Universidad Carlos III de Madrid

Escuela Politécnica Superior

Departamento de Tecnología Electrónica

Proyecto fin de carrera.

SISTEMA RELACIONAL DE DATOS EN PROCESOS DE EVALUACIÓN DE SEGURIDAD EN TECNOLOGÍAS DE IDENTIFICACIÓN.

1ª PARTE: TRANSMISION DE DATOS Y SEGURIDAD EN TECNOLOGÍAS DE IDENTIFICACIÓN CON DNIE.

INGENIERÍA TECNICA ELECTRONICA INDUSTRIAL

Autor: Diego Ismael Muñoz Martín.

Tutor: Raúl Sánchez Reíllo.

Fecha: 28-10-2011.

Dedico todo mi trabajo a mi mujer Alicia y a mis hijos Izan y Jone. También lo dedico a mi familia, a mis amigos, a mis compañeros, profesores y a todos los que me conocen, ya que siempre han confiado en mí, incluso más de lo que yo mismo confió. Por eso les doy las gracias a todos y espero poder cumplir con sus expectativas.

Índice.

Tablas.

Figuras.

Acrónimos.

AAP: Autoridad de **A**probación de **P**olíticas.

AC: Autoridad de **C**ertificación.

AES: Advanced **E**ncryption **S**tandard (también conocido como algoritmo Rijndael).

ANSI: American **N**ational **S**tandards **I**nstitute (instituto nacional estadounidense de estándares).

API: Application **P**rogramming **I**nterface (interfaz de programación de aplicaciones).

AR: Autoridad de **R**egistro.

ASP: Active **S**erver **P**ages.

AV: Autoridad de **V**alidación.

C: Country (País). Atributo dentro de DN.

CA: Certification **A**uthorities (lo mismo que AC).

CC: Common **C**riteria.

CCN: Centro **C**riptológico **N**acional.

CEN: Comité **E**uropeo de **N**ormalización.

CEPT: **C**onférence **e**uropéenne des administrations des **p**ostes et des **t**élécommunications (conferencia europea de administraciones de correos y telecomunicaciones).

CLI: Imagen **L**áser **C**ambiante.

CN: Common **N**ame (nombre común). Atributo dentro de DN.

CPD: **D**ata **P**rocessing **C**enter.

CPS: Certificate **P**ractise **S**tatements (la declaración de prácticas de certificación).

CRL: Certificate **R**evocation **L**ist (lista de certificados revocados).

CSP: Crypto **S**ervice **P**rovider (módulo Windows de acceso a proveedor de servicios criptográficos de Tarjetas Inteligentes).

CWA: CEN **W**orkshop **A**greement (grupo de trabajo que CEN creó para normalizar situaciones de mercado rápidamente).

C#: (también C Sharp) es un lenguaje de programación de alto nivel orientado a objetos, que Microsoft ha diseñado dentro de su plataforma .NET.

DES: Data **E**ncriptation **S**tandard (estándar de cifrado de datos).

DN: Distinguished **N**ame (Nombre Distintivo). Identificación unívoca de una entrada dentro de la estructura de directorio X.500.

DNI: Documento **N**acional de **I**dentidad.

DNIe: Documento **N**acional de **I**dentidad **E**lectrónico.

DSCF: Dispositivo **S**eguro para la **C**reación de **F**irma.

EAL: Evaluation **A**ssurance **L**evel.

EEPROM: Electrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory (dispositivo de memoria programable y borrable eléctricamente).

EMV: Europay, **M**asterCard y **V**ISA.

ETSI: European **T**elecommunications **S**tandards **I**nstitute (instituto europeo de normas de telecomunicaciones).

FIPS: Federal **I**nformation **P**rocessing **S**tandards (estándares del gobierno norteamericano de procesamiento de la información).

GN: Given**N**ame (nombre). Atributo dentro de DN.

HTTP: Hyper**t**ext **T**ransfer **P**rotocol (protocolo de transferencia de hipertexto).

HTTPS: Hyper**t**ext **T**ransfer **P**rotocol **S**ecure.

IDE: Integrated **D**evelopment **E**nvironment (entorno integrado de desarrollo de software).

IDEA: International **D**ata **E**ncryption **A**lgorithm (algoritmo internacional de cifrado de datos).

IEC: International **E**lectrotechnical **C**omision (Subgrupo de ISO).

IETF: Internet **E**ngineering **T**ask **F**orce (grupo especial sobre ingeniería de Internet encargado de regular los estándares de internet).

IIS 7: Internet **I**nformation **S**erver versión 7.

IP: Internet **P**rotocol.

ISBN: International **S**tandard **B**ook **N**umber (número estándar internacional de libro).

ISO: International **O**rganization for **S**tandardization (organización internacional para la estandarización).

ITU-T: International **T**elecommunication **U**nion – "**T**elecommunication Standardization Sector" (unión internacional de las telecomunicaciones).

MD5: Message **D**igest **5** (compendio o resumen de mensaje).

Ms_PL: Licencia **P**ública de **M**icro**s**oft.

O: Organization. Atributo dentro de DN.

OACI: Organización de **A**viación **C**ivil **I**nternacional.

OAEP: Optimal **A**symmetric **E**ncryption **P**adding (Algoritmo de relleno óptimo de cifrado asimétrico).

OASIS: Organization **A**dvancing **O**pen **S**tandards for the **I**nformation **S**ociety (organización para el avance de estándares las normas de información estructurada).

OCR-B: Optical **C**haracter **R**ecognition-Type **B**. (reconocimiento óptico de caracteres).

OCSP: Online **C**ertificate **S**tatus **P**rotocol (protocolo que permite comprobar en línea el estado del certificado electrónico).

OID: Object **i**dentifier (Identificador de objeto único).

OU: Organizacional **U**nit. Atributo dentro de DN.

PC/SC: Personal **C**omputer/**S**mart **C**ard.

PDA: Personal **D**igital **A**ssistant.

PDF: Portable **D**ocument **F**ormat (formato de documento portátil).

PDM: Product **D**ata **M**anagement (sistemas de gestión de datos del producto).

PFC: Proyecto **F**inal de **C**arrera.

PIN: Personal **I**dentification **N**umber (número de identificación personal).

PKCS: Public **K**ey **C**ryptography **S**tandards (estándares de PKI desarrollados por RSA).

PKI: Infraestructura de **C**lave **P**ública.

PKIX: Grupo de trabajo del IETF (Public Key Infrastructure X509 IEFT Working Group) constituido con el objeto de desarrollar las especificación relacionadas con las PKI e Internet.

PP: Perfil de **P**rotección.

RA: Lo mismo que AR.

RF: Radio **f**recuencias.

RFC: Request **F**or **C**omments (estándar emitido por la IETF).

RG: Requisitos **G**enerales.

RO: Requisitos **O**peracionales.

RS: Requisitos de **S**eguridad.

RSA: Rivest, **S**hamir and **A**dleman (de los nombres de los primeros desarrolladores del algoritmo criptográfico, se han convertido en un centro de normalización criptográfica).

SFP: Security **F**unction **P**olicy (requisitos funcionales de seguridad).

SHA: Secure **H**ash **A**lgorithm (algoritmo seguro de hash).

SN: Sur**N**ame (apellido). Atributo dentro de DN.

SSCD: Secure **S**ignature **C**reation **D**evice (dispositivo seguro de creación de firma electrónica).

SSL: Secure **S**ockets **L**ayer (protocolo de capa de conexión segura).

SSL3: Secure **S**ockets **L**ayer versión 3.

SSPI: Security **S**upport **P**rovider **I**nterface.

ST: Security **T**arget (declaración de seguridad).

TDT: Televisión **D**igital **T**errestre.

TOE: Target **o**f **E**valuation (objeto de evaluación).

TSA: Time **S**tamping **A**uthority (autoridad de sellado de tiempo).

TSC: TSF Scope of Control.

TSF: TOE Security Functions.

TSFI: TSF Interface.

TSP: TOE Security Policy.

TSU: Time **S**tamping **U**nit (unidad de sellado de tiempo).

WWW: World **W**ide **W**eb.

XML: E**x**tensible **M**arkup **L**anguage (lenguaje de marcas o etiquetas extensible).

3DES: Algoritmo que hace triple cifrado DES.

.NET: Es un framework de Microsoft diseñado para redes.

1.- Introducción.

1.1.- Estableciendo el ámbito del proyecto.

En ingeniería uno de los cometidos más importantes y amplios al que nos podemos enfrentar es el acceso y la gestión de la información. Por gestión de la información podemos entender la generación, la transmisión, el procesamiento o el archivado de datos.

En un ámbito profesional de trabajo, los datos que se manejan en la empresa pueden ser su principal bien o capital. El control de acceso a estos datos, necesariamente tiene que estar protegido. Además, internet es la herramienta de mayor difusión y versatilidad que tenemos, sobre todo en lo que a puntos de acceso se refiere. Por lo tanto, no es extraño tener que solucionar el acceso a los datos desde cualquier localización con conexión a internet como medio de enlace.

En este proyecto final de carrera (PFC) nos centraremos en la creación de un portal de acceso a un sitio web. Trataremos dos aspectos que son el acceso y la transmisión de información de forma segura. Concretamente estableceremos un canal seguro de transmisión de datos mediante las tecnologías Secure Sockets Layer (SSL), y utilizaremos el documento nacional de identidad de España (DNIe) como medio de autenticación del usuario que intenta acceder.

En un control de acceso, tenemos dos interlocutores cliente y servidor, que necesitan cubrir fundamentalmente tres servicios:

- El cifrado de la comunicación, busca evitar que otras personas puedan acceder a la comunicación.
- La validación de la identidad (autenticación), es decir, la identificación de los interlocutores que establecen la comunicación.
- La integridad del mensaje, este apartado queda resuelto con la firma digital, ya que sirve para detectar modificaciones en el contenido de un mensaje.

Para conseguir la trasmisión de datos segura, emplearemos las funcionalidades que nos proporciona el DNIe, de autenticación y firma digital. Está claro que en estas operaciones de creación de canales seguros de comunicación, intervienen otros factores (gestión y acceso a las claves de seguridad, gestión de recursos y bases de datos, etc.) e intentaremos proporcionar herramientas que faciliten estos procedimientos.

Por último destacar que la base de todo el PFC utiliza la tecnología .NET de Microsoft, que en la actualidad es el sistema de desarrollo más utilizado en el mundo. Esto nos permite emplear los sistemas de seguridad más extendidos, probados y atacados de la actualidad y, nos da una evolución en la seguridad razonablemente actualizada.

1.2.- ¿Qué objetivos buscamos con este proyecto?

Este PFC pretende conseguir un sistema seguro para realizar nuestro acceso a la información con la tecnología disponible en estos momentos. La base de nuestra solución es la utilización

de sistemas de cifrado asimétricos de clave pública, combinados con sistemas de cifrado simétricos.

Estamos seguros de que este sistema podría ser roto mediante procedimientos que están en constante evolución. Sin embargo, pensamos que la base en que se apoya la seguridad no es en crear sistemas infalibles, sino sistemas que sean capaces de conseguir que su rotura implique recursos y unidades de tiempo superiores al valor y al tiempo de caducidad de la información de nuestra red. Además nos vamos a basar en los sistemas más extendidos actualmente, por lo que tenemos garantizados aspectos como su evolución, al mismo tiempo que se desarrollen los sistemas de ataque. Solo hay que mantener el sistema con las actualizaciones de seguridad vigentes para, garantizar un nivel de seguridad alto.

Deberemos hacer un diseño abierto y fácil de actualizar, que nos permita cambiar claves y sistemas de codificación, de forma razonablemente sencilla y adaptarnos a las evoluciones del mercado. Para conseguir esto último, hemos desarrollado un programa de configuración de nuestro sistema, con el que gestionar todos los aspectos de seguridad que permiten mantener actualizado el sistema.

1.3.- Organización de la memoria.

El PFC presenta la siguiente estructura:

1.- Introducción. En este primer capítulo explicamos las características del PFC. En concreto, se identifica una necesidad en seguridad y control de acceso qué pretendemos cubrir con nuestro PFC. Se establece un ámbito tecnológico que nos indica las soluciones que actualmente se adoptan frente a este tipo de problemas y, del conjunto de soluciones se escogen una serie de recursos para utilizar en nuestra implementación. Por último se describe la estructura del documento que conforma el PFC.

2.- Introducción a las herramientas de desarrollo. Este capítulo está dedicado a realizar una pequeña introducción a los recursos que pretendemos emplear en el desarrollo del PFC. En concreto comentamos el tipo de lenguaje de programación que vamos a emplear y hacemos una pequeña descripción de la evolución de los sistemas criptográficos hasta la actualidad, centrándonos en los dispositivos de tarjeta inteligente. Dentro de estos dispositivos podemos encontrar el DNIe. Por último se comenta una serie de pautas para la elaboración de aplicaciones de seguridad.

3.- Especificaciones y Requisitos. En esta sección desarrollamos todas las características que pretendemos que cumpla nuestro control de acceso. Se definen tanto las opciones de uso, su funcionamiento y características en el ámbito funcional y de seguridad, insistiendo en que el sistema sólo debe realizar dichas acciones y quedando prohibida cualquier otra respuesta fuera de estas especificaciones.

4.- Diseño. Este capítulo contiene todos los aspectos que definen nuestra solución, es decir, tanto la descripción de los modos de funcionamiento de las distintas partes y elementos del portal de acceso, como, la definición de los distintos módulos del conjunto de utilidades que conforman la solución.

5.- Desarrollo. En este capítulo se exponen las herramientas finales de implementación de la solución, se explica el ámbito de funcionamiento y su configuración, se describe el portal y su codificación y por último se comenta el completo funcionamiento del portal.

6.- Estudio Económico. Esta sección incluye todo el contexto económico, y los gastos generados por la elaboración del PFC.

7.- Conclusiones. Aquí podemos ver las impresiones que ha tenido el autor al elaborar este PFC, así como el ciclo de vida esperado y posibles progresiones futuras.

- Además al final del documento se adjuntan una serie de anexos como apoyo para el PFC.

2.- Introducción a las herramientas de desarrollo.

Aunque el servicio de cifrado podría realizarse a nivel de red, para conseguir los servicios de autenticación y firma debemos trabajar a nivel de aplicación. No nos hace falta crear un programa que haga de intérprete entre nuestro proceso y el proceso remoto, sino que nos apoyaremos en las plataformas ya existentes. Como la principal plataforma de comunicación de datos hoy en día es internet, podemos utilizar las capacidades de seguridad establecidas en world wide web (WWW) para aplicarlas a nuestro sistema. Solo tendremos que crear un portal de acceso web con funcionalidades de seguridad que se encargue de controlar el acceso. Este agente lo vamos a escribir con un lenguaje de programación de alto nivel denominado C Sharp (a partir de ahora C#), que es una de las últimas creaciones de Microsoft para tecnologías .NET. Como editor utilizaremos el propio entorno de desarrollo de Microsoft, Visual Studio 2008.

Además contaremos con el DNIe sistema para identificación de personas que desarrolla, gestiona e implementa el gobierno de España. Este recurso se encuentra entre los mejores sistemas de seguridad de la actualidad y cuenta con muchos medios y recursos para garantizar un sistema seguro.

Combinando estos dos medios y aplicando una serie de reglas básicas para la creación de aplicaciones criptográficas podremos desarrollar un sistema de comunicaciones cliente (usuario) y servidor (sistema de datos) seguro.

2.1.- C#.

C# es un lenguaje de programación de alto nivel orientado a objetos, que Microsoft ha diseñado dentro de su plataforma .NET. La página de referencia del fabricante es:

http://msdn.microsoft.com/es-es/vcsharp/default.aspx

Se ha denominado framework .NET a un conjunto de recursos (aplicaciones y bibliotecas) que Microsoft está desarrollando y que se pueden utilizar para crear cualquier aplicación, desde sistemas operativos, aplicaciones tradicionales, de ventana, para sistemas operativos portátiles, o de consola, etc.

Microsoft lo define como:

.NET Framework es un componente integral de Windows que admite la compilación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- *Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.*
- *Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.*

- *Ofrecer un entorno de ejecución de código que promueva la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.*
- *Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan scripts o intérpretes de comandos.*
- *Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.*
- *Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.*

La utilización de este lenguaje de programación es uno de los requisitos que nos viene impuesto en las especificaciones iniciales de este PFC.

2.2.- Sistemas Criptográficos.

La **criptografía** es la ciencia que se dedica al estudio de los medios para el cifrado y descifrado de mensajes. Como sitios de referencia para obtener información básica tenemos:

http://es.wikipedia.org/wiki/Criptograf%C3%ADa

http://www.kriptopolis.org/

En principio, como podemos ver en la Figura 1, se trata de transformar un texto normal mediante una función parametrizada con una clave, obteniendo como salida una texto cifrado.

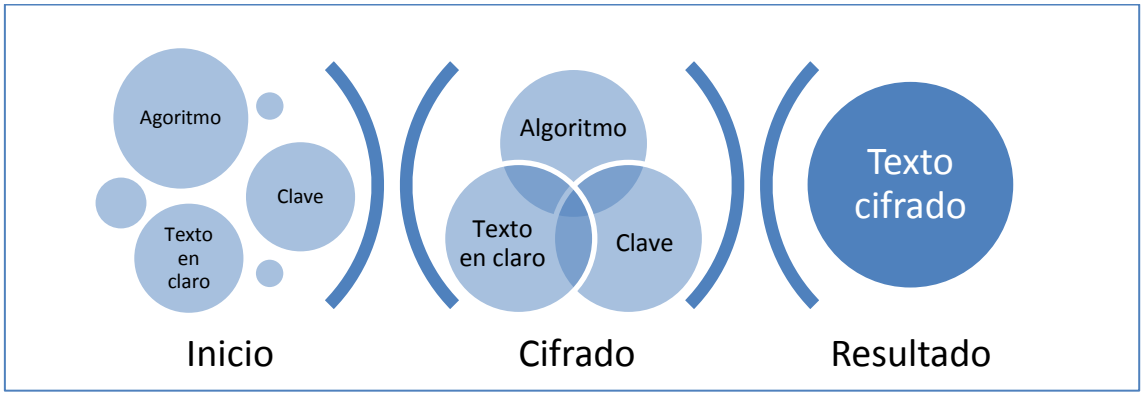

Figura 1. Método genérico de cifrado simétrico.

Generalmente la función de parametrización es conocida y está ampliamente extendida y comprobada. A este campo se le dedican grandes recursos y es bastante costoso. La criptología generalmente está desarrollada por matemáticos que trabajan para innovar y crear algoritmos de cifrado y, sobre todo probar y descifrar (criptoanálisis).

Por otro lado tenemos la clave. Consiste en una cadena corta que actúa sobre la función principal de cifrado. Con ella seleccionamos uno de los cifrados sobre muchas posibles. Podemos cambiar la clave las veces que necesitemos para garantizar la seguridad de nuestro mensaje.

Mientras que el algoritmo general de cifrado está expuesto públicamente, a fin de ser atacado por todo tipo de personas que quieran ponerlo a prueba (con cinco años de resistencia se

consideraría bastante robusto); la clave permanecerá secreta y será la llave que utilicemos aplicada a nuestro algoritmo para cifrar nuestros mensajes. Esta llave puede ser cambiada cuando queramos y su gestión debe de ser cuidadosa. Según la aplicación y el nivel de seguridad que necesitemos, seleccionaremos la longitud de la clave mayor o menor. Una clave corta nos serviría para un ámbito doméstico, ya que aunque proporciona un nivel de seguridad menor también nos implicaría un tiempo de cálculo pequeño; una clave larga nos protegería contra intrusiones más sofisticadas, aunque haciendo uso de mayores recursos de cálculo.

2.2.1.- Sistemas Criptográficos Simétricos.

Es el sistema de cifrado más común y básico. Consiste en utilizar un algoritmo y una única clave que sirve tanto para encriptar como para desencriptar un mensaje. Ha ido evolucionando a lo largo de la historia, igual que el lenguaje humano, hasta lo que hoy se conoce, por lo que existen multitud de algoritmos. Todos estos algoritmos se basan en permutaciones y sustituciones de las letras que componen el mensaje a encriptar. La forma de construir maquinas que realicen estas operaciones es sencilla.

Históricamente destacan por su implantación el sistema DES (1977), sustituido más adelante por el 3DES (1978), el sistema desarrollado por una entidad independiente suiza IDEA (1990) y el sistema AES o Rijndael (2001) nacido de un gran concurso para buscar un algoritmo de cifrado que sustituyera a DES, es de los más seguros y el más utilizado en la actualidad ya que utiliza pocos recursos y es muy rápido de calcular.

Las dos ideas básicas para controlar que no se produzcan intrusiones serían:

- Introducción de redundancia en el mensaje. Por ejemplo es más difícil violar una codificación en la que uno de los campos del mensaje fueran del tipo “cliente número 10000963” que si el campo lo definiéramos como “cliente número 963”. Evidentemente tampoco se debe de introducir redundancia solo de ceros y unos, es preferible, introducir datos aleatorios.
- Marcas de tiempo que impidan el uso de mensajes válidos por parte de enemigos que intercepten nuestros mensajes y luego quieran emplearlos.

2.2.2.- Sistemas Criptográficos Asimétricos.

Nos basamos en pares de claves (una clave pública y otra clave privada) asociadas de tal forma que si una codifica el mensaje solo sirve la otra para descodificar dicho mensaje, y viceversa. La clave pública se utiliza para codificar mensajes que solo queremos que sean leídos por el usuario que tenga la clave privada (y secreta) que corresponde a esa clave pública. De esta manera conseguimos confidencialidad, ya que nos aseguramos de que solo el poseedor de la clave privada puede leer el mensaje. Para conseguir autenticación de la identidad origen deberemos utilizar la clave privada, cualquiera que emplee la clave pública del usuario origen sabrá que el mensaje solo puede venir de él. Simplificando podríamos representarlo con una notación matemática como muestra la Figura 2, donde, C sería el mensaje cifrado que obtendríamos al aplicar la clave secreta del usuario A (Ks_A) sobre el mensaje m, que siempre podríamos recuperar aplicando la clave pública del usuario A (Kp_A)al mensaje cifrado C. También, podríamos realizar el proceso inverso cifrando m con la clave pública del individuo A y descifrando con su clave secreta.

$$C = Ks_A(m) \rightarrow m = Kp_A(C)$$

$$C = Kp_A(m) \rightarrow m = Ks_A(C)$$

Figura 2. Elementos básicos de cifrado asimétrico.

Aunque parece un buen sistema, tiene un coste temporal caro. Es decir, el tiempo de procesado de la información es muy alto lo que limita su uso y en la actualidad obliga a utilizar sistemas mixtos de criptografía asimétrica (para intercambiar unas claves de sesión), con criptografía simétrica.

2.2.2.1.- Algoritmo RSA.

El algoritmo RSA es uno de los más conocidos y utilizados. Sus iniciales provienen de sus diseñadores (Rivest, Shamir, Adleman) que en 1978. Se basa en ciertas propiedades de la teoría de los números, y consiste a grandes rasgos en escoger dos números primos elevados, a los que denominamos p y q ($>10^{100}$). Luego se calcularían dos números n=p * q y z=(p-1)*(q-1). Después de esto, seleccionamos un número primo con respecto a z, llamándolo d y encontramos e tal que e * d = 1 mod z. Con estos parámetros calculados de antemano, podríamos realizar el cifrado. Para ello dividimos la cadena de bits que forman el mensaje a cifrar P en bloques de k bits donde debe cumplirse que 2^k<n.

Para cifrar el mensaje P, calculamos $C=P^e \pmod n$. Para descifrar C, calculamos $P=C^d \pmod n$. Puede demostrarse que para todos los P del intervalo especificado las funciones de cifrado y descifrado son inversas. Para el cifrado necesitaremos el par (e, n) que llamaremos clave pública y para el descifrado, utilizaremos (d, n) que es lo que llamamos clave privada.

La seguridad del método se basa en la dificultad de factorizar números grandes. Los matemáticos llevan más de 300 años tratando de resolverlo.

2.2.2.2-Otros Algoritmos.

Podemos encontrar otros algoritmos de clave secreta basados en la dificultad para calcular logaritmos discretos. El Gamal (1985) y Schgnorr (1991) son ejemplos de algoritmos basados en estos principios. Neal Koblitz y Victor Miller en 1985

Además también existen otros algoritmos basados en curvas elípticas propuestos por Neal Koblitz y Victor Miller (1985), y tratados posteriormente por Menezes y Vanstone (1993).

2.2.3.- Protocolo de Validación de identificación.

El método que generalmente se usa para establecer canales de comunicación seguros combina el uso de claves simétricas y asimétricas. El inicio de la sesión consistiría en la verificación de los interlocutores mediante intercambio de mensajes con sus correspondientes claves públicas y su lectura con las claves privadas, estableciendo en este proceso una clave de sesión que serviría para la comunicación posterior. En La Figura 3 podemos ver como el proceso A envía su identidad (A) y un número al azar (R_A) cifrado con la clave pública del proceso B (E_B). El proceso B que, todavía no confía en el proceso A, le devuelve al proceso A un mensaje con Ra, su propio número al azar (R_B), y una clave de sesión (K_S), todo ello codificado con la clave pública del proceso A. Cuando el proceso A abre el mensaje con su clave privada tiene la seguridad de que está hablando con el proceso B y le devuelve el número reto (R_B) al

proceso b, pero, utilizando la clave de sesión K_s. En el momento en que el proceso B recibe este último mensaje reconoce que está tratando con el proceso A, iniciando la comunicación utilizando la clave de sesión K_s.

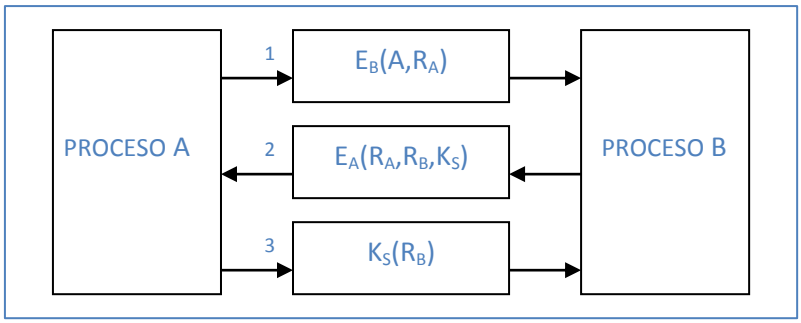

Figura 3 Validación de identidad mutua.

Otro modo de realizar la validación de una forma alternativa sería la variante para la identificación que resumimos en la Figura 4.

En un primer mensaje, además de establecer la identidad del proceso A, le damos al proceso B una clave de sesión para iniciar la comunicación, que sirve a su vez de reto hacia el proceso B.

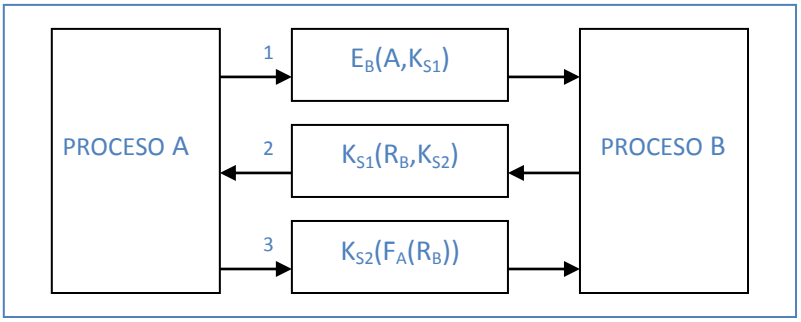

Figura 4. Validación de identidad mutua alternativa.

Que el proceso B pueda utilizar la clave de sesión K_{S1} nos confirma la identidad del proceso B. Además nos propone un reto y una clave de sesión definitiva. Nuestra identidad quedará validada para el proceso B cuando le devolvemos su reto firmado digitalmente por nuestro DNIe. Esta firma y el estado del DNIe puede ser comprobada por el proceso B en un servidor de la Dirección general de Policía, que es la autoridad que emite el certificado DNIe que estamos utilizando para comprobar al usuario.

2.3.- Infraestructura de clave pública.

La infraestructura de clave pública (PKI) encuadra a todos los medios: hardware, software, las políticas y los procedimientos que intervienen en la seguridad en un ámbito de transacciones electrónicas. Para ello se crea una figura independiente de confianza denominada **entidad o autoridad de certificación (AC)**, que se encarga certificar la seguridad de la comunicación. Esta entidad de certificación crea un **certificado electrónico** único para cada usuario, con el que podrá acreditar su identidad.

2.3.1. Autoridades de certificación.

Una autoridad de certificación (AC) es una entidad de confianza que, basándose en los estándares internacionales publicados por el Internet Engineering Task Force (IETF), el European Telecommunications Standards Institute (ETSI) y el Comité Europeo de Normalización (CEN), establece unas políticas de seguridad y funcionamiento (políticas de autorregulación) que deben ser aceptadas por los usuarios de sus servicios. Esta entidad se encarga además de realizar las funciones de expedición de certificados, verificación de identidad, información de las formas de uso y validez de los certificados y, gestión de listas de certificados revocados. Para facilitar la gestión de las funciones y servicios de los certificados, como puede verse en la Figura 5, se pueden crear las siguientes autoridades:

- **Las Autoridades de Registro (AR).** Las autoridades de registro se encargan de la expedición, revocación y trámites de los certificados electrónicos y los usuarios.
- **Las Autoridades de Validación (AV).** Las autoridades de validación se encargan de gestionar los medios para la validación de todos los certificados electrónicos por parte de cualquier usuario.

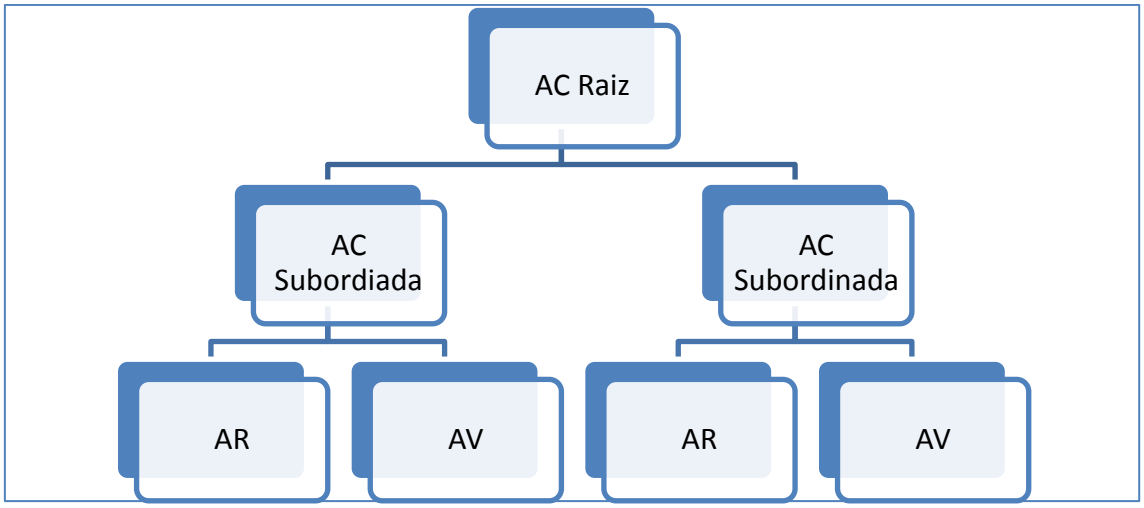

Figura 5. Modelo de jerarquía de la infraestructura de clave pública.

