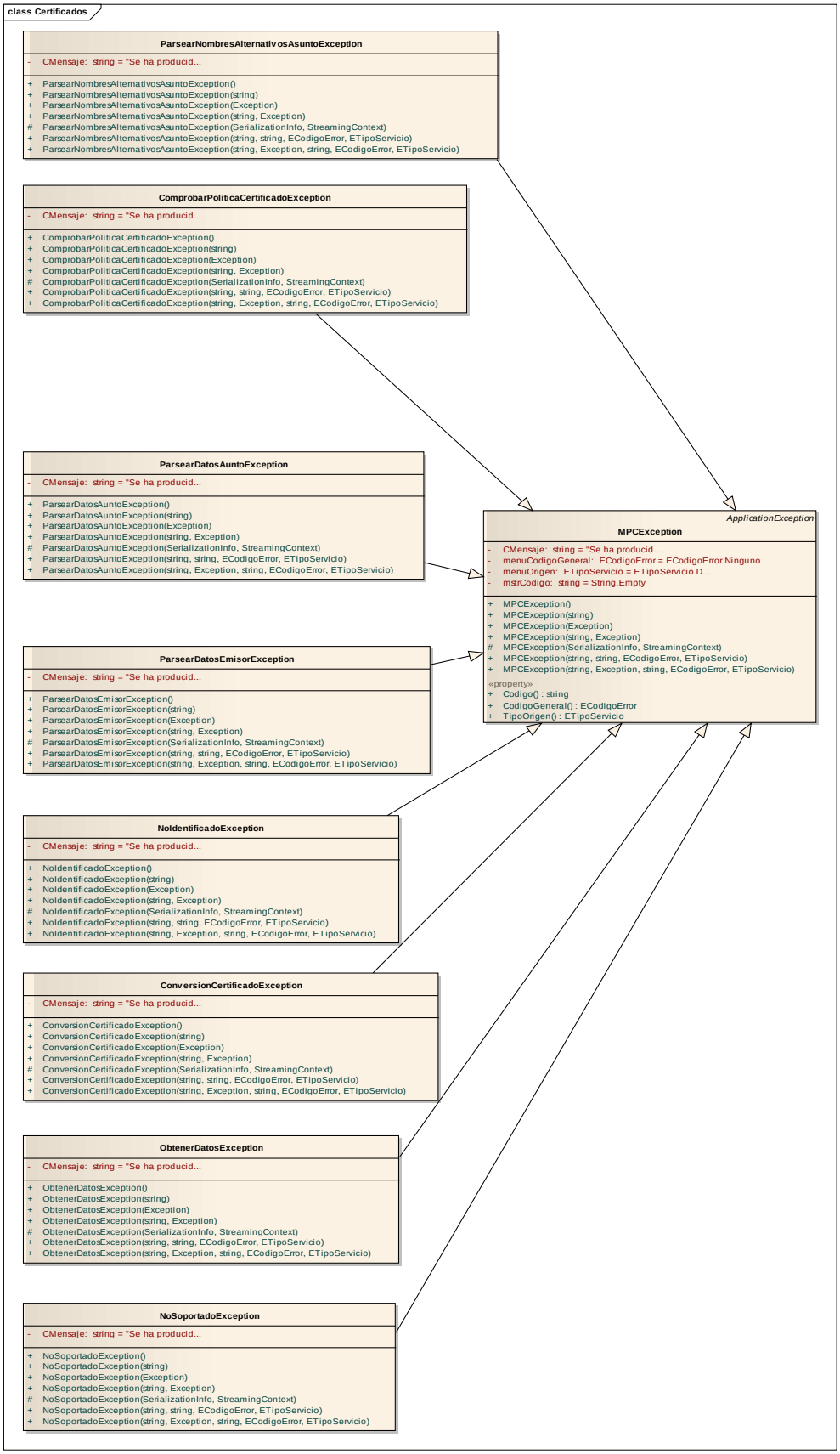


Ilustración 77: Clases para Excepciones de obtención de datos

Este conjunto de excepciones están relacionadas con errores producidos durante la extracción de datos o parseo del certificado. Todas heredan también de la clase *MpcException* y establecen los siguientes mensajes de error:

- *ParsearNombresAlternativosAsuntoException:* "Se ha producido un error al parsear los nombres alternativos del asunto del certificado."
- *ComprobarPoliticaCertificadoException:* "Se ha producido un error al comprobar la política del certificado."
- *ParsearDatosAuntoException:* "Se ha producido un error al parsear los datos del asunto del certificado."
- *ParsearDatosEmisorException:* "Se ha producido un error al parsear los datos del emisor del certificado."
- *NoIdentificadoException:* "Se ha producido un error al identificar el prestador al que pertenece el certificado"
- *ConversionCertificadoException:* "Se ha producido un error al intentar convertir el certificado a un formato válido"
- *ObtenerDatosException:* "Se ha producido un error al obtener los datos del certificado."
- *NoSoportadoException:* "Se ha producido un error de certificado no soportado por el sistema."

Excepciones de Validación de datos:

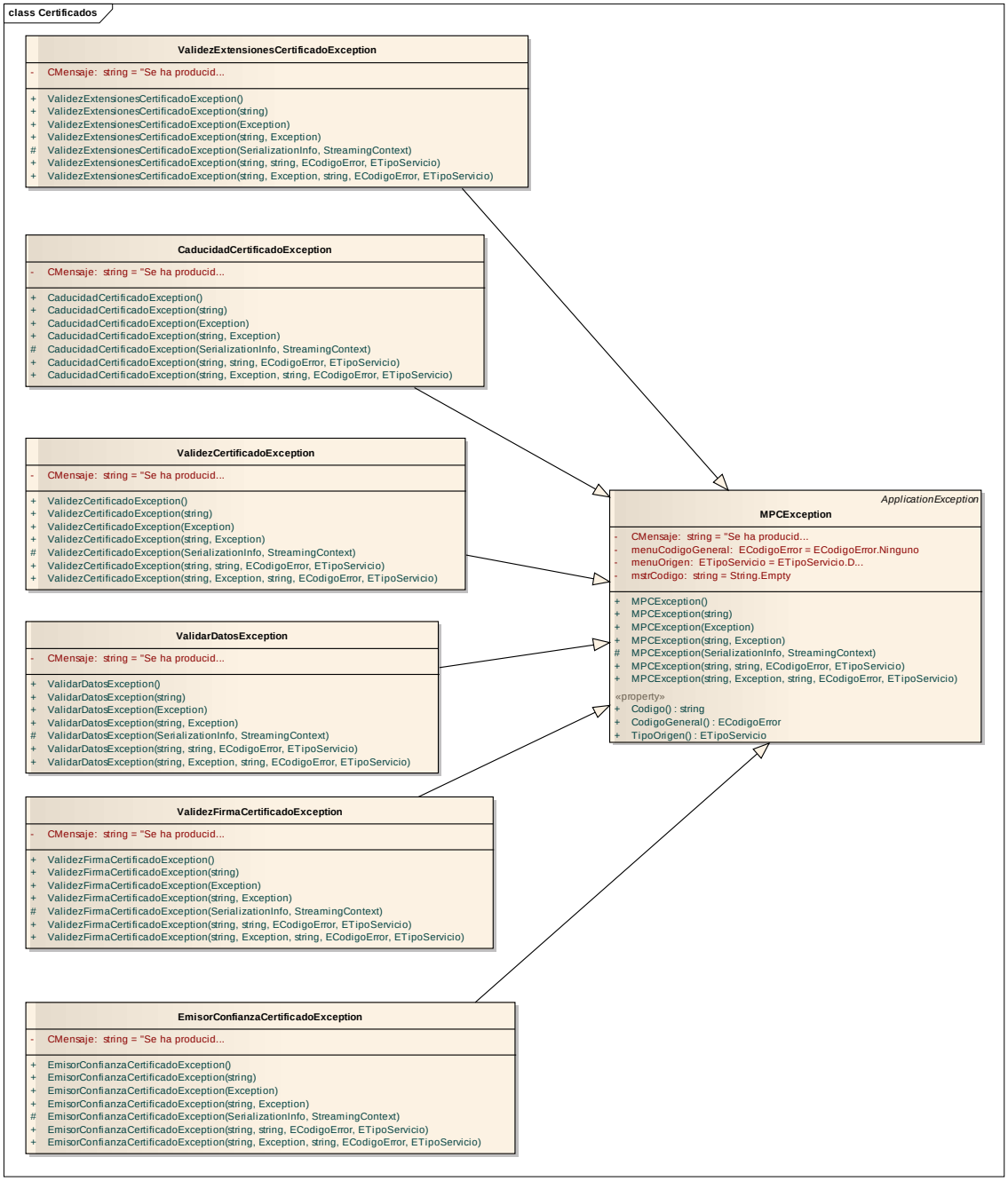


Ilustración 78: Clases para Excepciones de validación de datos

Este conjunto de excepciones están relacionadas con errores producidos durante la validación del certificado. Todas heredan también de la clase *MpcException* y establecen los siguientes mensajes de error:

- *ValidezExtensionesCertificadoException:* "Se ha producido un error al comprobar la validez de las extensiones del certificado."
- *CaducidadCertificadoException:* "Se ha producido un error al intentar validar el estado de caducidad del certificado."

- *ValidezCertificadoException:* "Se ha producido un error al intentar validar que la fecha actual no sea anterior a la fecha de comienzo del periodo de validez del certificado."

- *ValidarDatosException:* "Se ha producido un error al validar los datos del certificado."

- *ValidezFirmaCertificadoException:* "Se ha producido un error al comprobar la validez de la firma del certificado."

- *EmisorConfianzaCertificadoException:* "Se ha producido un error al intentar validar la confianza del emisor del certificado"

Clases principales del espacio de nombres Certificados:

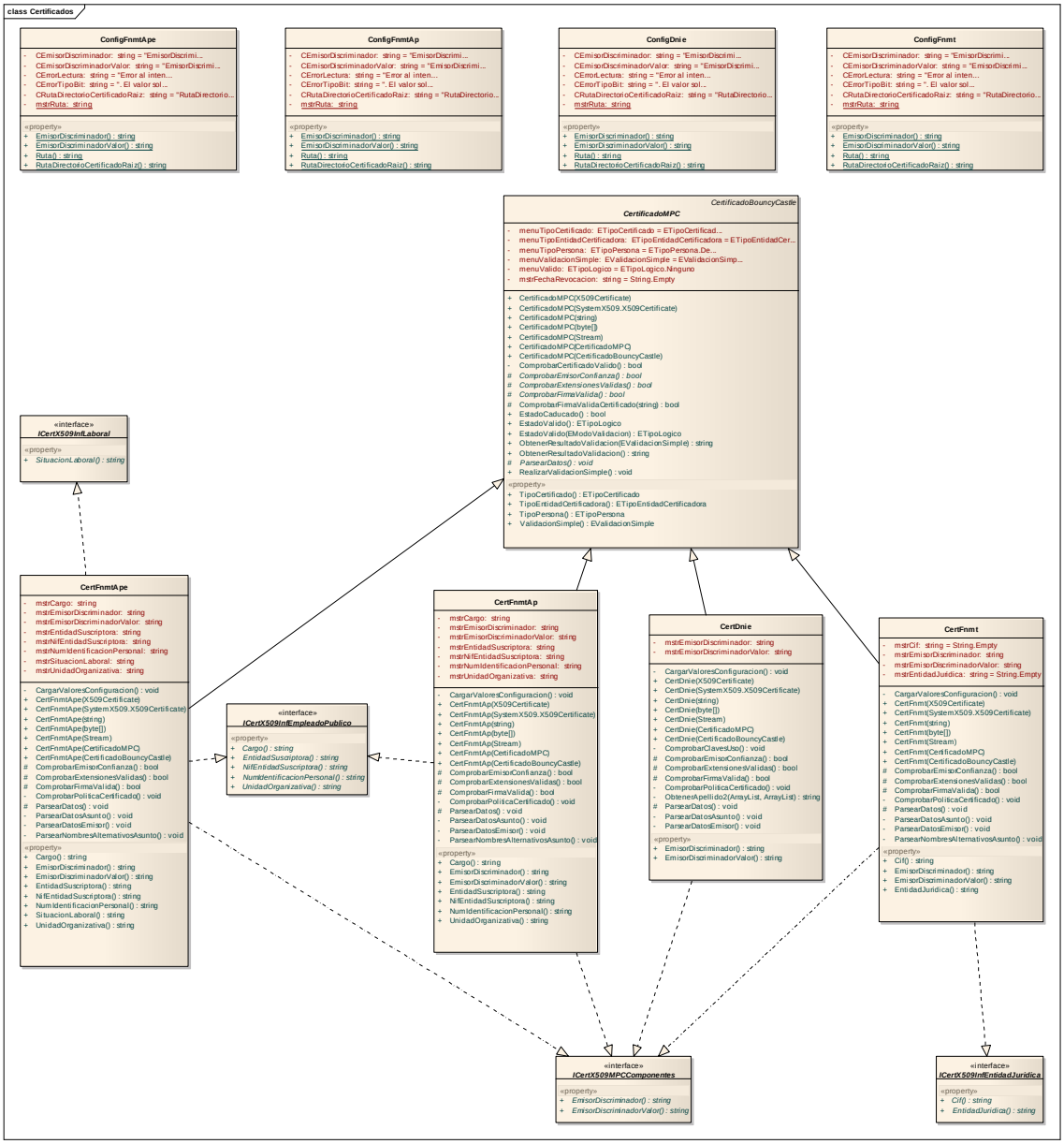


Ilustración 79: Diagrama de clases principales para el manejo de los certificados soportado por MPC

Cada librería particular de cada certificado posee 2 clases y un archivo de configuración. La clase principal hereda de la clase CertificadoMpc de la librería **MpcCertificadosLib** y es la encargada de realizar el parseo y la validación del certificado en cuestión. Esta clase implementa distintas características dependiendo de cada prestador. La otra clase de la librería se utiliza para poder leer información de un archivo de configuración particular llamado App.config dónde se establecen los discriminadores para identificar el certificado emisor por parseo y las rutas a los certificados emisores físicos para la distinción del certificado emisor mediante verificación. Si desgranamos el diagrama anterior por cada librería tenemos lo siguiente:

FnmtCertLib:

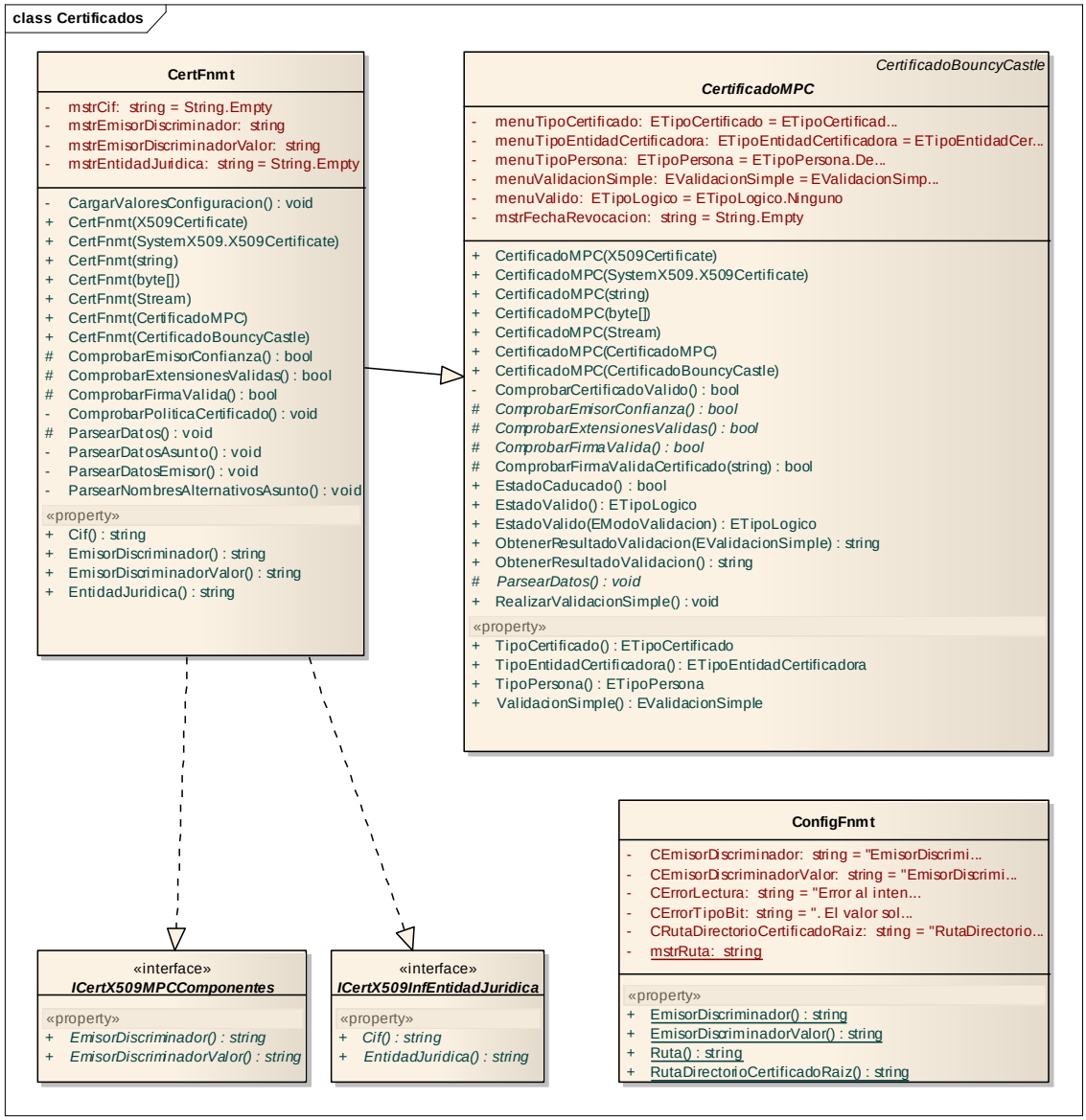


Ilustración 80: Diagrama de clases de la librería FnmtCertLib

La clase *CertFnmt* es la que tiene en este caso por clase base a la clase *CertificadoMPC*. Implementa dos interfaces. La interface *ICertX509InfEntidadJuridica añade los campos Cif y Entidad jurídica que son utilizados cuando el certificado es de tipo persona jurídica.* La interface *ICertX509MPCComponentes* tiene las propiedades EmisorDiscriminador (discriminador para conocer el prestador dentro del emisor {O/OU}) y EmisorDiscriminadorValor (valor del discriminador usado para conocer el prestador dentro del emisor {O/OU}). Estas dos propiedades se establecen en el constructor utilizando la clase estática ConfigFnmt para leer los valores, los cuales están especificados en el archivo de configuración App.config. Este fichero tiene los siguientes valores para la librería **FnmtCertLib**:

```
<?xml version="1.0" encoding="utf-8" ?>

<configuration>

    <appSettings>

        <!-- Discriminador de la entidad dentro del emisor -->

        <add key="EmisorDiscriminador" value="OU"/>

        <add key="EmisorDiscriminadorValor" value="FNMT Clase 2 CA"/>

        <add key="RutaDirectorioCertificadoRaiz" value="CertificadosRaices\Fnmt"/>

    </appSettings>

</configuration>
```

Ilustración 81: Archivo App.config de la librería FnmtCertLib

Esta información se utilizará en la clase *SelectorCertificado* para determinar ya sea mediante parseo o mediante verificación cual es el emisor del certificado que recibe como parámetro la aplicación y así poder determinar qué librería se debe usar para su parseo y validación.