Existen medios que sirven para relacionar entidades de certificación. De esta manera se puedan establecer niveles de confianza entre certificados. Para ello las entidades de certificación establecen entre ellas unos mecanismos de reconocimiento mutuo como pueden ser la certificación cruzada, AC puente (Bridge CA), las listas de certificados de confianza, etc.

2.3.2. Certificados Electrónicos.

La AC provee a cada usuario de un documento electrónico en el que figuran los datos de identificación y la clave pública del usuario codificados con la clave privada de la propia AC. De esta forma todo usuario que dispone de a la clave pública de la AC, puede extraer los datos de este certificado. Por otro lado, nadie puede falsificar certificados al no disponer de la clave privada de la AC.

Existe gran cantidad de tipos de certificados electrónicos, que dependen fundamentalmente de la aplicación que vamos a darles:

- **Certificados personales.** Permiten la identificación de un usuario.
- **Certificados web o SSL.** Permiten el establecimiento de una conexión SSL con un servidor.
- **Certificados de firma de código.** Sirven para establecer un nivel de confianza respecto al origen del código que estamos manejando con el fin de descartar código malicioso.
- **Certificados de componentes.** Se emplean en el establecimiento de canales de comunicación seguros y confiables con dispositivos criptográficos.
- **Certificados de sello de tiempo.** Establecen una marca de tiempo para certificar el momento en que se realizó determinada acción. Incluyen una identificación de la TSU(Time Stamping Unit) o a la TSA (Time Stamping Authority) que firma la marca temporal utilizada.
- Etc.

Para indicar el tipo de uso de un certificado electrónico, se utilizan las extensiones "key usage" y "extended key usage" del certificado. Son accesibles por los usuarios del certificado para indicarles qué acciones están permitidas con el certificado electrónico que manejan.

2.3.2.1. Certificados X509.

Aunque existen muchos estándares que definen certificados electrónicos, nosotros nos vamos a centrar en el **estándar X509**, que es el utilizado por la International Telecommunication Union – "Telecommunication Standardization Sector" (ITU-T) y por ISO/International Electrotechnical Comision (IEC). Es una derivación del estándar X.500 que establece la sintaxis y componentes que debe seguir un certificado electrónico. La primera versión es de 1988 y actualmente se utiliza la versión X509v3 (1996). La estructura básica de un certificado del tipo X509 es:

- Certificado.
  - Versión.
  - Número de serie.
  - ID del algoritmo.
  - Emisor.
  - Validez.
    - No antes de.
    - No después de.
  - Sujeto.
  - Información de clave pública del sujeto.
    - Algoritmo de clave pública.
    - Clave pública del sujeto.
  - Identificador único de emisor (opcional).
  - Identificador único de sujeto (opcional).
  - Extensiones (opcional).
- Algoritmo usado para firmar el certificado.
- Firma digital del certificado.

El estándar X509v3 es el formato elegido en el DNIe para los certificados electrónicos de usuario que contiene. En concreto podemos ver en la Tabla 1 contenido del certificado de autenticación de usuario incluido en el DNIe.

Tabla 1. Descripción del contenido del certificado electrónico de autenticación del 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. ubjectUniqueIdentifier	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	
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	

2.3.2.2. Comprobación de Certificados mediante OCSP.

Aunque DNIe soporta tanto CRLs(Certificate Revocation List) como respuestas OCSP(Online Certificate Status Protocol), estas últimas tienen numerosas ventajas sobre las listas de revocación (actualización, mantenimiento externo, etc.). Nosotros vamos a utilizar OCSP como método de comprobación del estado de validez del DNIe.

El protocolo OCSP se describe en la norma RFC 2560. Sirve para comprobar el estado de revocación de un certificado electrónico. Consiste en una serie de mensajes codificados en ANS.1 intercambiados entre el usuario que realiza la comprobación y una AV (en el caso del DNIe http://ocsp.dnie.es). En la petición del usuario se incluye la información del certificado que se quiere validar. La respuesta de la AV es un certificado firmado en el que encontramos el estado de validez de nuestro certificado ("bueno" (good), "revocado" (revoked) o "desconocido" (unknown). En caso de algún fallo de comunicación o formato de petición nos devolverá un mensaje de error.

2.3.3. Normas de regulación.

Un resumen de las normas que intervienen en la infraestructura de clave pública en España podría ser el esquema de la Figura 6. Normativa sobre PKI en España., en el que vemos que existe una legislación propia respaldada por una legislación a nivel europeo y una normalización a nivel internacional. El tercer bloque sería los documentos de autorregulación que genera cada PKI.

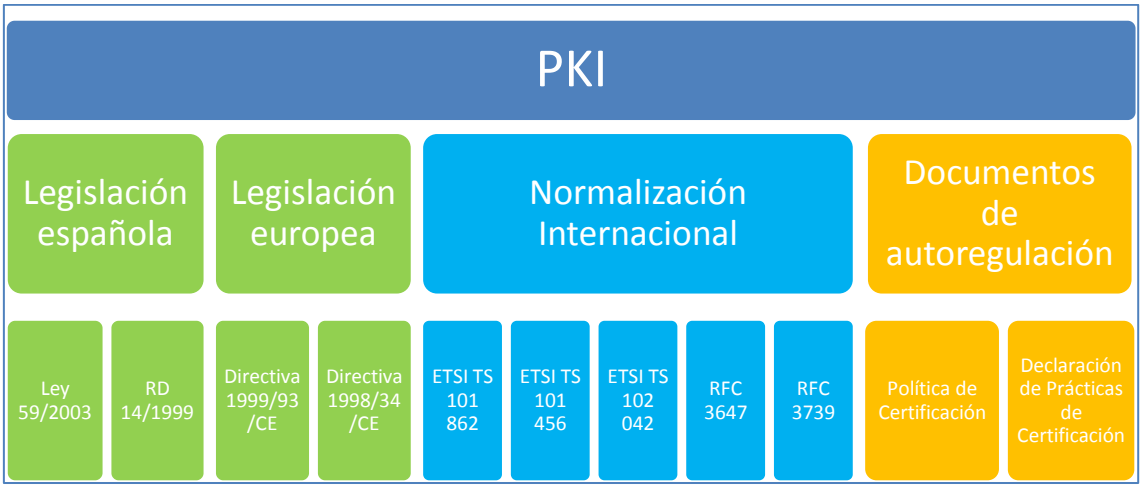

Figura 6. Normativa sobre PKI en España.

2.3.3.1. Legislación.

El estado español inicialmente publicó el Real Decreto 14/1999 de 17 de septiembre, en el que se trataba la firma electrónica en respuesta a la Directiva 98/34/CE del Parlamento Europeo. Este documento ha sido ampliado con la Ley 59/2003 de 19 de diciembre, emitido a su vez como respuesta a la Directiva 99/93/CE.

La ley 59/2003 tiene entre otras, las siguientes características:

- Establece un marco jurídico para desarrollar el DNIe.
- Regula la emisión de certificados de personas jurídicas, es decir, permite el uso de firma electrónica a las empresas.
- Regula la liberalización de los servicios de certificación. Esto permitirá impulsar los sistemas PKI privados que podrán coexistir con los de ámbito estatal.

2.3.3.2. Organismos de normalización.

Existen varios grupos que se dedican al tema de regular y normalizar PKI y en concreto a todos los aspectos referentes a su aplicación en la firma digital. Los principales grupos de regulación son:

- **CEN:** El Comité Europeo de Normalización, es un organismo privado no lucrativo que se encarga de fomentar y mejorar la economía, el bienestar y el medio ambiente en el ámbito europeo por medio de la creación de infraestructuras para la creación, estudio y mantenimiento de estándares y especificaciones.
- **ETSI**: Instituto Europeo de normas de Telecomunicaciones, es una organización de normalización europea de la industria de las telecomunicaciones creada 1988 por la Conferencia Europea de Administraciones de Correos y Telecomunicaciones (CEPT).
- **IETF**: Grupo de Trabajo en Ingeniería de Internet, creada en estados unidos en 1986, es una organización internacional abierta de normalización de tecnologías de internet. Existe un grupo de trabajo (PKIX) que es el encargado de la elaboración de la documentación referente a la infraestructura de clave pública.
- **OASIS:** Organización para el Avance de las Normas de Información Estructurada, consorcio sin ánimo de lucro, que busca el establecimiento de acuerdos de desarrollo, convergencia y adopción de estándares abiertos aplicados al comercio electrónico y los servicios web.
- **RSA:** existe una serie de especificaciones denominadas PKCS# (Public Key Cryptography Standards) establecidas por la empresa RSA para el uso del algoritmo RSA, que son reguladas por IETF para aplicarse en los mecanismos de firma digital.

2.3.3.3. Documentos de autorregulación.

Estos documentos son redactados por los prestadores de servicios PKI para la regulación de sus actividades y servicios, con lo que establece el entorno en que se basa el nivel de confianza para sus usuarios.

Existen principalmente dos tipos de documentos utilizados, que son las políticas de certificación y la declaración de prácticas de certificación o CPS (Certificate Practise Statements).

La **Política de Certificación** es el conjunto de normas y reglas que definen el ámbito de aplicación de los distintos tipos de certificados disponibles, indicando los requisitos de seguridad y las formas de utilización.

La **Declaración de Prácticas de Certificación** es el conjunto de procedimientos que la AC debe seguir en la emisión, seguridad, gestión, soporte, etc., de los certificados.

2.4.- Tarjetas inteligentes.

Se definen como tarjetas que contienen un circuito integrado embebido con capacidad para ejecutar lógica programada. Genéricamente podemos distinguir entre tarjetas con circuito de memoria no volátil y tarjetas con memoria y microcontrolador. Dentro de estas últimas existe un conjunto de tarjetas con capacidades criptográficas que incorporan un hardware y un software diseñados para contener certificados electrónicos e implementar funcionalidades y una seguridad de acceso específicos.

La comunicación con estos dispositivos puede hacerse con un lector de contacto físico o con algún interface remoto, vía radio frecuencia (RF).

2.4.1.- Estándares de tarjetas inteligentes.

Existen muchos aspectos que definen a una tarjeta inteligente (tamaño, grosor, materiales, tipo de circuito integrado, posición de los contactos, etc.). Algunos estándares y normas que regulan las características de las tarjetas inteligentes se resumen en los siguientes apartados:

1.- ISO 7810. Es el estándar que define los tamaños para tarjetas de identidad o identificación.

- **ID-1**. El formato ID-1 especifica un tamaño de 85,6x53,98 mm. Es el formato más utilizado. Se utiliza comúnmente en tarjetas bancarias (tarjetas de crédito, tarjetas de débito, de fidelización, etc.), así como en permisos de conducir en determinados países.
- **ID-2**. El formato ID-2 especifica un tamaño de 105x74 mm. Dicho tamaño es el formato A7. Se utiliza en algunos documentos de identidad.
- **ID-3**. El formato ID-3 especifica un tamaño de 125x88 mm, que corresponde al formato B7. Se utiliza en pasaportes.

2.- ISO 7811. Tiene nueve partes donde se describen las técnicas de grabación en la banda magnética de tarjetas de identificación. Así mismo, se tratan las marcas de identificación táctiles de las tarjetas para personas con discapacidades visuales.

3.- ISO/IEC 7816. Lo componen 15 partes:

- **Partes 1, 2 y 3**. Tratan solo las tarjetas de contacto. En estos apartados se describe los distintos aspectos de su interface y aspecto (dimensiones de la tarjeta, interface eléctrico y protocolo de comunicaciones).
- **Partes 4, 5, 6, 8, 9, 11, 13 y 15.** Se aplicado a todo tipo de tarjetas, describe la estructura lógica, varios comandos utilizados por el interface de programación de aplicaciones de uso básico, gestión de aplicaciones, servicios criptográficos, …
- **Parte 7.** Encontramos la descripción de SCQL (Structured Card Query Language) es decir un leguaje de base de datos seguro basado en SQL (Structured Query Language).

- **Parte 10 y 12.** Está dedicada a las señales eléctricas intercambiadas en las comunicaciones y procedimientos de conexión y desconexión.

4.- CEN 726. Este estándar de ETSI define un conjunto de comandos (APDUs), una estructura de ficheros, requisitos de acceso y, métodos de conexión entre los terminales. Consta de 7 partes:

- **Parte 1.** Supervisión del sistema.
- **Parte 2.** Marco de trabajo de seguridad.
- **Parte 3.** Requisitos de tarjeta independientes de aplicación.
- **Parte 4**. Requisitos de terminal relativo a tarjetas independientes de aplicación.
- **Parte 5.** Métodos de pago.
- **Parte 6.** Características de telecomunicaciones.
- **Parte 7.** Módulo de seguridad.

El CEN726 concuerda en gran parte con el estándar 7816-4.

5.- EMV. Es un estándar publicado en 1996 por **E**uropay International S.A., **M**asterCard International Incorporated, y **V**ISA International Service Association, también denominado, "Especificaciones de Tarjetas IC para Métodos de Pago". Consta de tres libros en los que se especifican los aspectos de diseño y funcionamiento de terminales y tarjetas inteligentes para aplicaciones de crédito/débito.

6.- PC/SC. Es un grupo de trabajo que está actualmente a cargo de Microsoft Corp., que se encarga de mantener unos estándares para las tarjetas inteligentes en la plataforma de ordenadores personales. Trabajan sobre tres aspectos principales:

- InterFaceDevice (IFD). El interfaz de terminales de tarjeta hacia el PC.
- Application Programming Interface (API). El interfaz de programación de aplicaciones de alto nivel para acceder a las funcionalidades de la tarjeta inteligente.
- Los mecanismos para un acceso múltiple de varias aplicaciones hacia una tarjeta inteligente.

La especificación PS/SC está basada en el estándar ISO/IEC 7816 y soporta estándares de aplicación EMV y GSM.

7.- OPENCARD. Es un consorcio de doce empresas del sector, que se unieron para sacar una nueva especificación de la tecnología para ayudar a programadores a desarrollar soluciones informáticas. Consiste es un conjunto de APIs y herramientas de desarrollo basadas en Java, que permiten interactuar con gran variedad de lectores y tarjetas, incluidos los compatibles con PS/SC.

8.- PKCS #11 (CRYPTOKI). Dentro de los estándares que RSA Laboratories ha desarrollado para la aplicación de los algoritmos de cifrado de clave público esta cryptoki (Cryptographic Token Interfaz) que es un interface de programación en el que la tarjeta inteligente se encapsula en un modelo denominado token criptográfico al que se pueden conectar varias aplicaciones accediendo a un slot o punto de conexión.

9.- PKCS#15. Define el almacenaje y el formato de certificados electrónicos, claves, etc., en tarjetas inteligentes.

2.5.- DNIe.

Es un tipo de tarjeta inteligente emitido por la dirección general de la policía para que cada ciudadano acredite su identidad. La página oficial de DNIe es:

http://www.DNIe.es/

Podemos encontrar un portal para desarrolladores de software denominado ZonaTic:

https://zonatic.usatudni.es/

Aquí el soporte DNIe procura suministrar cualquier información útil para facilitar dicho desarrollo, como cursos, drivers, código fuente, avances, etc.

La tarjeta inteligente DNIe tiene dos partes:

- Por un lado está la parte física que consiste en una tarjeta en la que figuran los datos identificativos del individuo, datos biométricos (firma manuscrita, huella dactilar, fotografía), y un chip ST19WL34 que contiene y gestiona los datos lógicos del individuo.
- Tenemos también una parte lógica (certificados electrónicos), que nos van a permitir realizar funciones de autenticación, utilizada para las comunicaciones por ordenador de manera segura y, el certificado de firma digital, para que los ciudadanos puedan firmar documentos. Esta firma, en determinadas circunstancias, tiene todas las características legales igual que tendría una firma manuscrita.

Asociado a este tipo de tarjeta, el estado tiene la obligación de crear y mantener a disposición pública tanto una lista en la que figuren los DNIe revocados como un medio de consulta online según las normas para consultas OCSP.

2.5.1.- ¿Qué ventajas obtenemos de utilizar el DNIe?

En primer lugar la seguridad de este tipo de tarjeta viene avalada por la dirección general de la policía, la cual se encarga del suministro y gestión de este tipo de documento. En segundo lugar tenemos que es un tipo de documento que todo el mundo tiene (aunque en estos momentos estamos en proceso de conversión de los DNI antiguos por modernos con tarjeta inteligente).

2.5.2.- Descripción.

Se trata de una tarjeta de policarbonato que, contiene los datos impresos del titular, así como, un chip que realizaría todos los trámites electrónicos. Esta tarjeta sigue la norma ISO-7816-1 y está diseñada para soportar 10 años de uso.

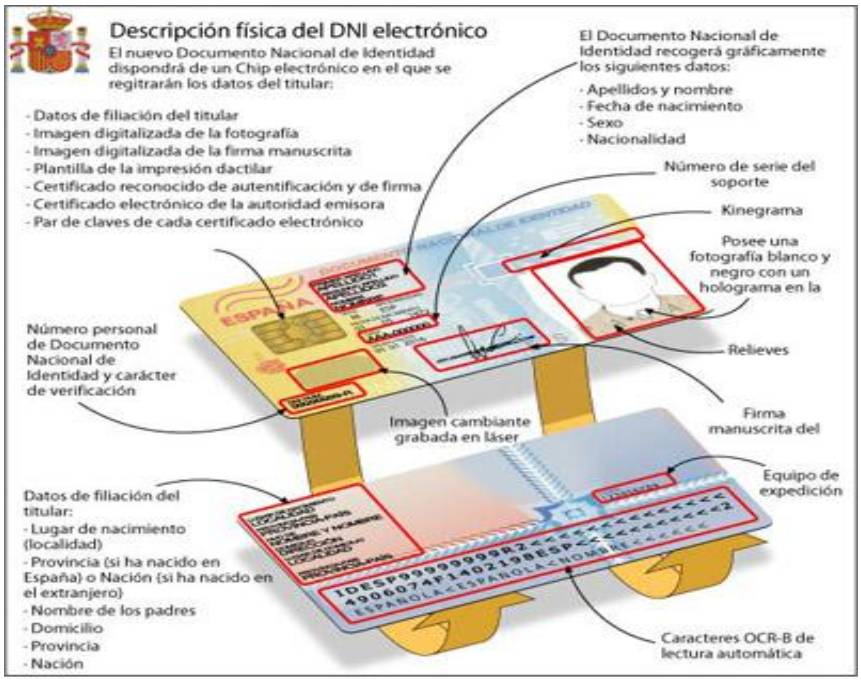

Figura 7. Descripción física del DNIe. (Dirección General de la Policia.).

Podemos ver en la Figura 7 los siguientes datos en la parte delantera de la tarjeta:

- PRIMER APELLIDO
- SEGUNDO APELLIDO
- NOMBRE
- SEXO Y NACIONALIDAD
- FECHA DE NACIMIENTO
- IDESP(Número de serie del soporte físico de la tarjeta)
- VÁLIDO HASTA
- DNI NUM

También en la parte frontal del DNIe tenemos un espacio destinado a la impresión de imagen láser cambiante (CLI) donde figura:

- La fecha de expedición en formato DDMMAA.
- La primera consonante del primer apellido + primera consonante del segundo apellido + primera consonante del nombre (del primer nombre en caso de ser compuesto).

El chip criptográfico que posee y gestiona los siguientes documentos electrónicos:

- Un certificado electrónico para autenticar la personalidad del ciudadano.
- Un certificado electrónico para firmar electrónicamente, con la misma validez jurídica que la firma manuscrita.
- Certificado de la Autoridad de Certificación emisora.
- Claves para su utilización.
- La plantilla biométrica de la impresión dactilar.
- La fotografía digitalizada del ciudadano.
- La imagen digitalizada de la firma manuscrita.

- Datos de la filiación del ciudadano, correspondientes con el contenido personalizado en la tarjeta.

Tenemos los elementos de seguridad del documento, para impedir su falsificación:

- Medidas de seguridad físicas:
  - Visibles a simple vista (tintas ópticamente variables, relieves, fondos de seguridad).
  - Verificables mediante medios ópticos y electrónicos (tintas visibles con luz ultravioleta, micro escrituras).
- Medidas de seguridad digitales:
  - Cifrado de los datos del chip.
  - Acceso a la funcionalidad del DNI electrónico mediante clave personal de acceso (PIN).
  - Las claves nunca abandonan el chip.
  - La Autoridad de Certificación es el la Dirección General de la Policía.

El reverso de la tarjeta contiene los siguientes elementos:

- Información impresa (y visible a simple vista) sobre la identidad del ciudadano en la parte superior:
  - LUGAR DE NACIMIENTO
  - PROVINCIA-PAÍS
  - HIJO DE
  - DOMICILIO
  - LUGAR DE DOMICILIO
  - PROVINCIA-PAÍS y EQUIPO
- Información impresa OCR-B para lectura mecanizada sobre la identidad del ciudadano según normativa OACI para documentos de viaje.

2.5.3.-Certificados electrónicos del DNIe.

El DNIe cuenta con un chip que se encarga de realizar y gestionar todas las operaciones criptológicas necesarias para la identificación de la persona a la que se ha asignado dicho DNIe. De esta manera podemos acreditar electrónicamente nuestra identidad, que es una de las funcionalidades que necesitamos en este PFC.

- Características generales del Chip:
  - Modelo del Chip: st19wl34 y ICC ST19wl34.
  - Sistema operativo: DNIe v1.1.
  - Capacidad: 34Kbytes EEPROM.
- Contenido. La información en el chip está distribuida en tres zonas con diferentes niveles de seguridad y condiciones de acceso:
  - ZONA PÚBLICA: Accesible para lectura sin restricciones, contiene:
    - Certificado CA intermedia emisora.
    - Claves Diffie-Hellman.
    - Certificado X509 de componente.
  - ZONA PRIVADA: Accesible en lectura por el ciudadano, mediante la utilización de la Clave Personal de Acceso o PIN, conteniendo:

        - Certificado de Firma (No Repudio).
        - Certificado de Autenticación (Digital Signature).
    - ZONA DE SEGURIDAD: Accesible en lectura por el ciudadano, en los Puntos de Actualización del DNIe:
        - Datos de filiación del ciudadano (los mismos que están en el soporte físico).
        - Imagen de la fotografía.
        - Imagen de la firma manuscrita.

Los datos con los que cuenta el chip en su memoria son:

- DATOS CRIPTOGRÁFICOS: Claves de ciudadano.
    - Clave RSA pública de autenticación (Digital Signature).
    - Clave RSA pública de no repudio (ContentCommitment).
    - Clave RSA privada de autenticación (Digital Signature).
    - Clave RSA privada de firma (ContentCommitment).
    - Patrón de impresión dactilar.
    - Clave Pública de root CA para certificados card-verificables.
    - Claves Diffie-Hellman.
- DATOS DE GESTIÓN:
    - Traza de fabricación.
    - Número de serie del soporte.

El chip de la tarjeta almacena los siguientes certificados electrónicos:

- Certificado de Componente. Su propósito es la autenticación de la tarjeta del DNIe mediante el protocolo de autenticación mutua definido en CWA 14890.
    - Permite el establecimiento de un canal cifrado y autenticado entre la tarjeta y los Drivers.
    - Este certificado solo estará accesible directamente por los interfaces estándar (PKCS#11 o CSP).
- Certificado de Autenticación. Este certificado es el que utilizaremos para el establecimiento del Canal Seguro con autenticación de Cliente y Servidor.
    - Es un certificado X509v3 estándar, que tiene activo en el Key Usage el bit de Digital Signature (Firma Digital) y que está asociado a un par de claves pública y privada, generadas en el interior del CHIP del DNI.
- Certificado de firma. Este certificado es el que utiliza para la firma de documentos garantizando la integridad del Documento y el No repudio de origen.
    - Es un certificado X509v3 estándar, que tiene activo en el Key Usage el bit de ContentCommitment (No Repudio) y que está asociado a un par de claves pública y privada, generadas en el interior del CHIP del DNI.
    - Es este Certificado expedido como certificado reconocido y creado en un Dispositivo Seguro de Creación de Firma (DSCF), el que convierte la firma electrónica avanzada en firma electrónica reconocida, permitiendo su

equiparación legal con la Firma Manuscrita (Ley 59/2003 y Directiva 1999/93/CE).

2.5.4.- Funcionamiento del DNIe.

Ley 59/2003, de 19 de diciembre, de firma electrónica, regula las funciones y el formato del DNIe. En principio se trata de generar unos certificados de ciudadano según los estándares internacionales para PKI que se desarrollan en las normas:

- ETSI TS 101 862: Qualified Certificate Profile.
- RFC 3739 Internet X.509 Public Key Infrastructure: Qualified Certificates Profile.

Para el buen funcionamiento de estos certificados, la Dirección General de la Policía (Ministerio del Interior) que es el órgano competente de la expedición y gestión del DNIe, crea unas autoridades de gestión:

- **La Autoridad de Aprobación de Políticas (AAP).** Es la encargada de establecer los criterios funcionamiento y las modificaciones que se realicen, establece los términos para interactuar con AC externas, regula los servicios de Validación externos. También se encargada de gestionar y resolver las auditorias que se lleven a cabo.
- **Las Autoridades de Certificación (AC).** Es la encargada de certificar la relación entre los pares de clave y el ciudadano concreto asociado a esas claves. En la Figura 8 podemos ver los siguientes niveles de certificación:

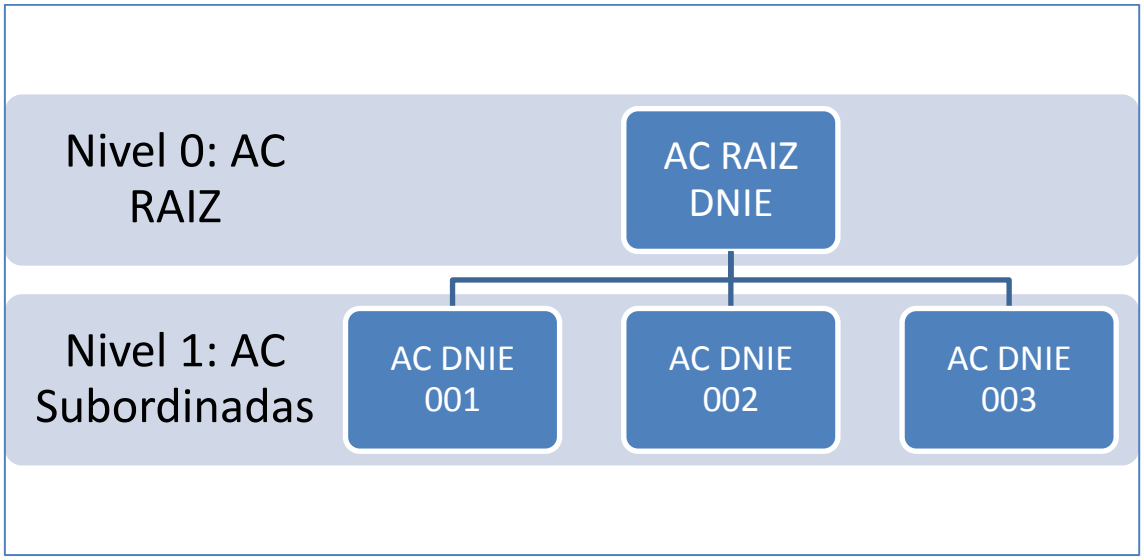

Figura 8. Niveles de las autoridades de certificación del DNIe.

  - **AC Raíz**: Autoridad de Certificación de primer nivel. Esta AC sólo emite certificados para sí misma y sus AC Subordinadas.
  - **AC Subordinadas**: Autoridades de Certificación subordinadas de "AC Raíz". Su función es la emisión de certificados para los titulares de DNIe.
- **Las Autoridades de Registro (AR).** La autoridad de registro está formada por todas las oficinas de expedición del DNIe de la Dirección General de la Policía. Su misión es la de la expedición y tramites del DNIe.

- **Las Autoridades de Validación (AV).** Las Autoridades de Validación (AV) tienen como función la comprobación del estado de los certificados emitidos por DNIe, mediante el protocolo OCSP, que determina el estado actual de un certificado electrónico a solicitud de un tercero aceptante sin requerir el acceso a listas de certificados revocados por éstas.

2.6.- Normas para la creación de aplicaciones criptográficas.

2.6.1.- Nociones básicas.

Empezaremos con una frase típica utilizada en criptografía sería la siguiente. *"Saber poco de criptografía es peor que no saber nada."*

Lo primero para ponernos en antecedentes sería consultar las páginas webs recomendadas:

- Organización para el Avance de las Normas de Información Estructurada (OASIS):

  http://www.oasis-open.org/

- Área de seguridad y comunicación de la Universidad Carlos III de Madrid:

  http://asyc.uc3m.es

Hemos destacado los siguientes puntos como guía para aplicaciones criptográficas:

1.- Los desarrolladores no son criptógrafos, es decir, no inventan sistemas criptográficos. Es mejor aplicar algoritmos conocidos que son sistemas públicos ampliamente implantados y puestos a prueba.
2.- Una vez seleccionado un algoritmo o método de cifrado, tendremos que seleccionar la mayor longitud de clave que se pueda permitir nuestro sistema. Es importante la valoración que se realiza entre tiempos de procesos y seguridad de la información con la que se trabaja.
3.- Hay que realizar comprobaciones de la integridad de los datos. No podemos validar información corrupta o alterada. Se suelen utilizar resúmenes (hash o digest) y marcas de tiempo (timestamping). Se ha revelado una vulnerabilidad en el resumen hash MD5 que hace no recomendable su utilización.
4.- En los cifrados por bloques, hay que utilizar rellenos (padding). Existen algunas soluciones como PKCS5, OAEP, SSL3, etc.
5.- En las comunicaciones hay que comprobar los certificados, en el caso de DNIe disponemos del sistema público de OCSP.
6.- Utilizar números aleatorios de buena calidad.
7.- El certificado del servidor es irremplazable, tenemos que salvaguardar su integridad a toda costa.
8.- Hay que cifrar toda la información sensible en nuestras comunicaciones, y la información que se archiva (no almacenar datos innecesarios).
9.- También se puede proteger el programa, pero recordar que ofuscar no es cifrar.

Todas estas normas son de ámbito general. También utilizaremos en las distintas secciones otras técnicas que hagan de nuestra aplicación un sistema seguro. La descripción de estas características de seguridad se describe en dichas secciones.