La clase *CertFnmt* sobrescribe los métodos abstractos de su clase base (CertificadoMPC) ComprobarEmisorConfianza(), ComprobarFirmaValida(), y ComprobarExtensionesValidas() y ParsearDatos(). Estos métodos deben ser implementados aquí porque a este nivel ya se conoce el tipo concreto de certificado que se está procesando, en este caso un certificado de tipo Fnmt y se pueden consultar los OIDs propios del certificado especificados en su política de certificación. El método ParsearDatos() para el tipo Fnmt es el siguiente:

```
/// <summary>
/// Parsea los datos del certificado
/// </summary>
protected override void ParsearDatos()
{
    try
    {
        // Se establece que el tipo de entidad es la FNMT
        base.TipoEntidadCertificadora = ETipoEntidadCertificadora.Fnmt;

        // Se comprueba el tipo de persona (física/jurídica)
        ComprobarPoliticaCertificado();

        // Parsear los datos del emisor
        ParsearDatosEmisor();

        // Parsear los datos del asunto que se puedan obtener
        ParsearDatosAsunto();

        // Parsear el nombre, apellidos y nif a través de los nombres alternativos
        ParsearNombresAlternativosAsunto();
    }
    catch (NoSoportadoException)
    {
        throw;
    }
    catch (ComprobarPoliticaCertificadoException)
    {
        throw;
```

```
    }
    catch (ParsearDatosEmisorException)
    {
        throw;
    }
    catch (ParsearDatosAuntoException)
    {
        throw;
    }
    catch (ParsearNombresAlternativosAsuntoException)
    {
        throw;
    }
    catch (Exception)
    {
        throw;
    }
}
```

Ilustración 82: Método ParsearDatos() de la clase CertFnmt

En este método se establece directamente el tipo de entidad certificadora puesto que ya se sabe que es de tipo Fnmt y se llama a los siguientes métodos:

```
/// <summary>
/// Método que establece el TipoPersona del certificado dependiendo del tipo exacto de certificado
que sea.
/// Para los certificados de tipo FNMT se admiten cuatro tipos concretos de certificados
/// </summary>
private void ComprobarPoliticaCertificado()
{
    string strTipoPersona;

    // Constante con la politica del tipo de persona
    const string CPoliticaTipoPersona = "1.3.6.1.4.1.5734.1.33";
    // Constantes con los valores posibles del tipo de persona
    const string CPersonaFisica = "PERSONA FISICA";
    const string CPersonaJuridicaTributario = "CERTIFICADO EXCLUSIVO PARA EL AMBITO TRIBUTARIO";
    const string CPersonaJuridicaVarios = "CERTIFICADO CPJ FNMT/ÁMBITO PÚBLICO/OTROS ÁMBITOS
AUTORIZADOS-DESTINATARIO";
    const string CEntidadNoJuridicaTributario = "CERTIFICADO DE ENTIDAD SIN PERSONALIDAD JURÍDICA
EXCLUSIVO PARA EL ÁMBITO TRIBURARIO";

    try
    {
        if (base.ExisteExtension(CPoliticaTipoPersona))
        {
            strTipoPersona = base.ObtenerValorExtension(CPoliticaTipoPersona);
            if (!String.IsNullOrEmpty(strTipoPersona))
            {
                switch (strTipoPersona)
                {
                    case CPersonaFisica:
                        base.TipoPersona = ETipoPersona.Fisica;
                        break;

                    case CPersonaJuridicaTributario:
                        base.TipoPersona = ETipoPersona.Juridica;
                        break;

                    case CPersonaJuridicaVarios:
                        base.TipoPersona = ETipoPersona.Juridica;
                        break;

                    case CEntidadNoJuridicaTributario:
                        base.TipoPersona = ETipoPersona.Juridica;
                        break;

                }

                if (base.TipoPersona == ETipoPersona.Desconocido) throw new NoSoportadoException("El
tipo de certificado " + strTipoPersona + " del prestador FNMT no está soportado por la
aplicación.");
            }
        }
    }
    catch (NoSoportadoException)
    {
        throw;
```

```
    }
    catch (Exception ex)
    {
        throw new ComprobarPoliticaCertificadoException(ex);
    }
}
```

Ilustración 83: Método ComprobarPoliticaCertificado de la clase CertFnmt

El método ComprobarPoliticaCertificado() se encarga de establecer el valor del campo TipoPersona, el cual solo puede ser Física o Jurídica según los requisitos de la aplicación. Por tanto el método lleva a cabo una equivalencia entre los distintos tipos aceptados para determinar cuáles son personas físicas y cuales personas jurídicas. Este método también lanza una excepción de certificado no soportado cuando se procesa algún tipo de certificado de Fnmt Clase 2 CA no admitido, como por ejemplo los certificados de Sede o de Sello.

```
/// <summary>
/// Parsea los datos del emisor
/// </summary>
private void ParsearDatosEmisor()
{
    X509Name objEmisor;
    ArrayList colValores;
    ArrayList colOids;
    string strEmisor;

    try
    {
      strEmisor = base.Emisor;
      if (!String.IsNullOrEmpty(strEmisor))
      {
          objEmisor = new X509Name(strEmisor);
          if (objEmisor != null)
          {
            colValores = objEmisor.GetValues();
            colOids = objEmisor.GetOids();

            base.EntidadCertificadora = base.ParsearValorDN(X509Name.O, colValores, colOids);
            base.UnidadEntidadCertificadora = base.ParsearValorDN(X509Name.OU, colValores, colOids);
            base.DatosEntidadCertificadora = base.ParsearValorDN(X509Name.CN, colValores, colOids);

          }
      }
    }
    catch (Exception ex)
    {
        throw new ParsearDatosEmisorException(ex);
    }
}
```

Ilustración 84: Método ParsearDatosEmisor() de la clase CertFnmt

El método ParsearDatosEmisor() extrae del certificado los campos que deben mostrarse en la salida contenidos en la estructura Emisor del certificado. Para este certificado de tipo Fnmt Clase 2 CA se extraen los campos *entidad certificadora*, *unidad de la entidad certificadora* y *datos de entidad certificadora*.

```
/// <summary>
/// Parsea los datos del asunto
/// </summary>
private void ParsearDatosAsunto()
{
    X509Name objAsunto;
    ArrayList colValores;
```

```
    ArrayList colOids;
    string strAsunto;
    //string strDatosNombre;

    try
    {
        strAsunto = base.Asunto;
        if (!String.IsNullOrEmpty(strAsunto))
        {
            objAsunto = new X509Name(strAsunto);
            if (objAsunto != null)
            {
                colValores = objAsunto.GetValues();
                colOids = objAsunto.GetOids();

                base.Pais = base.ParsearValorDN(X509Name.C, colValores, colOids);
            }
        }
        }
    catch (Exception ex)
    {
        throw new ParsearDatosAuntoException(ex);
    }
}
```

Ilustración 85: Método ParsearDatosAsunto() de la clase CertFnmt

El método ParsearDatosAsunto() extrae del certificado los campos que deben mostrarse en la salida contenidos en la estructura Asunto del certificado. Para este certificado de tipo Fnmt Clase 2 CA se extrae solo el campo *país*.

```
/// <summary>
/// Parsea los datos de la extensión de nombres alternativos del asunto
/// </summary>
private void ParsearNombresAlternativosAsunto()
{
    ArrayList colExtensiones = null;
    ArrayList colExtension = null;
    string strExtensiones = null;
    string strValor = null;
    int intIdExt;

    // Constantes de parseo
    const int CExtId = 0;
    const int CExtValor = 1;
    // Constantes con los OID con los campos
    const string CNombre = "1.3.6.1.4.1.5734.1.1";
    const string CApellido1 = "1.3.6.1.4.1.5734.1.2";
    const string CApellido2 = "1.3.6.1.4.1.5734.1.3";
    const string CNif = "1.3.6.1.4.1.5734.1.4";
    const string CEntidad = "1.3.6.1.4.1.5734.1.6";
    const string CCif = "1.3.6.1.4.1.5734.1.7";
    // Constantes para expresiones regulares
    const string CPatron = @"=(?<oid>[^,]*)";
    const string COid = "oid";

    try
    {
        colExtensiones = base.ObtenerNombresAlternativosAsunto();
        if (colExtensiones != null)
        {
            for (int i = 0; i < colExtensiones.Count; i++)
            {
                colExtension = (ArrayList)colExtensiones[i];

                intIdExt = (int)colExtension[CExtId];
                strValor = colExtension[CExtValor].ToString();

                switch (intIdExt)
                {
                    case GeneralName.Rfc822Name:
                        if (String.IsNullOrEmpty(base.Email))
                        {
                            base.Email = strValor;
                        }
                        break;
```

```
                    case GeneralName.DirectoryName:
                        strExtensiones = strValor;
                        break;
                }
            }

            if (!String.IsNullOrEmpty(strExtensiones))
            {
                if (String.IsNullOrEmpty(base.Nombre))
                {
                    base.Nombre = base.ParsearValorNombreAlternativo(strExtensiones, CNombre);
                }
                if (String.IsNullOrEmpty(base.Apellido1))
                {
                    base.Apellido1 = base.ParsearValorNombreAlternativo(strExtensiones, CApellido1);
                }
                if (String.IsNullOrEmpty(base.Apellido2))
                {
                    base.Apellido2 = base.ParsearValorNombreAlternativo(strExtensiones, CApellido2);
                }
                if (String.IsNullOrEmpty(base.Nif))
                {
                    base.Nif = base.ParsearValorNombreAlternativo(strExtensiones, CNif);
                }

                // Certificado de entidad jurídica
                if (base.TipoPersona == ETipoPersona.Juridica)
                {
                    if (String.IsNullOrEmpty(this.EntidadJuridica))
                    {
                        this.EntidadJuridica = base.ParsearValorNombreAlternativo(strExtensiones,
CEntidad);
                    }
                    if (String.IsNullOrEmpty(this.Cif))
                    {
                        this.Cif = base.ParsearValorNombreAlternativo(strExtensiones, CCif);
                    }
                }

                #region Expresiones regulares

                if (base.DatosPersonalesIncompletos())
                {
                    if (String.IsNullOrEmpty(base.Nombre))
                    {
                        base.Nombre = ParsearValorRegEx(strExtensiones, CNombre + CPatron, COid);
                    }
                    if (String.IsNullOrEmpty(base.Apellido1))
                    {
                        base.Apellido1 = ParsearValorRegEx(strExtensiones, CApellido1 + CPatron,
COid);
                    }
                    if (String.IsNullOrEmpty(base.Apellido2))
                    {
                        base.Apellido2 = ParsearValorRegEx(strExtensiones, CApellido2 + CPatron,
COid);
                    }
                    if (String.IsNullOrEmpty(base.Nif))
                    {
                        base.Nif = ParsearValorRegEx(strExtensiones, CNif + CPatron, COid);
                    }

                    // Certificado de entidad jurídica
                    if (base.TipoPersona == ETipoPersona.Juridica)
                    {
                        if (String.IsNullOrEmpty(this.EntidadJuridica))
                        {
                            this.EntidadJuridica = ParsearValorRegEx(strExtensiones, CEntidad +
CPatron, COid);
                        }
                        if (String.IsNullOrEmpty(this.Cif))
                        {
                            this.Cif = ParsearValorRegEx(strExtensiones, CCif + CPatron, COid);
                        }
                    }
                }

                #endregion

            }

            // Nombre completo
            if (String.IsNullOrEmpty(base.NombreCompleto))
            {
                base.EstablecerNombreCompleto();
            }
        }
    }
```

```
    catch (CertificateParsingException ex)
    {
        throw new ParsearNombresAlternativosAsuntoException(ex);
    }
    catch (Exception ex)
    {
        throw new ParsearNombresAlternativosAsuntoException(ex);
    }
}
```

Ilustración 86: Método ParsearNombresAlternativosAsunto() de la clase CertFnmt

El método ParsearNombresAlternativosAsunto() se encarga de extraer el resto de campos relevantes del asunto del certificado pero no los extrae directamente del mismo. En ocasiones los certificados incluyen extensiones, de forma que cada campo tiene un oid propio y directamente accesible y no es necesario parsear con expresiones regulares el campo asunto. De esta forma este método obtiene los campos *nombre, apellido 1, apellido 2, nif, entidad, y Cif.* También se establece el campo *nombre completo*.

Referente a los métodos que intervienen en la validación y para que sirva de muestra se muestran a continuación los 3 métodos sobrescritos ComprobarEmisorConfianza(), ComprobarFirmaValida(), y ComprobarExtensionesValidas(). Alguno como ComprobarFirmaValida() utiliza a su vez métodos implementados en la clase base *CertificadoMPC.*

```
/// <summary>
/// Comprobar si el emisor es de confianza y por lo tanto soportado por el sistema
/// </summary>
/// <returns>Verdadero si el emisor es de confianza, Falso en caso contrario</returns>
protected override bool ComprobarEmisorConfianza()
{
    bool bolCorrecto = false;

    try
    {
        bolCorrecto =
this.ComprobarEmisor(ConfigDnie.EmisorDiscriminador.Trim().ToUpper(),
ConfigDnie.EmisorDiscriminadorValor.Trim().ToUpper());
        return bolCorrecto;
    }
    catch (Exception ex)
    {
        throw new EmisorConfianzaCertificadoException(ex);
    }
}



public bool ComprobarEmisor(string pstrDiscriminador, string pstrDiscriminadorValor)
{
    string strEmisorParseado = null;
    bool bolTipoIdentificado = false;
    const string CDiscriminadorO = "O";
    const string CDiscriminadorOU = "OU";

    try
    {
        switch (pstrDiscriminador)
        {
```

```
            case CDiscriminadorO:
                strEmisorParseado = ParsearValorEmisor(X509Name.O);
                break;

            case CDiscriminadorOU:
                strEmisorParseado = ParsearValorEmisor(X509Name.OU);
                break;
        }

        if (!String.IsNullOrEmpty(strEmisorParseado))
        {
            if (strEmisorParseado.Trim().ToUpper().Equals(pstrDiscriminadorValor))
            {
                bolTipoIdentificado = true;
            }
        }
    }
    catch (Exception)
    {
        throw;
    }

    return bolTipoIdentificado;

}
```

Ilustración 87: Método ComprobarEmisorConfianza() de la clase CertFnmt

```
/// <summary>
/// Comprobar si la firma del certificado es con la clave pública del emisor
/// </summary>
/// <returns>Verdadero si la firma es válida, Falso en caso contrario</returns>
protected override bool ComprobarFirmaValida()
{
    bool bolCorrecto = false;

    try
    {
        bolCorrecto =
base.ComprobarFirmaValidaCertificado(ConfigFnmt.RutaDirectorioCertificadoRaiz);
        return bolCorrecto;
    }
    catch (IOException ex)
    {
        throw new ValidezFirmaCertificadoException(ex);
    }
    catch (Exception ex)
    {
        throw new ValidezFirmaCertificadoException(ex);
    }
}
```

Ilustración 88: Método ComprobarFirmaValida() de la clase CertFnmt

```
/// <summary>
/// Comprueba que el certificado contenga las extensiones críticas admitidas en su
politica de certificación
/// </summary>
/// <returns>Verdadero si las extensiones son válidas, Falso en caso
contrario</returns>
protected override bool ComprobarExtensionesValidas()
{
    bool bolCorrecto = false;

    try
    {
        //No tiene extensiones críticas por lo tanto siempre pasará ésta validación
        bolCorrecto = true;
        return bolCorrecto;
    }
    catch (Exception ex)
    {
        throw new ValidezExtensionesCertificadoException(ex);
    }
}
```

Ilustración 89: Método ComprobarExtensionesValidas() de la clase CertFnmt

A continuación se muestran los diagramas de las clases para el parseo y validación de los certificados FnmtAp, FnmtApe y Dnie. Para no resultar redundante no se va a entrar a explicar cada clase ya que su funcionamiento e implementación son similares a los ya explicados para los certificados Fnmt.

FnmtApCertLib:

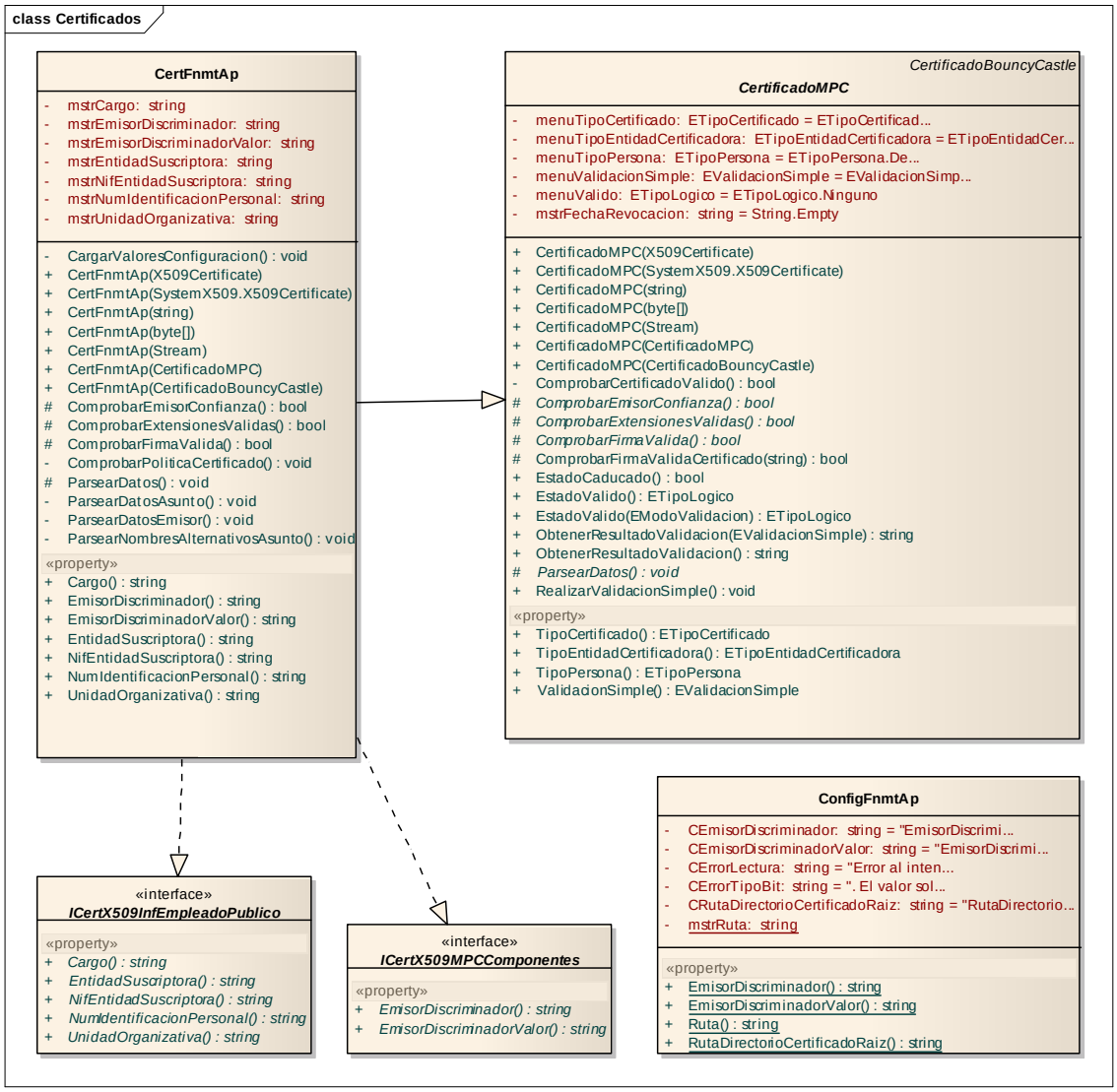


Ilustración 90: Diagrama de clases de la librería FnmtApCertLib

FnmtApeCertLib:

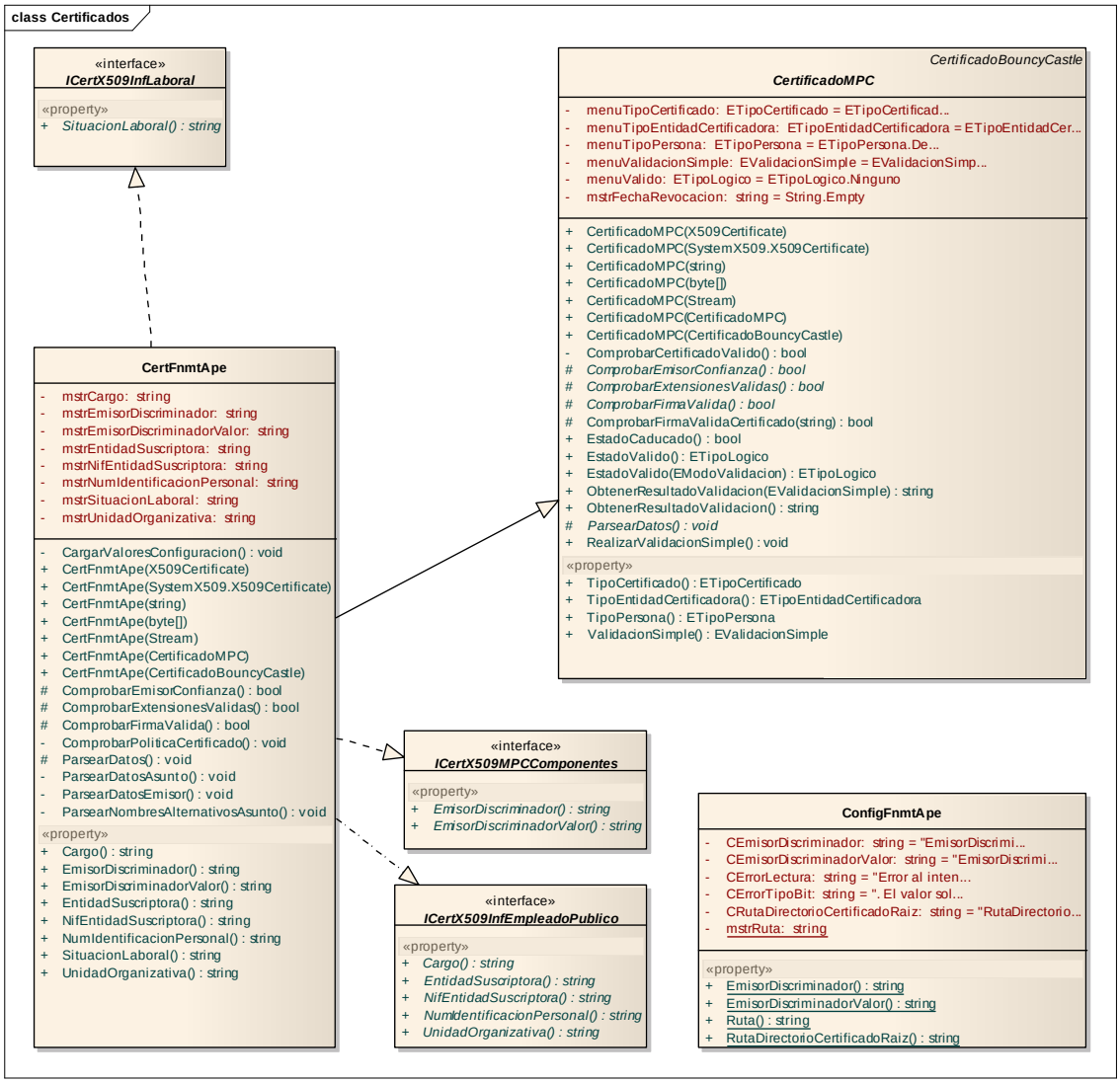


Ilustración 91: Diagrama de clases de la librería FnmtApeCertLib

DnieCertLib:

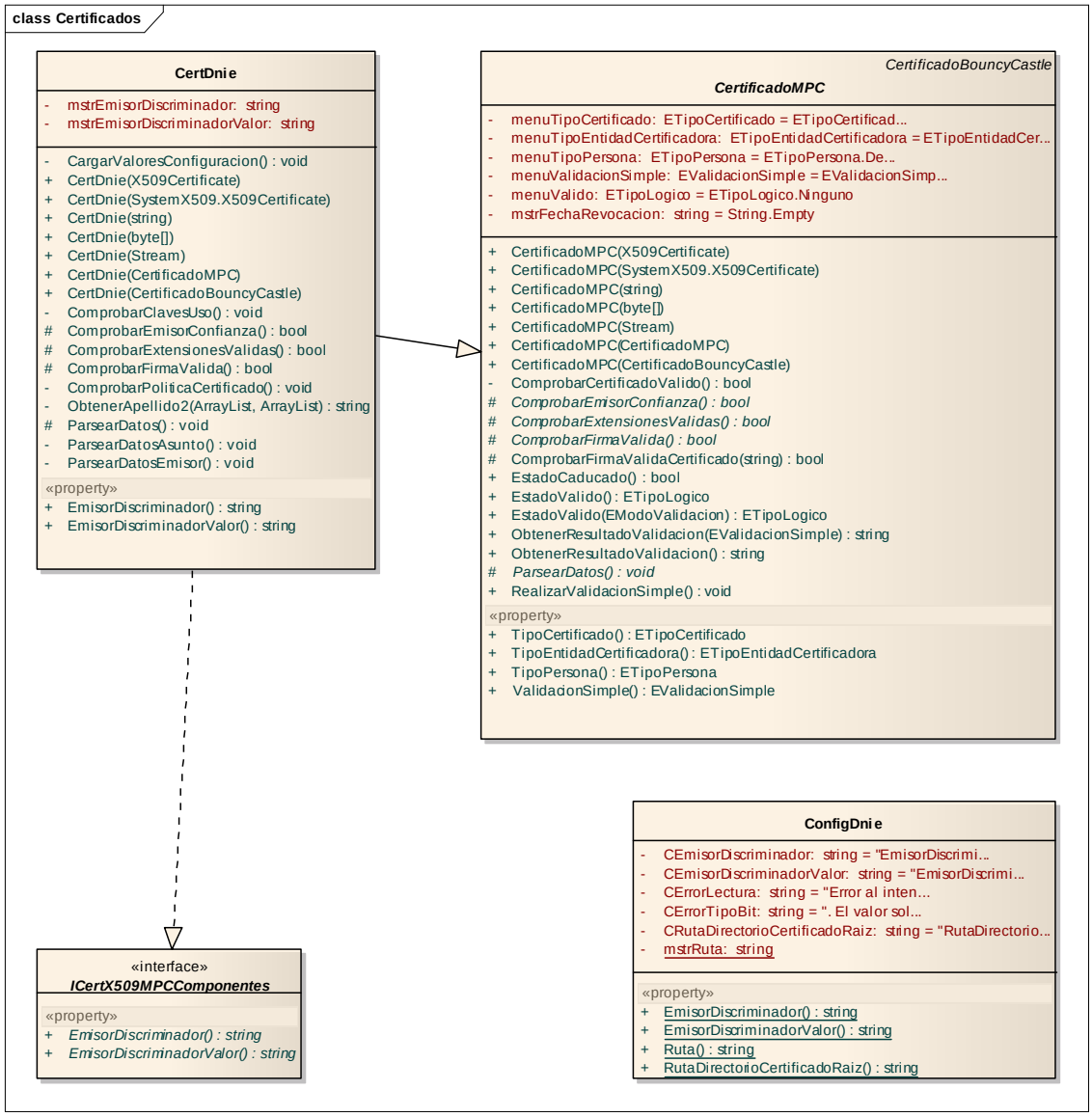


Ilustración 92: Diagrama de clases de la librería DnieCertLib

Servicio Web MPC

A continuación se mostrarán las clases que forman parte del propio proyecto MPC, y que por tanto no están separadas en componentes en forma de librerías DLL. Las clases del servicio web MPC están también estructuradas en varios espacios de nombres, aunque para tener una visión más global se mostrarán juntas en varios diagramas.