2.6.2. Common Criteria.

La mejor forma de crear una aplicación informática es utilizar una serie de reglas formales y una estructura ordenada que permita la evaluación clara e inequívoca del nivel de seguridad aplicable a nuestra solución. Uno de los mejores métodos es el denominado Common Criteria que podemos consultar en:

http://www.commoncriteriaportal.org

Common Criteria regulado en el estándar internacional ISO/IEC 15408, es un certificado de seguridad para aplicaciones informáticas según unos perfiles de protección. Es un estándar reconocido por casi todos los países del mundo entre ellos España (2000), que también sirve para catalogar el nivel de seguridad del DNIe. En la elaboración de un producto informático bajo estos criterios, intervienen tanto el cumplimiento de unos requisitos y metodología de trabajos definidos como la verificación por parte de un laboratorio independiente de certificación, de nuestro producto. En el caso de España en la actualidad operan tres laboratorios con licencia del CCN (Centro Criptológico Nacional):

- APPLUS+. www.applus.com .
- CESTI. www.inta.es/certificaciones.aspx .
- EPOCHE&ESPRI. www.epoche.es .

3.- Especificaciones y Requisitos.

Nuestra aplicación consistirá en un servicio web, lo que permitirá que sea multiplataforma. En concreto vamos a utilizar el servidor de páginas web IIS 7 de Microsoft que está incorporado en sus sistemas operativos. Se podrá conectar desde cualquier sistema operativo ya que el encargado de la comunicación con el DNIe será el navegador web del cliente. El desarrollo estará basado en la tecnología .NET de Microsoft utilizando el lenguaje C# para la programación.

El servidor web deberá mediante la conexión estándar SSL de internet autenticarnos con nuestro DNIe dándonos acceso a una zona segura dentro de nuestro portal web. En esta zona segura se pueden encontrar desde páginas web clásicas con tablas que nos den la información procesada, hasta las últimas tecnologías de portal, donde, se gestionarán para cada usuario una serie de Widgets (agenda, datos personales, proyectos, etc.), de esta manera tendríamos un portal de trabajo en el que podemos configurar a nuestro gusto todos los recursos a los que tenemos acceso. Así mismo podremos desarrollar diferentes Widgets que proporcionarán distintas funcionalidades y accesos. Por ejemplo existirá un Widget de proyectos en el que, para los distintos proyectos que llevemos a cabo en el laboratorio, podremos configurar el acceso a los recursos como: normas, ensayos, usuarios asignados, laboratorios asignados, etc.; lo que haría de nuestra web una herramienta consistente, en constante evolución, a la vez que, multiplicará su funcionalidad e interactividad con el cliente.

Para conseguir una comunicación segura vamos a establecer cuatro puntos críticos que resolver. Estos puntos son:

1.- Establecimiento de canal SSL seguro. En el modelo elegido, la comunicación se establecerá entre el servidor de páginas web IIS 7 que deberemos configurar para funcionar en modo seguro y un navegador de páginas web en el lado del cliente. Así mismo en esta parte podríamos englobar el establecimiento de canal seguro entre DNIe y Navegador que gestionan las librerías que nos suministran la plataforma del DNIe.

El comportamiento como servidor de IIS7 es negociar con el cliente la versión de SSL que se va a establecer en la comunicación, utilizando la versión mayor compatible con el navegador. Podemos intervenir en el registro de Windows para que no se admitan versiones inferiores a SSL v.3 y que podrían considerarse no seguras, esto ocurre en aplicaciones que necesitan un nivel de seguridad máximo, pero no vamos a hacerlo para no limitar el uso de los navegadores ya que nuestra aplicación no se considera crítica. Este bloque se considerará suficientemente seguro ya que es público, actualmente es el más utilizado y continuamente está siendo revisado, comprobado y actualizado en todo el mundo. Para su gestión bastará con incidir en tres puntos:

- Actualizaciones de seguridad de IIS 7.
- Actualizaciones de seguridad del Navegador.
- Actualización de los certificados de validación.

2.- Autenticación en nuestra aplicación. Lo primero que nos va a llegar a través del canal SSL va a ser el certificado de autenticación de nuestro cliente. Debemos ser capaces de analizar este certificado y establecer si es correcto y el usuario al que representa está autorizado a acceder a nuestro portal.

En este segundo bloque vamos a seguir algunas de las directrices establecidas en Common Criteria para asegurar la seguridad. De esta manera utilizaremos un método estructurado de comprobación de la seguridad y podremos llegar a certificar la aplicación.

3.- Administración del Sistema. La administración del sistema se realizará a través de una aplicación en modo local. Además estará dotada de sistemas de protección mediante DNIe, y solo se permitirá su uso por usuarios con rol de administrador. De esta forma se garantiza que la gestión de todos los recursos se realice en un entorno seguro.

4.- Seguridad de la Aplicación en el servidor. Este apartado se refiere a la seguridad y el acceso en la aplicación y en las distintas bases de datos que conforman el proyecto. Hay que evaluar las necesidades de seguridad tanto del contenido (la aplicación y las bases de datos, a las que podemos proteger con cifrado, servidores espejo, etc.) como, del contenedor, es decir, el servidor, la instalación e infraestructuras, la protección del acceso al CPD, etc.

Este PFC se va a centrar en el apartado 2, es decir, en el proceso de autenticación de nuestra aplicación, aunque la seguridad global siempre se apoyará en todos los apartados. Para resolver el apartado 2, crearemos un portal de acceso denominado Server1 que definimos como el objeto de seguridad (TOE) en nuestro informe de Common Criteria. También, tendremos que resolver el apartado 3 con el fin de configurar nuestro sistema para su correcto funcionamiento. Para ello crearemos una aplicación denominada AdministradorServer1.

3.1.- Catalogar el tipo de aplicación.

Lo primero que tenemos que definir según Common Criteria son los perfiles de protección. Con ellos definiremos el esquema y señalaremos las características que definen el diseño de nuestra aplicación.

Se consideran dos tipos de aplicaciones: Tipo1 y Tipo 2. Estos tipos hacen referencia a la plataforma en la que se ejecuta la aplicación. Mientras que las aplicaciones Tipo 1 están dirigidas a sistemas operativos de móviles, PDAs, TDTs, etc., en el Tipo 2 se orienta a sistemas operativos generales.

Nuestra aplicación será en principio de Tipo 2 ya que pretendemos utilizar las funcionalidades de los navegadores ejecutables en sistemas operativos de tipo general. Además, aunque el portal se ejecutará sobre IIS 7 de Microsoft, el lenguaje .NET es compatible con otros servidores de páginas web del mercado como Apache, etc., que se ejecutan también en los sistemas operativos de ámbito general.

3.2.- Elección del nivel de garantía.

Por otro lado elegiremos el nivel de garantía que vamos a exigir a nuestra aplicación. Aunque Common Criteria distingue 7 niveles, las aplicaciones basadas en el DNIe (EAL 4+) suelen estar entre los niveles de seguridad EAL 1 (probado funcionalmente) y EAL 3 (chequeado y probado

con metodología). Cuanto mayor sea el nivel de garantía exigido al proyecto, mayor será el coste de desarrollo de la aplicación.

En este PFC inicialmente utilizaremos EAL 1. Common Criteria nos permite escalar hasta el nivel EAL3 o los niveles intermedios añadiendo algunas características adicionales. En el Anexo A encontramos la versión original de Common Criteria para los requisitos de garantía de seguridad nivel EAL 1.

En resumen tenemos una aplicación con un perfil de protección Tipo 2 y un nivel de garantía EAL 1 **(PP-Tipo2-EAL1)**.

3.3.- Requisitos generales.

En la Tabla 2 se han enumerado todos los requisitos que deberá cumplir nuestra aplicación, es decir, describimos la forma de operar de la aplicación, la cual debe cumplir y solo cumplir estos requisitos.

Tabla 2. Requisitos Generales.

Requisito	Descripción
RG-001	La aplicación nos permitirá acceder a la zona segura al autenticarnos.
RG-002	Se debe establecer un canal seguro SSL cliente-servidor mediante HTTPS.
RG-003	Se debe poder autenticar al cliente mediante DNIe.
RG-004	La aplicación será multiplataforma.
RG-005	Se comprobará la caducidad del certificado cliente.
RG-006	Se comprobará la validez del certificado cliente mediante OCSP.
RG-007	Se comprobará la autenticación del cliente en la base de datos propia.
RG-008	Funcionará solo con el DNIe.
RG-009	La aplicación se desarrollará en C#. Framework 3.5.

3.4.- Requisitos operacionales.

Dentro de la Tabla 3 establecemos los requisitos operacionales de nuestro diseño. Al ser una aplicación ASP solo tenemos que definir la plataforma .NET sobre la que haremos correr nuestro portal web. En este caso hemos decidido que será .NET 3.5 que sirve para todos los sistemas Windows desde Windows XP. El navegador no tendrá ninguna limitación de versión ni sistema operativo.

Tabla 3. Requisitos Operacionales.

Requisito	Descripción
RO-001	Se requiere tener instalado el módulo .NET 2.0 o superior.
RO-002	Se requiere tener instalados el middleware de DNIe.
RO-003	Se requiere tener instalado y configurado IIS.
RO-004	Se necesita conexión a internet para acceder a la AV con el protocolo OCSP.
RO-005	Se requiere navegador compatible con HTTP y XML.

Hemos limitado la aplicación con RO-003 al servidor de servicios IIS 7, ya que es el que se utiliza y describe en este PFC. Este punto podría ser ampliado con otros servidores siempre que se revise y se detallen los métodos de configuración de seguridad en esas plataformas.

3.5.- Requisitos de seguridad.

En este apartado veremos las funcionalidades de seguridad que deberemos seguir en nuestra aplicación para que esta sea certificable. En la Tabla 4 vemos la descripción de los requisitos de seguridad que luego serán desarrollados en la declaración de seguridad para esta aplicación que se encuentra en el anexo B.

Tabla 4. Requisitos de seguridad.

Requisito	Descripción	Requisito PP
RS-001	Se extraerán los datos de usuario (UD) por un canal seguro.	FTP_ITC.1.1
		FTP_ITC.1.2
		FTP_ITC.1.3
		FTP_TPR.1.1
		FTP_TPR.1.2
		FTP_TPR.1.3
RS-002	Se validará el certificado electrónico de usuario: Tipo de certificado (autenticación), fecha y OCSP.	FDP_DAU.2.1
		FDP_DAU.2.2
RS-003	Se validará la identidad del usuario en base de datos.	FDP_DAU.1.1
		FDP_DAU.1.2
		FCS_COP.1.1
RS-004	Se dará acceso a zona segura del servidor web.	FDP_ACC.2.1
		FDP_ACC.2.2
		FDP_ITC.1.1
		FDP_ITC.1.2
		FDP_ITC.1.3
		FIA_UAU.2.1
		FIA_UAU.4.1
		FIA_UID.2.1
RS-005	Se podrá validar el entorno de operación. Revisión de historial de eventos.	FAU_ARP.1.1
		FAU_GEN.1.1
		FAU_GEN.1.2
		FAU_GEN.2.1

En el Anexo B, encontramos la documentación original de Common Criteria con los comentarios que describen cómo se aplica a cada RS los distintos requisitos del perfil de protección.

3.6.- Casos de Uso.

En este PFC el caso de uso está muy limitado y bien definido. Consiste en que un usuario se conecta a nuestra página web con un certificado. La página lo comprueba y le da acceso a una zona segura. Todas las características del acceso se reflejan en la Tabla 5.

Tabla 5. Casos de uso. Acceso a nuestro portal.

Actores	Usuario externo.
Trigger	Usuario intenta acceder a nuestra página de inicio.(Server1)
Descripción	Control de acceso mediante DNIe.
Precondiciones	La configuración del entorno operativo es el adecuado.
	Los certificados de validación están en uso.
	El entorno operativo establece un canal SSL con el usuario externo.

	El entorno operativo solicita un certificado DNIe de acceso.
Postcondiciones	Se obtiene acceso a zona segura.
Flujo normal	El entorno operativo realiza una petición de acceso con un certificado.
	Se comprueba la fecha del certificado de acceso.
	Se comprueba la validez del certificado de acceso vía OCSP.
	Se comprueba la autenticación del certificado en nuestra base de datos.
	Se crea un Ticket para acceder a zona segura.
	Se realiza un registro log del acceso.
Excepciones	El tipo de certificado no es correcto.
	El certificado de usuario está caducado.
	El certificado de usuario es Revocado.
	El certificado de usuario no está registrado en la base de datos.
Tratamiento Excepciones	No se dará acceso a la zona segura.
	Se mostrará información al usuario.
	Se realiza un registro log del intento de acceso.

Así mismo, en la Tabla 6, se describen los casos de uso del sistema administrador de aplicación.

Tabla 6. Casos de uso del Administrador de nuestro portal.

Actores	Usuario interno.
Trigger	Ejecuta la aplicación de configuración. (AdministradorServer1)
Descripción	Administración de recursos del Control de acceso mediante DNIe.
Precondiciones	La configuración del entorno operativo es el adecuado.
Postcondiciones	Se modifican los registros locales del sistema.
Flujo normal	El usuario realiza una petición de acceso con un DNIe.
	Se comprueba la existencia de un certificado electrónico.
	Se comprueba la fecha del certificado de acceso.
	Se comprueba la validez del certificado de acceso vía OCSP.
	Se comprueba la existencia de usuario con nivel de administrador en la base de datos.
	Se comprueba la autenticación del certificado en la base de datos.
	Se realizan las funciones de administración del sistema.
	Se realiza un registro log del acceso.
Excepciones	No existe certificado electrónico.
	El tipo de certificado no es correcto.
	El certificado de usuario está caducado.
	El certificado de usuario es Revocado.
	No existe ningún usuario administrador en la base de datos.
	El certificado de usuario no está registrado en la base de datos.
	Error en alguna función de administración
Tratamiento Excepciones	Se intenta subsanar el problema. (Solo en el caso de ausencia de certificados AV y en el caso de ausencia de administrador).
	No se realizará la operación solicitada.
	Se mostrará información al usuario.
	Se realiza un registro log del intento de acceso.

4.- Diseño.

Utilizaremos la tecnología .NET de Microsoft, ya que nos provee de todas las herramientas necesarias para el desarrollo del proyecto, así mismo, es una de las tecnologías más modernas y utilizadas en la actualidad, lo que, mejora el ciclo de vida del producto, y lo flexibiliza para posibles actualizaciones. Por último, que sea tan extendido y conocido, aumenta la robustez de sus sistemas de cifrado y seguridad. Como lenguaje base usaremos C#, que es la evolución actual de leguaje C. Es uno de los más apreciados y utilizados en el ámbito de la ingeniería. La versión que actualmente está más implantada es la Framework 3.5 aunque, ya está disponible la versión 4. Se ha elegido utilizar v.3.5 para conseguir una mayor estabilidad del proyecto.

4.1.- Diagramas de Flujo.

4.1.1.- Diagrama de Flujo Conexión Remota mediante DNIe.

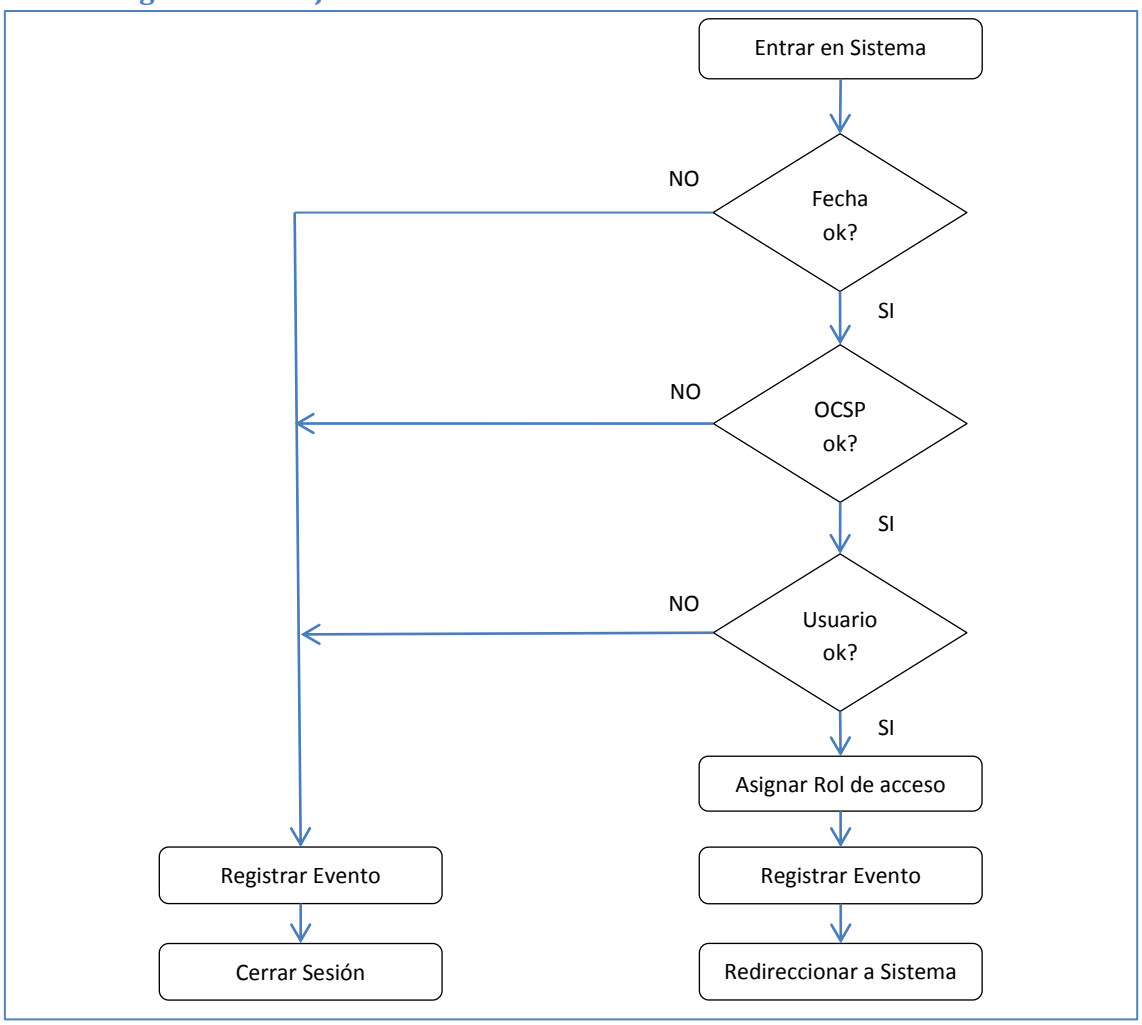

Figura 9. Diagrama de flujo de conexión remota mediante DNIe.

La primera parte que analizamos es el portal de acceso web. Será además el objeto de nuestro TOE de seguridad. El TOE comienza con una petición de la página de acceso desde el exterior.

Esta petición entregará el certificado de autenticación del DNIe del usuario en formato X509v3. Hemos configurado el sistema para que primero se re direccione a una página de Login donde se realizarán las funciones de autenticación del usuario que intenta conectarse. La página de acceso implementa un proceso de autenticación consistente en una verificación de fecha del certificado, una comprobación online OCSP del certificado y una comprobación en una base de datos de gestión propia en la que se registran los usuarios autorizados (en esta comprobación obtenemos también el rol del usuario). Si se cumple con todos las comprobaciones, se registra el acceso, se asignará a la sesión un ticket de acceso y, se redireccionará hacia la página de inicio en la zona segura del nuestra web. En la Figura 9 podemos ver además que, si no se cumple con alguno de los requisitos de acceso, el sistema crea un apunte en nuestro registro sobre el acceso indebido y cierra la sesión.

4.1.2.- Diagrama de Flujo de Administrador del Sistema.

La aplicación de administración del sistema se encarga de gestionar los elementos que necesita el portal para realizar las operaciones de autenticación y comprobación de los certificados DNIe que tenemos que administrar.

Modo de funcionamiento.

Como queda reflejado en la Figura 10, al ejecutar la aplicación se comprueba si disponemos de los certificados de la entidad de emisión del DNIe. Es lo primero que necesita el sistema ya que las comprobaciones del DNIe mediante OCSP requieren de estos certificados. En caso de no estar disponibles, se nos presentará una pantalla para realizar la instalación de estos certificados. Estos archivos están disponibles en www.dnie.es y tienen un periodo de vigencia de 15 años. Luego se lanza una función que nos solicita el DNIe. Con ella accedemos a los certificados electrónicos del DNIe mediante el PIN. En este punto del programa pueden suceder varias condiciones:

- El DNIe no es correcto (esta caducado, está en la lista de certificados revocados, etc.) no se nos permitirá utilizar la aplicación.
- Si el DNIe es correcto pero, no existe ningún administrador en nuestra base de datos (ya sea porque se trata de una nueva base de datos de usuarios y está vacía o, porque hayamos borrado todos los usuarios con rol de administrador por error, etc.) el sistema añadirá a este usuario en la base de datos como administrador.
- Si el DNIe es correcto, ya existen usuarios definidos como administradores y, el usuario actual que realiza el acceso no está en la base de datos como administrador del sistema. En este caso no se podrá acceder a las funcionalidades de la aplicación.
- Si el DNIe es correcto y el usuario está en la base de datos como administrador, se nos dará acceso a los menús de la aplicación ya que cumplimos con los requisitos de autenticación.

Desde este momento podemos acceder al menú de administración de usuarios y al menú de administración de certificados, y podremos realizar las siguientes funciones:

- Gestión de certificados de las AC intermedias del DNIe en nuestro almacén de certificados.
- Gestión de usuarios y roles de nuestra base de datos.

Todo el funcionamiento de la aplicación quedará registrado en un archivo de control denominado trace.txt, con la información de las acciones y los usuarios que las realizan, junto con una marca de tiempo. Este archivo tiene como misión posibilitar una auditoria y un control de eventos.

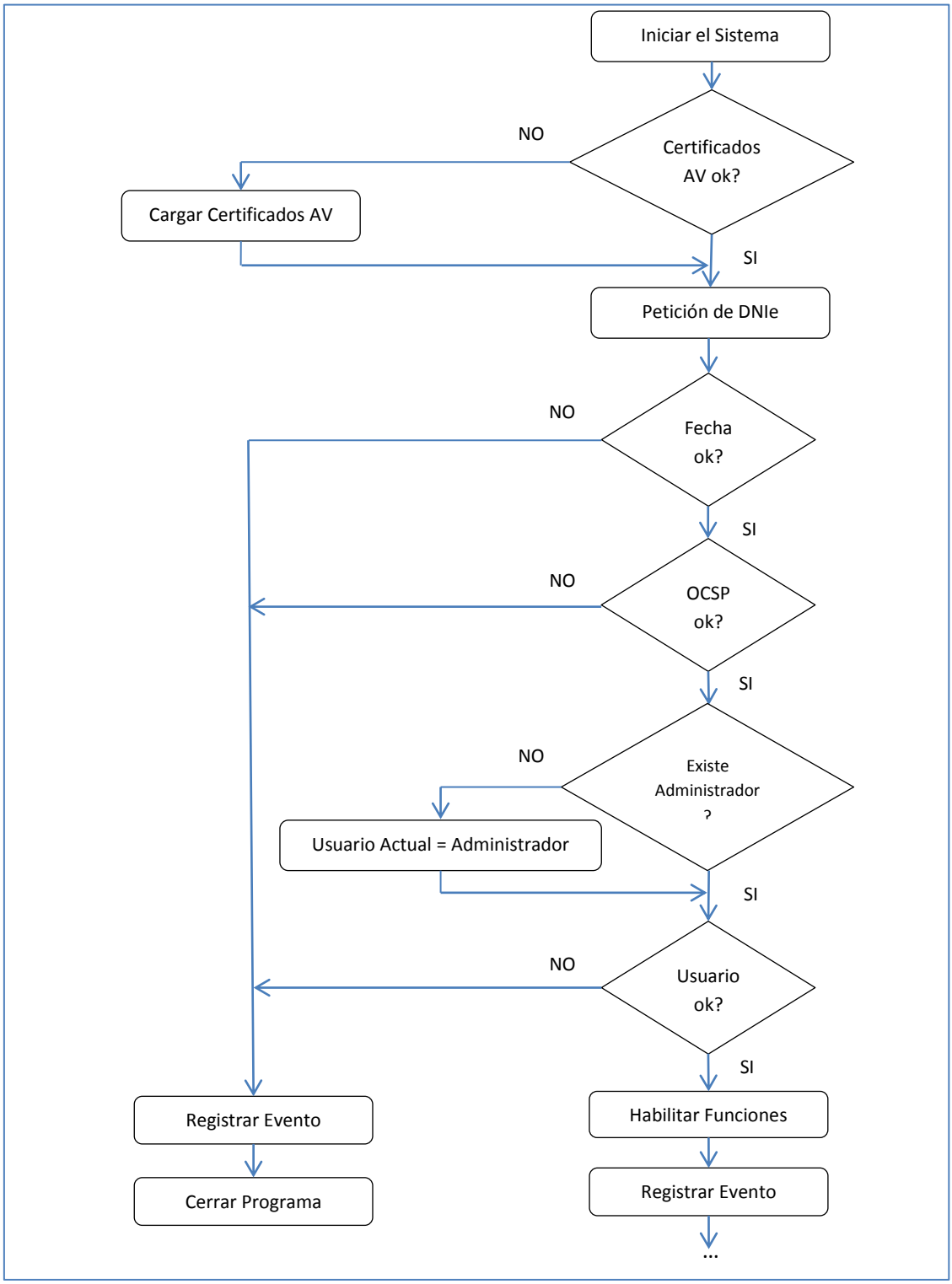

Figura 10. Diagrama de flujo administración del sistema.

4.2.- Definición de Funciones en Server1.

Nuestro portal de acceso está en la carpeta del proyecto Server1. Englobará una serie de páginas web "paginas.aspx", con un archivo *"paginas.aspx.cs"* de código asociado. Todas estas páginas están gestionadas desde el archivo de configuración *"web.config"*. Además se apoyarán en unas clases con distintas funcionalidades (del tipo *"archivos.cs")* y en unas clases definidas en el módulo *"org.bouncycastle"* como utilidades para trabajar con certificados X509. Por último para la función Log, se ha integrado un bloque predefinido de Enterprise Library Configuration de Microsoft denominado *Logging Aplication Block*, que permite la creación de un archivo Log de una manera sencilla.

4.2.1.- Diagrama de Clases en Server1.

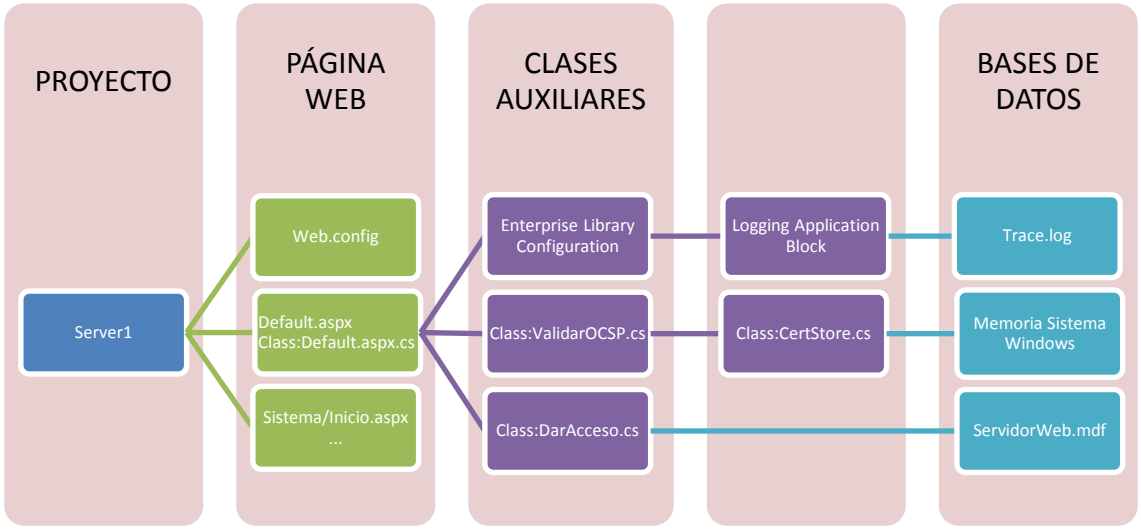

Figura 11. Diagrama de clases en la aplicación Server1.

Podemos comprobar en la Figura 11 que, tenemos una serie de clases todas dependientes de una clase principal denominada *default.aspx.cs* que centralizará toda la aplicación, de esta manera se consigue un sistema más seguro. Cuando el usuario pulsa el botón "E*ntrar en el portal"* se produce una excepción en la página default que comienza invocando la función interna *ComprobarFecha,* para continuar utilizando la clase *ValidarOCSP* y, por último comprueba el usuario con la clase *DarAcceso*. Llegados a este punto genera un rol de acceso y redirecciona al usuario a la página de inicio de la zona segura. En el caso de un error en las comprobaciones el usuario será redireccionado a una página de error. Cada evento y llamada a una clase produce a través del *Logging Application Block* un mensaje que se guarda en el archivo Trace.log. La clase *CertStore* devuelve un certificado guardado en el registro que el sistema operativo Windows tiene para catalogar todos los certificados electrónicos disponibles en nuestra máquina. Esta información podemos gestionarla y actualizarla mediante la aplicación *AdministradorServer1* que comentaremos más adelante. El certificado que se nos devuelve será el correspondiente a un nombre de AC que nos proporciona el certificado de DNIe que estemos utilizando.

Cuando, estamos comprobando el usuario mediante el método *VerificarAcceso* utilizamos una función resumen SHA1 para codificar el usuario y buscarlo en la base de datos. Como índice de

búsqueda utilizamos el número de serie del certificado que será sobre el que se aplica la función resumen. Hemos visto que hay diferencias entre el número de serie que obtenemos en el certificado estándar de la clase *X509Certificate2* y el que nos proporciona la clase *HttpClientCertificate*, así que, tenemos que dar formato a este dato ya que la función resumen distingue entre mayúsculas y minúsculas y que los certificados *HttpClientCertificate* incluyen unos guiones entre los números que tendremos que eliminar.

La base de datos de usuarios se administra desde la aplicación AdministradorServer1, que comentaremos más adelante. En Server1 lo único que se permite es hacer consultas en la base de datos a través del procedimiento de la base de datos StoredProcedure1 que nos sirve de motor para realizar la búsqueda del usuario y recuperar la información en forma de un número que nos indicará el tipo de usuario que está registrado en ella:

- 0 ó -1. Si no está registrado.
- 1. Si es usuario con rol de administrador.
- 2. Si el usuario tiene rol de usuario.

4.2.2.- Definición de Clases y Funciones en la Aplicación Server1.

4.2.2.1.- Web.config.

Es el archivo de configuración de nuestro sitio web y en él se definen varios apartados importantes:

1.- <loggingConfiguration> Corresponde a todo el apartado de Enterprise Library y se introduce y configura de forma automática al invocar esta utilidad.

2.- <connectionStrings> Indica a la aplicación la dirección de enlace con la base de datos.

3.- <system.web> <authentication> Es donde definiremos los aspectos del tipo de autenticación que vamos a emplear. En este apartado tenemos:

- *mode=”Forms”* Indicamos el módulo de autenticación que vamos a utilizar.
- *loginUrl="Default.aspx"* Donde indicamos la página que sirve para comprobar el usuario.
- *defaultUrl="Sistema/Inicio.aspx"* Es nuestra página de inicio donde nos redirigirá si somos usuario válido.

4.- <system.web> <authorization> Donde definiremos los tipos de usuario que tienen acceso a las distintas partes de la web.

- *<deny users="?"/>* Por defecto no dejaremos que acceda ningún usuario sin registrar.

5.- <system.web> <customErrors> Donde se define el tratamiento de los errores.

- *mode="RemoteOnly"* Aquí le indicamos al sistema que realice el plan de tratamiento de errores en accesos remotos, permitiendo que se dé por pantalla toda la información del error en accesos locales y se restrinja en los accesos remotos. Esto nos ayudará en la depuración del programa.

- *defaultRedirect="Error.aspx"* Todos los errores, serán direccionados a una página de error común, de esta manera, limitamos la información a cualquier acceso indebido.

Existen más parámetros que el sistema gestiona e introduce de forma automática, pero, los parámetros que hemos comentado son los fundamentales.

4.2.2.2.- Defaul.aspx.

Es la página principal de nuestra utilidad y en ella se gestiona de forma centralizada todas las funciones de autenticación. Cuando alguien intenta acceder a nuestro sitio web, es direccionado a esta página, que le muestra, mediante una serie de mensajes la situación de su certificado en caso de error y le redirecciona hacia la página inicio de la carpeta sistema en caso de usuario válido. Además realiza una serie de anotaciones Log en el archivo de trace del sistema en las que indica la situación de la ejecución. Podemos encontrar dos partes:

- **Defaul.aspx** En esta parte se define la interface de la página web que consistirá en una serie de logos e imágenes que se encuentran en la carpeta recursos; y una serie de mensajes con los que la aplicación comunicará al usuario en su caso, por qué ha sido desestimada su petición.
- **Defaul.aspx.cs** Aquí encontramos toda la lógica que rige nuestra página de Login. Básicamente se realizará una comprobación del usuario que intenta acceder a nuestra zona segura, anotando en el log todos los pasos.

Modo de funcionamiento.

Primero comprobamos si el usuario ha proporcionado un certificado X509, esto se realiza haciendo una llamada *Request.ClientCertificate* que nos proporciona un certificado que meteremos en una clase como variable local *HttpClientCertificate cs*. Luego haremos uso de una de las funciones que implementa esta clase, en concreto *cs.IsPresent* que nos devolverá *true* si hay un certificado válido. En caso de no tener un certificado adecuado, termina nuestro intento de acceso, realizando la anotación correspondiente en el log, poniendo los iconos de error y mensajes en la página, y deshabilitando el botón de acceso.

Seguidamente, si existe certificado adecuado, comprobamos la fecha del certificado con la función interna *ComprobarFecha()* que nos devolverá un *true* para la fecha correcta. Cabe recordar que los certificados de los DNIe tienen un periodo de vigencia de 30 meses. Hacemos la anotación de Log correspondiente, y seguimos. Si la fecha actual estuviera por encima de la fecha de caducidad del certificado o la fecha actual estuviera por debajo de la fecha de entrada en vigor del certificado, tendríamos un retorno *false* con lo que también terminaría nuestro intento de acceso, realizando la anotación correspondiente en el log, poniendo los iconos de error y mensajes en la página, y deshabilitando el botón de acceso.

Si el certificado es adecuado y estamos en fecha correcta, la siguiente comprobación es una llamada a la clase *validarOCSP.* La instanciamos en nuestra aplicación con *ValidarOCSP validar = new ValidarOCSP(),* y utilizamos el método *validar.OCSP_validacion(cs)* pasándole nuestro certificado *cs* para, encapsulado en *ValidarOCSP.CertificateStatus Cstatus,* obtener una respuesta de la situación de nuestro certificado. Podemos tener como respuesta:

- Good, si el certificado es correcto.
- Revoked, si el certificado ha sido anulado o dado de baja.
- Unknown, si el formato del certificado no es el adecuado.
- NoCertee, si nuestra aplicación no ha podido acceder a los certificados de autoridad de validación, necesarios, para encapsular la consulta y comprobar la respuesta.

En cualquiera de los tres últimos casos daríamos por terminado el intento de acceso, con la correspondiente anotación en Log y la presentación de un mensaje de error y cierre de botón de la página web.

Con el certificado totalmente comprobado tanto en fecha, como, con una autoridad de validación, pasaríamos a realizar la comprobación del usuario en nuestra base de datos. Para ello utilizamos la clase *DarAcceso().* Con el método x.VerificarAcceso(SerialNumber) donde proporcionamos el número de serie del certificado que ha presentado el usuario que pretende acceder. Esta función nos devolverá 1 si se trata de un usuario registrado y 2 si se trata de un administrador registrado. En cualquier otro caso estaremos hablando de un usuario no registrado o dado de baja y no tendrá acceso, con la consiguiente anotación el Log, la presentación de un mensaje de error y el cierre del botón acceso en la página web.

Para el caso de usuario registrado, hacemos anotación en el Log, modificamos la página web indicando que todas las comprobaciones son correctas, y con la sentencia *FormsAuthentication.RedirectFromLoginPage(cs.Subject , false)* redireccionamos la petición de acceso hacia la página de inicio dentro de nuestro sistema y el atributo de usuario autenticado pasará a ser *true*.

4.2.2.3.- ValidarOCSP.cs.

Lo primero a destacar es el uso de Bouncycastle, que es un proveedor de librerías criptográficas para Java y C# localizado en www.bouncycastle.com de código libre y abierto. Su uso está ampliamente extendido y consolidado y, nos facilitará una serie de herramientas para la comprobación OCSP de una forma estandarizada y segura. Al tratarse de un PFC universitario preferimos incluir la API completa de Bouncycastle.

Esta clase, además de apoyarse en la API Bouncycastle define una variable *CertificateStatus* donde poder indicar el estado del certificado que estamos analizando. Representará valores de Good, Revoked, Unknown, NoCertee en función de que el estado del certificado sea bueno, anulado, desconocido o no se disponga de los certificados adecuados para comprobarlo.

Por otro lado implementa un método *OCSP_validacion()* con el que proporcionando un certificado en formato *HttpClientCertificate* nos hace una comprobación OCSP y obtenemos un objeto *CertificateStatus* con la indicación, según su valor, del estado de uso del certificado cuya comprobación hemos solicitado.

Modo de funcionamiento.

Resolvemos cuál es el certificado de entidad emisora que necesitamos para este certificado de usuario. Mediante la llamada a la clase *CertStore* que implementa el método *BuscarCertificado* obtenemos dicho certificado de entidad emisora buscándolo en el store de Windows. Previamente tendremos que haber descargado e instalado dichos certificados, que tienen una

duración de 15 años desde la fecha de expedición, con nuestra aplicación de gestión de certificados.

Se genera la petición OCSP con el certificado de la entidad emisora del certificado y el número de serie del certificado cliente, utilizando una plantilla de Bouncycastle. Previamente hemos convertido los certificados de cliente y Autoridad de validación (entidad emisora) que estaban en formatos de Microsoft FrameWork 3.5 a formato Bouncycastle que nos permitirán trabajar con sus librerías. Esto se realiza exportando los certificados a una array de bytes en formato ANS1, que es un estándar que se puede leer con la clase de Bouncycastle *x509CertificateParser()*. La función *BuscarCertificado* también nos devolverá un certificado de AC en formato array de bytes encapsulado en ANS1 para facilitar la conversión del certificado a tipo Bouncycastle.

Se establece la conexión HTTP con el OCSP del DNIe y se compone una petición con una serie de cabeceras y con la plantilla que hemos generado con Bouncycastle. Se envía esa petición y se recoge una respuesta.

Se verifica si la respuesta es de un formato reconocido por la plantilla de respuesta de Bouncycastle. Esta respuesta OCSP contiene unas cabeceras normalizadas, con la respuesta a nuestra consulta y, está firmada por la autoridad de validación. Tras verificar todos estos puntos podemos recuperar la respuesta de la entidad emisora del certificado de nuestro usuario objeto de la consulta, y según sea esta respuesta estableceremos en nuestra variable certstatus un valor u otro. Finalmente devolveremos esta variable como resultado de nuestro método invocado.

4.2.2.4.- CertStore.cs.

Esta clase proporciona los certificados cargados en nuestra máquina, en la ubicación Windows que denominamos *"Servidor", StoreLocation.LocalMachine*. Recordemos que nuestra herramienta administración del portal podremos instalar y gestionar los certificados en esa localización, para disponer siempre de los certificados de autoridad de validación correctos. La respuesta que obtendremos al invocar esta clase será un certificado X509 en modo array de bytes ANS1 para certificados X509. El formato de codificación para este tipo de certificados es DER (Distinguish Encoding Rules).

4.2.2.5.- DarAcceso.cs.

Normalmente el acceso a bases de datos es un punto sensible en la programación de los controles de acceso. En nuestro caso es menos probable llegar a la comprobación del usuario en la base de datos con un certificado manipulado para producir errores que beneficien un posible acceso indebido. Sin embargo seguiremos una serie de reglas genéricas que se suelen seguir a la hora de realizar este módulo.

La conexión con la base de datos que contiene los datos de autenticación se define en el archivo *web.config* en el apartado *<connectionStrings>.* Donde encontramos las siguientes definiciones:

name="SQLServidor" Define la etiqueta de referencia de esta conexión.

connectionString="Data Source=.\SQLEXPRESS;AttachDbFilename=C:\inetpub\temp\base de datos\ServidorWeb.mdf;Integrated Security=sspi;User Instance=True" Contiene la ruta física de nuestra base de datos. Recordemos que tiene que estar en un directorio con permisos de uso definidos para IIS 7("IIS_IUSR" suele ser el usuario por defecto). Con *Integrated Security=sspi,* habilitaremos la seguridad en la comunicación. Se podrían añadir más propiedades de seguridad como conexión SSL, user, pasword, etc. aunque, en principio con este nivel consideramos que es suficiente.

El segundo punto es la creación de un procedimiento en la base de datos que utilizando el motor y los recursos de la base de datos realizará nuestra consulta. De esta manera se gana en seguridad ya que evitamos el movimiento de datos innecesarios y optimizamos la consulta.

Por último, también queremos depositar la información en la base de datos codificada. En este caso vamos a realizar una función hash SHA1 sobre el indicador de usuario que será el número de serie de su certificado de autenticidad. De esta manera además aseguramos que se mantiene la integridad de los datos contenidos en el DNIe de los usuarios registrados.

Modo de funcionamiento.

El funcionamiento de nuestra clase es sencillo. Al invocar el método *VerificarAcceso,* proporcionamos el número de serie del certificado a verificar en la base de datos. Por un lado mediante la función interna *GenerarClaveSHA1* obtenemos un resumen normalizado del número de serie proporcionado. Seguidamente se abre un enlace con la base de datos. Instanciamos el procedimiento *StoredProcedure1* de la base de datos, definiendo los parámetros de entrada y salida que se deberán utilizar. Ejecutamos la consulta, y procesamos el resultado, que dejaremos registrado en una variable tipo entero *Autorizado*. Terminamos cerrando la conexión con la base de datos.

4.3.- Definición de Funciones en AdministradorServer1.

4.3.1.- Diagrama de Clases en AdministradorServer1.

El punto de inicio de nuestro programa será la clase Program.cs, que llamará al formulario Form1.cs. Este será el nudo central de nuestro programa, desde esta clase se accederá a todos los recursos de nuestro sistema. Esta clase la podemos distinguir tres partes. Una primera parte comprueba los inputs del sistema (certificados de AC y certificados de usuario que inicia la aplicación). Una segunda parte, que se habilita si las comprobaciones en la primera parte son correctas, con los controles generales: el botón Salir, el botón para cargar el panel Administrar Usuario y, el botón de carga del panel Administrar Certificados; una tercera parte son los botones que contienen funciones en el panel Administrar Usuario y el panel Administrar Certificados que realizan todas la funcionalidades de administración disponibles en cada caso.

Modo de funcionamiento.

Al comenzar el programa en la carga del formulario, se ocultan todos los paneles de administración del sistema hasta comprobar el usuario que intenta acceder. Lo primero es comprobar si están los certificados de entidad certificadora, invocamos la función *ListarCertificados* que los busca en la memoria de Windows. Si están los imprime en pantalla y

continua, en caso contrario, se procederá a cargar cada certificado desde un fichero seleccionable por el usuario con la función *openFileDialog1* y la utilidad *ReadFile*. Para realizar estas operaciones con los certificados AV, tenemos que emplear las utilidades de *CertStore.cs* que nos van a permitir comprobar, extraer, agregar y eliminar cualquier certificado en la memoria Windows de certificados que utilizamos. Luego se imprime por pantalla la información general y se continúa con el programa.

Utilizando el proyecto **PlainConcepts.Fx.DNIe** (ver el capítulo 4.3.2.4 y 4.6) cargamos el DNIe y accedemos a sus certificados digitales. Cogemos el certificado de autenticación, y le sometemos a una tabla de pruebas: comprobación de fecha, comprobación de OCSP, y finalmente comprobación en nuestra base de datos local. Para ello emplearemos la función *ComprobarFecha()* y las clases *ValidarOCSP.cs* y *BaseDatos.cs* que se apoyan en las clases *CertStore.cs* y *Conexión.cs* respectivamente. Además todos los procesos en la base de datos se realizarán con los StoredProcedure correspondientes. Estos procedimientos son descritos en el capítulo 4.4.2.- Procedimientos de la base de datos.

Una vez cargado el certificado de usuario, antes de comprobar si está en la base de datos se realiza una comprobación de existencia de administrador en la base de datos, en caso de que no exista ningún administrador, se entenderá que es la primera vez que se activa el servicio y se definirá como Administrador al usuario actual. Además conviene indicar que no se ha puesto restricciones en el número de usuarios que puedan tener el rol de administrador.

Tras cubrir estos pasos con éxito, accedemos a los menús de la aplicación con dos bloques distintos: Administración de Certificados, donde podemos eliminar y cargar los certificados de la entidad AV y, Administración de Usuarios, donde podremos eliminar, modificar, o añadir usuarios. La relación entre todas estas funciones y clases se representa en la Figura 12.

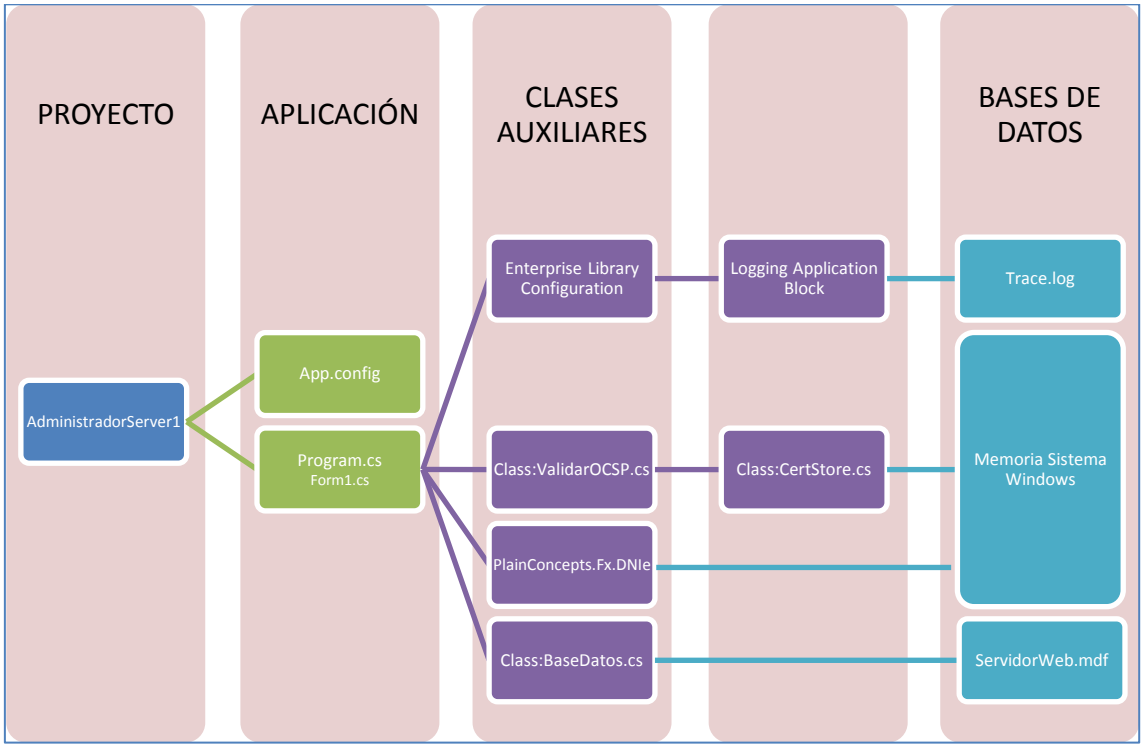

Figura 12. Diagrama de clases de la aplicación AdministradorServer1.

4.3.2.- Definicion de Clases y Funciones en la Aplicación AdministradorServer1.

4.3.2.1.- App.config.

Es el archivo de configuración de nuestra aplicación, en él se definen varios apartados importantes: Enterprise Library configura automáticamente sus bloques y, en el apartado <connectionStrings> se indica a la aplicación la dirección de enlace con la base de datos.

4.3.2.2.- Program.cs.

Es el punto de entrada de la aplicación. Básicamente ejecuta nuestro formulario de inicio y se genera automáticamente al crear una aplicación de formularios.

4.3.2.3.- Form1.cs.

Es el punto principal de la aplicación. Al arrancar, realiza todas las gestiones relacionadas con los certificados de las entidades AV, así como la gestión y creación en su caso, del primer administrador del sistema. Mediante el módulo de acceso al DNIe PlainConceps.Fx.Dnie realizamos las funciones de acceso a los certificados electrónicos de la tarjeta DNIe mediante el PIN. Luego comprueba si el certificado electrónico del usuario que quiere acceder está registrado. En el caso afirmativo, mostrará el panel de Administrador de certificados con un TextBox de información desde la aplicación al usuario. Tenemos un Segundo panel que servirá para la Administración de usuarios. En este caso cargará además una tabla donde se mostrarán todos los usuarios que existen en nuestra base de datos. Para seleccionar entre una utilidad de administración a otra tendremos dos botones rotulados en azul, y así mismo, dispondremos en todo momento de un botón de salida de la aplicación rotulado en color rojo.

Además del control de selección entre paneles y de todos los eventos que disparan nuestros botones, en Form1.cs tenemos una serie de funciones que son:

- ListarCertificados() nos permite presentar por pantalla la lista de certificados AV, así como la fecha de caducidad. Si falta alguno nos mostrará una ventana de dialogo para poder seleccionarlo y cargarlo como cualquier fichero.
- openFileDialog() es la parte del código donde cogemos el fichero seleccionado como certificado válido de entidad AV y lo cargamos.
- ComprobarFecha() es una utilidad para comprobar si cualquier certificado está en fecha o esta caducado.
- ReadFile() es la utilidad que se utiliza para abrir un fichero.

4.3.2.4.- Proyecto PlainConceps.Fx.Dnie .

Hemos añadido a nuestro proyecto AdministradorServer1 el proyecto PlainConceps.Fx.Dnie. Luego instanciamos las clases *DnieHelper* y la estructura de datos *DnieIdentity*. Con el método de *DnieHelper, IsSmartCardPresents()* se realiza la operación de inserción del DNIe en el lector y la extracción de los certificados electrónicos del DNIe hacia la memoria de Windows previa introducción del PIN. En la estructura de datos *DnieIdentity* colocamos la información del usuario del DNIe con el método de *DnieHelper*, *GetIdentity().* Y con el método *GetDnieCertToAutenticacion()* también de la clase *DnieHelper*, conseguimos el certificado electrónico de autenticación.

4.3.2.5.- ValidarOCSP.cs.

Es exactamente igual que la clase ValidarOCSP.cs descrita en el punto 4.2.2.3.- ValidarOCSP.cs. de la aplicación Server1. Con esta clase obtenemos para un certificado X509, su estado actual remitido desde la AV y conseguido mediante una consulta OCSP.

4.3.2.6.- CertStore.cs.

En CertStore.cs implementamos una clase para manejar certificados situados en la zona de memoria de Windows, para acceder a dichos certificados como consulta o para modificaciones y borrados. Es una ampliación de la clase *CertStore.cs* comentada en el punto 4.2.2.4.

Además del método *BuscarCertificado()* que teníamos en la aplicación Server1, hemos añadidos los métodos:

- *ComprobarCertificado()* que nos informa de la existencia de un certificado.
- *LeerStore()* que nos da la colección de todos los certificados del almacén.
- *AñadirStore()* para añadir un certificado a la colección de certificados existente.
- *BorrarStore()* para borrar un certificado especifico se la colección de certificados.
- *BorrarTodoStore()* con este método se borrará toda la colección de certificados de la memoria.

4.3.2.7.- Conexión.cs.

Esta clase simplemente devuelve el string de conexión que tenemos en *app.config* para cargar en la línea de comando de conexión con la base de datos de nuestro sistema.

4.3.2.8.- BaseDatos.cs.

Esta clase es una utilidad para realizar todas las funciones básicas de la base de datos, como son consultas, modificaciones, creación de registros y eliminación de registros. Dispondremos de los siguientes métodos:

- *Comprobar()* Que nos indicará invocando el StoredProcedure2 si existen algún registro en la base de datos que tenga configurado el tipo como 1, es decir, como administrador.
- *VerificarAcceso()* Esta función es la misma que utilizábamos en Server1 y sirve para buscar un usuario en la base de datos, para ello utilizaremos su número de serie, solo que en este caso no tenemos que darle formato antes de aplicarle la función hash (*GenerarClaveSHA1()*).
- *CrearUsuario()* Con esta función nos conectaremos a la base de datos y utilizando el StoredProcedure3 pasaremos los parámetros de nombre, usuario y tipo para crear un nuevo usuario. Como precaución primero comprobamos si el usuario ya existe, es decir si se trata de una modificación (StoredProcedure4) o de un nuevo usuario, esto se hace para no duplicar usuarios.
- ModificarUsuario() Desde aquí nos conectaremos a la base de datos para realizar una modificación en un registro. Solo podremos modificar el tipo y el nombre del usuario. Utilizaremos el usuario para encontrar el registro a modificar. Esta función es inversa a la función CrearUsuario, ya que si el usuario que intentamos modificar no existe, lo crearíamos.

- *GenerarClaveSHA1()* Esta función nos permite obtener un resumen del tipo SHA1 aplicándolo a cualquier cadena de caracteres. Nosotros lo vamos a aplicar al número de serie del DNIe para almacenar en la base de datos información no esencial y además en forma cifrada.

4.4.- Base de Datos.

Hemos creado una base de datos solo para el control de acceso, independiente de la base de datos de trabajo del portal. De esta forma podemos administrarla de forma separada. Tiene el nombre de ServidorWeb.mdf y podemos ver su estructura en la Figura 13. Para la autenticación de los usuarios solo se hace uso de la Tabla *AccesoUsuarios* y del procedimiento *StoredProcedure1*. Considerando que la base de datos se encuentra en una ubicación local, la configuración del enlace con ella tiene un nivel de seguridad básico. Por otro lado no tiene ningún dato que pueda considerarse sensible, así que, no serán necesarias mayores medidas de seguridad.

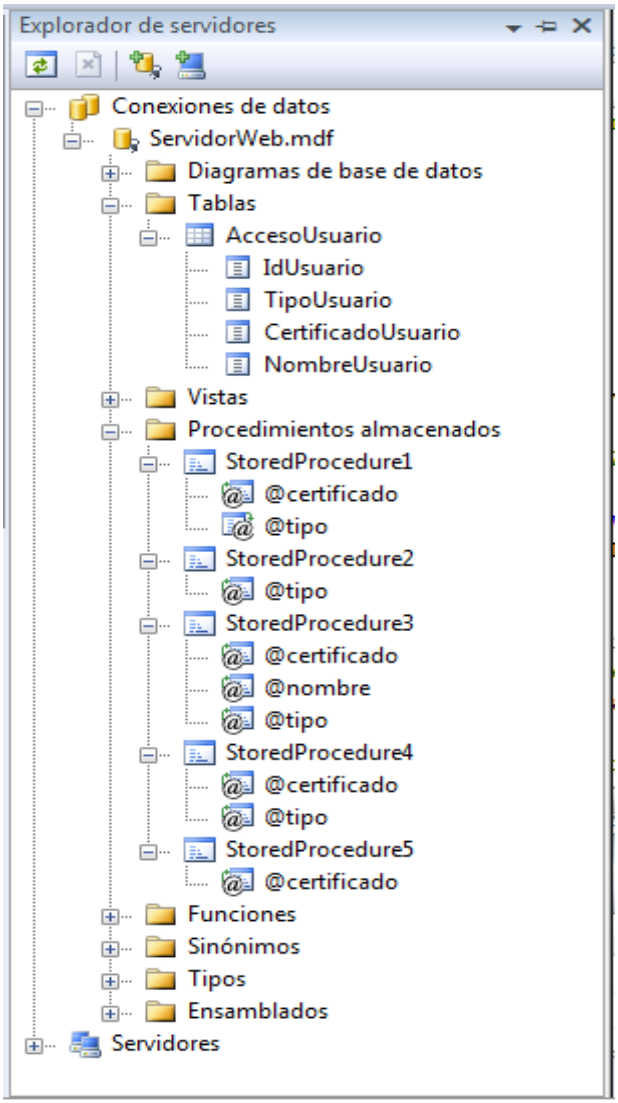

Figura 13. Base de datos ServidorWeb.mdf.

El resto de storedprocedures se emplean en la Administración de los Usuarios, que se realiza en la aplicación AdminitradorServer1.

4.4.1.- Tablas de la base de datos.

En la Tabla 7 podemos distinguir las cuatro variables que necesitaremos para cada usuario. En primer lugar un identificador único de usuarios (IdUsuario) que nos servirá para enlazar con el resto de bases de datos.

Tabla 7. AccesoUsuario.

	Nombre de la columna	Tipo de datos	Permitir Valores nulos
*	IdUsuario	Int	No
	TipoUsuario	Int	Si
	CertificadoUsuario	Nvarchar(50)	Si
	NombreUsuario	Nvarchar(50)	Si

En una variable tipo de usuario (TipoUsuario) identificaremos el Rol del usuario ya sea administrador del sistema (1) o usuario básico (2) y en el campo CertificadoUsuario tendremos el hash del identificador de certificado de usuario. El último campo pondremos el nombre y apellidos del usuario. Es una información muy general y no necesitamos codificarla.

4.4.2.- Procedimientos de la base de datos.

4.4.2.1.-VerificarAcceso (StoredProcedure1).

A continuación tenemos el código del procedimiento en la Figura 14, en el que declaramos dos variables, una de entrada *@certificado* en la que proporcionaremos el código del certificado que pretendemos buscar, y en la variable *@tipo* asignaremos la respuesta con el tipo de usuario encontrado. La consulta consistirá en colocar en la variable de salida el valor del campo *TipoUsuario* de los registros donde el campo *CertificadoUsuario* se igual que el valor proporcionado en *@certificado*.

```
ALTER PROCEDURE dbo.StoredProcedure1(
        @certificado nvarchar(50),
        @tipo int OUTPUT )
AS
        SELECT @tipo = max(TipoUsuario)
        FROM AccesoUsuario
        WHERE CertificadoUsuario = @certificado
```

Figura 14. Código del procedimiento StoredProcedure1.

4.4.2.2.-Comprobar (StoredProcedure2).

En este procedimiento de la Figura 15, solo utilizaremos una variable para buscar, en este caso el tipo de usuario. Al ejecutar el comando ExecuteScalar() nos devolverá un entero donde nos indica el número de coincidencias que existen con nuestra variable de referencia.

```
ALTER PROCEDURE dbo.StoredProcedure2
        @tipo int = 1
AS
        SELECT COUNT (*)
        FROM AccesoUsuario
        WHERE TipoUsuario = @tipo
RETURN
```

Figura 15. Código del procedimiento StoredProcedure2.

4.4.2.3.-CrearUsuario (StoredProcedure3).

Este procedimiento descrito en la Figura 16, sirve para crear un registro. Le pasamos las tres variables que se definen en nuestra base de datos, certificado, nombre y tipo de usuario. La base de datos asignará automáticamente el identificado del usuario.

```
ALTER  PROCEDURE dbo.StoredProcedure3(
        @certificado nvarchar(50),
        @nombre nvarchar(50),
        @tipo int )
AS
BEGIN
        SET NOCOUNT ON;
        INSERT INTO
AccesoUsuario(TipoUsuario,CertificadoUsuario,NombreUsuario)
        VALUES(@tipo,@certificado,@nombre)
END
```

Figura 16. Código del procedimiento StoredProcedure3.

4.4.2.4.-ModificarUsuario (StoredProcedure4).

El procedimiento de la Figura 17 nos ayuda a modificar las variables TipoUsuario, y NombreUsuario usando como índice el certificado de Usuario.

```
ALTER PROCEDURE dbo.StoredProcedure4(
        @certificado nvarchar(50),
        @nombre nvarchar(50),
        @tipo int)
AS
BEGIN
        SET NOCOUNT ON;
        UPDATE AccesoUsuario
        SET TipoUsuario = @tipo,NombreUsuario = @nombre
        WHERE (CertificadoUsuario = @certificado)
END
```

Figura 17. Código del procedimiento StoredProcedure4.

4.4.2.5.-BorrarUsuario (StoredProcedure5).

El último procedimiento, el de la Figura 18, nos permite borrar un registro de la base de datos. Como indicador utilizamos el certificado de usuario.

```
ALTER  PROCEDURE dbo.StoredProcedure5
        @certificado nvarchar(50)
AS
BEGIN
        SET NOCOUNT ON;
        DELETE FROM AccesoUsuario
        WHERE (CertificadoUsuario = @certificado)
END
```

Figura 18. Código del procedimiento StoredProcedure5.

4.5.- Bloque Enterprise Library Configuration.