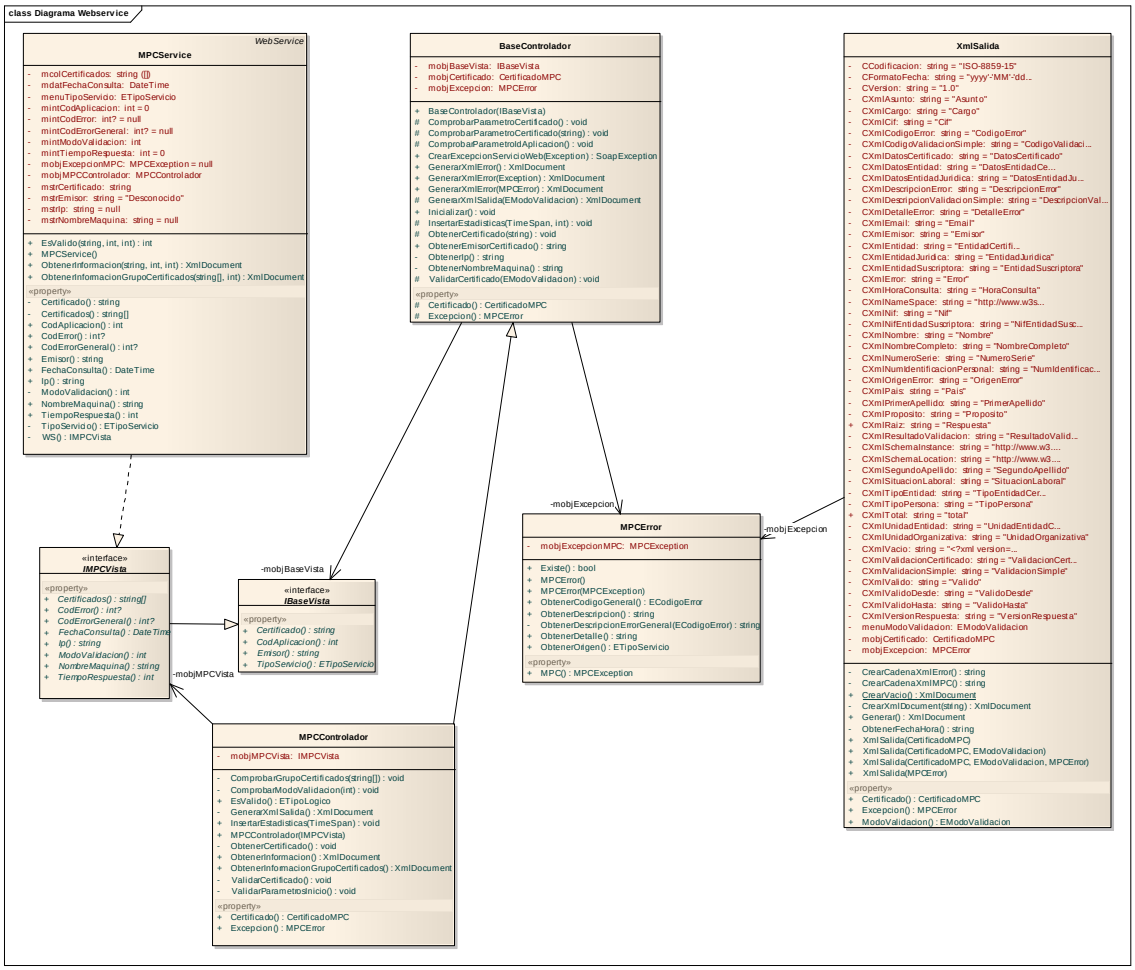


Ilustración 93: Diagrama de clases servicio web MPC

Como se ha comentado anteriormente se ha utilizado el patrón vista controlador para separar la lógica de negocio de la capa de presentación. En este caso al ser un servicio web, el sistema MPC no tiene parte visible como tal pero el patrón resulta igualmente útil ya que permite presentar el código de una forma más limpia, separando los métodos principales del servicio web - que son los que invocarán las aplicaciones - de las operaciones internas de parseo, validación e inserción de estadísticas.

Así pues, la propia clase *MPCService* usa una variable de tipo *MPCcontrolador* que será la encargada de invocar a todos los métodos del sistema.

La clase *MPCService* implementa también la interface *IMPCVista*, completando así el patrón vista-controlador. Los parámetros no son pasados al controlador a través de las funciones sino que accede a ellos a través del objeto vista.

La clase *MPCError* se utiliza para almacenar los errores de la aplicación. En ella se almacenará la excepción en caso de que exista y contiene métodos para obtener la descripción, el código, el detalle y el origen del error.

La clase *XmlSalida* es la encargada de generar el XML de salida del servicio MPC. Crea todo la estructura xml cumpliendo el xsd definido previamente y completando los datos de parseo, validación y excepciones producidas según el caso concreto.

El resto de clases del WebService son las siguientes:

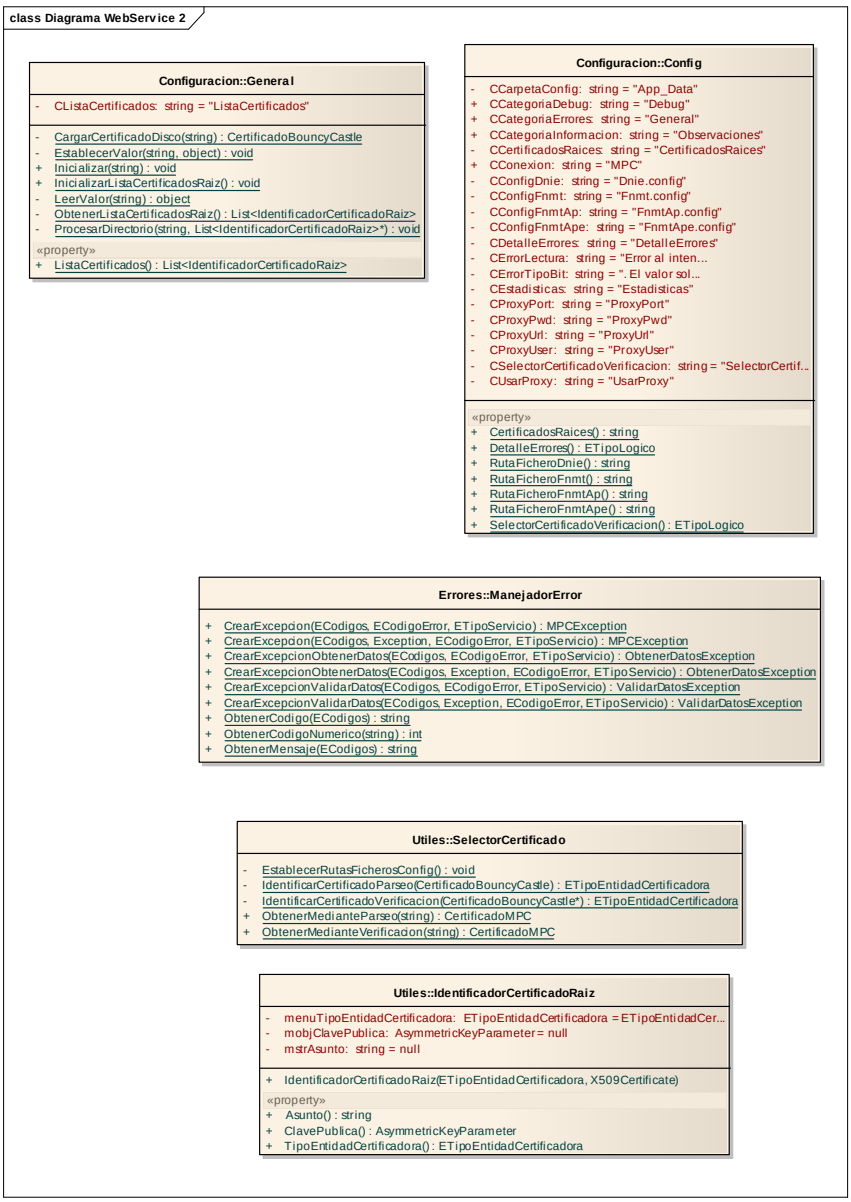


Ilustración 94: Clases del servicio web MPC

Las clases *General* y *Config* se agrupan dentro del espacio de nombres Configuración. La clase *General* tiene métodos que se utilizan para inicializar el sistema la primera vez (cadena de conexión, inicialización de lista de certificados raíces, etc) mientras que la clase Config se utiliza para leer información del archivo de configuración de la aplicación. En este archivo, denominado Web.config se parametrizan valores del sistema MPC como son:

- Configuración del Log
- Configuración nivel de detalle de errores
- Configuración cadena de conexión de la base de datos
- Ubicación de ficheros de configuración de prestadores
- Ubicación directorio certificados raíces
- Configuración del método selector de certificado (mediante verificación o mediante parseo)

La clase *ManejadorError* se encuentra dentro del espacio de nombres Errores. Es una clase estática que se utiliza para el manejo de excepciones generales de MPC.

La clase *IdentificadorCertificadoRaiz* es una clase auxiliar enmarcada dentro del espacio de nombres Útiles. Se utiliza para formar la lista de certificados raíces admitidos. Cuando se carga en memoria la lista de certificados raíces se guardan en una lista de clases de este tipo. Para cada certificado raíz se almacenan: el tipo de entidad certificadora, la clave pública y el asunto. Esta información se utiliza para verificar los certificados que se reciban como parámetros de entrada al servicio web.

La clase SelectorCertificado es quizá la más importante de este grupo. Está también dentro del espacio de nombres Útiles y actúa como “factoría” de clases. La clase SelectorCertificado es una clase estática que determina a que tipo de prestador pertenece el certificado que se le pasa y por tanto es capaz de crear la clase correspondiente para poder llevar a cabo su parseo y validación. Para ello puede utilizar dos métodos diferentes según el que esté establecido en el archivo de configuración de la aplicación (Web.config) Los dos métodos son los siguientes:

- ObtenerMedianteParseo: Compara ciertos valores del campo emisor del certificado con los configurados como discriminadores para cada tipo de prestador. Estos valores discriminadores están establecidos en los archivos de configuración de cada prestador. Por ejemplo, para el DNIe, en el archivo dnie.config se establece que el discriminador es el campo “OU” y que el valor

de dicho campo es “DNIE”. Si estos datos coinciden con los del certificado que se está analizando se determinará que es de tipo DNIe y se creará una clase de tipo CertDnie. Si no coincide con ningún prestador se lanzará una excepción de certificado no soportado.

- ObtenerMedianteVerificacion: En este caso lo que se hace es comprobar si el certificado es verificado por algún prestador soportado. Puesto que los certificados se firman con la clave privada de su emisor, y para verificar dicha firma se necesitan las claves públicas, se prueba con todos los certificados raíces que se tienen almacenados en el sistema y que se corresponden con las cadenas de certificación de los prestadores soportados por MPC. Si en alguno de los casos se obtiene una verificación positiva se determina que dicho certificado es de ese tipo concreto y en consecuencia se llama al constructor de la clase correspondiente. En caso de no ser verificado por ninguna clave pública se lanzará una excepción de certificado no soportado.

Métodos del servicio Web

Los 3 métodos del Servicio Web son los siguientes: *ObtenerInformacion*, *EsValido* y *ObtenerInformacionGrupoCertificados*.

```
[WebMethod(Description = "Extrae y/o parsea los datos de un certificado. Si se usa validación, comprueba que sea correcto.")]
public XmlDocument ObtenerInformacion(string pstrCertificado, int pintModoValidacion, int pintCodigoAplicacion)
{
  ETipoServicio enuTipoServicio = ETipoServicio.Desconocido;
  XmlDocument objXmlSalida = new XmlDocument();
  TimeSpan objTiempoRespuesta = TimeSpan.Zero;
  Stopwatch objTemporizador = null; ;

  try
  {
    objTemporizador = Stopwatch.StartNew();

    #region Determinar el tipo de servicio

    switch (pintModoValidacion)
    {
      case 0:
        enuTipoServicio = ETipoServicio.ObtenerInformacion;
        break;
      case 1:
        enuTipoServicio = ETipoServicio.ObtenerInformacionConValidacion;
        break;
     }

     #endregion

     #region Guardar las variables de entrada para usar el MVC

     WS.TipoServicio = enuTipoServicio;
     WS.Certificado = pstrCertificado;
     WS.ModoValidacion = pintModoValidacion;
     WS.CodAplicacion = pintCodigoAplicacion;

     #endregion
```

```
     objXmlSalida = mobjMPCControlador.ObtenerInformacion();


  }
  catch (Exception ex) // Error no controlado
  {
    LogHelper.Error(ex);
    objXmlSalida = mobjMPCControlador.GenerarXmlError(ex);
  }

  #region Estadísticas
  finally
  {
    objTiempoRespuesta = objTemporizador.Elapsed;

    try
    {
      mobjMPCControlador.InsertarEstadisticas(objTiempoRespuesta);
    }
    catch (InsertarEstadisticasException ex)
    {
      LogHelper.Error(ex);
    }
  }
  #endregion

return objXmlSalida;

}
```

Ilustración 95: Método ObtenerInformacion() del servicio web MPC

```
[WebMethod(Description = "Comprueba si un certificado es válido.")]
public int EsValido(string pstrCertificado, int pintModoValidacion, int
pintCodigoAplicacion)
{
  ETipoLogico enuValido = ETipoLogico.Ninguno;
  TimeSpan objTiempoRespuesta = TimeSpan.Zero;
  Stopwatch objTemporizador = null;


  try
  {
    objTemporizador = Stopwatch.StartNew();

    #region Guardar las variables de entrada en la vista

    WS.TipoServicio = ETipoServicio.ValidarCertificado;
    WS.Certificado = pstrCertificado;
    WS.ModoValidacion = pintModoValidacion;

    #endregion

    enuValido = mobjMPCControlador.EsValido();

  }
  catch (Exception ex) // Error no controlado
  {
    LogHelper.Error(ex);
    // Se devuelve el valor por defecto (-1)
  }

  #region Estadísticas
  finally
  {
    objTiempoRespuesta = objTemporizador.Elapsed;

    try
    {
      mobjMPCControlador.InsertarEstadisticas(objTiempoRespuesta);
    }
    catch (InsertarEstadisticasException ex)
    {
      LogHelper.Error(ex);
    }
  }
```

```
  #endregion

return (int)enuValido;

}
```

Ilustración 96: Método EsValido() del servicio web MPC

```
[WebMethod(Description = "Extrae y parsea los datos de un grupo de certificados.")]
public XmlDocument ObtenerInformacionGrupoCertificados(string[] pstrCertificados, int
pintCodigoAplicacion)
{
  XmlDocument objXmlSalida = new XmlDocument();


  try
  {
    #region Guardar las variables de entrada en la vista

    WS.TipoServicio = ETipoServicio.ObtenerInformacion;
    WS.Certificados = pstrCertificados;
    WS.CodAplicacion = pintCodigoAplicacion;

    #endregion

    objXmlSalida = mobjMPCControlador.ObtenerInformacionGrupoCertificados();

  }
  catch (Exception ex) // Error no controlado
  {
    LogHelper.Error(ex);
    objXmlSalida = mobjMPCControlador.GenerarXmlError(ex);
  }

return objXmlSalida;

}
```

Ilustración 97: Método ObtenerInformacionGrupoCertificados() del servicio web MPC

5.5.1 Ejemplo de llamada al método ObtenerInformacion() de MPC

A continuación se van a mostrar los métodos a los que llamaría el sistema MPC si recibe parámetros válidos (Certificado, Validación y aplicación autorizada) para el método ObtenerInformacion() de MPC. De esta forma daremos una idea del camino que llevaría el programa para una consulta sin errores de un certificado.

Antes de entrar en el método ObtenerInformación del controlador se debe mencionar lo que se lleva a cabo la primera vez que se invoca al servicio web. Lo primero que hace el sistema MPC cuando arranca -mediante su constructor- es invocar al método inicializar del controlador.

```
public MPCService()
{
  ServicePointManager.ServerCertificateValidationCallback =
  new RemoteCertificateValidationCallback
  (TrustAllCertificatePolicy.CheckValidationResult);

  mobjMPCControlador = new MPCControlador(this);

  mobjMPCControlador.Inicializar();
}
```

Ilustración 98: Constructor del servicio web MPC

Este método es el encargado de crear la cadena de conexión de la base de datos utilizando para ello la información del archivo de configuración de la aplicación (Web.config). También se encarga de volcar en memoria la lista de certificados raíces soportados por el sistema. Los certificados se encuentran en la carpeta “CertificadosRaices” dentro de subcarpetas, una por cada entidad certificadora soportada. De esta forma no se tienen que leer de disco todas las veces que se quieran utilizar.

Después se llama al método ObtenerInformacion() del controlador y finalmente se escriben las estadísticas en la base de datos mediante el método InsertarEstadisticas().

Método ObtenerInformacion() de la clase MPCControlador

```
/// <summary>
/// Extrae y/o parsea los datos de un certificado. Si se usa validación, comprueba que
sea correcto, pudiéndose comprobar en una fecha concreta, sino se validará con el día
actual.
/// </summary>
/// <returns>XmlDocument</returns>
public XmlDocument ObtenerInformacion()
{
  XmlDocument objXmlSalida = new XmlDocument();

  try
  {
    ValidarParametrosInicio();
```

```
    // Se carga parseando el certificado y se valida

    ObtenerCertificado();

    if ((ETipoServicio)mobjMPCVista.TipoServicio ==
    ETipoServicio.ObtenerInformacionConValidacion)
    {
      ValidarCertificado();
    }

    objXmlSalida = GenerarXmlSalida();

  }
  catch (MPCException ex)
  {
    LogHelper.Error(ex);
    objXmlSalida = GenerarXmlError(ex);
  }
  catch (Exception ex)
  {
    LogHelper.Error(ex);
    objXmlSalida = GenerarXmlError(ex);
  }

return objXmlSalida;

}
```

Ilustración 99: Método ObtenerInformacion() de la clase MPCControlador

Lo primero que se hace es comprobar que los parámetros de entrada sean válidos llamando al método ValidarParametrosInicio(). En caso de que todo sea correcto se obtienen los datos del certificado mediante el método ObtenerCertificado(). Si se ha elegido validación se invocará también al método ValidarCertificado(). Finalmente se generará el resultado mediante el método GenerarXmlSalida() si no ha ocurrido ninguna excepción en el proceso.