El uso de esta utilidad es muy sencillo. Hemos habilitado la edición de *web.config* mediante la opción *edit Enterprise Library Configuration*. De esta manera accedemos a una ventana donde podemos añadir a nuestro proyecto cualquiera de los módulos de esta librería. Podemos distinguir en la Figura 19 el uso de un módulo específico que sirve para capturar eventos y registrarlos, que se denomina ***Logging Aplication Block.***

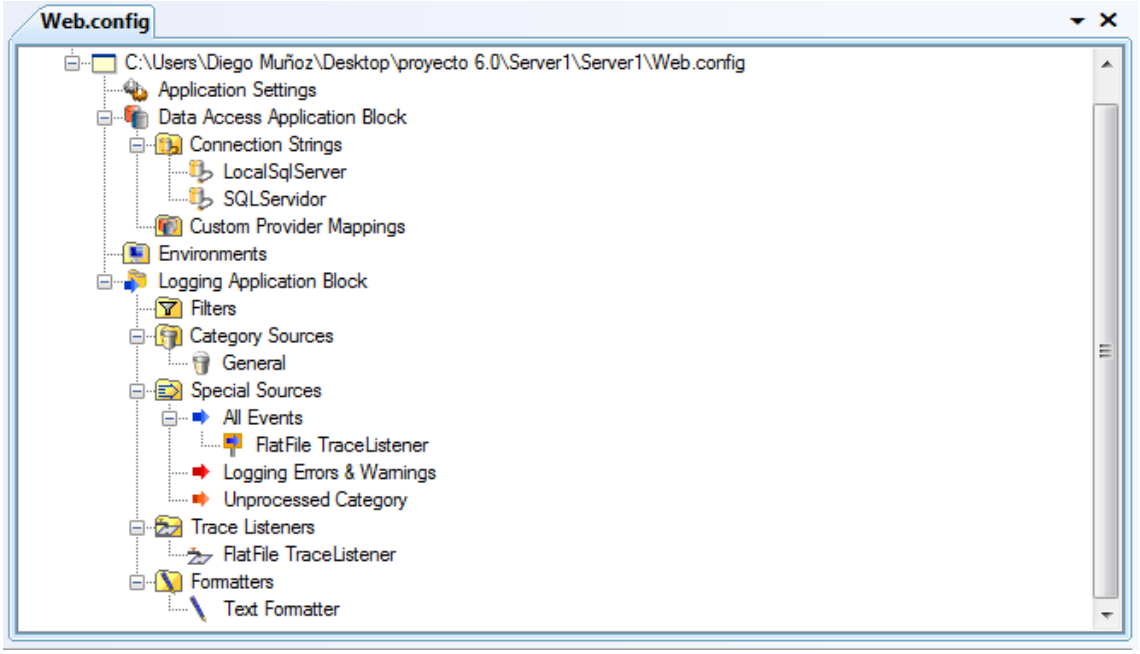

Figura 19. Panel de edición de Web.config con Edit Enterprise Library Configuration.

Para poder configurar la salida de la información, es decir dónde registramos todos los eventos que se van a producir, debemos acceder a la carpeta *TraceListener,* donde crearemos un módulo *Flatfile TraceListener.* Editando las propiedades de este módulo podemos:

FileName = Asignar el archivo en el que se realizará el registro. En nuestro caso en LogServer1/Trace.log.

Formatter = Se asigna el formato a utilizar, en nuestro caso *<template>* que luego editaremos.

Para definir los eventos que se quieren monitorizar, en la carpeta *Special Sources/All Events*, donde añadiremos la opción *Flatfile TraceListener.* Dentro de sus propiedades tendremos que referenciar a *Flafile TraceListener,* que heredará la configuración de *TraceListener.*

Por último en la carpeta *Formatters/TextFormater* podemos editar la propiedad <Template> para definir qué información queremos registrar, además del *menssage,* como puede ser un *TimeStamp*, y otra serie de datos; aunque se recomienda no introducir demasiada información por dos motivos, seguridad y facilidad para el análisis posterior. Nuestro perfil de protección Common Criteria (Anexo B. Capítulo 5) define los elementos mínimos que debemos incluir que son:

- Fecha y hora del evento.
- Tipo de evento.

- Identidad del Sujeto que generó el evento.
- Resultado (éxito o fracaso).

<Template> quedaría así configurado:

Fecha:{timestamp} Evento:{category} {message}

Una peculiaridad de la función timestamp que queremos reflejar, es que hace un acceso externo para establecer el tiempo de aplicación, de esta manera proporcionamos mayor seguridad a nuestro sistema respecto a alteraciones en el registro.

El tipo de evento obtenemos el nivel de prioridad que ha generado el evento. Y en el apartado mensaje comentaremos, el evento que ha generado la llamada, el resultado y la identidad del usuario que ha realizado la acción.

Una vez configurado el bloque, solo falta añadir al proyecto los puntos de registro de eventos, que tendrán la forma:

Logger.Write("Comprobación Acceso Usuario (PASA: OCSP OK). Usuario:" + cs.Subject);

Ya que, Default.aspx.cs actúa como centro de todo el proceso de autenticación, es el mejor sitio para realizar la función Logger para, poder visualización de desarrollo del proceso, y es aquí donde realizaremos esa función.

También en la página de la zona segura Sistema/Inicio.aspx podemos indicar cuándo se produce el inicio y la salida del sistema, y cualquier evento que podamos considerar de utilidad.

Para el caso de la aplicación **AdministradorServer1**, el bloque Enterprise library se aplica sobre el archivo *App.config* y añadimos en el proyecto los puntos de registro, en este caso en *Form1.cs*, no olvidando los includes en references y en el propio *Form1.cs*.

4.6.- Bloque DNIe Toolkit.

Existe un proyecto impulsado por Microsoft que se llama CodePlex para la publicación de código fuente abierto, libre y que puede ser utilizado por la comunidad de desarrolladores. Dentro de CodePlex encontramos DnieToolkit que es una API (denominada PlainConceps.Fx.Dnie), para trabajar con el DNIe para Microsoft .NET y Visual Studio 2008. Permite autenticar y firmar documentos partiendo de la información contenida en los DNIe así como validar el acceso a los lectores SmartCard a través del uso de la librería WinSCard.dll. En nuestro proyecto básicamente lo utilizamos para acceder al certificado electrónico de autenticación del DNIe, para lo cual hemos tenido que hacer una pequeña modificación en sus componentes:

- En *Resources.String.resx* es una tabla de strings claves de búsqueda en los certificados del DNIe, asignamos para CertSubjectAuthentication = AUTENTICACIÓN que es la cadena de caracteres que figura en el certificado electrónico de autenticación del DNIe dentro del campo CN del apartado subject como podemos ver en la Tabla 1.
- En la clase *CertStoreHelper,* que se encarga de manejar los certificados electrónicos de la zona de memoria Windows donde, previamente se instalan al ejecutar el método

IsSmartCardPresents() de la clase *DniHelper,* hemos añadido el método *GetCertificateToAutenticar().* Este método se encarga de recorrer todos los certificados de la memoria hasta dar con el certificado que contiene la cadena de caracteres AUTENTICACIÓN, que es el certificado electrónico que devuelve este método como respuesta.

- En la clase *DniHelper,* hemos añadido el método *GetDnieCertToAutenticacion()* que se encarga de conseguir a través de *DnieHelper* el certificado que nos sirve para realizar una autenticación.

Aunque no hemos utilizado las funcionalidades de firma electrónica están disponibles en este módulo tanto para documentos del tipo DOC, PDF y XML.

5.- Desarrollo.

5.1.- Herramientas.

El entorno de desarrollo empleado para el desarrollo estaba impuesto en las especificaciones del PFC y ha sido **Microsoft Visual Studio 2008** configurado para C#. Hemos utilizado una versión con licencia para estudiantes de la universidad Carlos III de Madrid.

Además hemos empleado las siguientes APIs:

1.- Bouncycastle 1.6.1 para C#. Esta librería criptográfica podemos encontrarla en:

www.bouncycastle.org

Actualmente tenemos publicada la versión 1.7. Esta API es totalmente libre y la más utilizada en la actualidad para el desarrollo de software criptográfico.

2.- Enterprise Library 4.0. Podemos descargarla desde el enlace:

http://msdn.microsoft.com/en-us/library/ff650810.aspx

Es una librería de utilidades desarrollada por Microsoft. Y que podemos utilizar libremente bajo la licencia de Microsoft Ms_PL.

3.- IIS 7. Es la plataforma para servicios web integrada en los últimos sistemas operativos de Microsoft. Solo hay que activar esa funcionalidad en nuestro equipo si cuenta con dicho sistema operativo, y su uso es libre.

4.- DNIe ToolKit. Esta utilidad la podemos encontrar en:

http://DNIetoolkit.codeplex.com

Consiste en una API para acceder a nuestros certificados DNIe y firmar documentos PDF, DOC, XML. El código está en C# y es de mucha utilidad para comprender como se debe hacer un acceso a los certificados de un Smart Card. Codeplex es un proyecto de Microsoft para desarrollo de software libre y trabajamos bajo licencia Ms_PL.

5.- DNIe_v6_0_2.exe. Lo podemos conseguir en:

http://www.DNIelectronico.es/descargas/win_comp_vista.html.

Es la utilidad que el portal oficial del DNIe tiene para instalar las librerías de acceso al DNIe para Windows. Con esta aplicación se cargaran los módulos criptográficos PKCS11 con el que operan Netscape y Firefox Mozilla sobre Windows y, el módulo criptográfico CSP de Microsoft.

5.2.- Configuración del entorno de desarrollo.

5.2.1.- Configuración en Microsoft Visual Studio.

Una de las precauciones que debemos seguir es la de instalar en nuestro entorno de desarrollo la versión correcta de Enterprise Library. Recordaremos que nuestro entorno de desarrollo es Visual Basic 2008 sobre una plataforma .NET framework 3.5. Revisando la documentación de Enterprise Library descubrimos que la versión adecuada será Enterprise Library 4. Tenemos que recordar también añadir las referencias:

- Microsoft.Practices.EnterpriseLibrary.*
- *using Microsoft.Practices.EnterpriseLibrary.Logging* en el archivo *Defaul.aspx.cs*.

En cuanto a Bouncycastle 1.6.1 está desarrollado sobre una plataforma .NET framework 2.0. Pero, es compatible con nuestro proyecto y no necesita re compilación hacia .NET framework 3.5., tendremos que incluirlo en nuestras referencias de proyecto, y hacer un using de algunas librerías que utilizamos en la clase *ValidarOCSP*. Concretamente haremos las siguientes llamadas:

- *using Org.BouncyCastle.OCSP;*
- *using Org.BouncyCastle.Asn1.X509;*
- *using Org.BouncyCastle.X509;*

El resto son llamadas a diferentes librerías dentro de la plataforma de trabajo framework 3.5.

5.2.2.- Configuración de IIS 7.

Toda la documentación para la configuración de IIS 7 la hemos recogido del siguiente enlace: https://zonatic.usatudni.es/es/aprendizaje/aprende-sobre-el-DNIe/58-desarrolla-con-el-dni-electronico/226-internet-information-services-iis.html.

Pertenece a ZonaTic que es un portal para profesionales y expertos sobre DNIe. Sin embargo, repasaremos todos los pasos ya que existen algunos aspectos particulares y errores en la información de esta web.

5.2.2.1.- Instalación de IIS 7.

IIS 7 es un módulo que viene por defecto desactivado en nuestro sistema operativo. Para activarlo debemos seguir la siguiente secuencia: Inicio -> Panel de control -> Programas -> Activar o desactivar las características de Windows. Una vez aquí debemos seleccionar la casilla Internet Information Services. Además tendremos que acceder dentro de esta casilla a otro directorio Servicios World wide web -> Características de desarrollo de aplicaciones y seleccionar las casillas: ASP, ASP .NET y extensibilidad de .NET que le permitirá interactuar con nuestro proyecto.

Todas las opciones de configuración se harán desde la utilidad *Administrador de Internet Information Services* que se creará en nuestro ordenador.

5.2.2.2.- Usuarios y Directorios.

Al activar este módulo de Microsoft nos creará un directorio por defecto para instalar nuestras aplicaciones web: *C:/inetpub*. Aunque podremos configurar el acceso a otros directorios.

Además se crea un usuario de trabajo que será ISS_IUSR y con el que se accederá por defecto a los diferentes recursos. Es conveniente o crear un nuevo usuario o darle permisos de control a este usuario dentro de nuestros directorios de trabajo y base de datos.

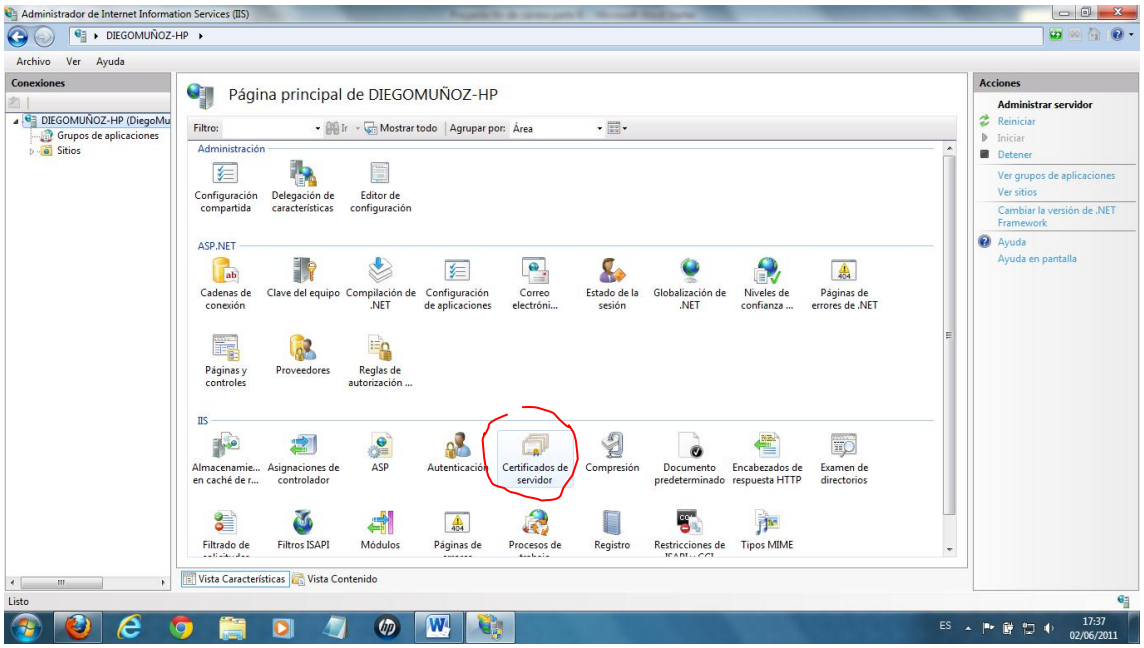

Figura 20. Administrador de Internet Information Services.

Dentro de la aplicación *Administrador de Internet Information Services* podremos configurar todos los aspectos de IIS 7. Abrimos en la opción IIS-> Certificados de Servidor, como se distingue en la Figura 20 y, vemos en la Figura 21 que existen varias formas de instalar un certificado en nuestro servidor:

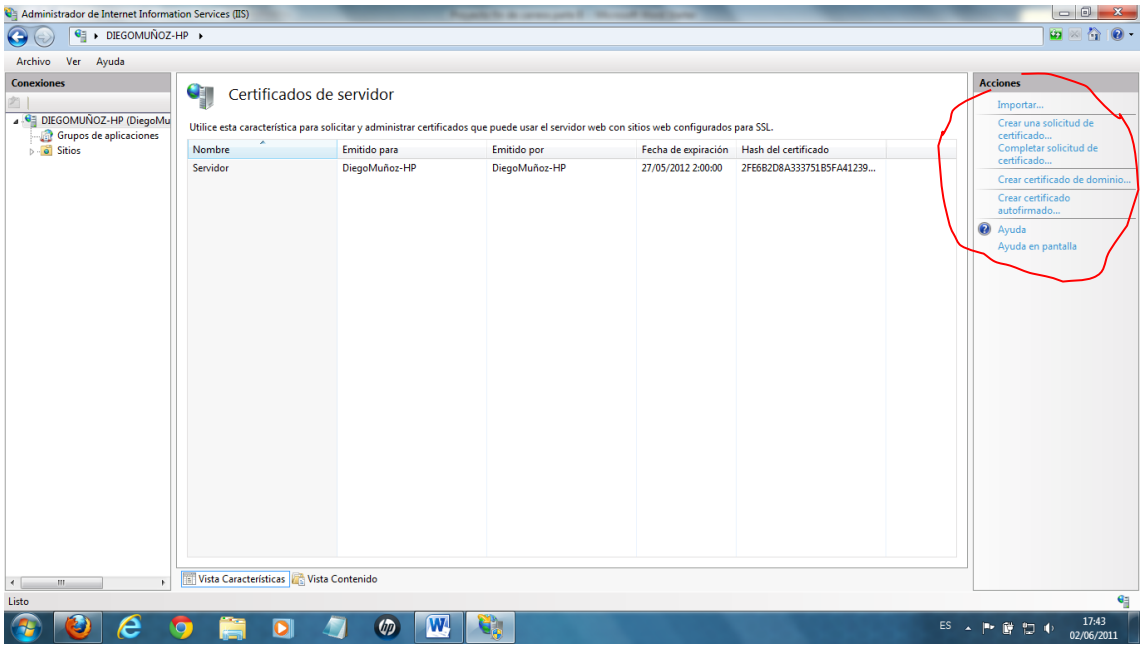

Figura 21. Ventana de administración de los certificados de servidor de IIS 7.

- Importando un certificado. Existen aplicaciones como la aplicación Shareware ABylon SelfCert con la que crear certificados en formato X509 para nuestro servidor. Podemos descargarlo de:

  http://www.abylonsoft.com.

- Podemos solicitar un certificado a Microsoft mediante las opciones *Crear una solicitud de certificado…* De esta forma se rellena unos formularios y se reenvían a Microsoft para que nos facilite un certificado del sitio web.
- Crear un certificado autofirmado. La propia aplicación IIS 7 se encargará de generar un certificado. Esta opción es la que utilizaremos nosotros de momento.

5.2.2.4.- Configuración de sitios web.

Nuestro proyecto tiene dos accesos uno, http://localhost, que será una dirección a página web normal sin seguridad, la cual nos redireccionará abriendo otra ventana a una dirección https://localhost esta vez sí en modo seguro y con petición de certificados de seguridad. De esta forma al terminar nuestro acceso por HTTPS podemos cerrar la ventana y nos quedaremos en la ventana HTTP general desde donde podremos acceder a otros sitios web.

La página HTTP está en nuestro proyecto InicioServer1/InicioServer1/Default.aspx y podemos configurarlo en IIS 7 desde el administrador. En el menú lateral de la izquierda, seleccionar el elemento Sitios. En el menú que se muestra a continuación, se debe hacer clic derecho y seleccionar la opción Agregar sitio web. Seguidamente, como puede verse en la Figura 22 se despliega un menú para configurar el nuevo sitio que se debe configurar con los siguientes valores:

- Ruta de acceso físico: C:\..\ InicioServer1/InicioServer1/Default.aspx
- Conectar como... -> Especificar usuario con permisos de acceso.
- Tipo: HTTP.
- Dirección IP: Todas las no asignadas.
- Puerto: 80.

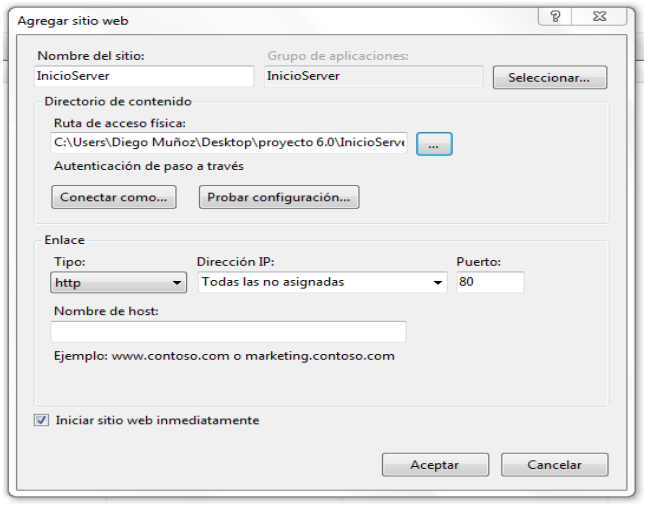

Figura 22. Menú para agregar nuestro sitio web HTTP.

Del mismo modo agregaremos un sitio web para el proyecto Server1. En este caso seleccionaremos como tipo HTTPS y como certificado el certificado autofirmado que hemos llamado Servidor. Como podemos observar en la Figura 23, tendremos los siguientes campos:

- Ruta de acceso físico: C:\..\ Server1/Server1/Default.aspx
- Conectar como... -> Especificar usuario con permisos de acceso.
- Tipo: HTTPS.
- Dirección IP: Todas las no asignadas.
- Puerto: 443.
- Certificado SSL. El certificado autofirmado que hemos denominado Servidor.

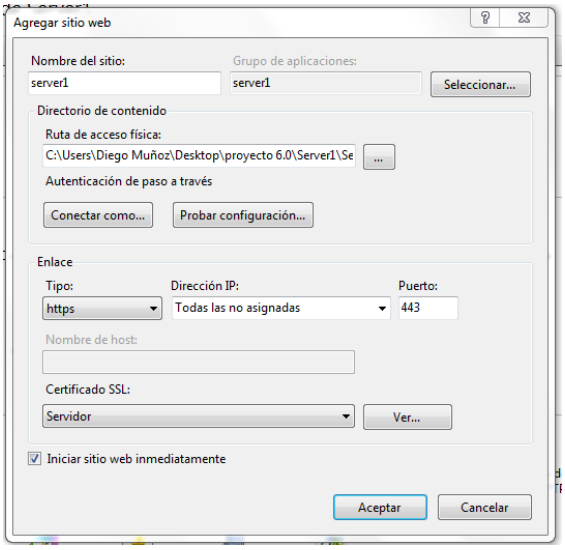

Figura 23. Menú para agregar nuestro sitio web HTTPS.

5.2.2.5.- Configuración de SSL.

Una vez realizado las conexiones a los sitios web, para nuestro sitio HTTPS deberemos configurar la conexión SSL y el método de autenticación a través de los iconos señalados en la Figura 24.

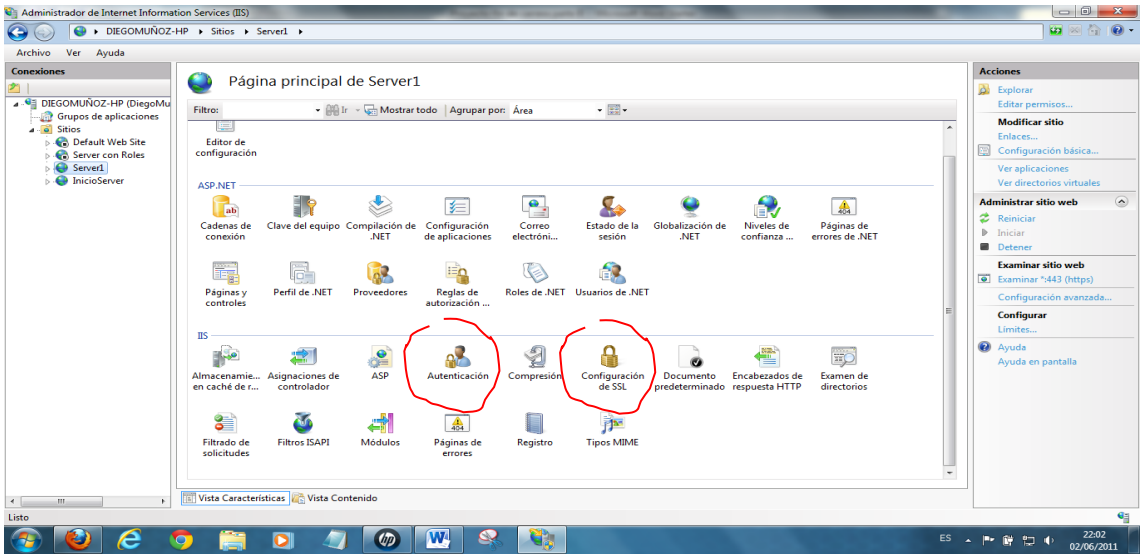

Figura 24. Menú de IIS7 para la configuración del sitio Server1.

Como podemos distinguir en la Figura 25 tenemos un acceso a Configuración de SSL donde seleccionaremos Requerir SSL y en certificados de cliente Requerir.

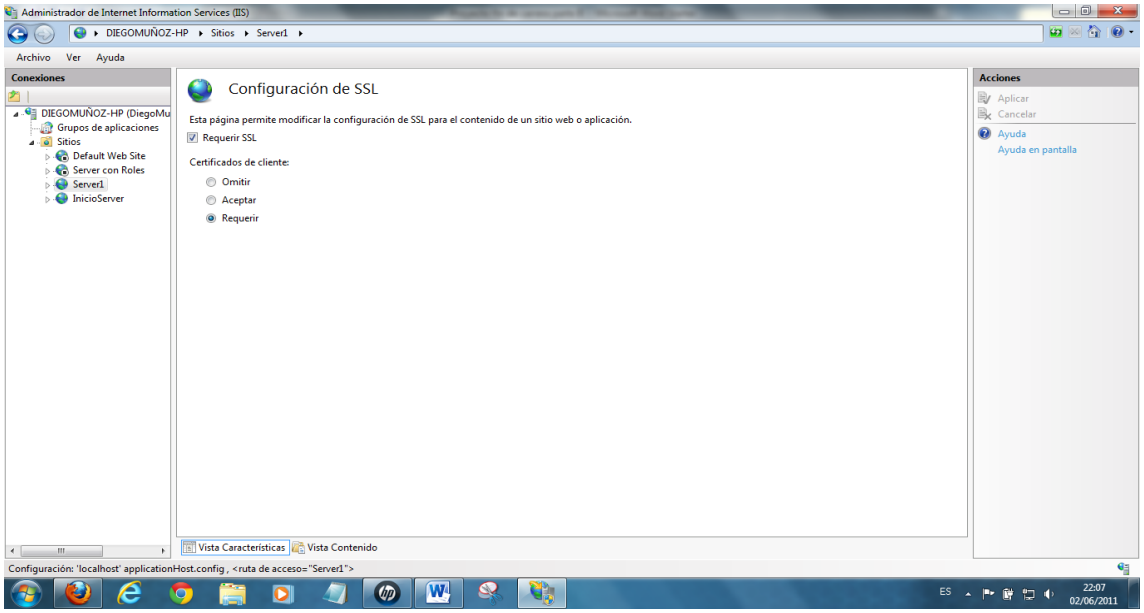

Figura 25. Menú para configuración de SSL en nuestro sitio web Server1.

En el icono Autenticación, Figura 26, tendremos que seleccionar la autenticación anónima y configurar un usuario con derecho de acceso a ese directorio.

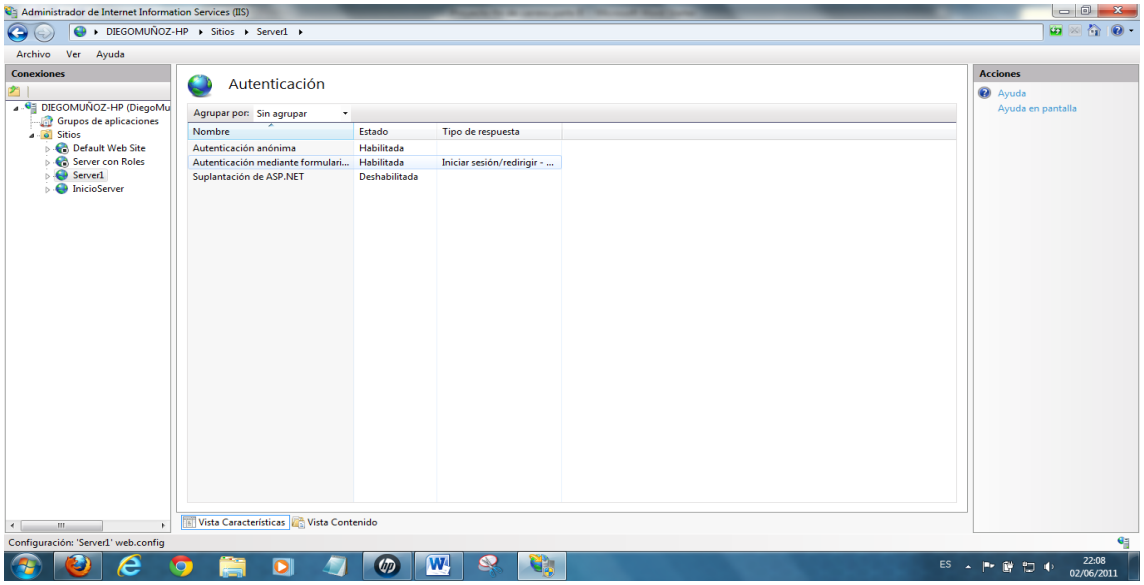

Figura 26. Menú para configuración de la Autenticación en nuestro sitio web Server1.

5.2.2.6.- Configuración de los certificados.

En primer lugar tendremos que configurar los certificados suministrados con el DNIe para que tengan una correspondencia con nuestro usuario local esto lo podemos configurar dentro de IIS 7 en el Editor de configuración, en la ruta que podemos ver en la Figura 27. *system.webServer/security/authentication/iisClientCertificateMappingAuthentication*

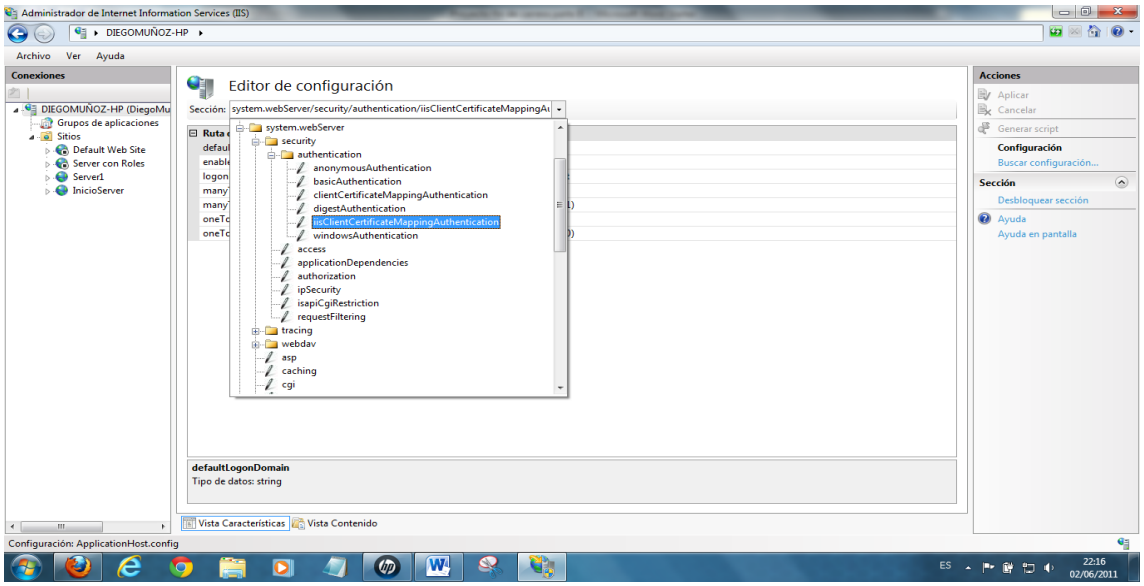

Figura 27. Editor de configuración de IIS7.