Método **ValidarParametrosInicio** de la clase *MPCcontrolador*

```
/// <summary>
/// Valida los parámetros introducidos por la aplicación al llamar al servicio web
/// </summary>
private void ValidarParametrosInicio()
{
  ETipoServicio enuMetodo;

  try
  {
    enuMetodo = mobjMPCVista.TipoServicio;
    //Se comprueba que la aplicación este autorizada
    base.ComprobarParametroIdAplicacion();

    switch (enuMetodo)
    {
      case ETipoServicio.ObtenerInformacion:
        if (!String.IsNullOrEmpty(mobjMPCVista.Certificado))
        {
          base.ComprobarParametroCertificado();
        }
        else // Es un grupo de certificados
        {
          ComprobarGrupoCertificados(mobjMPCVista.Certificados);
        }
        break;

      case ETipoServicio.ObtenerInformacionConValidacion:
        base.ComprobarParametroCertificado();
        break;
```

```
      case ETipoServicio.ValidarCertificado:
        base.ComprobarParametroCertificado();
        break;

      case ETipoServicio.Desconocido:
        throw new ArgumentException("El modo de validación debe ser uno de los
siguientes {0} si no se quiere validación o {1} para validación simple");
    }
  }
  catch (SqlException ex)
  {
    throw new ComprobarIdAplicacionException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.ErrorLeerIdAplicacionBaseDatos),
ECodigoError.ErrorBaseDatos, mobjMPCVista.TipoServicio);
  }
  catch (ArgumentOutOfRangeException ex)
  {
    throw new FormatoEntradaException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.ParametroAplicacionError),
ECodigoError.ParametrosIncorrectos, mobjMPCVista.TipoServicio);
  }
  catch (ArgumentNullException ex)
  {
    throw new FormatoEntradaException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.ParametroCertificadoNulo),
ECodigoError.ParametrosIncorrectos, mobjMPCVista.TipoServicio);
  }
  catch (ArgumentException ex)
  {
    throw new FormatoEntradaException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.ParametroValidacionError),
ECodigoError.ParametrosIncorrectos, mobjMPCVista.TipoServicio);
  }
  catch (FormatException ex)
  {
    throw new FormatoEntradaException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.ParametroCertificadoMalFormado),
ECodigoError.ParametrosIncorrectos, mobjMPCVista.TipoServicio);
  }
  catch (Exception ex)
  {
    throw new FormatoEntradaException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.NoIdentificado),
ECodigoError.ParametrosIncorrectos, mobjMPCVista.TipoServicio);
  }
}
```

Ilustración 100: Método ValidarParametrosInicio() de la clase MPCControlador

Lo primero que se comprueba es que la aplicación que invoca al sistema MPC tenga autorización para usar el servicio. Esto se lleva a cabo mediante una consulta a base de datos a través del método ComprobarParametroIdAplicacion() del controlador base. La aplicación estará autorizada si su identificador se encuentra dado de alta en la tabla Aplicaciones.

Para nuestro ejemplo puesto que queremos obtener la información y validación de un certificado se invocará al método ComprobarParametroCertificado(), el cual comprobará que el parámetro Certificado no sea nulo y que esté codificado en base64.

Método ObtenerCertificado() de la clase MPCControlador

```
/// <summary>
/// Carga el certificado X509
/// </summary>
private void ObtenerCertificado()
{
  try
  {
    base.ObtenerCertificado(mobjMPCVista.Certificado);
  }
  catch (ObtenerDatosException)
  {
    throw;
  }
  catch (Exception)
  {
    throw;
  }
}
```

Ilustración 101: Método ObtenerCertificado() de la clase MPCControlador

El método ObtenerCertificado() del controlador invocará al método ObtenerCertificado() del controlador base.

Método ObtenerCertificado() de la clase BaseControlador:

```
/// <summary>
/// Carga el certificado X509
/// </summary>
/// <param name="pstrCertificado">Certificado</param>
protected void ObtenerCertificado(string pstrCertificado)
{
  CertificadoMPC objCertificado;

  try
  {
    if (Config.SelectorCertificadoVerificacion == ETipoLogico.Si)
    {
      objCertificado = SelectorCertificado.ObtenerMedianteVerificacion(pstrCertificado);
    }
    else
    {
      objCertificado = SelectorCertificado.ObtenerMedianteParseo(pstrCertificado);
    }

    if (objCertificado != null && objCertificado.CertificadoX509BC != null)
    {
      mobjCertificado = objCertificado;
    }
  }
  catch (ComprobarPoliticaCertificadoException ex)
  {
    throw
ManejadorError.CrearExcepcionObtenerDatos(ManejadorError.ECodigos.ErrorComprobarPolitica
, ex, ECodigoError.ErrorObtenerDatos, mobjBaseVista.TipoServicio);
  }
  catch (ParsearDatosEmisorException ex)
  {
    throw
ManejadorError.CrearExcepcionObtenerDatos(ManejadorError.ECodigos.ErrorParsearEmisor,
ex, ECodigoError.ErrorObtenerDatos, mobjBaseVista.TipoServicio);
  }
  catch (ParsearDatosAuntoException ex)
  {
    throw
ManejadorError.CrearExcepcionObtenerDatos(ManejadorError.ECodigos.ErrorParsearAsunto,
ex, ECodigoError.ErrorObtenerDatos, mobjBaseVista.TipoServicio);
```

```
  }
  catch (ParsearNombresAlternativosAsuntoException ex)
  {
    throw
ManejadorError.CrearExcepcionObtenerDatos(ManejadorError.ECodigos.ErrorParsearNombresAlt
ernativos, ex, ECodigoError.ErrorObtenerDatos, mobjBaseVista.TipoServicio);
  }
  catch (NoSoportadoException ex)
  {
    throw
ManejadorError.CrearExcepcionObtenerDatos(ManejadorError.ECodigos.CertificadoNoSoportado
, ex, ECodigoError.CertificadoNoSoportado, mobjBaseVista.TipoServicio);
  }
  catch (ConversionCertificadoException ex)
  {
    throw
ManejadorError.CrearExcepcionObtenerDatos(ManejadorError.ECodigos.ConversionCertificadoE
rror, ex, ECodigoError.ErrorObtenerDatos, mobjBaseVista.TipoServicio);
  }
  catch (Exception ex)
  {
    throw ManejadorError.CrearExcepcionObtenerDatos(ManejadorError.ECodigos.Desconocido,
ex, ECodigoError.ErrorObtenerDatos, mobjBaseVista.TipoServicio);
  }
}
```

Ilustración 102: Método ObtenerCertificado() de la clase BaseControlador

El método ObtenerCertificado() utiliza la clase estática SelectorCertificado para devolver el certificado parseado. Para ello primero consulta el archivo Web.config y en función de si está configurado el método de parseo o el método de verificación se llamará al método correspondiente. Para nuestro ejemplo vamos a suponer que está configurado el método por verificación, por tanto se llamará a SelectorCertificado.ObtenerMedianteVerificacion().

Método ObtenerMedianteVerificacion():

```
/// <summary>
/// Obtiene la clase personalizada verificando con los certificados raíces en disco
/// </summary>
/// <param name="pstrCertificado">Certificado</param>
/// <returns>CertificadoMPC</returns>
public static CertificadoMPC ObtenerMedianteVerificacion(string pstrCertificado)
{
  CertificadoBouncyCastle objCertificado;
  CertificadoMPC objCertificadoMPC = null;
  ETipoEntidadCertificadora enuTipo = ETipoEntidadCertificadora.Desconocido;

  try
  {
    // Establece las rutas de los ficheros de configuración de los prestadores
    EstablecerRutasFicherosConfig();

    // Se crea un certificado BC genérico
    objCertificado = new CertificadoBouncyCastle(pstrCertificado);

    // Se crea el objeto de la clase correspondiente según el tipo
    enuTipo = IdentificarCertificadoVerificacion(ref objCertificado);
    switch (enuTipo)
    {
      case ETipoEntidadCertificadora.Fnmt:
        objCertificadoMPC = new CertFnmt(objCertificado);
        break;

      case ETipoEntidadCertificadora.FnmtApe:
        objCertificadoMPC = new CertFnmtApe(objCertificado);
        break;

      case ETipoEntidadCertificadora.FnmtAp:
        objCertificadoMPC = new CertFnmtAp(objCertificado);
        break;

      case ETipoEntidadCertificadora.Dnie:
        objCertificadoMPC = new CertDnie(objCertificado);
        break;
    }

    if (objCertificadoMPC == null)
    {
      throw new NoSoportadoException();
    }
  }
  catch (FormatException ex)
  {
     throw new ConversionCertificadoException(ex);
  }
  catch (CertificateParsingException ex)
  {
    throw new ConversionCertificadoException(ex);
  }
  catch (CertificateException ex)
  {
    throw new ConversionCertificadoException(ex);
  }
  catch (ComprobarPoliticaCertificadoException)
  {
    throw;
  }
  catch (NoSoportadoException)
  {
    throw;
  }
  catch (Exception)
  {
    throw;
  }

  return objCertificadoMPC;
}
```

Ilustración 103: Método ObtenerMedianteVerificacion() de la clase SelectorCertificado

Se establecen las rutas de los archivos de configuración y se crea un certificado de tipo *CertificadoBouncyCastle* a partir del certificado en base64. Como la clase *CertificadoBouncyCastle* contiene elementos comunes a todos los certificados de tipo x509 válidos, se puede convertir a este tipo aún sin saber a cuál de los 4 de tipos de certificados pertenece. Acto seguido se llama al método IdentificarCertificadoVerificación que será el encargado de indicarnos a que tipo pertenece. Una vez conocido el tipo se llama al constructor correspondiente para llevar a cabo el parseo de datos tal y como se ha explicado anteriormente, en el apartado correspondiente a las librerías de cada prestador.

Así pues el método ObtenerMedianteVerificacion devolverá un certificado de tipo CertificadoMPC el cual será una generalización del tipo concreto que ha sido necesario crear para realizar el parseo correspondiente.

Método IdentificarCertificadoVerificacion():

```
/// <summary>
/// Verifica que el certificado ha sido emitido por una de las CA de confianza y
devuelve su tipo
/// </summary>
/// <param name="pobjCertificado">Certificado</param>
/// <returns>ETipoEntidadCertificadora</returns>
private static ETipoEntidadCertificadora IdentificarCertificadoVerificacion(ref
CertificadoBouncyCastle pobjCertificado)
{
  ETipoEntidadCertificadora enuTipoCertificado = ETipoEntidadCertificadora.Desconocido;

  try
  {
    if ((pobjCertificado != null) && (pobjCertificado.CertificadoX509BC != null))
    {
      List<IdentificadorCertificadoRaiz> colCertificadosRaiz =
General.ListaCertificados;

      if (colCertificadosRaiz != null)
        {
          foreach (IdentificadorCertificadoRaiz objIdentificadorCertificadoRaiz in
colCertificadosRaiz)
          {
            if
(pobjCertificado.VerificarConfianza(objIdentificadorCertificadoRaiz.ClavePublica))
            {
              pobjCertificado.ClavePublicaEmisor =
objIdentificadorCertificadoRaiz.ClavePublica;
              enuTipoCertificado =
objIdentificadorCertificadoRaiz.TipoEntidadCertificadora;
            }
          }
        }
    }
  }
  catch (Exception)
  {
    throw;
  }

  return enuTipoCertificado;

}
```

Ilustración 104: Método IdentificarCertificadoVerificacion() de la clase SelectorCertificado

En este método se utiliza la lista de certificados raíces que ha sido previamente cargada en memoria. Mediante un bucle se van recorriendo todos los certificados raíces y se va invocando sucesivamente al método VerificarConfianza() del propio certificado de tipo CertificadoBouncyCastle pasándole la clave pública del emisor. Este método devolverá “true” cuando esa clave pública verifique ese certificado. En ese caso ya se puede determinar qué prestador ha emitido ese certificado.

Método VerificarConfianza() de la clase CertificadoBouncyCastle

```
/// <summary>
/// Comprueba la integridad del certificado mediante la clave pública de la CA emisora
/// </summary>
/// <param name="pobjClavePublicaCa">Clave pública certificado Raíz</param>
/// <returns>False si el certificado no es verificado por ninguna CA</returns>
public bool VerificarConfianza(AsymmetricKeyParameter pobjClavePublicaCa)
{
  bool bolResultado = true;

  try
  {
    if (mobjCertificado != null && pobjClavePublicaCa != null)
    {
      mobjCertificado.Verify(pobjClavePublicaCa);
    }
  }
  catch (InvalidKeyException)
  {
    bolResultado = false;
  }
  catch (CertificateException)
  {
    bolResultado = false;
  }
  catch (Exception)
  {
    bolResultado = false;
  }

  return bolResultado;

}
```

Ilustración 105: Método VerificarConfianza de la clase CertificadoBouncyCastle

Este método encapsula la funcionalidad de verificación de certificados x509 que proporciona la librería BouncyCastle a través del método Verify(). Este método recibe como parámetro una clave pública de tipo AsymetricKeyParameter.

Si volvemos al principio, al método ObtenerInformacion() de la clase MPCControlador, vemos que ya hemos explicado los métodos ValidarParametrosInicio() y ObtenerCertificado() junto con sus métodos internos. Para completar los métodos que forman parte de ObtenerInformacion() deberemos explicar también ValidarCertificado() y GenerarXmlSalida(). Se explican a continuación.

Método ValidarCertificado() de la clase MPCControlador:

```
/// <summary>
/// Valida el certificado
/// </summary>
private void ValidarCertificado()
{
  try
  {
    base.ValidarCertificado((EModoValidacion)mobjMPCVista.ModoValidacion);
  }
  catch (ValidarDatosException)
  {
    throw;
  }
  catch (Exception)
  {
    throw;
  }
}
```

Ilustración 106: Método ValidarCertificado() de la clase MPCControlador

Si se ha elegido validación se llamará al método ValidarCertificado() del controlador. Este método se limita a llamar al método homónimo del controlador base.

Método ValidarCertificado() de la clase BaseControlador:

```
/// <summary>
/// Valida el certificado
/// </summary>
/// <param name="penuModoValidacion">Modo de validación</param>
protected void ValidarCertificado(EModoValidacion penuModoValidacion)
{
  CertificadoMPC objCertificado;

  try
  {

    objCertificado = this.Certificado;
    if (objCertificado != null)
    {
      objCertificado.RealizarValidacionSimple();
    }

  }
  catch (CaducidadCertificadoException ex)
  {
    throw
ManejadorError.CrearExcepcionValidarDatos(ManejadorError.ECodigos.ErrorValidarCaducidad,
ex, ECodigoError.ErrorValidar, mobjBaseVista.TipoServicio);
  }
  catch (ValidezCertificadoException ex)
  {
    throw
ManejadorError.CrearExcepcionValidarDatos(ManejadorError.ECodigos.ErrorValidarFechaInici
oPeriodoValidez, ex, ECodigoError.ErrorValidar, mobjBaseVista.TipoServicio);
  }
  catch (EmisorConfianzaCertificadoException ex)
  {
    throw
ManejadorError.CrearExcepcionValidarDatos(ManejadorError.ECodigos.ErrorValidarEmisorConf
ianza, ex, ECodigoError.ErrorValidar, mobjBaseVista.TipoServicio);
  }
  catch (ValidezFirmaCertificadoException ex)
  {
    throw
ManejadorError.CrearExcepcionValidarDatos(ManejadorError.ECodigos.ErrorValidarFirma, ex,
ECodigoError.ErrorValidar, mobjBaseVista.TipoServicio);
  }
  catch (ValidezExtensionesCertificadoException ex)
```

```
  {
    throw
ManejadorError.CrearExcepcionValidarDatos(ManejadorError.ECodigos.ErrorValidarExtensione
s, ex, ECodigoError.ErrorValidar, mobjBaseVista.TipoServicio);
  }
  catch (Exception ex)
  {
    throw ManejadorError.CrearExcepcionValidarDatos(ManejadorError.ECodigos.Desconocido,
ex, ECodigoError.ErrorValidar, mobjBaseVista.TipoServicio);
  }
}
```

Ilustración 107: Método ValidarCertificado() de la clase BaseControlador

El Método ValidarCertificado() de la clase BaseControlador recibe como parámetro el modo de validación que se quiere. MPC solo puede realizar la validación simple, que consiste en comprobar la caducidad, integridad y confianza del certificado. Aún así, se ha escrito el código pensando en la posibilidad de querer ampliar la funcionalidad del proyecto en el futuro y hacer que se realicen validaciones más avanzadas (como la validación por OCSP o la validación de la cadena de certificación).

Este método también recibe el certificado que se quiere validar en formato clase CertificadoMPC. Dentro de esta clase general se encuentra el método RealizarValidacionSimple() el cual como se ha explicado anteriormente utiliza varios métodos para comprobar dicha validación, algunos de ellos de tipo abstracto que han sido implementados en la clase específica del certificado (CertDnie, CertFnmt, CertFnmtApe y CertFnmtAp).

Método GenerarXmlSalida() de la clase MPCControlador

```
/// <summary>
/// Genera el XML de salida
/// </summary>
/// <returns></returns>
private XmlDocument GenerarXmlSalida()
{
  XmlDocument objXmlSalida = new XmlDocument();

  if (!base.Excepcion.Existe())
  {
  objXmlSalida = base.GenerarXmlSalida((EModoValidacion)mobjMPCVista.ModoValidacion);
  }
  else
  {
    objXmlSalida = GenerarXmlError();
  }

  return objXmlSalida;

}
```

Ilustración 108: Método GenerarXmlSalida() de la clase MPCControlador

El método GenerarXmlSalida() de la clase MPCControlador comprueba si se ha generado alguna excepción en el proceso de parseo y validación del certificado, si ha

sido así se invoca al método GenerarXmlError, en caso de que todo sea correcto se invoca al método GenerarXmlSalida de la clase BaseControlador.

Método GenerarXmlSalida() de la clase BaseControlador:

```
/// <summary>
/// Genera el XML de salida
/// </summary>
/// <param name="penuModoValidacion">Modo de validación del certificado</param>
/// <returns>XmlDocument</returns>
protected XmlDocument GenerarXmlSalida(EModoValidacion penuModoValidacion)
{
  XmlSalida objXmlSalida;
  XmlDocument objXmlDocumento = new XmlDocument();

  try
  {
    objXmlSalida = new XmlSalida(this.Certificado, penuModoValidacion);
    objXmlDocumento = objXmlSalida.Generar();
  }
  catch (Exception ex)
  {
    if (objXmlDocumento == null)
    {
      objXmlDocumento = XmlSalida.CrearVacio();
    }
    LogHelper.Error(ex);
    throw;
  }

  return objXmlDocumento;

}
```

Ilustración 109: Método GenerarXmlSalida() de la clase BaseControlador

El método GenerarXmlSalida() de la clase BaseControlador crea un objeto de tipo XmlSalida, llamando a su constructor y pasándole el certificado de tipo CertificadoMPC que tiene almacenado en una variable el controlador así como el modo de validación elegido al invocar al servicio web. Es en la propia clase XmlSalida dónde se da formato al xml que devuelve como salida el sistema MPC mediante el método XmlSalida.Generar(). En caso de ocurrir un error en este paso se devolverá un documento xml vacío pero con sus encabezados mediante el método XmlSalida.CrearVacio().

Método XmlSalida.Generar()

```
/// <summary>
/// Genera un XML
/// </summary>
/// <returns>XmlDocument</returns>
public XmlDocument Generar()
{
  XmlDocument objXmlSalida = new XmlDocument();


  objXmlSalida = CrearXmlDocument(CrearCadenaXmlMPC());


  if ((objXmlSalida == null || String.IsNullOrEmpty(objXmlSalida.InnerXml)) &&
mobjExcepcion != null)
  {
    objXmlSalida = CrearXmlDocument(CrearCadenaXmlError());
  }

  if (objXmlSalida == null || String.IsNullOrEmpty(objXmlSalida.InnerXml))
  {
    objXmlSalida = CrearVacio();
  }

  return objXmlSalida;

}
```

Ilustración 110: Método Generar() de la clase XmlSalida

El método Generar() de la clase XmlSalida devuelve el xml de salida. Para crearlo, utiliza el método CrearXmlDocument el cual crea un fichero de tipo xml a partir de su representación en un string. Esto se lleva a cabo en el método CrearCadenaXmlMPC() que es realmente dónde se pasa toda la información recabada en el parseo y validación a formato xml siguiendo el esquema xsd defino en el apartado 5.1 de este documento. En caso de error se utiliza el método CrearCadenaXmlError().

Método CrearCadenaXmlMPC() de la clase XmlSalida

```
/// <summary>
/// Genera un XML de tipo cadena de texto a partir de un certificado X509 para el
Servicio Web del MPC
/// </summary>
/// <returns>XML de tipo cadena de texto</returns>
private string CrearCadenaXmlMPC()
{
  Encoding objEncoding;
  MemoryStream objMemoryStream = null;
  XmlTextWriter objXmlTextWriter = null;
  string strXml = String.Empty;

  try
  {
    if (mobjCertificado != null)
    {
      objMemoryStream = new MemoryStream();
      objEncoding = Encoding.GetEncoding(CCodificacion);

      objXmlTextWriter = new XmlTextWriter(objMemoryStream, objEncoding);
      objXmlTextWriter.Formatting = Formatting.Indented;
      objXmlTextWriter.Indentation = 3;
      objXmlTextWriter.WriteStartDocument();

      objXmlTextWriter.WriteStartElement(CXmlRaiz);

      objXmlTextWriter.WriteElementString(CXmlVersionRespuesta, CVersion);
      objXmlTextWriter.WriteElementString(CXmlHoraConsulta, ObtenerFechaHora());

      if (!mobjExcepcion.Existe() || mobjExcepcion.ObtenerCodigoGeneral() ==
ECodigoError.ErrorValidar)
      {
        objXmlTextWriter.WriteStartElement(CXmlDatosCertificado);

        objXmlTextWriter.WriteElementString(CXmlNumeroSerie,
mobjCertificado.NumeroSerie);
        objXmlTextWriter.WriteElementString(CXmlEmisor, mobjCertificado.Emisor);
        objXmlTextWriter.WriteElementString(CXmlAsunto, mobjCertificado.Asunto);
        objXmlTextWriter.WriteElementString(CXmlValidoDesde,
mobjCertificado.FechaDesde);
        objXmlTextWriter.WriteElementString(CXmlValidoHasta,
mobjCertificado.FechaHasta);
        objXmlTextWriter.WriteElementString(CXmlEntidad,
mobjCertificado.EntidadCertificadora);
        objXmlTextWriter.WriteElementString(CXmlUnidadEntidad,
mobjCertificado.UnidadEntidadCertificadora);

        if (!String.IsNullOrEmpty(mobjCertificado.DatosEntidadCertificadora))
        {
          objXmlTextWriter.WriteElementString(CXmlDatosEntidad,
mobjCertificado.DatosEntidadCertificadora);
        }

        objXmlTextWriter.WriteElementString(CXmlTipoEntidad,
mobjCertificado.TipoEntidadCertificadora.ToString("D"));
        objXmlTextWriter.WriteElementString(CXmlTipoPersona,
mobjCertificado.TipoPersona.ToString("D"));

        if (mobjCertificado is CertFnmt)
        {
          if (mobjCertificado.TipoPersona == ETipoPersona.Juridica)
          {
            objXmlTextWriter.WriteStartElement(CXmlDatosEntidadJuridica);
            objXmlTextWriter.WriteElementString(CXmlEntidadJuridica,
((CertFnmt)mobjCertificado).EntidadJuridica);
            objXmlTextWriter.WriteElementString(CXmlCif,
((CertFnmt)mobjCertificado).Cif);

            objXmlTextWriter.WriteEndElement(); // DatosEntidadJuridica
          }
        }
        if (mobjCertificado is CertDnie)
        {
          objXmlTextWriter.WriteElementString(CXmlProposito,
```

```
mobjCertificado.TipoCertificado.ToString("D"));
        }
        objXmlTextWriter.WriteElementString(CXmlNombreCompleto,
mobjCertificado.NombreCompleto);
        objXmlTextWriter.WriteElementString(CXmlPrimerApellido,
mobjCertificado.Apellido1);
        objXmlTextWriter.WriteElementString(CXmlSegundoApellido,
mobjCertificado.Apellido2);
        objXmlTextWriter.WriteElementString(CXmlNombre, mobjCertificado.Nombre);
        objXmlTextWriter.WriteElementString(CXmlNif, mobjCertificado.Nif);
        objXmlTextWriter.WriteElementString(CXmlPais, mobjCertificado.Pais);
        objXmlTextWriter.WriteElementString(CXmlEmail, mobjCertificado.Email);

        if (mobjCertificado is CertFnmtApe)
        {
          objXmlTextWriter.WriteElementString(CXmlEntidadSuscriptora,
((CertFnmtApe)mobjCertificado).EntidadSuscriptora);
          objXmlTextWriter.WriteElementString(CXmlNifEntidadSuscriptora,
((CertFnmtApe)mobjCertificado).NifEntidadSuscriptora);
          objXmlTextWriter.WriteElementString(CXmlNumIdentificacionPersonal,
((CertFnmtApe)mobjCertificado).NumIdentificacionPersonal);
          objXmlTextWriter.WriteElementString(CXmlUnidadOrganizativa,
((CertFnmtApe)mobjCertificado).UnidadOrganizativa);
          objXmlTextWriter.WriteElementString(CXmlCargo,
((CertFnmtApe)mobjCertificado).Cargo);
          objXmlTextWriter.WriteElementString(CXmlSituacionLaboral,
((CertFnmtApe)mobjCertificado).SituacionLaboral);
        }
        if (mobjCertificado is CertFnmtAp)
        {
          objXmlTextWriter.WriteElementString(CXmlEntidadSuscriptora,
((CertFnmtAp)mobjCertificado).EntidadSuscriptora);
          objXmlTextWriter.WriteElementString(CXmlNifEntidadSuscriptora,
((CertFnmtAp)mobjCertificado).NifEntidadSuscriptora);
          objXmlTextWriter.WriteElementString(CXmlNumIdentificacionPersonal,
((CertFnmtAp)mobjCertificado).NumIdentificacionPersonal);
          objXmlTextWriter.WriteElementString(CXmlUnidadOrganizativa,
((CertFnmtAp)mobjCertificado).UnidadOrganizativa);
          objXmlTextWriter.WriteElementString(CXmlCargo,
((CertFnmtAp)mobjCertificado).Cargo);
        }

        objXmlTextWriter.WriteEndElement(); // DatosCertificado
      }

      if (menuModoValidacion != EModoValidacion.Ninguno)
      {
        objXmlTextWriter.WriteStartElement(CXmlValidacionCertificado);

        objXmlTextWriter.WriteElementString(CXmlValido,
mobjCertificado.EstadoValido(menuModoValidacion).ToString("D"));
        objXmlTextWriter.WriteElementString(CXmlResultadoValidacion,
mobjCertificado.ObtenerResultadoValidacion());

        objXmlTextWriter.WriteStartElement(CXmlValidacionSimple);

        objXmlTextWriter.WriteElementString(CXmlCodigoValidacionSimple,
mobjCertificado.ValidacionSimple.ToString("D"));
        objXmlTextWriter.WriteElementString(CXmlDescripcionValidacionSimple,
mobjCertificado.ObtenerResultadoValidacion(mobjCertificado.ValidacionSimple));

        objXmlTextWriter.WriteEndElement(); // ValidacionSimple

        objXmlTextWriter.WriteEndElement(); // ValidacionCertificado
      }
      if (mobjExcepcion.Existe())
      {
        objXmlTextWriter.WriteStartElement(CXmlError);

        objXmlTextWriter.WriteElementString(CXmlOrigenError,
mobjExcepcion.ObtenerOrigen().ToString("D"));
        objXmlTextWriter.WriteElementString(CXmlCodigoError,
mobjExcepcion.ObtenerCodigoGeneral().ToString("D"));
        objXmlTextWriter.WriteElementString(CXmlDescripcionError,
mobjExcepcion.ObtenerDescripcion());
        if (Config.DetalleErrores == ETipoLogico.Si)
        {
```

```
          objXmlTextWriter.WriteElementString(CXmlDetalleError,
mobjExcepcion.ObtenerDetalle());
        }

        objXmlTextWriter.WriteEndElement(); // Error
      }

      objXmlTextWriter.WriteEndElement(); // Raiz

      objXmlTextWriter.Flush();

      using (StreamReader objStreamReader = new StreamReader(objMemoryStream,
objEncoding))
      {
        objMemoryStream.Seek(0, SeekOrigin.Begin);
        strXml = objStreamReader.ReadToEnd();
      }
    }
  }
  catch (Exception)
  {
    throw;
  }
  finally
  {
    if (objXmlTextWriter != null)
    {
      objXmlTextWriter.Close();
    }
    if (objMemoryStream != null)
    {
      objMemoryStream.Dispose();
    }
  }

  return strXml;
}
```

Ilustración 111: Método CrearCadenaXmlMPC() de la clase XmlSalida

En la clase XmlSalida están definidas constantes con los nombres de todos los nodos posibles que se pueden usar en el xml de salida. El método CrearCadenaXmlMPC() construye un xml dentro de un string utilizando estas constantes en función de los valores obtenidos en el certificado. No existe un método para cada tipo de certificado, por tanto hay diferentes bloques que añaden los campos específicos de parseo en función de cada tipo de certificado. En este punto resulta muy sencillo preguntar por el tipo de certificado que se está tratando y en función de eso devolver más o menos campos. El método también tiene en cuenta las excepciones, en caso de producirse se añade el nodo correspondiente para representarlas según está especificado en el Xsd.

Para terminar con la exposición de los métodos que forman parte de la llamada al método ObtenerInformacion() del servicio MPC, debemos comentar como se lleva a cabo la inserción de estadísticas. Como se ha comentado anteriormente, esto lo realiza el método InsertarEstadisticas() de la clase MPCControlador. A este método se le llama dentro de una cláusula Finally después de un bloque Try-Catch, esto quiere decir que tanto si se produce una excepción como si todo va correctamente se insertarán estadísticas en la base de datos. Esto es así porque se quiere tener un control de los errores producidos también en base de datos a parte de en el fichero log de la aplicación.

Método InsertarEstadisticas() de la clase MPCControlador:

```
/// <summary>
/// Inserta el registro de la petición en la BBDD
/// </summary>
/// <param name="ptmsTiempoTotal">Tiempo de respuesta total en la petición</param>
public new void InsertarEstadisticas(TimeSpan ptmsTiempoTotal)
{
  try
  {
    base.InsertarEstadisticas(ptmsTiempoTotal, (int)mobjMPCVista.ModoValidacion);
  }

  catch (SqlException ex)
  {
    throw new InsertarEstadisticasException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.ErrorInsertarEstadisticas),
ECodigoError.ErrorBaseDatos, mobjMPCVista.TipoServicio);
  }
  catch (Exception ex)
  {
    throw new InsertarEstadisticasException(ex.Message, ex,
ManejadorError.ObtenerCodigo(ManejadorError.ECodigos.ErrorInsertarEstadisticas),
ECodigoError.ErrorBaseDatos, mobjMPCVista.TipoServicio);
   }
}
```

Ilustración 112: Método InsertarEstadisticas() de la clase MPCControlador

El método InsertarEstadisticas() de la clase MPCControlador recibe como parámetro el tiempo que ha tardado el sistema MPC en parsear y validar el certificado. Esto se realiza mediante una variable de tipo TimeSpan que se comporta como un cronómetro. Se inicializa antes de hacer las llamadas y se para justo antes de insertar las estadísticas. El método llama a InsertarEstadisticas() de la clase BaseControlador pasándole además del tiempo total el modo de validación elegido por la aplicación.

Método InsertarEstadisticas() de la clase BaseControlador:

```
/// <summary>
/// Inserta el registro de la petición en la BBDD
/// </summary>
/// <param name="penuModoValidacion">Modo de validación</param>
/// <param name="ptmsTiempoTotal">Tiempo de respuesta total en la petición</param>
protected void InsertarEstadisticas(TimeSpan ptmsTiempoTotal, int pintModoValidacion)
{
  ConsultaData objNegConsulta;
  Consulta objConsulta;
  int? intCodError = null;
  int? intCodErrorGeneral = null;
  int intTiempoTotal = 0;

  try
  {
    if (ptmsTiempoTotal != null)
    {
      intTiempoTotal = Convert.ToInt32(ptmsTiempoTotal.TotalMilliseconds);
    }

    if (this.mobjExcepcion.Existe())
    {
      intCodErrorGeneral = (int)mobjExcepcion.ObtenerCodigoGeneral();
      intCodError = ManejadorError.ObtenerCodigoNumerico(mobjExcepcion.MPC.Codigo);
    }

    #region Insertar en BBDD

    objConsulta = new Consulta(intCodError,
                               intCodErrorGeneral,
                               (int) mobjBaseVista.TipoServicio,
                               pintModoValidacion,
                               mobjBaseVista.CodAplicacion,
                               DateTime.Now,
                               ObtenerEmisorCertificado(),
                               intTiempoTotal,
                               Environment.MachineName,

HttpContext.Current.Request.ServerVariables["REMOTE_ADDR"].ToString());

    // Se inserta el registro en la BBDD
    if (objConsulta != null)
    {
      objNegConsulta = new ConsultaData();
      objNegConsulta.Insert(objConsulta);
    }

    #endregion

  }
  catch (SqlException)
  {
    throw;
  }
  catch (Exception)
  {
    throw;
  }
}
```

Ilustración 113: Método InsertarEstadisticas() de la clase BaseControlador

En este punto es importante mencionar que puesto que nos encontramos dentro de un controlador estamos aún en la capa de negocio de la aplicación, pero en este método ya se utilizan objetos de la capa de acceso a datos ya que son necesarios para insertar un registro en la base de datos de la aplicación.

Lo primero que se hace en el método es obtener los milisegundos que ha tardado la aplicación en realizar la consulta. Después se comprueba si ha habido alguna

excepción, en caso de existir, se obtienen los códigos de error y de error general. Finalmente se crea un objeto de tipo Consulta, pasándole todos los datos que necesita el constructor. Los datos del objeto Consulta coinciden con los campos de la tabla Consultas de la base de datos, estos datos están explicados con más detalle en el diccionario de datos, en el apartado 4.5.6.1 Consultas.

Finalmente se crea un objeto de tipo ConsultaData, que servirá para invocar al método Insert() al que se le pasa como parámetro el objeto Consulta creado anteriormente. Este método es el encargado de insertar finalmente el registro en base de datos. Las clases Consulta y ConsultaData pertenecen a la librería MpcLogicaLib.

Método Insert() de la clase ConsultaData

```
/// <summary>
/// Inserta un registro en la tabla Consultas.
/// </summary>
public virtual void Insert(Consulta pobjConsulta)
{
  ValidationUtility.ValidateArgument("Consulta", pobjConsulta);

  SqlParameter[] colParameters = new SqlParameter[]
  {
    new SqlParameter("@CodError",
Base.BaseLogicaData.ObtenerValorCampo(pobjConsulta.CodError)),
    new SqlParameter("@CodErrorGeneral",
Base.BaseLogicaData.ObtenerValorCampo(pobjConsulta.CodErrorGeneral)),
    new SqlParameter("@CodTipoServicio", pobjConsulta.CodTipoServicio),
    new SqlParameter("@CodModoValidacion",
Base.BaseLogicaData.ObtenerValorCampo(pobjConsulta.CodModoValidacion)),
    new SqlParameter("@CodAplicacion", pobjConsulta.CodAplicacion),
    new SqlParameter("@Fecha", pobjConsulta.Fecha),
    new SqlParameter("@Emisor", pobjConsulta.Emisor),
    new SqlParameter("@TiempoRespuesta", pobjConsulta.TiempoRespuesta),
    new SqlParameter("@NombreMaquina", pobjConsulta.NombreMaquina),
    new SqlParameter("@Ip", pobjConsulta.Ip)
  };

  pobjConsulta.IdConsulta =
Convert.ToInt32(SqlClientUtility.ExecuteScalar(mstrConnectionStringName,
CommandType.StoredProcedure, "spConsultasInsert", colParameters));
}
```

Ilustración 114: Método Insert() de la clase ConsultaData

Este método se encuentra dentro de la Clase ConsultaData, la cual pertenece a la librería MpcLogicaLib, que como se ha explicado anteriormente se utiliza para insertar en la base de datos de la aplicación.

6 Aplicación resultante

6.1 Contenido compilado del sistema MPC

El sistema MPC, como se ha comentado a lo largo del documento es un servicio Web creado con la tecnología .NET Framework codificado en el lenguaje c#. Una vez compilado el código fuente los ficheros y carpetas que conforman la aplicación tienen el siguiente aspecto:

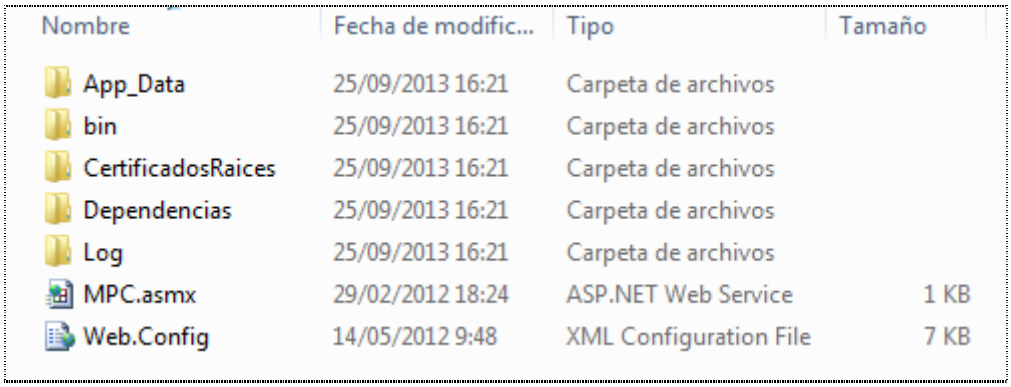


Ilustración 115: Archivos compilados de MPC

Las carpetas y archivos que se muestran en la imagen superior son las que deberán estar en el servidor web que aloje el sistema MPC. Deberá crearse un directorio virtual en el servidor IIS, el cual tendrá el contenido compilado del sistema.

A continuación se va explicar brevemente el contenido compilado del sistema MPC y la razón de ser de cada elemento.

Carpeta App_Data:

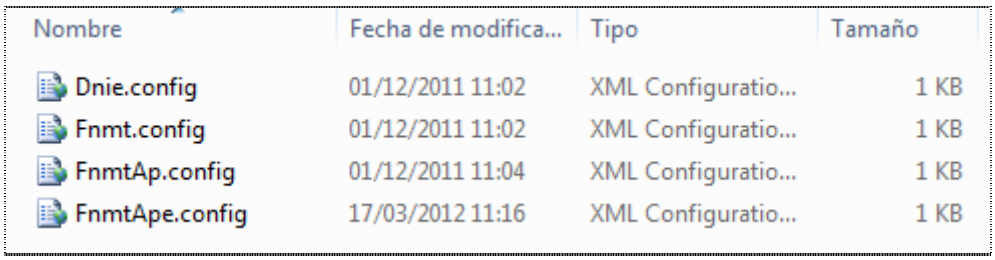


Ilustración 116: Contenido de la carpeta App_Data del sistema MPC

La carpeta App_Data contiene los archivos de configuración de las librerías DLL de cada prestador de certificados soportado. En estos archivos, como se ha explicado anteriormente, se puede configurar los valores EmisorDiscriminador (discriminador para conocer el prestador dentro del campo emisor {O/OU}) y EmisorDiscriminadorValor (valor del discriminador usado para conocer el prestador dentro del emisor {O/OU}). Estos valores son necesarios para que el sistema pueda saber qué tipo de certificado está tratando en caso de que este activada la selección de certificados mediante parseo. El contenido de los 4 ficheros de configuración es el siguiente:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <appSettings>
                <!-- Discriminador de la entidad dentro del emisor -->
                <add key="EmisorDiscriminador" value="OU"/>
                <add key="EmisorDiscriminadorValor" value="DNIE"/>
    <add key="RutaDirectorioCertificadoRaiz" value="CertificadosRaices\Dnie"/>
        </appSettings>
</configuration>
```

Ilustración 117: Fichero de configuración Dnie.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <appSettings>
                <!-- Discriminador de la entidad dentro del emisor -->
                <add key="EmisorDiscriminador" value="OU"/>
                <add key="EmisorDiscriminadorValor" value="FNMT Clase 2 CA"/>
    <add key="RutaDirectorioCertificadoRaiz" value="CertificadosRaices\Fnmt"/>
        </appSettings>
</configuration>
```

Ilustración 118: Fichero de configuración Fnmt.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <appSettings>
                <!-- Discriminador de la entidad dentro del emisor -->
                <add key="EmisorDiscriminador" value="OU"/>
                <add key="EmisorDiscriminadorValor" value="CERES"/>
    <add key="RutaDirectorioCertificadoRaiz" value="CertificadosRaices\FnmtAp"/>
        </appSettings>
</configuration>
```

Ilustración 119: Fichero de configuración FnmtAp.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <appSettings>
                <!-- Discriminador de la entidad dentro del emisor -->
                <add key="EmisorDiscriminador" value="OU"/>
                <add key="EmisorDiscriminadorValor" value="AC APE"/>
    <add key="RutaDirectorioCertificadoRaiz" value="CertificadosRaices\FnmtApe"/>
        </appSettings>
</configuration>
```

Ilustración 120: Fichero de configuración FnmtApe.config

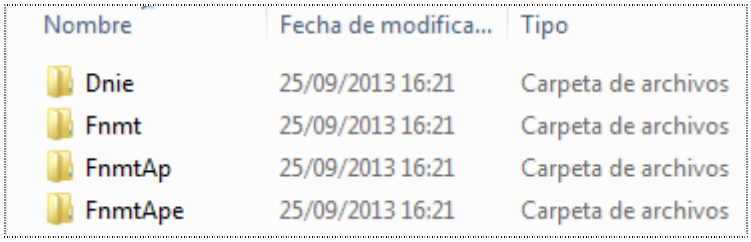
Carpeta bin:

Nombre	Fecha de modifica...	Tipo	Tamaño
App_Code.compiled	14/05/2012 9:48	Archivo COMPILED	1 KB
App_Code.dll	14/05/2012 9:48	Extensión de la aplicación	40 KB
BouncyCastle.Crypto.dll	07/04/2011 2:32	Extensión de la aplicación	1.468 KB
DnieCertLib.dll	14/05/2012 9:48	Extensión de la aplicación	10 KB
FnmtApCertLib.dll	14/05/2012 9:48	Extensión de la aplicación	13 KB
FnmtApeCertLib.dll	14/05/2012 9:48	Extensión de la aplicación	14 KB
FnmtCertLib.dll	14/05/2012 9:48	Extensión de la aplicación	13 KB
Microsoft.Practices.EnterpriseLibrary.Common.dll	11/05/2007 22:09	Extensión de la aplicación	158 KB
Microsoft.Practices.EnterpriseLibrary.Common.xml	22/05/2007 10:50	Documento XML	425 KB
Microsoft.Practices.EnterpriseLibrary.Logging.dll	11/05/2007 22:09	Extensión de la aplicación	214 KB
Microsoft.Practices.EnterpriseLibrary.Logging.xml	22/05/2007 10:51	Documento XML	469 KB
Microsoft.Practices.ObjectBuilder.dll	22/05/2007 10:47	Extensión de la aplicación	63 KB
MpcCertificadosLib.dll	14/05/2012 9:48	Extensión de la aplicación	13 KB
MpcComunLib.dll	14/05/2012 9:48	Extensión de la aplicación	54 KB

Ilustración 121: Contenido de la carpeta bin del sistema MPC

La carpeta bin es generada al compilar la aplicación, en ella se encuentran todas las librerías que utiliza el sistema MPC. Vemos como están las 4 librerías correspondientes a los cuatro prestadores soportados por la aplicación (*DnieCertLib.dll*, *FnmtCertLib.dll*, *FnmtApeCertLib.dll* y *FnmtApCert*). También están la librería *MpcCertificadosLib.dll* con la funcionalidad común referente a certificados y la librería *MpcComunLib.dll* con la funcionalidad común que puede ser utilizada en otras aplicaciones. Las clases del propio servicio web se agrupan compiladas en la librería *App_Code.dll*. Finalmente tenemos las librerías de terceros utilizadas por el sistema: la librería de código abierto *BouncyCastle.Crypto.dll* para el manejo de certificados y las librerías de *Microsoft Enterprise Library* para la capa de datos y el log de la aplicación.

Carpeta CertificadosRaices:



Nombre	Fecha de modifica...	Tipo
Dnie	25/09/2013 16:21	Carpeta de archivos
Fnmt	25/09/2013 16:21	Carpeta de archivos
FnmtAp	25/09/2013 16:21	Carpeta de archivos
FnmtApe	25/09/2013 16:21	Carpeta de archivos

Ilustración 122: Contenido de la carpeta CertificadosRaices del sistema MPC

En la carpeta CertificadosRaices hay una subcarpeta por cada prestador soportado por el sistema. Dentro la carpeta correspondiente a un prestador, se encuentran los certificados raíces e intermedios que componen la cadena de certificación de los certificados de usuarios soportados. Obviamente solo es tiene la parte pública de estos certificados, ya que para lo que se utilizarán es para verificar mediante su clave pública la firma de sus certificados emitidos, es decir, de los certificados de usuario que recibirá como parámetro el sistema. Estos certificados son públicos y están colgados en la página web de cada prestador. Sirva como muestra el contenido de la carpeta FnmtApe:

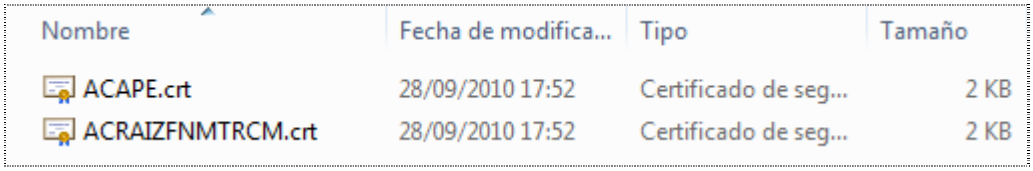


Ilustración 123: Contenido de la carpeta CertificadosRaices\FnmtApe del sistema MPC

En este caso la cadena de certificación está formada por 2 certificados, el certificado intermedio ACAPE.crt y el certificado raíz ACRAIZFNMTRCM.crt.