En el menú que aparece en la Figura 28, se han de configurar los siguientes parámetros:

- enabled: True.
- logonMethod: ClearText.
- manyToOneCertificateMappingsEnable: True.
- manyToOneMappings: pulsar botón.
- oneToOneCertificateMappingsEnable: False.

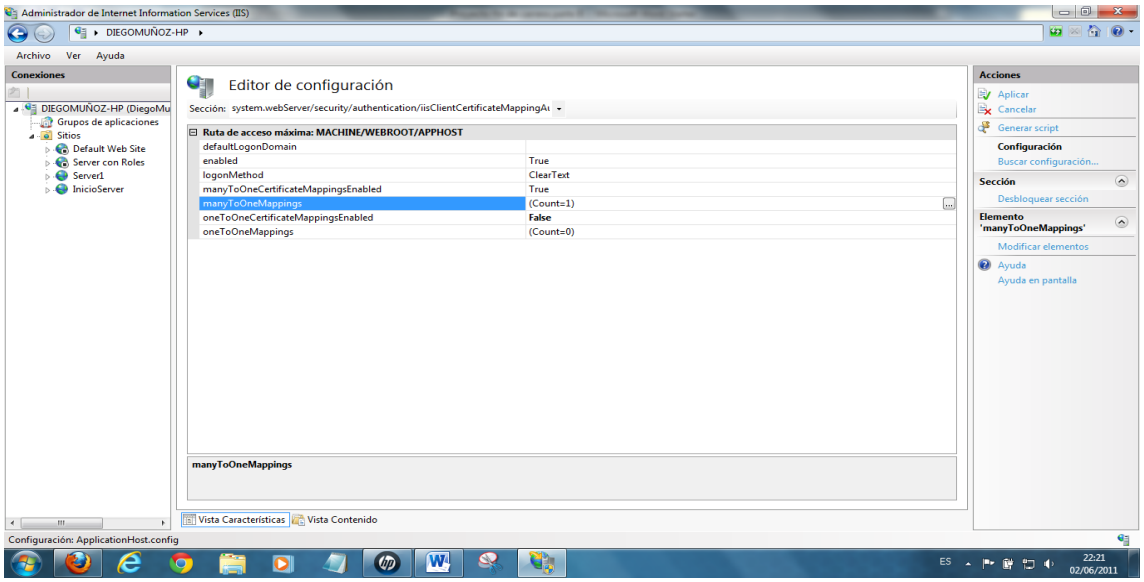

Figura 28. Parámetros de iisClientCertificateMappingAuthentication.

Además editamos la regla manyToOneMappings, Figura 29, en el que se tienen que definir las asociaciones entre certificados y cuenta de usuario. Se deben configurar los parámetros con los siguientes valores:

- description: DNIe.
- enabled: True.

- name: DNIe.
- pasword: contraseña_DNIe.
- permissionMode: Allow.
- userName: DNIe.

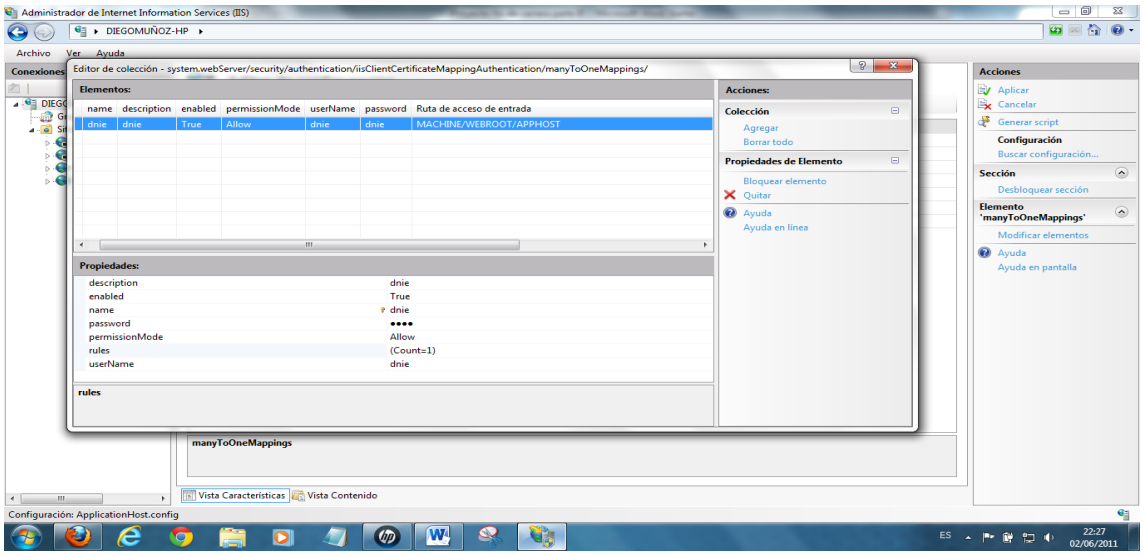

Figura 29. Menú de configuración dentro de manyToOneMappings.

Por último se crea una regla para definir qué certificados se aceptarán. En este caso, en la Figura 30, vemos que se aceptan los certificados de autenticación del DNIe, para ello se tendrán que configurar los parámetros con los siguientes valores:

- certificateField: Issuer.
- certificateSubField: OU.
- compareCaseSensitive: True.
- matchCriteria: DNIE.

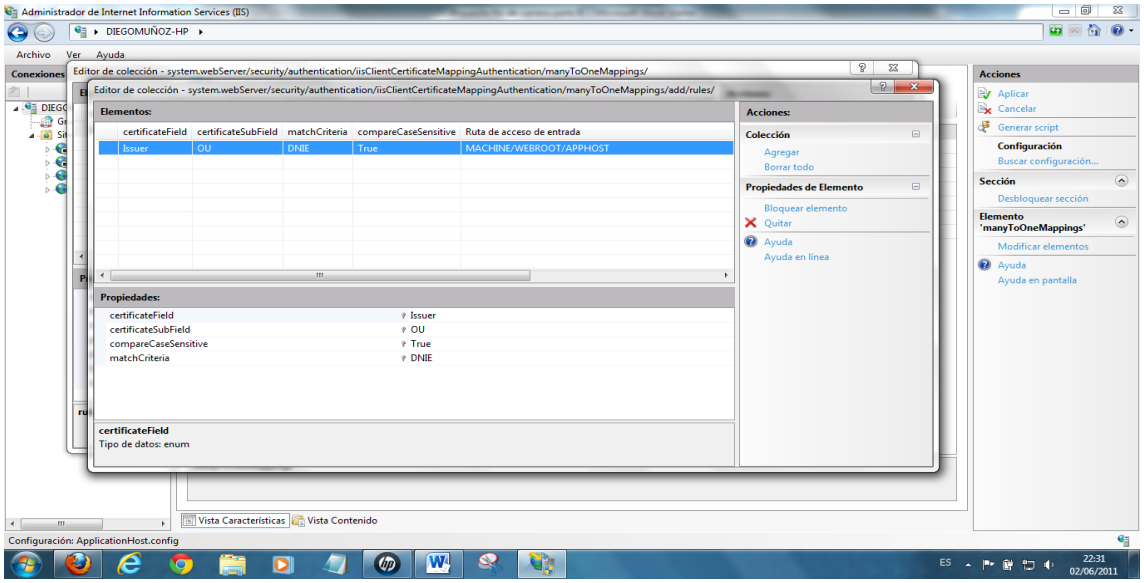

Figura 30. Reglas de los certificados aceptados.

IIS 7 hace una comprobación CRL de los certificados de cliente que recibe. Este sistema implica la creación y continua actualización de la lista CRL. Por nuestra parte va a ser la propia aplicación web la encargada de hacer una autenticación con comprobación OCSP, por lo que vamos a deshabilitar las comprobaciones que realiza IIS 7. Para ello utilizaremos la herramienta netsh.

Podemos iniciar netsh con Inicio->buscar programas y archivos e indicar el programa netsh. Recordar que deberemos ejecutar dicho programa como administradores. Tendremos una ventana de consola como la Figura 31. Para entrar en el contexto HTTP, teclearemos HTTP.

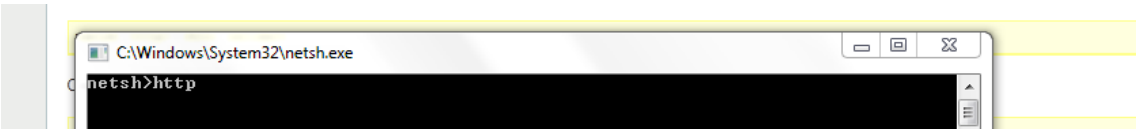

Figura 31. Consola en modo administrador ejecutando programa Netsh. Comando http.

Podemos hacer un listado de certificados de nuestro puerto con el comando show. Al realizarlo tendremos una respuesta parecida a la de la Figura 32.

Figura 32. Consola en modo administrador ejecutando programa Netsh. Comando Show.

El apartado *Comprobar revocación de certificado de cliente* debe figurar Disabled. Si no fuera el caso deberemos borrar el registro de ese puerto. En la Figura 33 tenemos este caso.

Figura 33. Consola en modo administrador ejecutando programa Netsh. Comando delete.

Luego volvemos a crear el registro utilizando el Hash del certificado (es el certificado autofirmado de IIS 7), y el identificador de la aplicación, que para IIS 7 siempre será el mismo en cualquier ordenador. En la Figura 34 vemos cómo se realiza esta función.

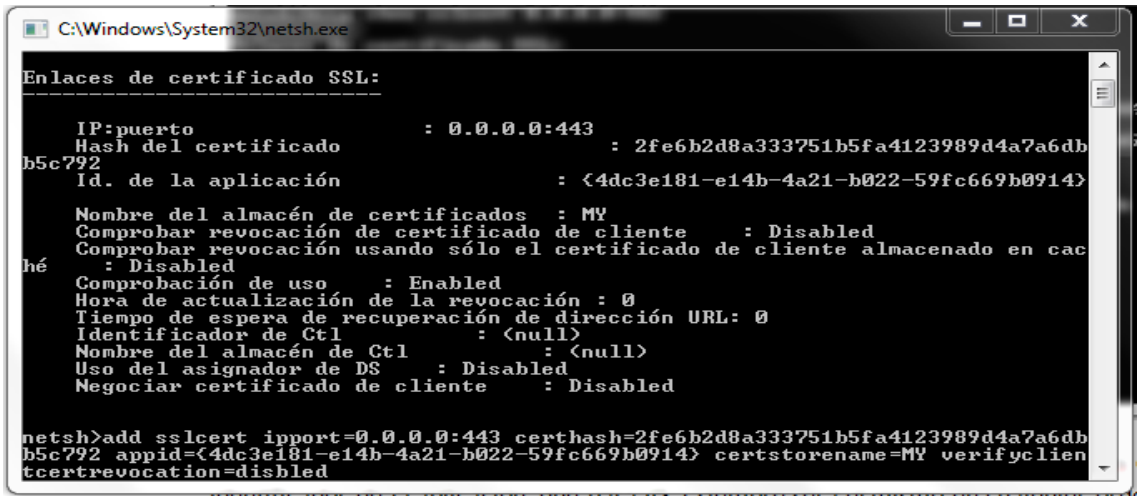

Figura 34. Consola en modo administrador ejecutando programa Netsh. Comando add.

Con esto quedarán configurada la forma de trabajo en IIS 7 para nuestro puerto de acceso respecto a los certificados electrónicos.

5.2.2.7. Configuración de los grupos de aplicaciones.

En este apartado deberemos configurar los dos grupos de aplicaciones con identidad NetworkService(Figura 35). De esta manera desactivamos incompatibilidades en el acceso a recursos, tanto a las bases de datos como a certificados y directorios con la identidad de usuario configurado. Esta operación se realiza desde el menú Configuración avanzada.

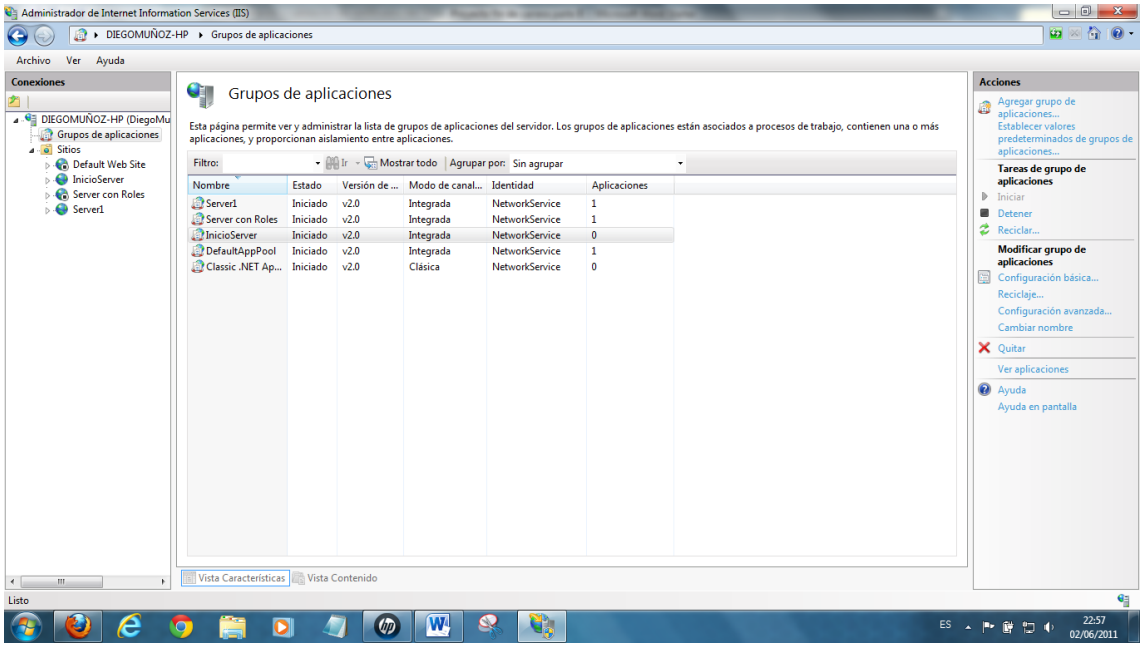

Figura 35. Menú de IIS 7 para configuración de grupos de aplicaciones.

5.3.- Codificación.

5.3.1.- Proyecto InicioServer1.

La conexión a de nuestra puerta de acceso se realiza a través de una conexión SSL, pero, es interesante inicialmente poder redireccionar desde un sitio web sin conexión SSL para poder admitir enmarcarse en sitios web de ámbito general. Por ello hemos diseñado una pequeña página denominada InicioServer1 que nos permite redireccionar desde una página por el puerto normal (HTTP), hacia la página de acceso por el puerto de conexión SSL (HTTPS). En la Figura 36 podemos comprobar la estructura de directorios que tiene esta página inicial a la que hemos denominado InicioServer1.

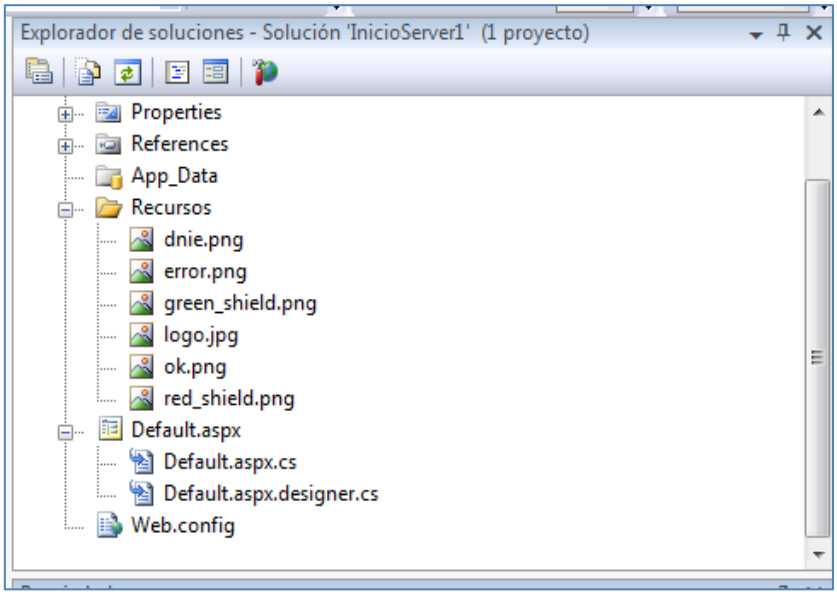

Figura 36. Estructura de archivos del proyecto InicioServer1.

Básicamente el proyecto consta de una página de inicio denominada Defaul.aspx, de la que cuelga el archivo Defaul.aspx.cs; un directorio de recursos con imágenes que utilizamos y un archivo de Web.config que en este caso no vamos a editar ya que se trata de una página web muy sencilla.

Defaul.aspx consiste en una página web de inicio en la se nos solicita que introduzcamos el Dnie y pulsemos un botón para acceder. El evento asociado a ese botón consiste en recuperar la dirección IP de nuestra página para crear una nueva ventana con la dirección conseguida, pero a través del puerto seguro (HTTPS). El conjunto de instrucciones que empleamos para abrir esta ventana, está codificado directamente en Java script ya que no se puede implementar en C#.

5.3.2.- Proyecto Server1.

Hemos denominado Server1 a la página web de acceso. Esta página se encargará de manipular el certificado de identidad que nos proporcionará el navegador del usuario, para comprobar la identidad y accesibilidad de la persona que intenta entrar en nuestro sistema. La estructura de directorios de nuestro proyecto principal, puede verse en la Figura 37.

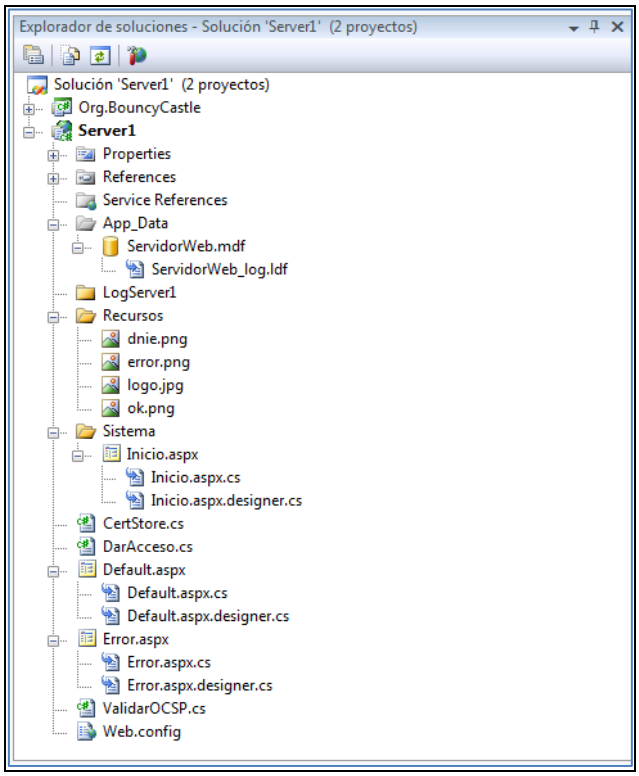

Figura 37. Estructura de archivos del proyecto Server1.

La solución tiene dos partes principales o directorios. Uno será Org.BouncyCastle y el otro será propiamente Server1.

El proyecto Org.BouncyCastle contiene todas las utilidades criptográficas que necesitamos para trabajar con certificados X509 entre las que destacamos la generación de una llamada OCSP y el tratamiento de la respuesta que es principalmente la que vamos a utilizar.

Dentro de Server1 tendremos las siguientes carpetas:

- References = Donde están todas las referencias a recursos del sistema.
- App_Data = Donde ubicamos nuestra base de datos.
- LogServer1 = Directorio donde podemos encontrar Trace.txt, que es el archivo de eventos.
- Recursos = Aquí están todas las imágenes que utilizamos en nuestra web.
- Sistema = El directorio donde instalaremos la web de aplicación, es decir, la zona a la que intentamos acceder desde este portal.

En el directorio raíz de Server1 encontramos además los siguientes archivos:

- CertStore.cs = Clase para acceder a certificados de autoridad de validación en nuestra memoria local.
- DarAcceso.cs = Clase que implementa la búsqueda en la base de datos de nuestros usuarios.
- ValidarOCSP.cs = Clase que se encarga de la validación mediante el protocolo OCSP, de los usuarios que pretenden acceder por este portal.

- Default.aspx = Página de Login a la que se direccionan las peticiones de acceso.
- Error.aspx = Página de error a la que se direccionan todas las salidas de errores del sistema.
- Web.config = Archivo de configuración del sitio web.

5.3.3.- Proyecto AdministradorServer1.

La estructura de directorios de nuestro proyecto principal, puede verse en la Figura 38.

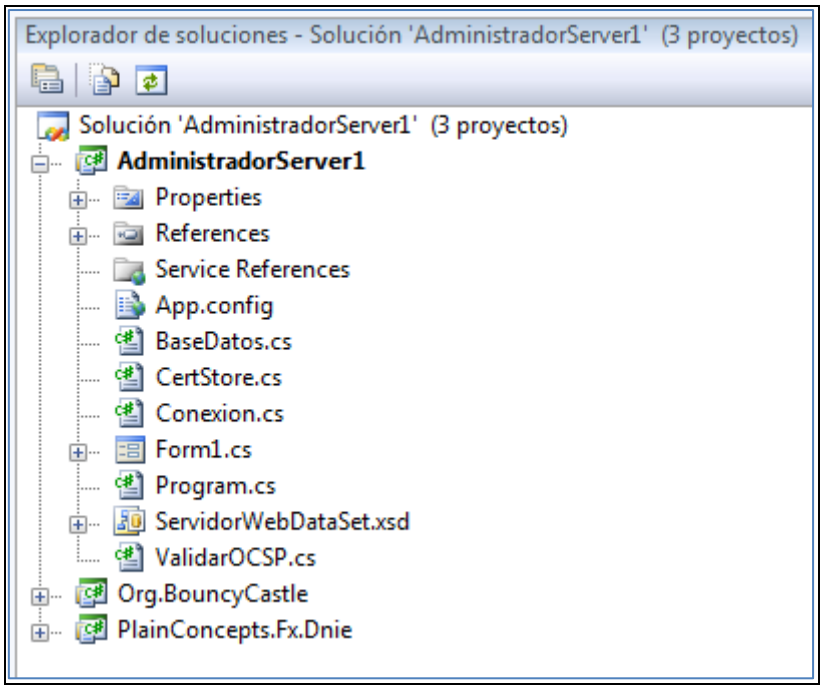

Figura 38. Estructura de archivos del proyecto AdministradorServer1.

5.4.- Operaciones de Uso.

Resumiendo lo visto hasta ahora, tenemos tres directorios de trabajo correspondientes a los tres proyectos que conforman el portal:

1.- **AdministradorServer1.** Es la aplicación de configuración del portal.
2.- **InicioServer1.** Es la página inicial de acceso al portal.
3.- **Server1.** Es la página que propiamente gestiona el acceso al portal.

Después configuramos IIS 7 para que utilice InicioServer1 y Server1 como páginas web de servicio. También configuramos su funcionamiento SSL y su acceso a los recursos del sistema.

En este momento estamos en condiciones de utilizar el portal de acceso a nuestro sistema:

- En primer lugar utilizaremos AdministradorServer1 para dar de alta los certificados de la AC y los usuarios con los distintos permisos que tienen acceso mediante este portal.
- Después podremos acceder con cualquier navegador tanto a la página InicioServer1 para ser redireccionados a Server1, como, directamente a Server1. De esta manera, tras autenticarnos, podremos entrar en la zona segura de nuestro servidor que es el objeto de este portal web.

5.4.1.- Configuración del portal con AdministradorServer1.

Al iniciar AdministradorServer1, lo primero que se realiza es una comprobación de la disponibilidad de los certificados de la autoridad de validación del DNIe. Estos certificados los podemos encontrar en la url: http://www.DNIe.es/seccion_integradores/auto_cert_sub.html y son los archivos: ACDNIE001-SHA1, ACDNIE002-SHA1 y ACDNIE003-SHA1. En un primer momento estos archivos no están cargados por lo que aparecerá un código de error como el de la Figura 39.

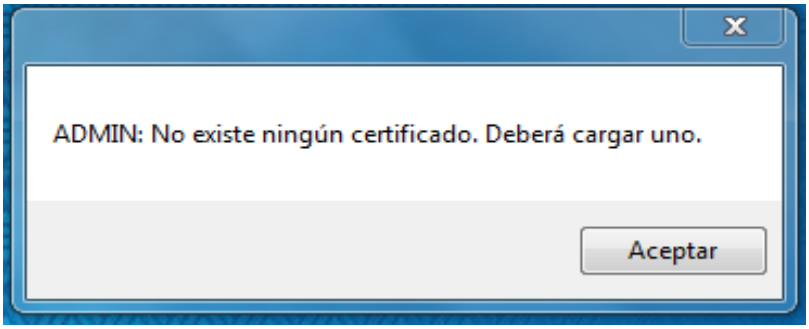

Figura 39. Ventana emergente solicitando Certificados de Entidad Emisora.

Si le indicamos que continúe, la aplicación desplegará un menú para localizar el archivo que necesitamos cargar, ver Figura 40.

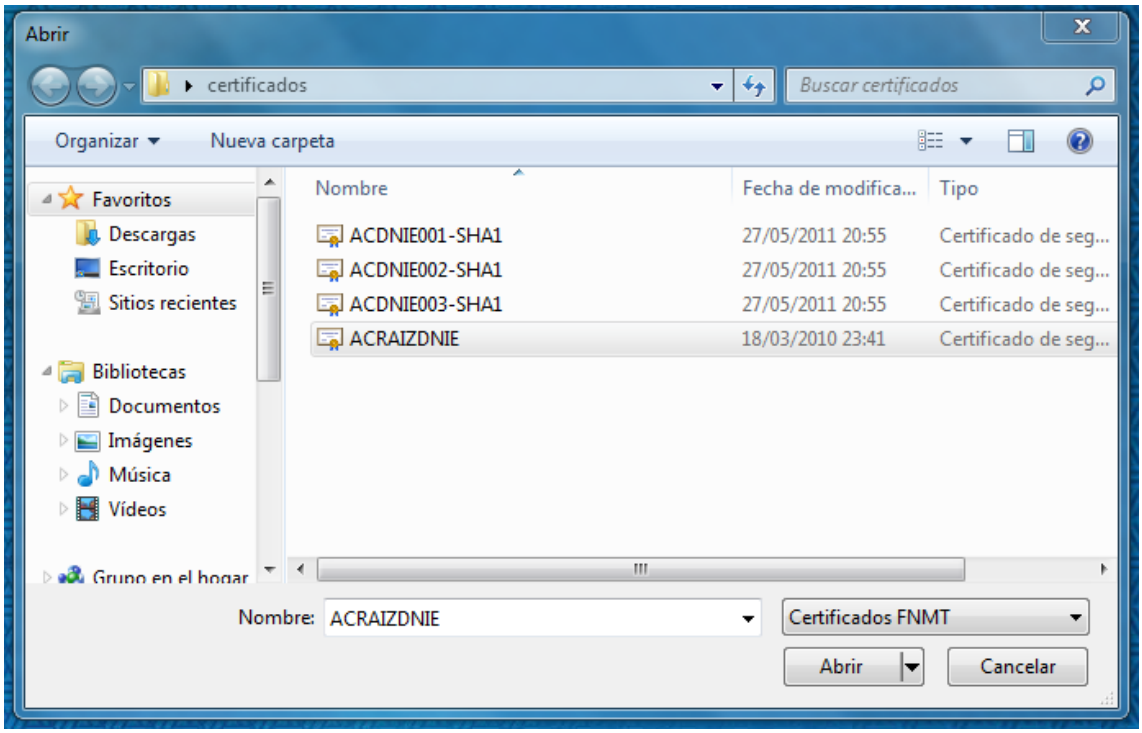

Figura 40. Menú para cargar Certificados de Entidad Emisora.

Para el correcto funcionamiento de nuestro programa necesitamos los tres archivos AC DNIE 001, AC DNIE 002 Y AC DNIE 003. Si el programa detecta en cualquier momento que falta alguno, nos lo comunicará para que procedamos a su carga, como se muestra en la Figura 41.

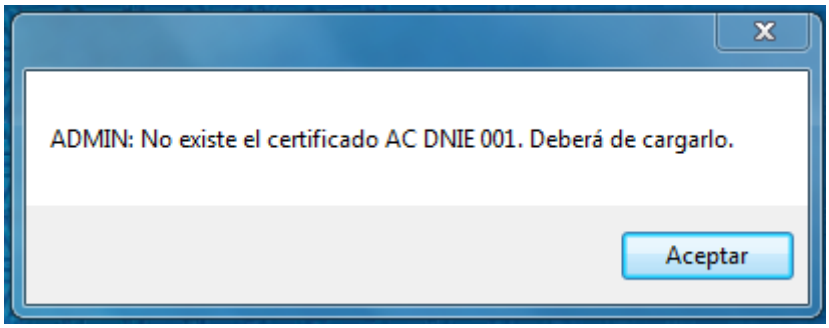

Figura 41. Aviso para indicar la necesidad de carga de Certificado AC DNIE 001.

Cuando el sistema dispone de todos los certificados de la entidad de validación, prosigue con su funcionamiento normal. Inicialmente pide el DNIe del usuario, como podemos ver en la Figura 42.

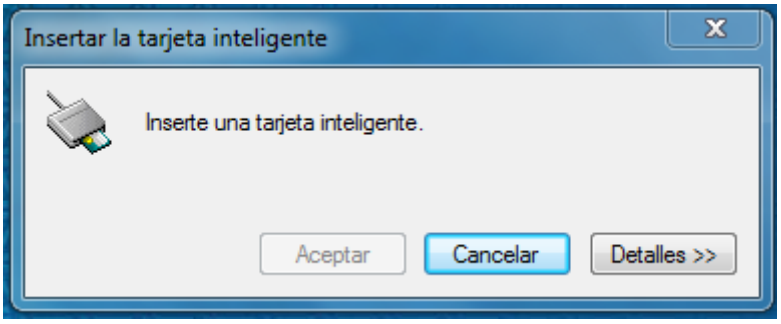

Figura 42. Ventana para la petición del DNIe.

Se nos pedirá el código PIN para acceder a los certificados del DNIe, ver Figura 43.

Figura 43. Petición del código PIN.

La aplicación realiza las funciones de comprobación de fecha y estado del certificado vía OCSP, si todo está bien, comprueba la base de datos para saber si hay algún administrador. Si no existiera ninguno, es decir, estamos inicializando la base de datos, asigna el DNIe actual como administrador. Para ello, como podemos ver en la Figura 44, nos muestra una notificación de las acciones que se van a realizar sobre la base de datos.