Carpeta Dependencias:

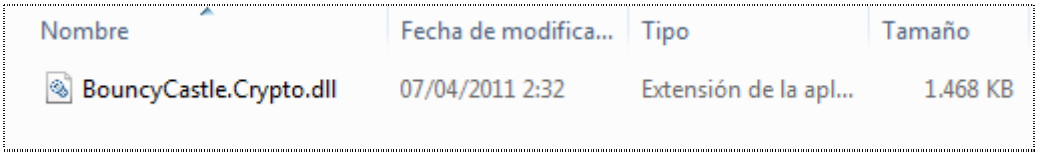


Ilustración 124: Contenido de la carpeta Dependencias del sistema MPC

La carpeta dependencias responde únicamente a una cuestión de diseño. Puesto que el sistema utiliza la librería de terceros BouncyCastleCrypto.dll, y esta librería debe ser añadida al proyecto como referencia, debe estar almacenada en algún sitio fuera de la carpeta bin, para que al compilar la añada a este. Se ha optado por crear la carpeta dependencias para añadir esta y futuras DLLs de terceros.

Carpeta Log:

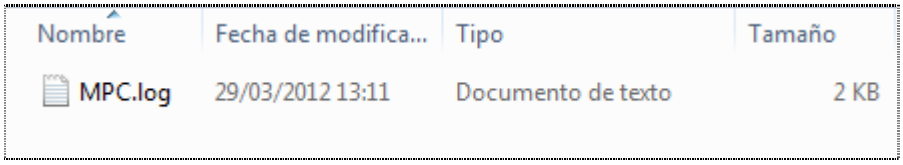


Ilustración 125: Contenido de la carpeta Log del sistema MPC

En la carpeta Log es dónde el sistema almacenará el archivo o archivos de Logs del sistema cuando ocurra una excepción durante la ejecución. La primera vez estará vacía y se creará cuando ocurra la primera excepción. Las siguientes excepciones se añadirán al mismo archivo aunque puede configurarse para que cree un fichero nuevo según la fecha o si se ha alcanzado cierto tamaño. El aspecto que tiene el archivo MCP.log es el siguiente:

```
----------------------------------------

Fecha: 22/03/2012 12:28:33

Mensaje: Certificado no soportado por el sistema.

Se ha producido un error de certificado no soportado por el sistema.

Categoría: General

Severidad: Error

Maquina: PC10287

----------------------------------------

Fecha: 22/03/2012 12:30:24

Mensaje: Error general de MPC.

Controlador no válido.

Categoría: General

Severidad: Error

Maquina: PC10287

----------------------------------------
```

Ilustración 126: Ejemplo de contenido del fichero log del sistema MPC

Fichero Web.config:

El fichero Web.config es el fichero de configuración de la aplicación. Aquí se pueden configurar los valores de ciertos parámetros para modificar el compartamiento del sistema MPC.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="loggingConfiguration"
type="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.LoggingSettings,
Microsoft.Practices.EnterpriseLibrary.Logging, Version=3.1.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" />
    <section name="dataConfiguration"
type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
Microsoft.Practices.EnterpriseLibrary.Data, Version=3.1.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" />
  </configSections>
  <loggingConfiguration name="Logging Application Block" tracingEnabled="true"
    defaultCategory="General" logWarningsWhenNoCategoriesMatch="true">
    <listeners>
      <add fileName="Log\MPC.log" header="----------------------------------------"
        footer="----------------------------------------" formatter="MPC"

listenerDataType="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.FlatFileTr
aceListenerData, Microsoft.Practices.EnterpriseLibrary.Logging, Version=3.1.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
        traceOutputOptions="None"
type="Microsoft.Practices.EnterpriseLibrary.Logging.TraceListeners.FlatFileTraceListener
, Microsoft.Practices.EnterpriseLibrary.Logging, Version=3.1.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
        name="FlatFile TraceListener" />
      <add source="Enterprise Library Logging" formatter="Text Formatter"
        log="Application" machineName=""
listenerDataType="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.FormattedE
ventLogTraceListenerData, Microsoft.Practices.EnterpriseLibrary.Logging,
Version=3.1.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
        traceOutputOptions="None"
type="Microsoft.Practices.EnterpriseLibrary.Logging.TraceListeners.FormattedEventLogTrac
eListener, Microsoft.Practices.EnterpriseLibrary.Logging, Version=3.1.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
        name="Formatted EventLog TraceListener" />
    </listeners>
    <formatters>
      <add template="Fecha: {timestamp}&#xA;Mensaje: {message}&#xA;Categoria:
{category}&#xA;Severidad: {severity}&#xA;Maquina: {machine}"
        type="Microsoft.Practices.EnterpriseLibrary.Logging.Formatters.TextFormatter,
Microsoft.Practices.EnterpriseLibrary.Logging, Version=3.1.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
        name="MPC" />
      <add template="Timestamp: {timestamp}&#xA;Message: {message}&#xA;Category:
{category}&#xA;Priority: {priority}&#xA;EventId: {eventid}&#xA;Severity:
{severity}&#xA;Title:{title}&#xA;Machine: {machine}&#xA;Application Domain:
{appDomain}&#xA;Process Id: {processId}&#xA;Process Name: {processName}&#xA;Win32 Thread
Id: {win32ThreadId}&#xA;Thread Name: {threadName}&#xA;Extended Properties:
{dictionary({key} - {value}&#xA;)}"
        type="Microsoft.Practices.EnterpriseLibrary.Logging.Formatters.TextFormatter,
Microsoft.Practices.EnterpriseLibrary.Logging, Version=3.1.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
        name="Text Formatter" />
    </formatters>
    <logFilters>
      <add categoryFilterMode="DenyAllExceptAllowed"
type="Microsoft.Practices.EnterpriseLibrary.Logging.Filters.CategoryFilter,
Microsoft.Practices.EnterpriseLibrary.Logging, Version=3.1.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
        name="Category Filter">
        <categoryFilters>
          <add name="General" />
          <add name="Observaciones" />
        </categoryFilters>
      </add>
    </logFilters>
```

```
    <categorySources>
      <add switchValue="Verbose" name="Debug">
        <listeners>
          <add name="FlatFile TraceListener" />
        </listeners>
      </add>
      <add switchValue="All" name="General">
        <listeners>
          <add name="FlatFile TraceListener" />
        </listeners>
      </add>
      <add switchValue="Information" name="Observaciones">
        <listeners>
          <add name="FlatFile TraceListener" />
        </listeners>
      </add>
    </categorySources>
    <specialSources>
      <allEvents switchValue="All" name="All Events" />
      <notProcessed switchValue="All" name="Unprocessed Category" />
      <errors switchValue="All" name="Logging Errors &amp;amp; Warnings">
        <listeners>
          <add name="FlatFile TraceListener" />
        </listeners>
      </errors>
    </specialSources>
  </loggingConfiguration>
  <dataConfiguration defaultDatabase="LocalSqlServer" />
  <connectionStrings>
    <add name="MPC" connectionString="Server=TEC-MHERNANDEZ; Database=MPC; User
ID=MPC_USER; Password=MPC_USER" providerName="System.Data.SqlClient"/>
   </connectionStrings>
  <appSettings>
    <add key="CertificadosRaices" value="CertificadosRaices" />
    <add key="DetalleErrores" value="1" />
    <add key="Estadisticas" value="1" />
    <add key="SelectorCertificadoVerificacion" value="0" />
  </appSettings>
  <system.web>
    <compilation debug="true" strict="true" explicit="true" defaultLanguage="c#"
targetFramework="4.0" />
    <authentication mode="Windows" />
    <globalization requestEncoding="ISO-8859-15" responseEncoding="ISO-8859-15" />
    <webServices>
      <protocols>
        <add name="HttpPost" />
        <add name="HttpGet" />
        <add name="HttpSoap" />
      </protocols>
    </webServices>
  </system.web>
</configuration>
```

Ilustración 127: Fichero Web.config del sistema MPC

Modificando el contenido de la sección <loggingConfiguration> se puede configurar el formato del archivo log del sistema, añadirle cabeceras y pie a cada entrada, configurar el formato de la hora, rotado etc.

En la sección <conectionStrings> se encuentra la cadena de conexión con la base de datos. Aquí es dónde se especifica el servidor, nombre de base de datos, usuario, contraseña y tipo de proveedor. De esta forma, cuando la aplicación cambie de entorno (por ejemplo, del entorno de desarrollo al entorno de producción) solo se tiene que modificar esta entrada para que la aplicación inserte estadísticas en otro servidor de base de datos diferente.

Dentro de la sección <appSettings> se encuentra diferentes claves que pueden configurarse:

- CertificadosRaices: En esta clave se debe indicar el nombre de la carpeta dónde se encuentran almacenados los certificados raíces de las distintas entidades de certificación soportadas por el sistema MPC.
- DetalleErrores: Esta clave indica si se quiere o no un mayor nivel de detalle (campo DetalleError) a la hora de mostrar los errores en el xml devuelto por el sistema MPC.
- Estadísticas: Esta clave indica si se quiere o no insertar estadísticas en la base de datos para cada llama a los métodos ObtenerInformacion() y EsValido() del sistema MPC.
- SelectorCertificadoVerificacion: En esta clave se activa la selección de certificado mediante verificación, en caso de tener un 0 se elegirá el método de selección de certificados mediante parseo.

Fichero MPC.asmx:

El fichero MPC.asmx es la página de entrada al servicio Web MPC desde un explorador web. Los servicios Web no tienen interfaz como tal, pero al crearlos con .NET, se puede obtener una página de ayuda autogenerada, la cual muestra una lista de los métodos que forman el servicio web y un enlace al documento de descripción del servicio web (WSDL)

6.2 Visualización del sistema MPC desde un navegador web

Al acceder al sistema MPC desde un navegador web lo que se muestra es la página de ayuda MPC.asmx, explicada en el apartado anterior. Esta página tiene el siguiente aspecto:

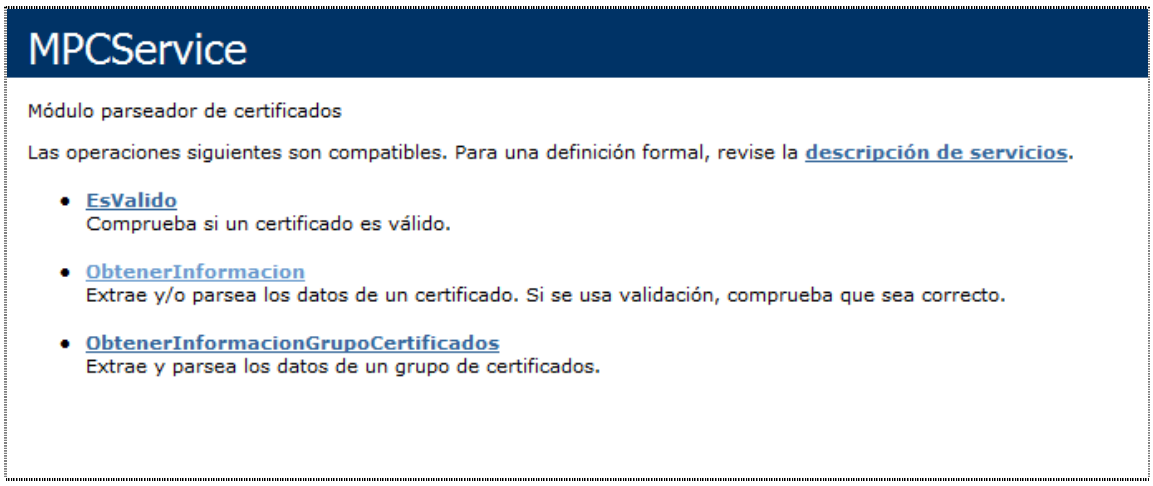


Ilustración 128: Página MPC.asmx del sistema MPC

Haciendo “click” en la descripción de servicios se nos muestra el fichero WSDL, que será el que leerán las aplicaciones clientes para conocer qué métodos están disponibles y qué parámetros reciben:

```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://intranet.es/webservices/MPC"
xmlns:s1="http://intranet.es/webservices/MPC/AbstractTypes"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://intranet.es/webservices/MPC">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Módulo parseador de
certificados</wsdl:documentation>
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://intranet.es/webservices/MPC">
      <s:element name="ObtenerInformacion">
        <s:complextype>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="pstrCertificado"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="pintModoValidacion"
type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="pintCodigoAplicacion"
type="s:int" />
          </s:sequence>
        </s:complextype>
      </s:element>
      <s:element name="ObtenerInformacionResponse">
        <s:complextype>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="ObtenerInformacionResult">
              <s:complextype mixed="true">
```

```
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complextype>
            </s:element>
          </s:sequence>
        </s:complextype>
      </s:element>
      <s:element name="EsValido">
        <s:complextype>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="pstrCertificado"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="pintModoValidacion"
type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="pintCodigoAplicacion"
type="s:int" />
          </s:sequence>
        </s:complextype>
      </s:element>
      <s:element name="EsValidoResponse">
        <s:complextype>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="EsValidoResult" type="s:int"
/>
          </s:sequence>
        </s:complextype>
      </s:element>
      <s:element name="ObtenerInformacionGrupoCertificados">
        <s:complextype>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="pstrCertificados"
type="tns:ArrayOfString" />
            <s:element minOccurs="1" maxOccurs="1" name="pintCodigoAplicacion"
type="s:int" />
          </s:sequence>
        </s:complextype>
      </s:element>
      <s:complextype name="ArrayOfString">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded" name="string" nillable="true"
type="s:string" />
        </s:sequence>
      </s:complextype>
      <s:element name="ObtenerInformacionGrupoCertificadosResponse">
        <s:complextype>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
name="ObtenerInformacionGrupoCertificadosResult">
              <s:complextype mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complextype>
            </s:element>
          </s:sequence>
        </s:complextype>
      </s:element>
      <s:element name="int" type="s:int" />
    </s:schema>
    <s:schema targetNamespace="http://intranet.es/webservices/MPC/AbstractTypes">
      <s:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <s:complextype name="StringArray">
        <s:complexcontent mixed="false">
          <s:restriction base="soapenc:Array">
            <s:sequence>
              <s:element minOccurs="0" maxOccurs="unbounded" name="String"
type="s:string" />
            </s:sequence>
          </s:restriction>
        </s:complexcontent>
      </s:complextype>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="ObtenerInformacionSoapIn">
    <wsdl:part name="parameters" element="tns:ObtenerInformacion" />
  </wsdl:message>
```

```
  <wsdl:message name="ObtenerInformacionSoapOut">
    <wsdl:part name="parameters" element="tns:ObtenerInformacionResponse" />
  </wsdl:message>
  <wsdl:message name="EsValidoSoapIn">
    <wsdl:part name="parameters" element="tns:EsValido" />
  </wsdl:message>
  <wsdl:message name="EsValidoSoapOut">
    <wsdl:part name="parameters" element="tns:EsValidoResponse" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionGrupoCertificadosSoapIn">
    <wsdl:part name="parameters" element="tns:ObtenerInformacionGrupoCertificados" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionGrupoCertificadosSoapOut">
    <wsdl:part name="parameters"
element="tns:ObtenerInformacionGrupoCertificadosResponse" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionHttpGetIn">
    <wsdl:part name="pstrCertificado" type="s:string" />
    <wsdl:part name="pintModoValidacion" type="s:string" />
    <wsdl:part name="pintCodigoAplicacion" type="s:string" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionHttpGetOut">
    <wsdl:part name="Body" />
  </wsdl:message>
  <wsdl:message name="EsValidoHttpGetIn">
    <wsdl:part name="pstrCertificado" type="s:string" />
    <wsdl:part name="pintModoValidacion" type="s:string" />
    <wsdl:part name="pintCodigoAplicacion" type="s:string" />
  </wsdl:message>
  <wsdl:message name="EsValidoHttpGetOut">
    <wsdl:part name="Body" element="tns:int" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionGrupoCertificadosHttpGetIn">
    <wsdl:part name="pstrCertificados" type="s1:StringArray" />
    <wsdl:part name="pintCodigoAplicacion" type="s:string" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionGrupoCertificadosHttpGetOut">
    <wsdl:part name="Body" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionHttpPostIn">
    <wsdl:part name="pstrCertificado" type="s:string" />
    <wsdl:part name="pintModoValidacion" type="s:string" />
    <wsdl:part name="pintCodigoAplicacion" type="s:string" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionHttpPostOut">
    <wsdl:part name="Body" />
  </wsdl:message>
  <wsdl:message name="EsValidoHttpPostIn">
    <wsdl:part name="pstrCertificado" type="s:string" />
    <wsdl:part name="pintModoValidacion" type="s:string" />
    <wsdl:part name="pintCodigoAplicacion" type="s:string" />
  </wsdl:message>
  <wsdl:message name="EsValidoHttpPostOut">
    <wsdl:part name="Body" element="tns:int" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionGrupoCertificadosHttpPostIn">
    <wsdl:part name="pstrCertificados" type="s1:StringArray" />
    <wsdl:part name="pintCodigoAplicacion" type="s:string" />
  </wsdl:message>
  <wsdl:message name="ObtenerInformacionGrupoCertificadosHttpPostOut">
    <wsdl:part name="Body" />
  </wsdl:message>
  <wsdl:porttype name="MPCServiceSoap">
    <wsdl:operation name="ObtenerInformacion">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Extrae y/o
parsea los datos de un certificado. Si se usa validación, comprueba que sea
correcto.</wsdl:documentation>
      <wsdl:input message="tns:ObtenerInformacionSoapIn" />
      <wsdl:output message="tns:ObtenerInformacionSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="EsValido">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Comprueba si un
certificado es válido.</wsdl:documentation>
      <wsdl:input message="tns:EsValidoSoapIn" />
      <wsdl:output message="tns:EsValidoSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="ObtenerInformacionGrupoCertificados">
```

```
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Extrae y parsea
los datos de un grupo de certificados.</wsdl:documentation>
      <wsdl:input message="tns:ObtenerInformacionGrupoCertificadosSoapIn" />
      <wsdl:output message="tns:ObtenerInformacionGrupoCertificadosSoapOut" />
    </wsdl:operation>
  </wsdl:porttype>
  <wsdl:porttype name="MPCServiceHttpGet">
    <wsdl:operation name="ObtenerInformacion">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Extrae y/o
parsea los datos de un certificado. Si se usa validación, comprueba que sea
correcto.</wsdl:documentation>
      <wsdl:input message="tns:ObtenerInformacionHttpGetIn" />
      <wsdl:output message="tns:ObtenerInformacionHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="EsValido">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Comprueba si un
certificado es válido.</wsdl:documentation>
      <wsdl:input message="tns:EsValidoHttpGetIn" />
      <wsdl:output message="tns:EsValidoHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="ObtenerInformacionGrupoCertificados">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Extrae y parsea
los datos de un grupo de certificados.</wsdl:documentation>
      <wsdl:input message="tns:ObtenerInformacionGrupoCertificadosHttpGetIn" />
      <wsdl:output message="tns:ObtenerInformacionGrupoCertificadosHttpGetOut" />
    </wsdl:operation>
  </wsdl:porttype>
  <wsdl:porttype name="MPCServiceHttpPost">
    <wsdl:operation name="ObtenerInformacion">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Extrae y/o
parsea los datos de un certificado. Si se usa validación, comprueba que sea
correcto.</wsdl:documentation>
      <wsdl:input message="tns:ObtenerInformacionHttpPostIn" />
      <wsdl:output message="tns:ObtenerInformacionHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="EsValido">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Comprueba si un
certificado es válido.</wsdl:documentation>
      <wsdl:input message="tns:EsValidoHttpPostIn" />
      <wsdl:output message="tns:EsValidoHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="ObtenerInformacionGrupoCertificados">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Extrae y parsea
los datos de un grupo de certificados.</wsdl:documentation>
      <wsdl:input message="tns:ObtenerInformacionGrupoCertificadosHttpPostIn" />
      <wsdl:output message="tns:ObtenerInformacionGrupoCertificadosHttpPostOut" />
    </wsdl:operation>
  </wsdl:porttype>
  <wsdl:binding name="MPCServiceSoap" type="tns:MPCServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="ObtenerInformacion">
      <soap:operation
soapAction="http://intranet.es/webservices/MPC/ObtenerInformacion" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EsValido">
      <soap:operation soapAction="http://intranet.es/webservices/MPC/EsValido"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ObtenerInformacionGrupoCertificados">
      <soap:operation
soapAction="http://intranet.es/webservices/MPC/ObtenerInformacionGrupoCertificados"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
```

```
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="MPCServiceSoap12" type="tns:MPCServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="ObtenerInformacion">
      <soap12:operation
soapAction="http://intranet.es/webservices/MPC/ObtenerInformacion" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EsValido">
      <soap12:operation soapAction="http://intranet.es/webservices/MPC/EsValido"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ObtenerInformacionGrupoCertificados">
      <soap12:operation
soapAction="http://intranet.es/webservices/MPC/ObtenerInformacionGrupoCertificados"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="MPCServiceHttpGet" type="tns:MPCServiceHttpGet">
    <http:binding verb="GET" />
    <wsdl:operation name="ObtenerInformacion">
      <http:operation location="/ObtenerInformacion" />
      <wsdl:input>
        <http:urlencoded />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EsValido">
      <http:operation location="/EsValido" />
      <wsdl:input>
        <http:urlencoded />
      </wsdl:input>
      <wsdl:output>
        <mime:mimexml part="Body" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ObtenerInformacionGrupoCertificados">
      <http:operation location="/ObtenerInformacionGrupoCertificados" />
      <wsdl:input>
        <http:urlencoded />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="MPCServiceHttpPost" type="tns:MPCServiceHttpPost">
    <http:binding verb="POST" />
    <wsdl:operation name="ObtenerInformacion">
      <http:operation location="/ObtenerInformacion" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
```

```
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EsValido">
      <http:operation location="/EsValido" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:mimexml part="Body" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ObtenerInformacionGrupoCertificados">
      <http:operation location="/ObtenerInformacionGrupoCertificados" />
      <wsdl:input>
        <mime:content type="application/x-www-form-urlencoded" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="Body" type="text/xml" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="MPCService">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Módulo parseador
de certificados</wsdl:documentation>
    <wsdl:port name="MPCServiceSoap" binding="tns:MPCServiceSoap">
      <soap:address location="http://localhost/PFC/MPCService.asmx" />
    </wsdl:port>
    <wsdl:port name="MPCServiceSoap12" binding="tns:MPCServiceSoap12">
      <soap12:address location="http://localhost/PFC/MPCService.asmx" />
    </wsdl:port>
    <wsdl:port name="MPCServiceHttpGet" binding="tns:MPCServiceHttpGet">
      <http:address location="http://localhost/PFC/MPCService.asmx" />
    </wsdl:port>
    <wsdl:port name="MPCServiceHttpPost" binding="tns:MPCServiceHttpPost">
      <http:address location="http://localhost/PFC/MPCService.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Ilustración 129: Contenido del fichero WSDL del sistema MPC

Si volvemos a la página anterior y se hace click en el nombre de cada método, nos lleva a otra página dónde se nos amplía la información sobre el mismo y se nos permite invocarlo, siempre y cuando el método reciba parámetros con tipos primitivos:

Método EsValido()

MPCService

Haga clic aquí para obtener una lista completa de operaciones.