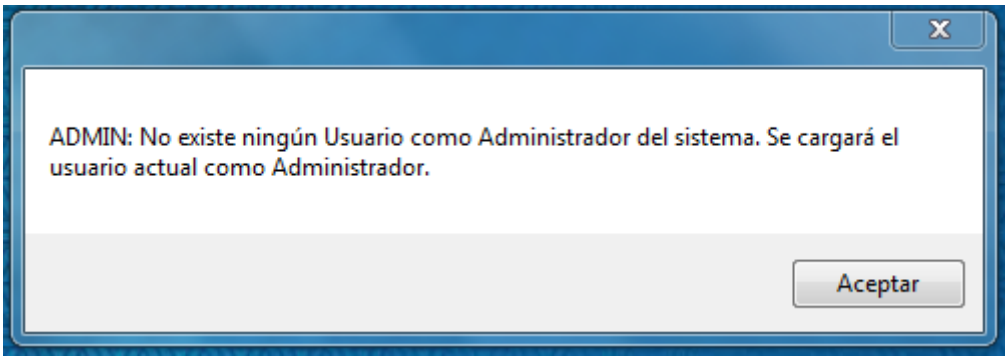

Figura 44. Notificación de acción automática de AdministradorServer1.

Tras tener configurados los certificados y el primer administrador, la aplicación nos da paso al panel de Administración de certificados de la Figura 45. En él podemos listar los certificados actualmente cargados, actualizarlos o borrarlos. También podemos pasar al panel de Administración de Usuarios.

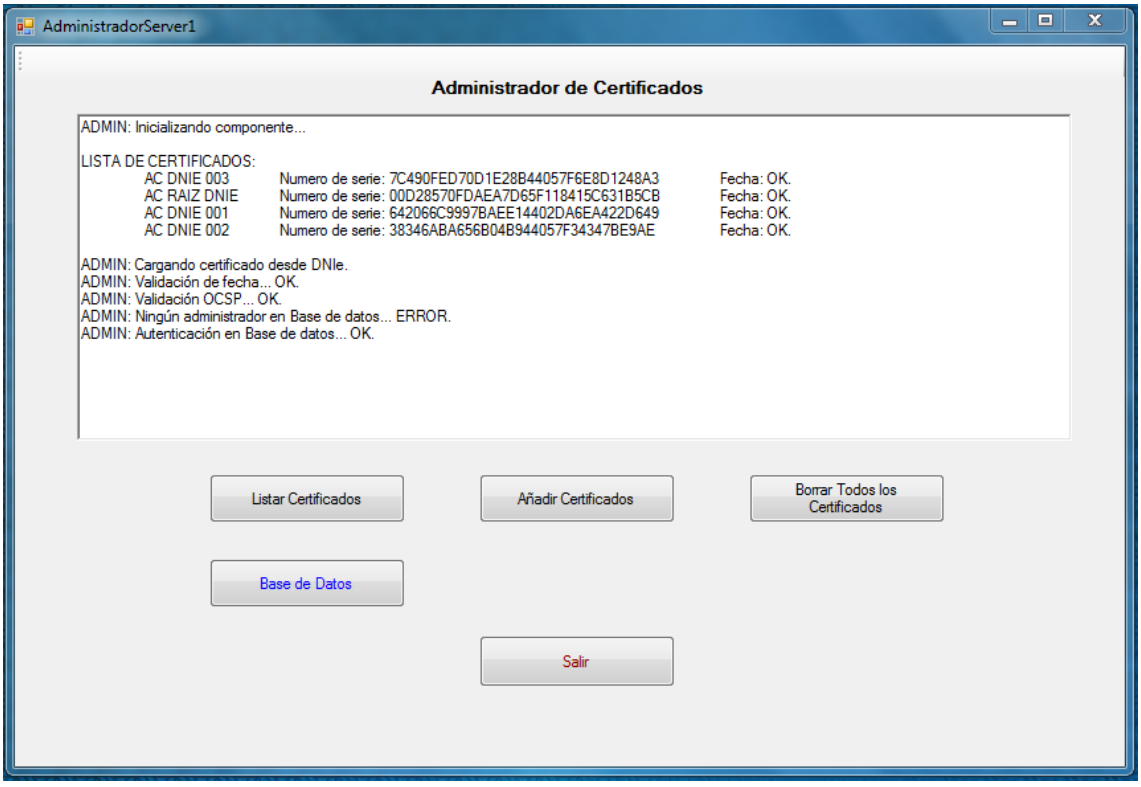

Figura 45. Panel Administrador de Certificados.

En la Figura 46 podemos ver el panel de Administrador de Usuarios donde tendremos una tabla que nos muestra todos los usuarios de la base de datos con sus datos más relevantes, así mismo veremos que podemos agregar, modificar o eliminar usuarios. Del mismo modo existe un botón para volver al panel Administrador de certificados. El tipo de usuario tendrá los valores 1 para Administrador y 2 para Usuario.

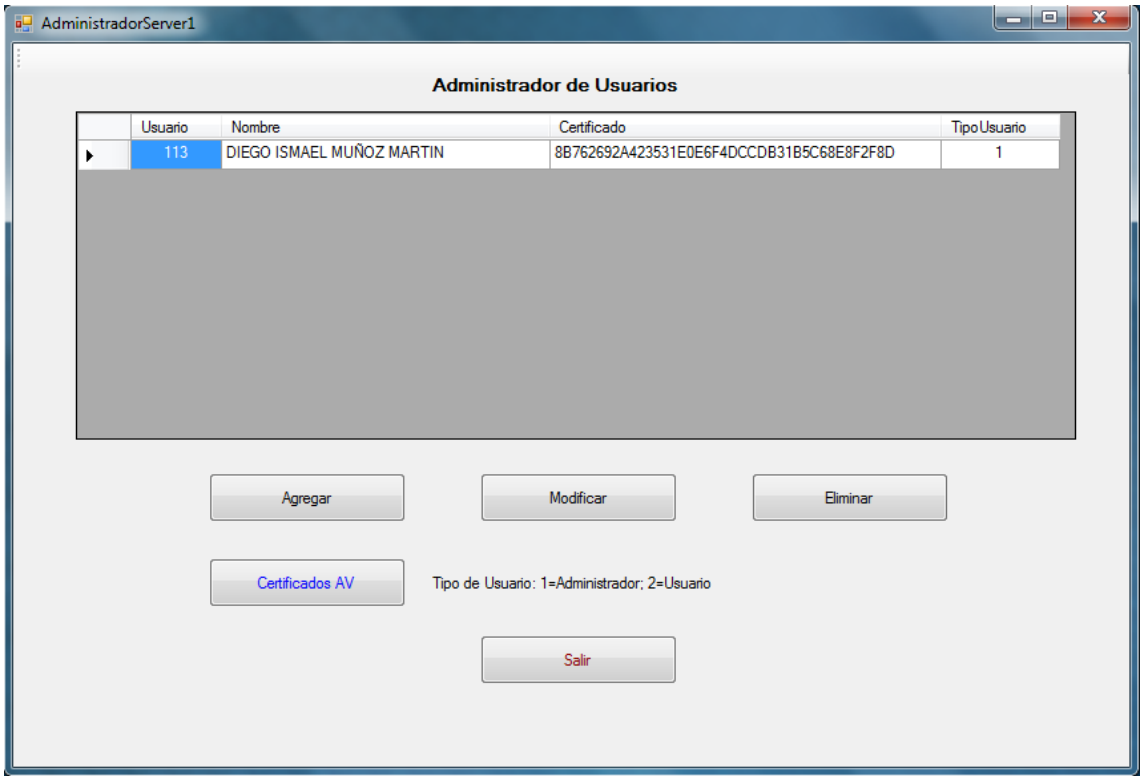

Figura 46. Panel Administrador de Usuarios.

Podemos seleccionar modificar. Nos cargará un panel de herramientas como el de la Figura 47, donde podremos cambiar algunos datos según seleccionemos un registro en la tabla principal.

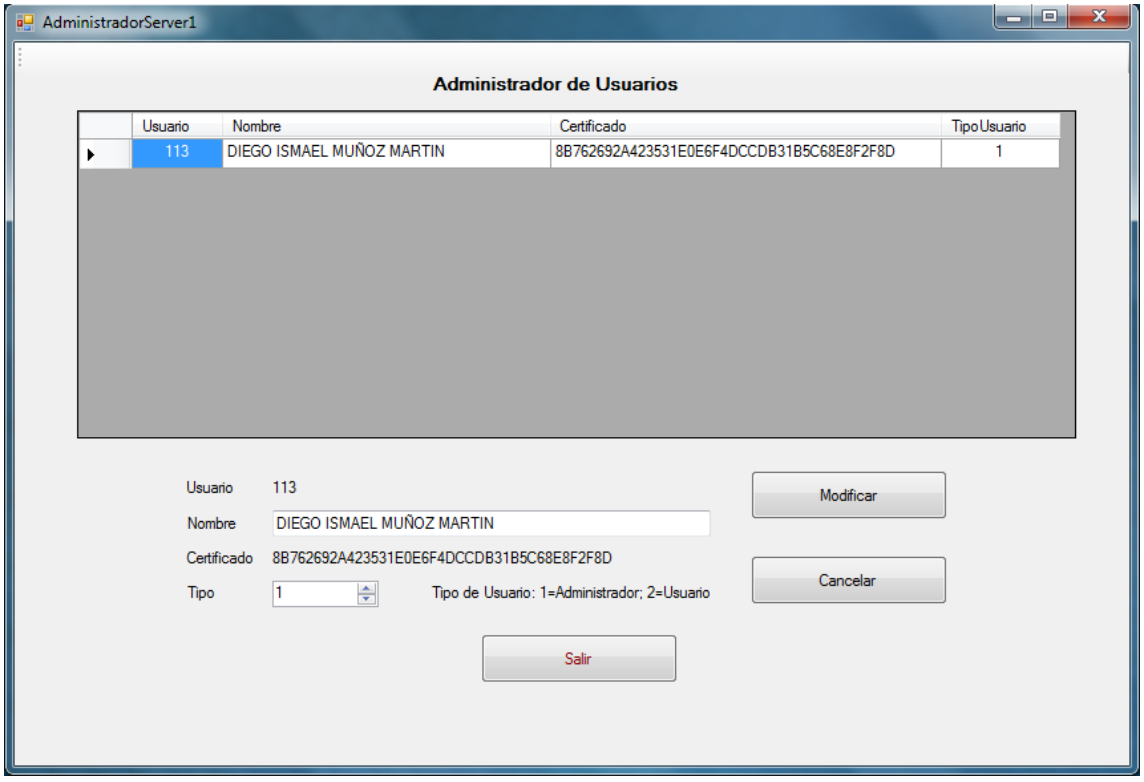

Figura 47. Panel para modificación de registros.

También podemos seleccionar la opción de borrado de usuarios. En este caso el programa nos mostrará las ventanas de la Figura 48 y Figura 49.

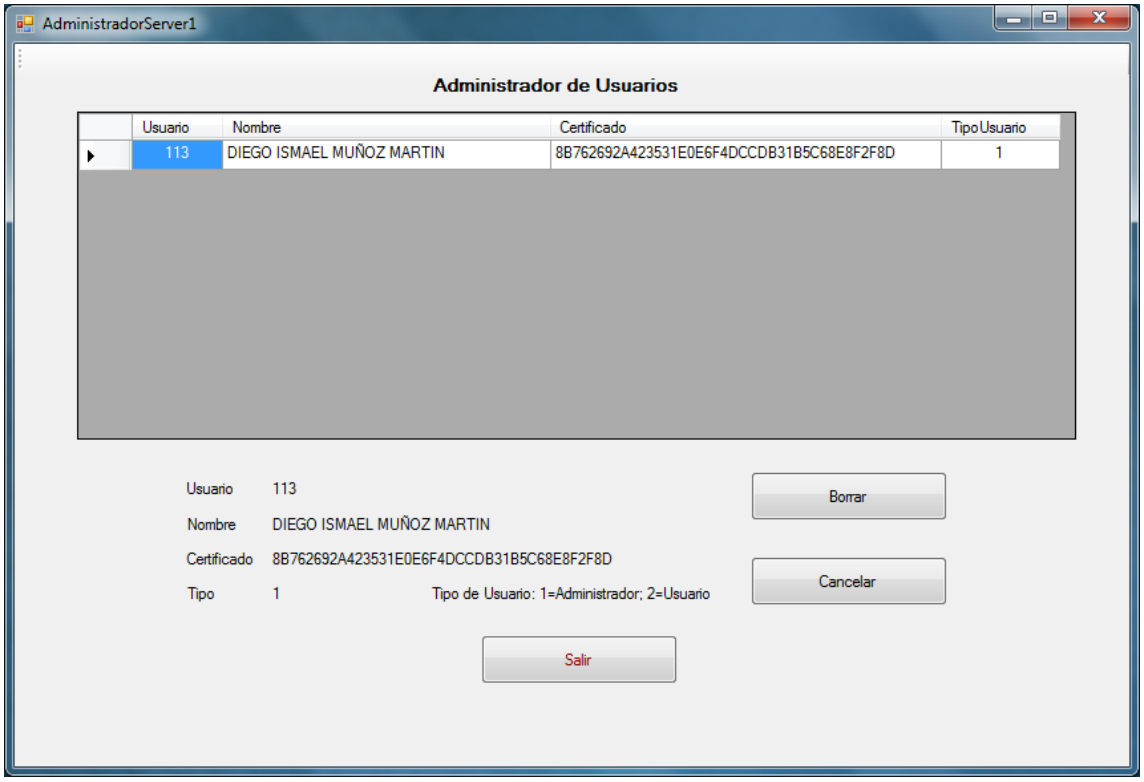

Figura 48. Ventana para borrar usuario de la base de datos.

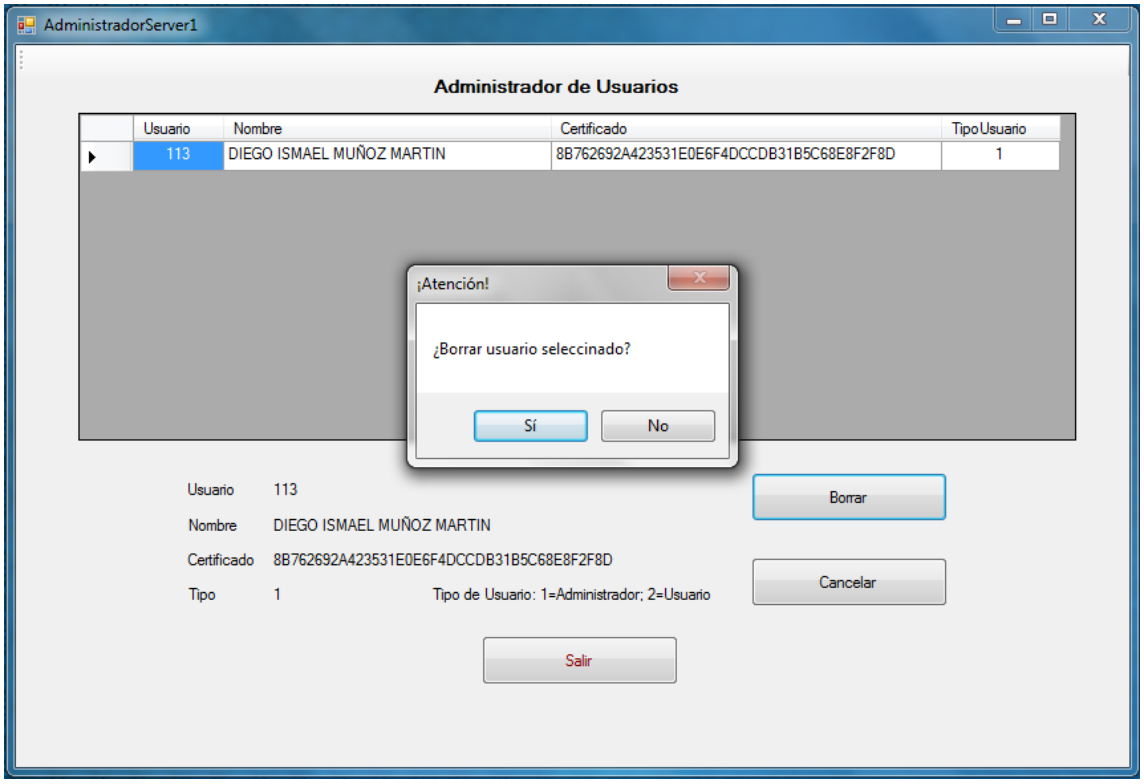

Figura 49. Mensaje de confirmación de borrado de usuario.

En todo momento disponemos de un botón rotulado en rojo, que indica Salir. Con él se cierra la aplicación.

Todos estos pasos quedan registrados en un archivo dentro del directorio LogAdministradorServer1, llamado Trace.txt.

La anotación en el archivo trace.txt, será la siguiente en caso de un acceso correcto:

Fecha: 23/06/2011 15:31:10 Evento:General Comprobación Acceso Usuario (PASA: Inicio OK). Usuario: CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)", G=DIEGO ISMAEL, SN=MUÑOZ, SERIALNUMBER=09457578R, C=ES
)}

Fecha: 23/06/2011 15:31:10 Evento:General Comprobación Acceso Usuario (PASA: Fecha OK). Usuario: CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)", G=DIEGO ISMAEL, SN=MUÑOZ, SERIALNUMBER=09457578R, C=ES
)}

Fecha: 23/06/2011 15:31:11 Evento:General Comprobación Acceso Usuario (PASA: OCSP OK). Usuario: CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)", G=DIEGO ISMAEL, SN=MUÑOZ, SERIALNUMBER=09457578R, C=ES
)}

Fecha: 23/06/2011 15:31:11 Evento:General Comprobación de Acceso Usuario (PASA: Usuario Registrado). Usuario: CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)", G=DIEGO ISMAEL, SN=MUÑOZ, SERIALNUMBER=09457578R, C=ES
)}

La anotación en el archivo trace.txt, será la siguiente en caso de administración de certificados (Borrado de todos los certificados):

Fecha: 23/06/2011 15:31:16 Evento:General Borrar Certificados AV. (PASA: Orden de borrar certificados AV). Usuario: CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)", G=DIEGO ISMAEL, SN=MUÑOZ, SERIALNUMBER=09457578R, C=ES
)}

Fecha: 23/06/2011 15:31:16 Evento:General Comprobación certificados AV. (ERROR: No existe ningún certificado AV). Usuario: ADMIN
)}

Fecha: 23/06/2011 15:31:19 Evento:General Comprobación certificados AV. (ERROR: No existe ningún certificado AC DNIE 002). Usuario: ADMIN
)}

Fecha: 23/06/2011 15:31:21 Evento:General Comprobación certificados AV. (PASA: Se ha cargado el certificado AC DNIE 002). Usuario: ADMIN
)}

Fecha: 23/06/2011 15:31:21 Evento:General Comprobación certificados AV. (ERROR: No existe ningún certificado AC DNIE 003). Usuario: ADMIN
)}

Fecha: 23/06/2011 15:31:23 Evento:General Comprobación certificados AV. (PASA: Se ha cargado el certificado AC DNIE 003). Usuario: ADMIN
)}

La anotación en el archivo trace.txt, será la siguiente en caso de administración de usuarios (Borrar usuario):

Fecha: 23/06/2011 15:31:31 Evento:General Modificar Usuario. (PASA: Orden de Modificar Usuario). Usuario: DIEGO ISMAEL MUÑOZ MARTIN Tipo:2
)}

Fecha: 23/06/2011 15:31:40 Evento:General Borrar Usuario. (PASA: Orden de Añadir Usuario). Usuario: 108 Nombre: 2
)}

La anotación en el archivo trace.txt, será la siguiente en caso de administración de usuarios (Añadir usuario):

Fecha: 23/06/2011 15:31:46 Evento:General Añadir Usuario. (PASA: Orden de Añadir Usuario). Usuario: CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)", G=DIEGO ISMAEL, SN=MUÑOZ, SERIALNUMBER=09457578R, C=ESTipo: 1
)}

La anotación en el archivo trace.txt, será la siguiente en caso de cerrar la aplicación:

Fecha: 23/06/2011 15:31:49 Evento:General Cerrando Aplicación. (PASA: Orden de cerrar la aplicación). Usuario: CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)", G=DIEGO ISMAEL, SN=MUÑOZ, SERIALNUMBER=09457578R, C=ES
)}

5.4.2.- Conexión Remota mediante DNIe.

Para la utilización de este portal, tras la configuración inicial con AdministradorServer1, ya podemos poner en marcha las páginas InicioServer1 y Server1 con el servidor de páginas web IIS 7. Cuando abrimos IIS 7 podemos acceder a las conexiones como puede verse en la Figura 50. Basta con iniciar las páginas InicioServer y Server1 configuradas previamente.

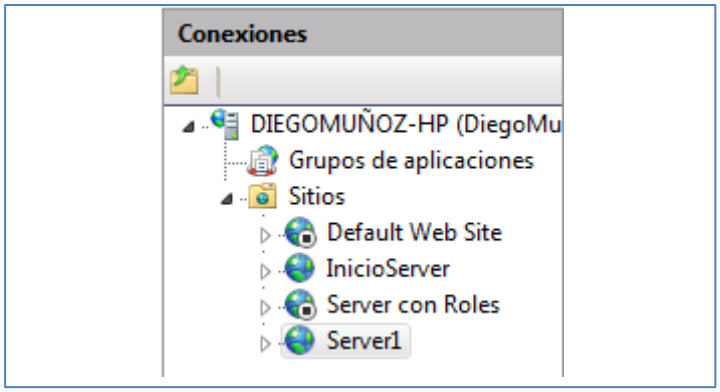

Figura 50. Programa IIS 7. Menú de conexiones.

Abrir un navegador, y poner en la barra de direcciones, la dirección de la página de inicio. En nuestro caso, como puede verse en la Figura 51, será “localhost”.

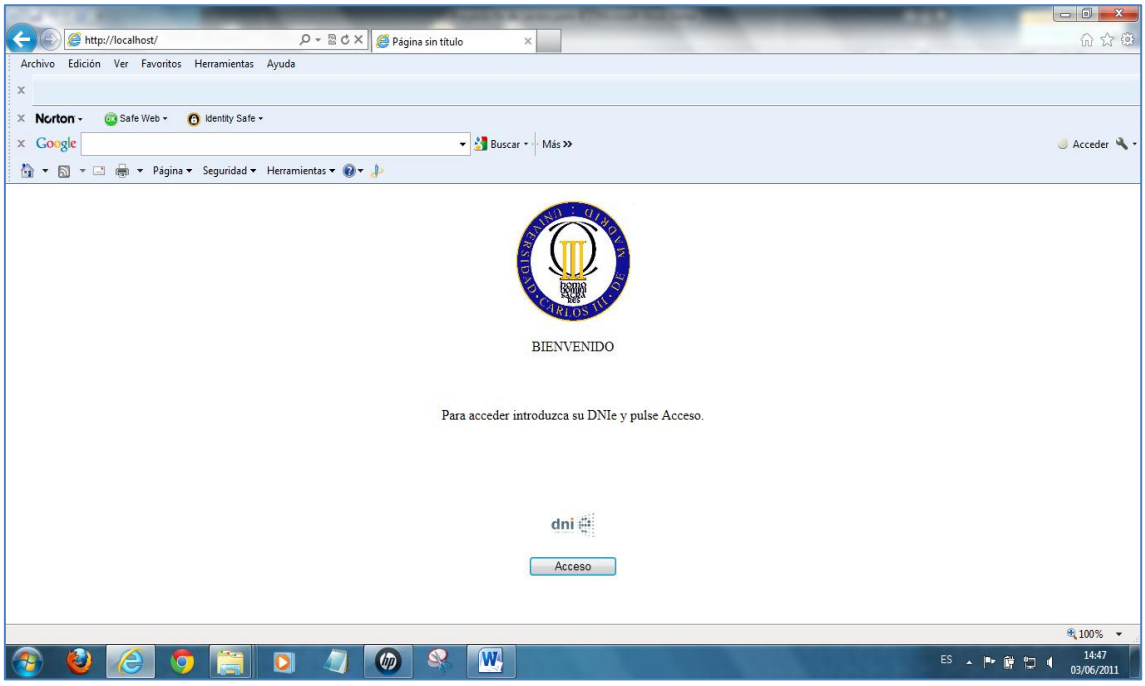

Figura 51. Página web InicioServer.

Al pulsar “Acceso”, IIS 7 establecerá una conexión SSL con nuestro navegador utilizando el certificado de autenticación de nuestro DNIe y el certificado del Servidor. Para acceder al certificado de nuestro DNIe el navegador nos pedirá el PIN. Explorer lo hará a través de CSP, como observamos en la Figura 52.

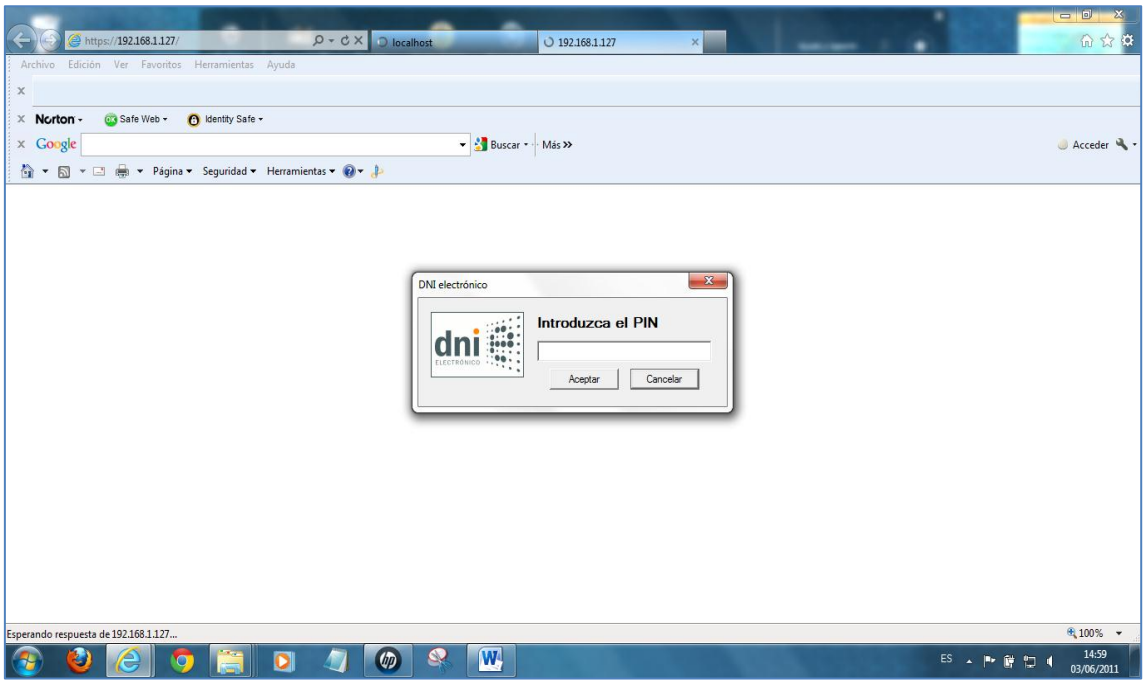

Figura 52. Inicio conexión con sitio web protocolo https. Acceso a DNIe mediante módulo CSP.

Mientras que cuando utilicemos Netscape o Mozilla Firefox se utilizará el módulo criptográfico PKCS11, como podemos observar en la Figura 53.

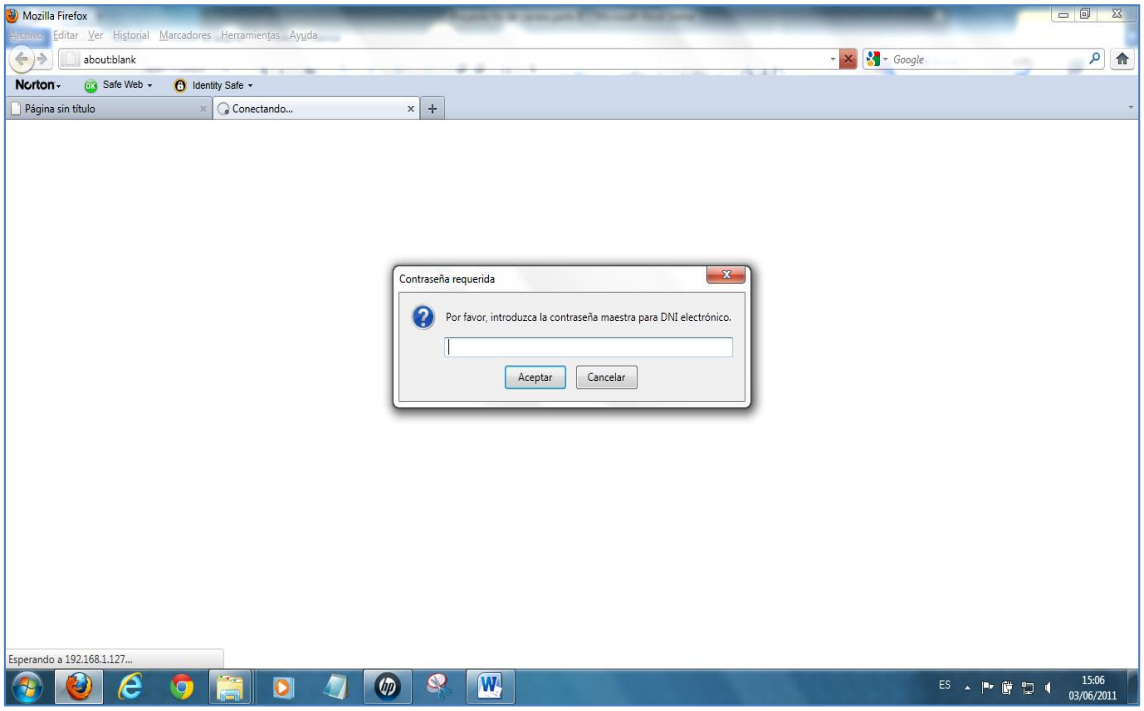

Figura 53. Inicio conexión con sitio web protocolo https. Acceso a DNIe mediante el módulo PKCS11.

Una vez establecido el canal SSL se suministra un certificado de usuario para que el portal haga su comprobación. Esto exigirá que seleccionemos nuestro certificado de autenticación DNIe e introduzcamos nuevamente la clave PIN, como podemos ver en la Figura 54. Internet Explorer de Microsoft seleccionará automáticamente el certificado de autenticación.