EsValido

Comprueba si un certificado es válido.

Prueba

Haga clic en el botón 'Invocar', para probar la operación utilizando el protocolo HTTP POST.

Parámetro	Valor
pstrCertificado:	
pintModoValidacion:	
pintCodigoAplicacion:	

Invocar

SOAP 1.1

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.1. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://intranet.es/webservices/MPC/EsValido"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <EsValido xmlns="http://intranet.es/webservices/MPC">
      <pstrCertificado>string</pstrCertificado>
      <pintModoValidacion>int</pintModoValidacion>
      <pintCodigoAplicacion>int</pintCodigoAplicacion>
    </EsValido>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <EsValidoResponse xmlns="http://intranet.es/webservices/MPC">
      <EsValidoResult>int</EsValidoResult>
    </EsValidoResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.2. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <EsValido xmlns="http://intranet.es/webservices/MPC">
      <pstrCertificado>string</pstrCertificado>
      <pintModoValidacion>int</pintModoValidacion>
      <pintCodigoAplicacion>int</pintCodigoAplicacion>
    </EsValido>
  </soap12:Body>
</soap12:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <EsValidoResponse xmlns="http://intranet.es/webservices/MPC">
      <EsValidoResult>int</EsValidoResult>
    </EsValidoResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP GET. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
GET /PFC/MPCService.asmx/EsValido?pstrCertificado=string&pintModoValidacion=string&pintCodigoAplicacion=string HTTP/1.1
```

```
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <EsValidoResponse xmlns="http://intranet.es/webservices/MPC">
      <EsValidoResult>int</EsValidoResult>
    </EsValidoResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP GET. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
GET /PFC/MPCService.asmx/EsValido?pstrCertificado=string&pintModoValidacion=string&pintCodigoAplicacion=string HTTP/1.1
Host: localhost
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://intranet.es/webservices/MPC">int</int>
```

HTTP POST

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP POST. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx/EsValido HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

pstrCertificado=string&pintModoValidacion=string&pintCodigoAplicacion=string
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://intranet.es/webservices/MPC">int</int>
```

Ilustración 130: Página de prueba para el método EsValido()

Método ObtenerInformacion()

MPCService

Haga clic aquí para obtener una lista completa de operaciones.

ObtenerInformacion

Extrae y/o parsea los datos de un certificado. Si se usa validación, comprueba que sea correcto.

Prueba

Haga clic en el botón 'Invocar', para probar la operación utilizando el protocolo HTTP POST.

Parámetro	Valor
pstrCertificado:	
pintModoValidacion:	
pintCodigoAplicacion:	

Invocar

SOAP 1.1

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.1. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://intranet.es/webservices/MPC/ObtenerInformacion"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ObtenerInformacion xmlns="http://intranet.es/webservices/MPC">
      <pstrCertificado>string</pstrCertificado>
      <pintModoValidacion>int</pintModoValidacion>
      <pintCodigoAplicacion>int</pintCodigoAplicacion>
    </ObtenerInformacion>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ObtenerInformacionResponse xmlns="http://intranet.es/webservices/MPC">
      <ObtenerInformacionResult>xml</ObtenerInformacionResult>
    </ObtenerInformacionResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.2. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ObtenerInformacion xmlns="http://intranet.es/webservices/MPC">
      <pstrCertificado>string</pstrCertificado>
      <pintModoValidacion>int</pintModoValidacion>
      <pintCodigoAplicacion>int</pintCodigoAplicacion>
    </ObtenerInformacion>
  </soap12:Body>
</soap12:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ObtenerInformacionResponse xmlns="http://intranet.es/webservices/MPC">
      <ObtenerInformacionResult>xml</ObtenerInformacionResult>
    </ObtenerInformacionResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP GET. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
GET /PFC/MPCService.asmx/ObtenerInformacion?pstrCertificado=string&pintModoValidacion=string&pintCodigoAplicacion=string HTTP/1.1
```

```
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ObtenerInformacionResponse xmlns="http://intranet.es/webservices/MPC">
      <ObtenerInformacionResult>xml</ObtenerInformacionResult>
    </ObtenerInformacionResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP GET. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
GET /PFC/MPCService.asmx/ObtenerInformacion?pstrCertificado=string&pintModoValidacion=string&pintCodigoAplicacion=string HTTP/1.1
Host: localhost
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0"?>
xml
```

HTTP POST

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP POST. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx/ObtenerInformacion HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

pstrCertificado=string&pintModoValidacion=string&pintCodigoAplicacion=string
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0"?>
xml
```

Ilustración 131: Página de prueba para el método ObtenerInformacion()

Método ObtenerInformacionGrupoCertificados()

MPCService

Haga clic aquí para obtener una lista completa de operaciones.

ObtenerInformacionGrupoCertificados

Extrae y parsea los datos de un grupo de certificados.

Prueba

El formulario de prueba sólo está disponible para métodos con tipos primitivos como parámetros.

SOAP 1.1

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.1. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://intranet.es/webservices/MPC/ObtenerInformacionGrupoCertificados"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ObtenerInformacionGrupoCertificados xmlns="http://intranet.es/webservices/MPC">
      <pstrCertificados>
        <string>string</string>
        <string>string</string>
      </pstrCertificados>
      <pintCodigoAplicacion>int</pintCodigoAplicacion>
    </ObtenerInformacionGrupoCertificados>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ObtenerInformacionGrupoCertificadosResponse xmlns="http://intranet.es/webservices/MPC">
      <ObtenerInformacionGrupoCertificadosResult>xml</ObtenerInformacionGrupoCertificadosResult>
    </ObtenerInformacionGrupoCertificadosResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.2. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ObtenerInformacionGrupoCertificados xmlns="http://intranet.es/webservices/MPC">
      <pstrCertificados>
        <string>string</string>
        <string>string</string>
      </pstrCertificados>
      <pintCodigoAplicacion>int</pintCodigoAplicacion>
    </ObtenerInformacionGrupoCertificados>
  </soap12:Body>
</soap12:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ObtenerInformacionGrupoCertificadosResponse xmlns="http://intranet.es/webservices/MPC">
      <ObtenerInformacionGrupoCertificadosResult>xml</ObtenerInformacionGrupoCertificadosResult>
    </ObtenerInformacionGrupoCertificadosResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP GET. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
GET /PFC/MPCService.asmx/ObtenerInformacionGrupoCertificados?pstrCertificados=string&pstrCertificados=string&pintCodigoAplicacion=string HTTP/1.1
Host: localhost
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
    </ObtenerInformacionGrupoCertificadosResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP GET. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
GET /PFC/MPCService.asmx/ObtenerInformacionGrupoCertificados?pstrCertificados=string&pstrCertificados=string&pintCodigoAplicacion=string HTTP/1.1
Host: localhost
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0"?>
xml
```

HTTP POST

A continuación se muestra un ejemplo de solicitud y respuesta para HTTP POST. Es necesario reemplazar los marcadores de posición que aparecen con valores reales.

```
POST /PFC/MPCService.asmx/ObtenerInformacionGrupoCertificados HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

pstrCertificados=string&pstrCertificados=string&pintCodigoAplicacion=string
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0"?>
xml
```

Ilustración 132: Página de prueba para el método ObtenerInformacionGrupoCertificados()

6.3 Ejemplos de salida del sistema MPC

6.3.1 Método ObtenerInformacion()

Ejemplo de salida para un certificado válido de tipo FNMT Clase 2 CA de persona física, eligiendo el modo de validación simple:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T16:38:00+02:00</HoraConsulta>
  <DatosCertificado>
    <NumeroSerie>1022483068</NumeroSerie>
    <Emisor>C=ES,O=FNMT,OU=FNMT Clase 2 CA</Emisor>
    <Asunto>C=es,O=fnmt,OU=fnmt clase 2 ca,OU=500650017,CN=NOMBRE HERNANDEZ TORREGROSA
MANUEL - NIF 99999999R</Asunto>
    <ValidoDesde>2013-03-12T07:14:54.00Z</ValidoDesde>
    <ValidoHasta>2016-03-12T07:14:54.00Z</ValidoHasta>
    <EntidadCertificadora>FNMT</EntidadCertificadora>
    <UnidadEntidadCertificadora>FNMT Clase 2 CA</UnidadEntidadCertificadora>
    <TipoEntidadCertificadora>0</TipoEntidadCertificadora>
    <TipoPersona>0</TipoPersona>
    <NombreCompleto>HERNANDEZ TORREGROSA, MANUEL</NombreCompleto>
    <PrimerApellido>HERNANDEZ</PrimerApellido>
    <SegundoApellido>TORREGROSA</SegundoApellido>
    <Nombre>MANUEL</Nombre>
    <Nif>99999999R</Nif>
    <Pais>es</Pais>
    <Email />
  </DatosCertificado>
  <ValidacionCertificado>
    <Valido>1</Valido>
    <ResultadoValidacion>Certificado ok</ResultadoValidacion>
    <ValidacionSimple>
      <CodigoValidacionSimple>0</CodigoValidacionSimple>
      <DescripcionValidacionSimple>Validación
satisfactoria</DescripcionValidacionSimple>
    </ValidacionSimple>
  </ValidacionCertificado>
</Respuesta>
```

Ilustración 133: Ejemplo de salida 1 para ObtenerInformacion()

Ejemplo de salida para un certificado válido de tipo FNMT Clase 2 CA de persona jurídica, eligiendo el modo de validación simple:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T17:06:59+02:00</HoraConsulta>
  <DatosCertificado>
    <NumeroSerie>1014931230</NumeroSerie>
    <Emisor>C=ES,O=FNMT,OU=FNMT Clase 2 CA</Emisor>
    <Asunto>C=ES,O=FNMT,OU=FNMT Clase 2 CA,OU=703007435,CN=ENTIDAD FUNDACION CAMPO -
CIF G99999999 - NOMBRE GALEOTE DIAZ HUGO - NIF 99999999R</Asunto>
    <ValidoDesde>2005-01-26T15:32:06.00Z</ValidoDesde>
    <ValidoHasta>2007-01-26T15:32:06.00Z</ValidoHasta>
    <EntidadCertificadora>FNMT</EntidadCertificadora>
    <UnidadEntidadCertificadora>FNMT Clase 2 CA</UnidadEntidadCertificadora>
    <TipoEntidadCertificadora>0</TipoEntidadCertificadora>
    <TipoPersona>1</TipoPersona>
    <DatosEntidadJuridica>
      <EntidadJuridica>FUNDACION CAMPO</EntidadJuridica>
      <Cif>G99999999</Cif>
    </DatosEntidadJuridica>
    <NombreCompleto>GALEOTE DIAZ, HUGO</NombreCompleto>
    <PrimerApellido>GALEOTE</PrimerApellido>
    <SegundoApellido>DIAZ</SegundoApellido>
    <Nombre>VICENTE</Nombre>
```

```
    <Nif>99999999R</Nif>
    <Pais>ES</Pais>
    <Email>FCAMPO@FCAMPO.ORG</Email>
  </DatosCertificado>
  <ValidacionCertificado>
    <Valido>1</Valido>
    <ResultadoValidacion>Certificado ok</ResultadoValidacion>
    <ValidacionSimple>
      <CodigoValidacionSimple>0</CodigoValidacionSimple>
      <DescripcionValidacionSimple>Validación
satisfactoria</DescripcionValidacionSimple>
    </ValidacionSimple>
  </ValidacionCertificado>
</Respuesta>
```

Ilustración 134: Ejemplo de salida 2 para ObtenerInformacion()

Ejemplo de salida para un certificado caducado de tipo DNIe de autenticación eligiendo el modo de validación simple:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T16:56:14+02:00</HoraConsulta>
  <DatosCertificado>
    <NumeroSerie>370003312712885416847982063081701506l4</NumeroSerie>
    <Emisor>C=ES,O=DIRECCION GENERAL DE LA POLICIA,OU=DNIE,CN=AC DNIE 001</Emisor>
    <Asunto>C=ES,SERIALNUMBER=99999999R,SURNAME=DIAZ,GIVENNAME=JUAN,CN=DIAZ PEREZ\,
JUAN (AUTENTICACIÓN)</Asunto>
    <ValidoDesde>2006-03-30T17:34:10.00Z</ValidoDesde>
    <ValidoHasta>2008-09-30T17:34:09.00Z</ValidoHasta>
    <EntidadCertificadora>DIRECCION GENERAL DE LA POLICIA</EntidadCertificadora>
    <UnidadEntidadCertificadora>DNIE</UnidadEntidadCertificadora>
    <DatosEntidadCertificadora>AC DNIE 001</DatosEntidadCertificadora>
    <TipoEntidadCertificadora>1</TipoEntidadCertificadora>
    <TipoPersona>0</TipoPersona>
    <Proposito>0</Proposito>
    <NombreCompleto>DIAZ PEREZ, JUAN</NombreCompleto>
    <PrimerApellido>DIAZ</PrimerApellido>
    <SegundoApellido>PEREZ</SegundoApellido>
    <Nombre>JUAN</Nombre>
    <Nif>99999999R</Nif>
    <Pais>ES</Pais>
    <Email />
  </DatosCertificado>
  <ValidacionCertificado>
    <Valido>0</Valido>
    <ResultadoValidacion>El certificado no pasó la validación Simple.
</ResultadoValidacion>
    <ValidacionSimple>
      <CodigoValidacionSimple>1</CodigoValidacionSimple>
      <DescripcionValidacionSimple>Certificado caducado</DescripcionValidacionSimple>
    </ValidacionSimple>
  </ValidacionCertificado>
</Respuesta>
```

Ilustración 135: Ejemplo de salida 3 para ObtenerInformacion()

Ejemplo de salida para un certificado de empleado público de tipo FNMT AP, sin elegir validación:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T17:25:08+02:00</HoraConsulta>
  <DatosCertificado>
    <NumeroSerie>147871037764999458671073112045995721474</NumeroSerie>
    <Emisor>C=ES,O=FNMT-RCM,OU=CERES,SERIALNUMBER=Q2826004J,CN=AC Administración
Pública</Emisor>
    <Asunto>C=ES,O=MINISTERIO DE INDUSTRIA ENERGIA Y TURISMO,OU=certificado electrónico
de empleado público,OU=SUBSECRETARIA,SERIALNUMBER=99999999R,SURNAME=PELAYO
VEGAS,GIVENNAME=MARIA PALOMA,CN=PELAYO VEGAS MARIA PALOMA - DNI 99999999R</Asunto>
    <ValidoDesde>2012-02-21T09:34:05.00Z</ValidoDesde>
    <ValidoHasta>2015-02-21T09:34:05.00Z</ValidoHasta>
    <EntidadCertificadora>FNMT-RCM</EntidadCertificadora>
    <UnidadEntidadCertificadora>CERES</UnidadEntidadCertificadora>
    <DatosEntidadCertificadora>AC Administración Pública</DatosEntidadCertificadora>
    <TipoEntidadCertificadora>3</TipoEntidadCertificadora>
    <TipoPersona>0</TipoPersona>
    <NombreCompleto>PELAYO VEGAS, MARÍA PALOMA</NombreCompleto>
    <PrimerApellido>PELAYO</PrimerApellido>
    <SegundoApellido>VEGAS</SegundoApellido>
    <Nombre>MARÍA PALOMA</Nombre>
    <Nif>99999999R</Nif>
    <Pais>ES</Pais>
    <Email>MPPELAYO@MINETUR.ES</Email>
    <EntidadSuscriptora>MINISTERIO DE INDUSTRIA ENERGIA Y TURISMO</EntidadSuscriptora>
    <NifEntidadSuscriptora>S2800214E</NifEntidadSuscriptora>
    <NumIdentificacionPersonal>1234568 A1234</NumIdentificacionPersonal>
    <UnidadOrganizativa>SUBSECRETARIA</UnidadOrganizativa>
    <Cargo>ANALISTA DE SISTEMAS</Cargo>
  </DatosCertificado>
</Respuesta>
```

Ilustración 136: Ejemplo de salida 4 para ObtenerInformacion()

Ejemplo de salida para un certificado con extensiones no válidas de empleado público de tipo FNMT APE con validación:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T17:35:22+02:00</HoraConsulta>
  <DatosCertificado>
    <NumeroSerie>10082</NumeroSerie>
    <Emisor>C=ES,O=FNMT-RCM,OU=AC APE</Emisor>
    <Asunto>C=ES,O=FNMT-RCM,OU=AC APE,OU=500780533,CN=NOMBRE RODRIGUEZ LOPEZ MANUEL -
NIF 99999999R</Asunto>
    <ValidoDesde>2010-04-29T07:17:49.00Z</ValidoDesde>
    <ValidoHasta>2014-04-29T07:17:49.00Z</ValidoHasta>
    <EntidadCertificadora>FNMT-RCM</EntidadCertificadora>
    <UnidadEntidadCertificadora>AC APE</UnidadEntidadCertificadora>
    <TipoEntidadCertificadora>2</TipoEntidadCertificadora>
    <TipoPersona>0</TipoPersona>
    <NombreCompleto>RODRIGUEZ LOPEZ, MANUEL</NombreCompleto>
    <PrimerApellido>RODRIGUEZ</PrimerApellido>
    <SegundoApellido>LOPEZ</SegundoApellido>
    <Nombre>ALFONSO</Nombre>
    <Nif>99999999R</Nif>
    <Pais>ES</Pais>
    <Email>MRODRIGUEZ@MINETUR.ES</Email>
    <EntidadSuscriptora>MINISTERIO DE ECONOMÍA Y HACIENDA</EntidadSuscriptora>
    <NifEntidadSuscriptora>S2800214E</NifEntidadSuscriptora>
    <NumIdentificacionPersonal>1234568 A1234</NumIdentificacionPersonal>
    <UnidadOrganizativa>SISTEMAS</UnidadOrganizativa>
    <Cargo>PROGRAMADOR</Cargo>
    <SituacionLaboral>FUNCIONARIO</SituacionLaboral>
  </DatosCertificado>
  <ValidacionCertificado>
    <Valido>0</Valido>
```

```
    <ResultadoValidacion>El certificado no pasó la validación
Simple</ResultadoValidacion>
    <ValidacionSimple>
      <CodigoValidacionSimple>5</CodigoValidacionSimple>
      <DescripcionValidacionSimple>El certificado posee extensiones que no son
válidas</DescripcionValidacionSimple>
    </ValidacionSimple>
  </ValidacionCertificado>
</Respuesta>
```

Ilustración 137: Ejemplo de salida 5 para ObtenerInformacion()

Ejemplo de salida para un certificado no soportado por el sistema:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T18:02:41+02:00</HoraConsulta>
  <Error>
    <OrigenError>3</OrigenError>
    <CodigoError>2</CodigoError>
    <DescripcionError>Certificado no soportado por el sistema</DescripcionError>
    <DetalleError>MPC08: Certificado no soportado por el sistema</DetalleError>
  </Error>
</Respuesta>
```

Ilustración 138: Ejemplo de salida 6 para ObtenerInformacion()

Ejemplo de salida para una llamada a ObtenerInformacion() pasando un parámetro incorrecto:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T18:05:00+02:00</HoraConsulta>
  <Error>
    <OrigenError>3</OrigenError>
    <CodigoError>1</CodigoError>
    <DescripcionError>Formato de entrada no válido</DescripcionError>
    <DetalleError>MPC01: El certificado presentado por el cliente tiene un formato
incorrecto.</DetalleError>
  </Error>
</Respuesta>
```

Ilustración 139: Ejemplo de salida 7 para ObtenerInformacion()

Ejemplo de salida con una excepción al parsear los nombres alternativos de un certificado:

```
<Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T18:12:45+02:00</HoraConsulta>
  <Error>
    <OrigenError>3</OrigenError>
    <CodigoError>3</CodigoError>
    <DescripcionError>Error al obtener los datos parseados del
certificado</DescripcionError>
    <DetalleError>MPC12: Error al parsear los nombres alternativos del
certificado</DetalleError>
  </Error>
</Respuesta>
```

Ilustración 140: Ejemplo de salida 8 para ObtenerInformacion()

6.3.2 Método EsValido()

Ejemplo de salida para un certificado válido:

```
<int xmlns="http://intranet.es/webservices/MPC">1</int>
```

Ilustración 141: Ejemplo de salida 1 para EsValido()

Ejemplo de salida para un certificado no válido:

```
<int xmlns="http://intranet.es/webservices/MPC">0</int>
```

Ilustración 142: Ejemplo de salida 2 para EsValido()

Ejemplo de salida en caso de producirse una excepción:

```
<int xmlns="http://intranet.es/webservices/MPC">-1</int>
```

Ilustración 143: Ejemplo de salida 3 para EsValido()

6.3.3 Método ObtenerInformacionGrupoCertificados()

Ejemplo de salida para el método ObtenerInformacionGrupoCertificados() para 2 certificados:

```
<Certificados total="2">
  <Respuesta>
  <VersionRespuesta>1.0</VersionRespuesta>
  <HoraConsulta>2013-09-30T18:32:51+02:00</HoraConsulta>
  <DatosCertificado>
          <NumeroSerie>1018937898</NumeroSerie>
          <Emisor>C=ES,O=FNMT,OU=FNMT Clase 2 CA</Emisor>
          <Asunto>C=es,O=fnmt,OU=fnmt clase 2 ca,OU=500650017,CN=NOMBRE HERNANDEZ
TORREGROSA MANUEL - NIF 47300252Q</Asunto>
          <ValidoDesde>2010-04-14T09:02:41.00Z</ValidoDesde>
          <ValidoHasta>2013-04-14T09:02:41.00Z</ValidoHasta>
          <EntidadCertificadora>FNMT</EntidadCertificadora>
          <UnidadEntidadCertificadora>FNMT Clase 2 CA</UnidadEntidadCertificadora>
          <TipoEntidadCertificadora>0</TipoEntidadCertificadora>
          <TipoPersona>0</TipoPersona>
          <NombreCompleto>HERNANDEZ TORREGROSA, MANUEL</NombreCompleto>
          <PrimerApellido>HERNANDEZ</PrimerApellido>
          <SegundoApellido>TORREGROSA</SegundoApellido>
          <Nombre>MANUEL</Nombre>
          <Nif>47300252Q</Nif>
          <Pais>es</Pais>
          <Email>MHERNANDEZT@MITYC.ES</Email>
  </DatosCertificado>
  </Respuesta>
  <Respuesta>
    <VersionRespuesta>1.0</VersionRespuesta>
    <HoraConsulta>2013-09-30T17:35:22+02:00</HoraConsulta>
    <DatosCertificado>
      <NumeroSerie>10082</NumeroSerie>
      <Emisor>C=ES,O=FNMT-RCM,OU=AC APE</Emisor>
      <Asunto>C=ES,O=FNMT-RCM,OU=AC APE,OU=500780533,CN=NOMBRE RODRIGUEZ LOPEZ MANUEL -
NIF 99999999R</Asunto>
      <ValidoDesde>2010-04-29T07:17:49.00Z</ValidoDesde>
      <ValidoHasta>2014-04-29T07:17:49.00Z</ValidoHasta>
      <EntidadCertificadora>FNMT-RCM</EntidadCertificadora>
      <UnidadEntidadCertificadora>AC APE</UnidadEntidadCertificadora>
      <TipoEntidadCertificadora>2</TipoEntidadCertificadora>
      <TipoPersona>0</TipoPersona>
      <NombreCompleto>RODRIGUEZ LOPEZ, MANUEL</NombreCompleto>
      <PrimerApellido>RODRIGUEZ</PrimerApellido>
      <SegundoApellido>LOPEZ</SegundoApellido>
      <Nombre>ALFONSO</Nombre>
      <Nif>99999999R</Nif>
      <Pais>ES</Pais>
      <Email>MRODRIGUEZ@MINETUR.ES</Email>
      <EntidadSuscriptora>MINISTERIO DE ECONOMÍA Y HACIENDA</EntidadSuscriptora>
      <NifEntidadSuscriptora>S2800214E</NifEntidadSuscriptora>
      <NumIdentificacionPersonal>1234568 A1234</NumIdentificacionPersonal>
      <UnidadOrganizativa>SISTEMAS</UnidadOrganizativa>
      <Cargo>PROGRAMADOR</Cargo>
      <SituacionLaboral>FUNCIONARIO</SituacionLaboral>
    </DatosCertificado>
  </Respuesta>
</Certificados>
```

Ilustración 144: Ejemplo de salida para ObtenerInformacionGrupoCertificados()

7 Planificación

7.1 Descripción

Para poder planificar correctamente el proyecto este ha sido dividido en distintas fases. Las etapas principales del proyecto han sido las siguientes:

- Fase de Análisis
- Fase de Diseño
- Fase de Implementación
- Fase de Pruebas
- Fase de Documentación

La fase de análisis es la de mayor importancia del proyecto ya que sienta las bases para la llegada a buen puerto del mismo. En esta fase se comenzó recopilando información sobre la propuesta inicial del cliente de cara a comprender mejor el objetivo del proyecto. Se estudió el alcance de la propuesta y se mantuvieron reuniones con el cliente para la obtención de los requisitos. Una vez definidos los requisitos se crearon los casos de uso y los diagramas de secuencia UML. En este proyecto en concreto se dedicó mucho tiempo al análisis las políticas de certificación de las principales emisores de certificados que están soportados por la aplicación. Es un punto importante del análisis ya que de aquí se obtiene toda la información sobre los campos que la aplicación debe devolver como salida. En esta parte también se concretó el modelo de datos de la aplicación.

La fase de diseño se desarrolló especificando aspectos como el formato concreto que debe tener el XML que se dará como salida mediante la definición de su XSD, diseñando la arquitectura que tiene que tener la aplicación, realizando el diseño de componentes teniendo presente los requisitos de escalabilidad y extensibilidad y creando los diagramas de clases detallados que servirían de guía a la hora de llevar a cabo la fase de implantación.

La fase de implantación es la que más tiempo consumió del proyecto. Primeramente se implementó un esqueleto básico del Servicio Web para posteriormente ir añadiéndole más funcionalidad. Luego se creó la base de datos con todos sus

procedimientos almacenados para insertar datos desde la aplicación, funciones y tablas maestras. Con la base de datos ya creada se implementó la librería de acceso a datos de la aplicación, que sería la encargada de insertar las estadísticas de cada petición procesada por el Servicio Web. A continuación se implementaron las clases del parseador genérico de certificados utilizando BouncyCastle. Estas clases son el corazón de la aplicación, ya que son las que llevan a cabo gran parte del parseo y validación de los certificados, el resto de librerías concretas de cada certificado (FNMT, FNMT AP, FNTM APE y DNIe) se desarrollaron sobre esta primera librería genérica. También se implementaron otros aspectos necesarios tales como utilidades para el manejo de XML, gestión de excepciones, manejo del log, etc… que se incluyen en una librería común. Finalmente se puso todo en común y se completó la funcionalidad del esqueleto de la aplicación incluyendo algunas clases necesarias que por su naturaleza no casaban en ninguna librería anterior completando el proceso de implementación del proyecto.

En la fase de pruebas se llevaron a cabo tests para comprobar el correcto funcionamiento de la aplicación. Principalmente las pruebas se centraron en el parseo y validación de certificados, inserción de estadísticas, seguridad de la aplicación y otras pruebas menos concretas como comprobar el correcto funcionamiento del log o la gestión de excepciones.

En la fase de documentación se generó toda la documentación del proyecto.

7.2 Diagrama de Gantt

Para poder crear el diagrama de Gantt de la aplicación se detallaron las tareas de cada fase y se asignaron tiempos en horas para cada tarea. Se ha tenido en cuenta jornadas laborables de 8 horas de duración y un calendario normal en el que solo se trabaja de lunes a viernes. A continuación se pueden observar las tareas junto a su duración y su fecha de comienzo y fin:

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**206 días**	**lun 02/01/12**	**lun 15/10/12**
Fase de Análisis	**39 días**	**lun 02/01/12**	**jue 23/02/12**
Recopilación información preliminar	4 días	lun 02/01/12	jue 05/01/12
Estudio del problema	4 días	vie 06/01/12	mié 11/01/12
Toma de requisitos	6 días	jue 12/01/12	jue 19/01/12
Creación casos de uso y diagramas de secuencia	7 días	vie 20/01/12	lun 30/01/12
Análisis de las políticas de certificación	10 días	mar 31/01/12	lun 13/02/12
Definición de los datos de salida	5 días	mar 14/02/12	lun 20/02/12
Análisis modelo de datos	3 días	mar 21/02/12	jue 23/02/12
Fase de Diseño	**16 días**	**vie 24/02/12**	**vie 16/03/12**
Diseño XML de salida	5 días	vie 24/02/12	jue 01/03/12
Diseño de la arquitectura	3 días	vie 02/03/12	mar 06/03/12
Diseño de componentes	3 días	mié 07/03/12	vie 09/03/12
Creación diagrama de clases	5 días	lun 12/03/12	vie 16/03/12
Fase de Implementación	**129 días**	**lun 19/03/12**	**jue 13/09/12**
Desarrollo esqueleto Servicio Web	15 días	lun 19/03/12	vie 06/04/12
Creación base de datos	4 días	lun 09/04/12	jue 12/04/12
Desarrollo librería acceso a base de datos	20 días	vie 13/04/12	jue 10/05/12
Desarrollo librería parseador BouncyCastle	10 días	vie 11/05/12	jue 24/05/12
Desarrollo librería parseador DNIe	10 días	vie 25/05/12	jue 07/06/12
Desarrollo librería parseador FNMT	10 días	vie 08/06/12	jue 21/06/12
Desarrollo librería parseador FNMT AP	10 días	vie 22/06/12	jue 05/07/12
Desarrollo librería parseador FNMT APE	10 días	vie 06/07/12	jue 19/07/12
Desarrollo librería funcionalidad común	15 días	vie 20/07/12	jue 09/08/12
Integración final de librerías en Servicio Web	25 días	vie 10/08/12	jue 13/09/12
Fase de Pruebas	**12 días**	**vie 14/09/12**	**lun 01/10/12**
Pruebas inserción estadísticas	3 días	vie 14/09/12	mar 18/09/12
Pruebas parseo certificados	3 días	mié 19/09/12	vie 21/09/12
Pruebas validación certificados	3 días	lun 24/09/12	mié 26/09/12
Pruebas seguridad sistema	3 días	jue 27/09/12	lun 01/10/12
Fase de Documentación	**10 días**	**mar 02/10/12**	**lun 15/10/12**
Documentación del sistema	10 días	mar 02/10/12	lun 15/10/12

Ilustración 145: Cuadro de tareas del proyecto

La duración total del proyecto es de 206 días. Se puede observar como la fase que más tiempo consume es la de Implementación y la que menos la de Documentación.

A continuación se muestra el diagrama de Gantt del proyecto:

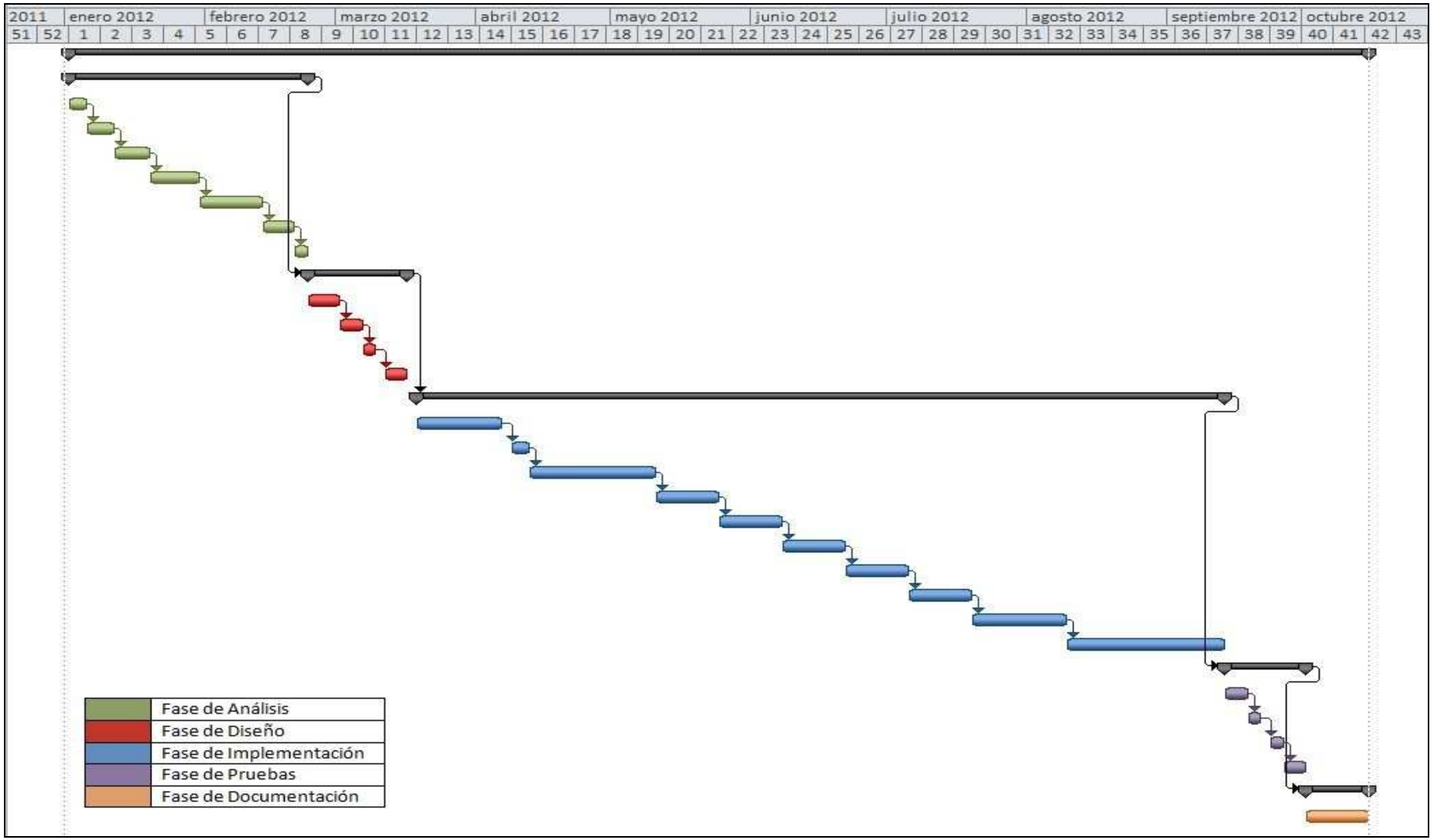


Ilustración 146: Diagrama de Gantt del proyecto

8 Presupuesto

8.1 Recursos

Para realizar el presupuesto del proyecto debemos identificar correctamente todos los recursos que se necesitan para completar el mismo. Esto incluye desde el coste humano hasta los equipos y licencias de programas que sean imprescindibles para completar correctamente el proyecto. Todo esto se detalla en los siguientes puntos.

8.1.1 Recursos Humanos

Aunque para la realización de este proyecto se ha tenido el apoyo y supervisión de un tutor de empresa, se considerará a la hora de calcular el presupuesto como que su coste ha sido cero y por tanto no se tendrá en cuenta. En consecuencia, los recursos humanos del proyecto serán 1 analista programador trabajando a jornada completa con un coste bruto a la hora de 25 euros. En la siguiente tabla aparece desglosado el coste humano para cada tarea:

ACTIVIDAD	DÍAS	HORAS	COSTE €
Fase de Análisis			
Recopilación información preliminar	4	32	800
Estudio del problema	4	32	800
Toma de requisitos	6	48	1200
Creación casos de uso y diagramas de secuencia	7	56	1400
Análisis de las políticas de certificación	10	80	2000
Definición de los datos de salida	5	40	1000
Análisis modelo de datos	3	24	600
Total fase análisis	39	312	7800

Fase de Diseño			
Diseño XML de salida	5	40	1000
Diseño de la arquitectura	3	24	600
Diseño de componentes	3	24	600
Creación diagrama de clases	5	40	1000
Total fase diseño	16	128	3200
Fase de Implementación			
Desarrollo esqueleto servicio web.	15	120	3000
Creación base de datos	4	32	800
Desarrollo librería acceso a base de datos	10	80	2000
Desarrollo librería parseador BouncyCastle	20	160	4000
Desarrollo librería parseador DNIe	10	80	2000
Desarrollo librería parseador FNMT	10	80	2000
Desarrollo librería parseador FNMT AP	10	80	2000
Desarrollo librería parseador FNMT APE	10	80	2000
Desarrollo librería funcionalidad común	15	120	3000
Integración final de librerías en servicio web.	25	200	5000
Total fase implementación	129	1032	25800
Fase de Pruebas			
Pruebas inserción estadísticas	3	24	600
Pruebas parseo certificados	3	24	600
Pruebas validación certificados	3	24	600

Pruebas seguridad sistema	3	24	600
Total fase pruebas	12	96	2400
Fase de Documentación			
Documentación del sistema	10	80	2000
Total fase documentación	10	80	2000

Ilustración 147: Coste humano de cada tarea

Si sumamos todas las fases del proyecto obtenemos finalmente el total de costes de recursos humanos empleados:

ACTIVIDAD	DÍAS	HORAS	COSTE €
Total fase análisis	39	312	7800
Total fase diseño	16	128	3200
Total fase implementación	129	1032	25800
Total fase pruebas	12	96	2400
Total fase documentación	10	80	2000
Total Recursos humanos	206	1648	41200

Ilustración 148: Total costes de recursos humanos del proyecto

Como vemos, el resultado del coste humano del proyecto son 42.200 euros repartidos en 206 jornadas de trabajo de 8 horas.

8.1.2 Recursos Software

Los recursos software utilizados en la realización del proyecto son los siguientes:

SOFTWARE	COSTE LICENCIA €
Microsoft Office 2007	200
Windows XP Professional Service Pack 3	250

Visual Studio 2010 Professional	500
SqlServer 2005	2000
Enterprise Architect 7.5	200
Total	3150

Ilustración 149: Recursos Software del proyecto

8.1.3 Recursos Hardware

Los recursos Hardware utilizados en la realización de este proyecto han sido un ordenador **IBM ThinkPad** con un coste total de 1.000 euros.

8.2 Resumen del presupuesto

Llegados a este punto podemos poner en común todos los recursos empleados junto con sus costes y obtener realmente el coste total del proyecto.

RECURSOS	COSTE €
Recursos humanos	41200
Recursos Software	3150
Recursos Hardware	1000
Total	45350

Ilustración 150: Coste total del proyecto

El proyecto tiene un coste total que asciende a 45.350 euros.

9 Futuras líneas de trabajo

La principal ampliación que sufrirá el sistema MPC en el futuro es la validación de certificados mediante el protocolo OCSP. Actualmente el sistema solo realiza la validación básica de un certificado, comprobando su caducidad, integridad y confianza. En la práctica es necesario también conocer el estado de revocación del mismo. Esta parte será una librería realizada por otro grupo de trabajo. El sistema MPC incluirá un nivel más de validación (validación intermedia), la cual realizará - a parte de la validación básica - la llamada al ocsp correspondiente a cada tipo de certificado para obtener el estado de revocación del mismo. Típicamente cada emisor tiene un servidor OCSP distinto, por tanto el sistema está preparado para almacenar en el archivo de configuración de la librería de cada emisor, un campo con la dirección OCSP contra la que se debe validar. En algunos casos, como ocurre con la Fábrica Nacional de Moneda y Timbre, el servicio de consulta OCSP no es gratuito, y su acceso está controlado mediante usuario y contraseña. Esta información también se almacenará en el archivo de configuración de la librería FnmtCertLib.dll

El sistema MPC está diseñado de forma que sea fácilmente ampliable. Lo más lógico es pensar que en el futuro se soportarán también nuevos prestadores y tipos de certificados. Así pues es muy probable que en el futuro se vayan añadiendo librerías al sistema - una por cada nuevo emisor- junto con sus certificados raíces para poder identificarlos.

10 Conclusiones

La realización de este proyecto me ha aportado mucho tanto a nivel profesional como personal. Al tratarse de un proyecto que iba a ser implantado y utilizado ha supuesto un reto, debido a la presión y la exigencia que supone saber que el sistema será probado por multitud de usuarios y aplicaciones.

He aprendido a realizar un análisis en un escenario real tomando los requisitos directamente del cliente mediante reuniones cara a cara. He sido consciente de lo difícil que es concretar y definir claramente los requisitos principales que deben cumplirse y lo importante que es saber identificar y descartar qué funcionalidades no aportarían nada al sistema. El cliente siempre sabe qué resultado quiere pero es mejor aconsejarle sobre cómo conseguirlo.

También quería comentar que con el desarrollo de este proyecto he comprendido que aunque siempre se deben seguir al píe de la letra los supuestos contemplados en el análisis, a la hora de la implementación es inevitable que surjan cambios, dificultades y mejoras que puedan variar ese primer planteamiento sobre papel. Por su puesto, lo que se debe sacar en conclusión de esto, es que siempre se deben cumplir los requisitos concretados con el cliente, pero el análisis y diseño deben ser lo suficientemente flexibles como para soportar modificaciones.

11 Bibliografía

Federico García. *Certificados X509*. En:

http://www.alu.ua.es/f/fgc10/

Carlos Verdes Blanco. *Certificados digitales y tarjetas inteligentes.* En:

http://es.scribd.com/doc/176820455/09-Autenticacion-Basada-en-Certificados-Digitales

Arturo López Guevara, Antonio Díaz Agudo. *La criptografía de clave pública y su aplicación a las tecnologías de la información*. En:

http://www.profesores.frc.utn.edu.ar/sistemas/ingsanchez/Redes/Archivos/Criptografia_de_clave_publica.pdf

BOE núm. 304. *LEY 59/2003, de 19 de diciembre, de firma electrónica*. En:

http://www.boe.es/boe/dias/2003/12/20/pdfs/A45329-45343.pdf

María Jesús Lamarca Lapuente. *Hipertexto: El nuevo concepto de documento en la cultura de la imagen.* En:

http://www.hipertexto.info/documentos/xml.htm

W3*. Tecnología XML.* En:

http://www.w3.org/standards/xml/

W3. Descripción Servicios Web. En:

http://www.w3.org/TR/ws-arch/

Wikipedia. *Definición de Servicio Web*. En:

http://es.wikipedia.org/wiki/Servicio_web

Wikipedia. *Expresiones Regulares*. En:

http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular

Declaración y prácticas de certificación del DNI Electrónico. En:

http://www.dnielectronico.es/PDFs/politicas_de_certificacion.pdf

Declaración y prácticas de certificación de la FNMT (General). En:

https://www.sede.fnmt.gob.es/documents/11614/137772/DPC_22_Junio_2010.pdf

Declaración y prácticas de certificación de la FNMT Clase 2 CA. En:

https://www.sede.fnmt.gob.es/documents/11614/137617/DPC_01_Agosto_2010.pdf

Declaración y prácticas de certificación de la FNMT para administraciones públicas. En:

https://www.sede.fnmt.gob.es/documents/11614/282563/DPC+APE+v13

Network Working Group R. Housley*. Internet X.509 Public Key Infrastructure*. En:

http://www.ietf.org/rfc/rfc3280.txt

The Legion of the Bouncy Castle. *Documentación y librería BouncyCastle.* En:

http://www.bouncycastle.org/csharp/index.html