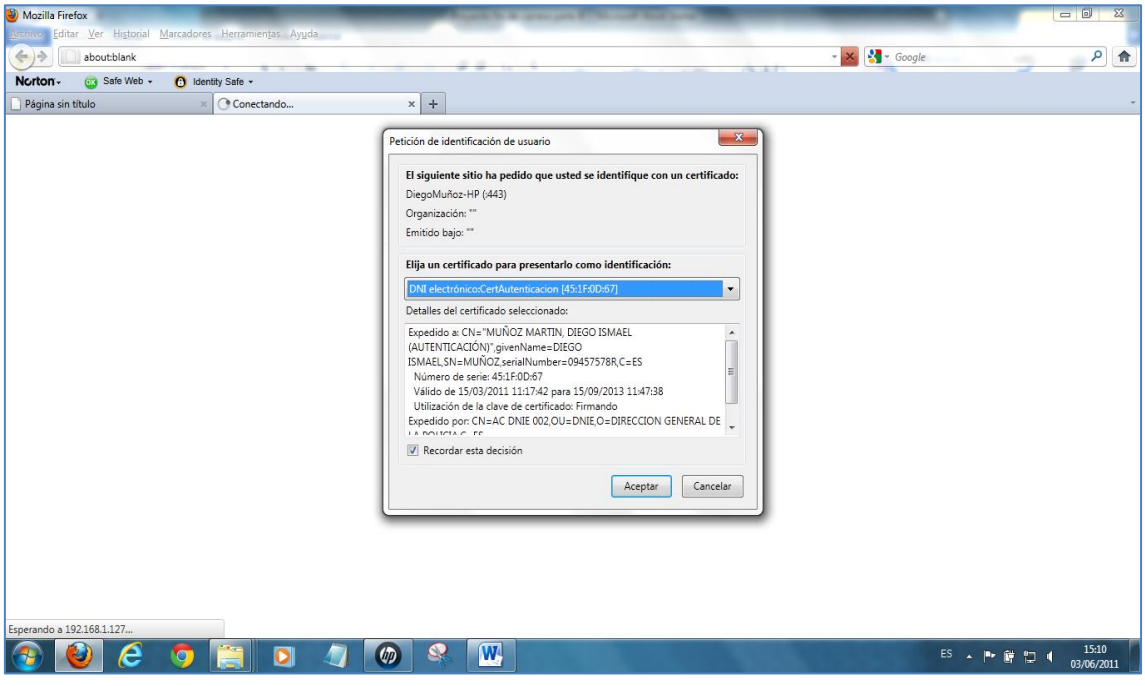

Figura 54. Selección de Certificado de Autenticación.

Llegados a este momento se carga la página de Login, Default.aspx, Figura 55. Si seleccionamos el botón Entrar, se realizarán las comprobaciones pertinentes que desencadenaran en el acceso al sistema, Figura 56, o la redirección a una página de error, Figura 57.

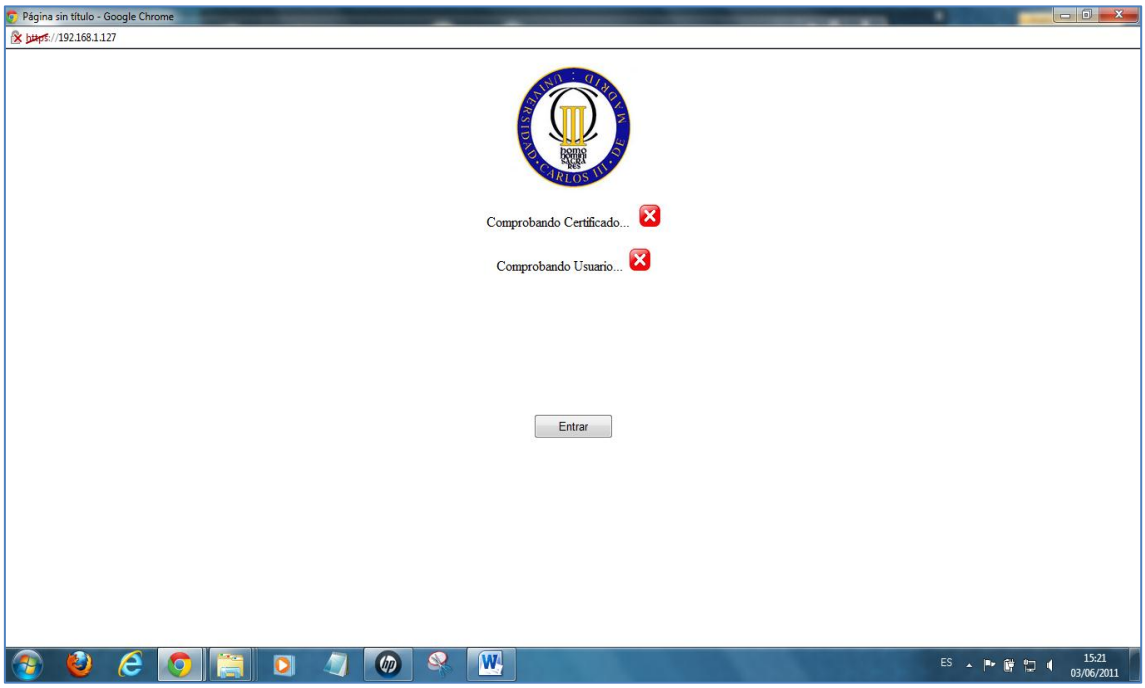

Figura 55. Página default.aspx en Google Chrome.

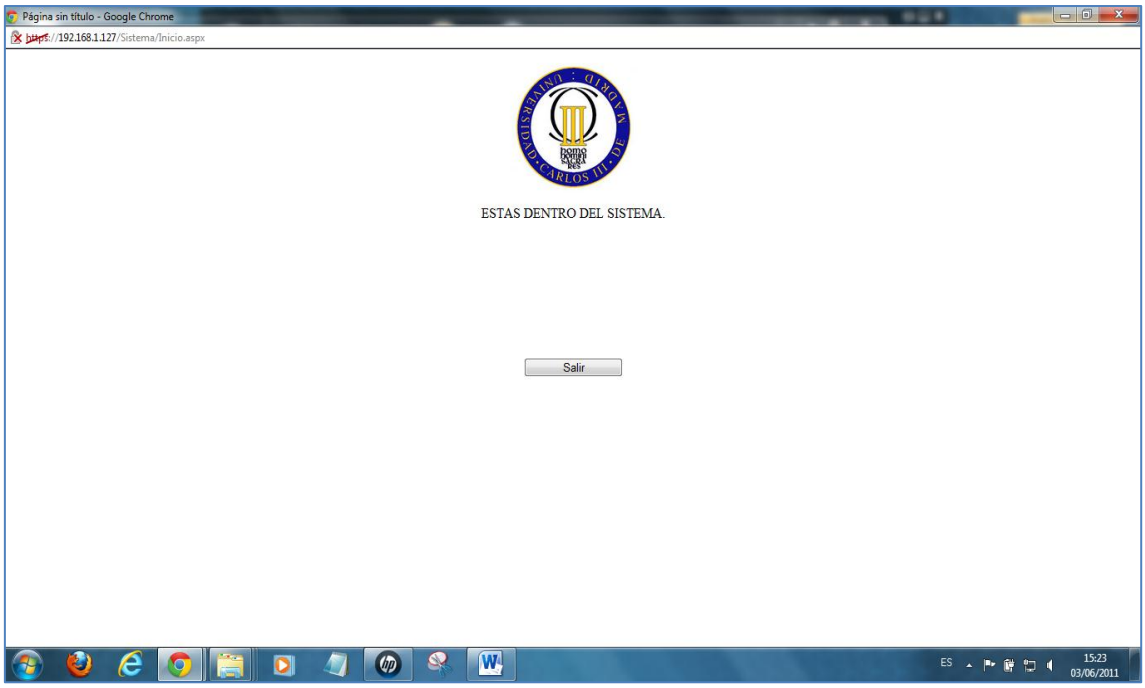

Figura 56. Dentro del Sistema.

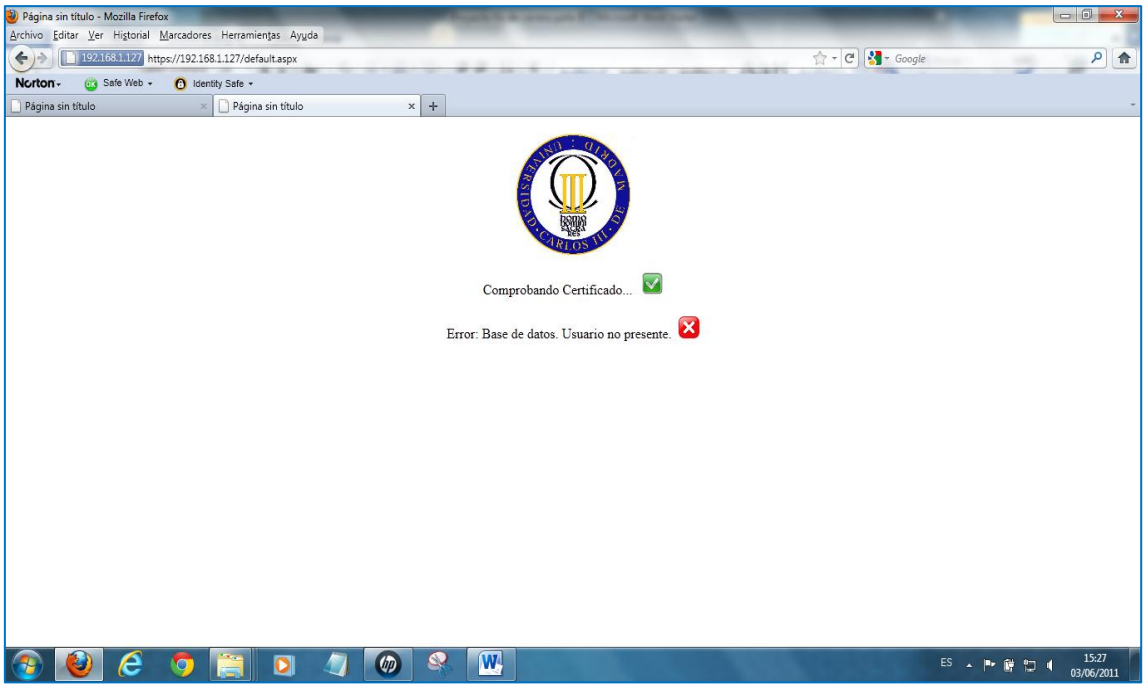

Figura 57. Error en el acceso con usuario no registrado.

Cuando estemos dentro del sistema, en la zona de acceso web privada, vamos a encontrar la web del laboratorio propiamente dicha. Esta web consistirá principalmente en páginas y utilidades de acceso a los recursos y bases de datos del laboratorio. Es un proyecto que se está realizando paralelamente con este PFC y que estará disponible en breve espacio de tiempo.

La anotación en el archivo trace.txt, será la siguiente en caso de un acceso correcto:

Fecha:03/06/2011 13:26:09 Evento:General Comprobación Acceso Usuario (PASA: Inicio OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:26:09 Evento:General Comprobación Acceso Usuario (PASA: Fecha OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:26:09 Evento:General Comprobación Acceso Usuario (PASA: OCSP OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:26:09 Evento:General Comprobación Acceso Usuario (PASA: Autenticación OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:26:09 Evento:General Sistema (PASA: Carga página Inicio OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:26:10 Evento:General Sistema (PASA: Carga página Inicio OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:26:10 Evento:General Sistema (PASA: Salida del Sistema OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

La anotación en el archivo trace.txt, será la siguiente en caso de un acceso incorrecto:

Fecha:03/06/2011 13:27:30 Evento:General Comprobación Acceso Usuario (PASA: Inicio OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (FIRMA)"

Fecha:03/06/2011 13:27:30 Evento:General Comprobación Acceso Usuario (PASA: Fecha OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:27:30 Evento:General Comprobación Acceso Usuario (PASA: OCSP OK). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (AUTENTICACIÓN)"

Fecha:03/06/2011 13:27:30 Evento:General Comprobación Acceso Usuario (ERROR: Usuario no registrado). Usuario: C=ES, SERIALNUMBER=09457578R, SN=MUÑOZ, G=DIEGO ISMAEL, CN="MUÑOZ MARTIN, DIEGO ISMAEL (FIRMA)"

5.4.3.- Operaciones de error del portal.

En la Tabla 8 tenemos todas las posibles operaciones de error que podemos encontrar en nuestro sistema.

Tabla 8. Operaciones de error de acceso a través del portal.

Error 1.0.	Acceso sin certificado Válido.	La comprobación de usuario no se realiza.
Error 1.1.	Acceso con DNIe caducado.	La comprobación de usuario no se realiza.
Error 1.2.	Acceso con DNIe Revocado.	La comprobación de usuario no se realiza.
Error 1.3.	Acceso con DNIe erróneo.	La comprobación de usuario no se realiza.
Error 2.0.	Acceso con DNIe en vigor.	Usuario no registrado.

En el caso de intentar acceder sin certificado válido, la conexión no se establecerá.

En cualquier caso en que el DNIe sea correcto, la aplicación realiza un apunte del intento de acceso y cierra el acceso.

En el caso de que el usuario no este registrado en la base de datos, la aplicación realizará un apunte del intento de acceso y no se permitirá el acceso.

6.-Estudio Económico.

En este apartado se evalúan los costes generados en la realización de este PFC.

En general, en la elaboración de un proyecto una de las primeras cosas que se deben tener en consideración es el estudio de viabilidad. De esta forma se puede establecer el coste aproximado de la inversión, quedando definidos además, los recursos y plazos que se utilizarán para desarrollar la solución.

Para nuestro PFC inicialmente tenemos definidos unos límites temporales que podríamos establecer en el marco de los 6 créditos de la asignatura, distribuidos en un año de curso académico.

6.1.- Actividades.

Tras la realización del PFC vemos que el proceso de elaboración ha pasado por tres fases.

Una primera parte consiste en el análisis de la situación tecnológica de la criptografía en estos momentos. Hemos tenido que hacer una introducción en los entornos de desarrollo aplicados a nuestro proyecto. Y como complemento, un acercamiento al DNIe desarrollando el curso online de Inteco denominado Curso de Desarrollo y Certificación de aplicaciones sobre DNI electrónico.

Como segunda parte tenemos el desarrollo del portal. La construcción de todos los aspectos que conforman el proyecto, desde la instalación y configuración de IIS para utilizar DNIe, pasando por la creación de la lógica de las páginas del portal, la utilización del código para manipular el certificado y la creación y acceso a la base de datos.

La tercera parte de este trabajo engloba la elaboración de toda la documentación. Es una parte que aunque parece trivial ha requerido de un tiempo casi similar al de cualquiera de las otras dos partes.

Por último existe una parte de finalización del proyecto, en la que se revisa el proyecto; se realiza una exposición de la solución; se pueden plantear futuras modificaciones y ampliaciones; tendríamos que tener en cuenta la certificación de la aplicación, etc.

En la Tabla 9 desglosamos los tiempos dedicados al proyecto. El coste de la mano de obra utilizada en el desarrollo de este PFC tiene un precio orientativo de 11,35€/h según convenio para la industria Siderometalúrgica para ingenieros técnicos industriales.

Tabla 9. Tiempo invertido en la elaboración del proyecto.

Descripción.	Tiempo estimado [Horas]
1.- Análisis y documentación	300
2.- Diseño y Desarrollo.	300
3.- Memoria.	250
4.- Revisión.	50
5.- Total.	900

6.2.- Recursos.

Otro punto importante a cuantificar es el uso de recursos como equipos, licencias de programas, infraestructuras, seguros, etc.

El equipo empleado tiene fecha de compra 5 de Agosto de 2010 a un precio de 668 Euros. Estaremos pues dentro del periodo de depreciación. El tiempo de utilización de este recurso será el mismo que el número de horas de mano de obra.

Al utilizar software de desarrollo con licencia de estudiante y software libre en este apartado podemos valorar un coste cero.

El editor de texto sin embargo con licencia de estudiante nos ha costado 69 euros y lo vamos a incluir como gastos directos del proyecto.

En cuanto a material de oficina, infraestructuras, desplazamientos, etc. vamos a hacer una estimación del 10% del total de mano de obra.

6.3.- Coste total.

En la Tabla 10 se describe el coste total del PFC. No parece un precio elevado 13.624 euros, gracias en gran medida a la utilización de recursos gratuitos.

En un caso fuera del ámbito académico, aunque el número de horas empleados debería ser mucho menor, ya que utilizaríamos personal más especializado en recursos software y sufriríamos unos incrementos de coste ya que el precio de la mano de obra seguramente podría elevarse y, el coste de los recursos se incrementaría sensiblemente al tener que incluir costes de infraestructuras, amortización de las inversiones de material, licencias, seguros, etc. Es decir, el precio del proyecto tiende a incrementarse.

Tabla 10. Costes totales del proyecto.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: Diego Ismael Muñoz

2.- Departamento: Tecnología Electrónica

3.- Descripción del Proyecto:

- Titulo: TRANSMISION DE DATOS Y SEGURIDAD EN TECNOLOGÍAS DE IDENTIFICACIÓN CON DNIE.
- Duración (meses): 6,84
- Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):

Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Diego Ismael Muñoz Martín	9457578 R	Alumno	6,84	1.489,69	10.189,48	
		Ingeniero Senior		4.289,54	0,00	
		Ingeniero		2.694,39	0,00	
					0,00	
					0,00	
		Hombres mes	**6,84**	**Total**	10.189,48	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador	668,00	100	7	60	76,15
		100		60	0,00
		100		60	0,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	76,15

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Editor de Texto		69,00
Material fungible, viajes y dietas		1.018,95
	Total	1.087,95

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	10.189
Amortización	76
Subcontratación de tareas	0
Costes Directos	1.088
Costes Indirectos	2.271
Total	13.624

7.- Conclusiones.

El tema de la seguridad en la transmisión de datos es complejo, pero, existen una serie de mecanismos, todos ellos públicos y ampliamente comprobados, que son los que debemos utilizar. En cualquier caso estos sistemas no están exentos de fallos (caso PlayStation 3, caso Debian, http://www.debian.org/security/2008/dsa-1571) que se pueden introducir en nuestro código, por ello es necesario hacer códigos adaptivos y fáciles de actualizar.

Como reflexión particular, indicar que el desarrollo en Java de soluciones software, para trabajo de transmisión de datos y utilización del DNIe está mucho más avanzado y que en la mayoría de los casos, las soluciones C# son adaptaciones a este lenguaje desde el lenguaje Java. Esto no es ningún problema en la línea académica de investigación que estamos siguiendo, pero, sería un punto importante a tener en cuenta en aplicaciones comerciales. Vemos de todas formas, que aunque Java fue la primera línea de desarrollo de estas tecnologías, la distancia entre Java y C# cada día es menor, y que las cosas cambian tanto que desde el comienzo de la elaboración de este PFC hasta el día de hoy esa línea de separación casi ha desaparecido.

7.1.- Evoluciones futuras.

Un asunto que no hemos tratado en este PFC es, la posibilidad de acceso para personas que no posean el DNIe. Si bien la plataforma DNIe nos da ventajas estratégicas en temas de seguridad y distribución en España, tendremos que ver cómo podemos adaptar nuestro portal de acceso a los distintos documentos nacionales de identidad de los posibles usuarios no nacionalizados.

Una solución evidente es la de proporcionarles tarjetas inteligentes implementando nuestras propia estructura de clave pública. En ese caso la Autoridad de certificación seremos nosotros mismos y nuestras tarjetas inteligentes tendrán que implementen las funcionalidades del DNIe.

La otra solución es que se nacionalicen españoles y accedan a un DNIe. El uso de este portal sería una buena razón subjetiva que, junto con otras más concretas como son las evidentes ventajas de trabajar en España (la protección, promoción, prestigio, proyección laboral, etc.), clima, cultura, desarrollo, etc.; hacen que los usuarios de este portal, no tengan problema en solucionar este punto. Pensamos que esta segunda solución es más sencilla u económica que crear una estructura independiente de clave pública y la suscribimos.

7.2.- Otras Tecnologías.

Existe un tipo de software comercial denominado Sistemas de Gestión del Producto (PDM). Este tipo de programas están orientados a la gestión de los recursos dentro del sistema de desarrollo de productos. Este tipo de software entendemos que podría ser el adecuado para implantar en un laboratorio como gestor de la base de datos de los distintos proyectos e investigadores.

8.- Bibliografía.

8.1.- Libros.

- ANDREW S., Tanenbaum. *Redes de Computadoras.* MORALES PEAKE, David (trad.); GUERRERO REYES, Gabriel (rev. tec.). 3ª. Ed. México: Prentice Hall Hiberoamericana, 1997. 812p. ISBN 968-880-958-6.

- DOMINGO SANDOVAL, Juan; BRITO GARCIA, Ricardo; MAYOR GONZALEZ, Juan Carlos. *Tarjetas inteligentes.* 1ª Ed. Madrid: Ed. Paraninfo, 1999. 209p. ISBN 84-283-2602-9.

- CHARTE OJEDA, Francisco. *ASP.NET 3.5.* 1ºEd.Madrid: Ediciones Anaya Multimedia, 2009. 320p. ISBN 978-84-415-2502-3.

- AL ZABIR, Omar. *Cómo crear un portal web 2.0 con ASP.NET 3.5.*1ª Ed. Madrid: Ediciones Anaya Multimedia, 2009. 350p. ISBN 978-84-415-2465-1.

- DOBSON, Rick. *Programación de Microsoft SQL Server 2000 con Microsoft Visual Basic .NET.* BAUTISTA, José Antonio (trad.). 1ª Ed. Madrid: Ediciones McGraw-Hill/Interamericana de España, 2002. 646p. ISBN 84-481-3721-3.

8.2.-Sitios Web.

[1] Centro de desarrolladores de visual C#:

http://msdn.microsoft.com/es-es/vcsharp/default.aspx

[2] Criptografía. Wikipedia, la enciclopedia libre:

http://es.wikipedia.org/wiki/Criptograf%C3%ADa

[3] Kriptopolis. Criptografía, Privacidad y Seguridad en Internet:

http://www.kriptopolis.org/

[4] Internet Engineering Task Force (IETF):

http://www.ietf.org/

[5] European Telecommunications Standards Institute (ETSI):

http://www.etsi.org/

[6] Comité Europeo de Normalización (CEN):

http://www.cen.eu/

[7] International Telecommunication Union – "Telecommunication Standardization Sector" (ITU-T):

http://itu.int/ITU-T/

[8] ISO/International Electrotechnical Comision (IEC):

http://www.iec.ch/

[9] DNIe. DECLARACIÓN DE PRÁCTICAS Y POLÍTICAS DE CERTIFICACIÓN (Versión 1.0–6 Marzo 2006):

http://www.dnie.es/dpc

[10] DNIe. Autoridad de Validación mediante OCSP.

http://ocsp.dnie.es/

[11] DNIe. Autoridad de Certificación. Certificado Electrónico AC Raíz:

http://www.dnie.es/certs/ACraiz.crt/

[12] Real Decreto 14/1999 de 17 de septiembre:

http://www.boe.es/boe/dias/1999/09/18/pdfs/A33593-33601.pdf

[13] Directiva 98/34/CE del Parlamento Europeo:

http://eur-ex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1998L0034:19980810:ES:PDF

[14] Ley 59/2003 de 19 de diciembre:

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

[15] Directiva 99/93/CE del Parlamento Europeo:

http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2000:013:0012:0020:ES:PDF

[16] Organización para el Avance de las Normas de Información Estructurada (OASIS):

http://www.oasis-open.org/

[17] Empresa RSA (The Security Division of EMC):

http://www.rsa.com/

[18] Portal Oficial sobre DNI Electrónico:

http://www.DNIe.es/

[19] Zona Tic (Usatudni):

https://zonatic.usatudni.es/

[20] Advancing Open Standards for the global information society:

http://www.oasis-open.org/home/index.php

[21] Debian Security Information.DSA-1571-1 openssl:

http://www.debian.org/security/2008/dsa-1571

[22]Portal oficial Common Criteria:

http://www.commoncriteriaportal.org

[23] En el caso de España en la actualidad operan tres laboratorios con licencia del CCN (Centro Cristológico Nacional):

APPLUS+. www.applus.com

CESTI. www.inta.es/certificaciones.aspx

EPOCHE&ESPRI. www.epoche.es

[24]Proyecto Bouncy Castle:

www.bouncycastle.com

[25]Enterprise Library 4.0. :

http://msdn.microsoft.com/en-us/library/ff650810.aspx

[26] DNIe ToolKit:

http://DNIetoolkit.codeplex.com

[27]Configuración de IIS 7:

https://zonatic.usatudni.es/es/aprendizaje/aprende-sobre-el-DNIe/58-desarrolla-con-el-dni-electronico/226-internet-information-services-iis.html

[28] Aplicaciones Shareware ABylon SelfCert con la que crear certificados en formato X509:

http://www.abylonsoft.com

[29] Área de seguridad y comunicación de la Universidad Carlos III de Madrid.

http://asyc.uc3m.es

8.3.- Referencias.

8.3.1. - Normas y guías técnicas de referencia.

FIPS:

- [FIPS 180-2] Secure Hash Signature Standard (SHA2).

RFC's:

- [RFC2311] S/MIME Version 2 - PKCS#7.
- [RFC2510] Internet X.509 Public Key Infrastructure Certificate Management Protocols.

- [RFC2511] Internet X.509 Certificate Request Message Format.
- [RFC2560] Internet X.509 PKI Online Certificate Status Protocol (OCSP).
- [RFC2986] PKCS #10: Certification Request Syntax Specification.
- [RFC3039] Internet X.509 Public Key Infrastructure Qualified Certificates Profile.
- [RFC3126] Electronic Signature Formats for long term electronic signatures.
- [RFC3161] Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP).
- [RFC3174] US Secure Hash Algorithm 1 (SHA1).
- [RFC3183] Domain Security Services using S/MIME.
- [RFC3261] SIP: Session Initiation Protocol.
- [RFC3279] Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.
- [RFC3280] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Sustituye al antiguo [RFC2459].
- [RFC3281] An Internet Attribute Certificate Profile for Authorization.
- [RFC3369] Cryptographic Message Syntax (CMS).
- [RFC3370] Cryptographic Message Syntax (CMS) Algorithms.
- [RFC3447] Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 -Generación de las Claves bajo el formato PKCS#1.
- [RFC3546] Transport Layer Security (TLS) Extensions.
- [RFC3565] Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS).
- [RFC3647] Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework Sustituyento al antiguo [RFC2527].

ETSI TS

- Certificados cualificados Profile ETSI TS 101 862 v1.1.1.
- Formatos de firma electrónica ETSI TS 101 733 v1.2.2.
- Requerimientos CA ETSI TS 101 456 v1.1.1 Time Stampin Profile -ETSI TS 101 861.
- Policy requirements for certification authorities issuing public key certificates ETSI TS 102 042 V1.1.1 (2002-04).

CWA

- CWA 14167-1.
- CWA 14167-2.
- CWA 14169.

ISO

- 15408 Common Criteria.
- 10181-4:1997, Information technology -- Open Systems Interconnection – “Security frameworks for open systems”.
- 17799:2000 Code of Practice for Information Security Management.
- 9594-8:2001 Information technology - Estructura de Certificados -Systems Interconnection – The Directory attribute certificate framework”.

- 8825-1 8825-1 Estructura de certificados - ASN.1 Basic Encoding Rules.
- 9796 Tecnología de la Información Técnicas de Seguridad -Mecanismo de Firma Digital ("Information Technology - Security Techniques - Digital Signature Scheme").
- 7816-15 Identification cards ("Integrated circuit(s) cards with contacts -- Part 1: Physical characteristics- Encargado de mantener el formato PKCS#15 en soporte SmartCard").
- 7498-2, Jul. 1988 Describe el modelo de referencia OSI, presenta en su parte 2 una Arquitectura de seguridad. ("Information Processing Sistems. OSI Reference model- Part 2: Security Architecture". ISO/IEC IS 7498-2, Jul. 1988).
- 3166-1 Códigos de país.

PKCS RSA

- PKCS# 15 v.1.1 en soporte software.
- PKCS# 11.
- PKCS# 10.
- PKCS# 7.

ITU - T

- AttributeCertificateDefinitions (X.509:03/2000).
- Module AuthenticationFramework (X.509:03/2000).
- Module CertificateExtensions (X.509:03/2000).
- Module AlgorithmObjectIdentifiers (X.509:03/2000).
- Module AlgorithmObjectIdentifiers (X.509:03/2000).
- ASN.1 Modules from Recommendation X.501.
- ASN.1 Modules from Recommendation X.520.

DNIe. *DECLARACIÓN DE PRÁCTICAS Y POLÍTICAS DE CERTIFICACIÓN* (Versión 1.0–6 Marzo 2006). http://www.dnie.es/dpc.

8.3.2.- Legislación vigente.

L21/92 Ley 21/1992, de 16 de julio, de Industria.

L59/03 Ley 59/2003, de 19 de diciembre, de firma electrónica.

RD2200/95 RD 2200/1995, de 28 de diciembre, por el que se aprueba el Reglamento de la Infraestructura para la calidad y seguridad industrial.

RD 1553/2005, de 23 de diciembre, permitirá al ciudadano establecer sus relaciones de confianza con terceros a través de las nuevas tecnologías.

D1999/93/CE Directiva de la CE sobre firma electrónica.

Anexos.

- En ellos se reflejan en versión original comentada, todos los aspectos que tendríamos que tener en cuenta a la hora de abordar la certificación de nuestra aplicación, que se encuentran en los manuales de CC v 3.1.

http://www.commoncriteriaportal.org .

Anexo A. Requisitos de garantía de seguridad. EAL 1.

El desarrollo y evaluación del TOE se realizará conforme al siguiente nivel de garantía:

- EAL 1.

En este apartado se nos indican los componentes básicos que necesitamos elaborar en un nivel de seguridad EAL 1. Podemos ver un resumen en la Tabla 11 de cómo se reflejan estos requisitos en el proyecto.

Tabla 11. Correspondencia entre los requisitos de garantía de seguridad y el proyecto.

CONDICIONES	ESPECIFICACIONES EN PROYECTO
ADV_FSP.1 Especificaciones de funcionamiento básicas.	Capítulo 3. Especificaciones y Requisitos.
AGD_OPE.1 Guía de Funcionamiento.	Capítulo 5.4. Operaciones de uso.
AGD_PRE.1 Guía de instalación.	Capítulo 5.2. Configuración del entorno de desarrollo.
ALC_CMC.1 Etiquetado del TOE.	Capítulo 3. Especificaciones y Requisitos.
ALC_CMS.1 Cobertura del TOE.	Capítulo 3. Especificaciones y Requisitos.
ASE_INT.1 Introducción de la declaración de seguridad.	Capítulo 3.1 Tipo de aplicación.
ASE_CCL.1 Declaración de conformidad.	Pendiente.
ASE_OBJ.1 Definición ampliada de los requisitos de seguridad.	Capítulo 3.5. Requisitos de Seguridad.
ASE_ECD.1 Definición extendida de los componentes.	Capítulo 3.4. Requisitos de Operacionales.
ASE_REQ.1 Requisitos de seguridad inicial.	Capítulo 3.5. Requisitos de Seguridad.
ASE_TSS.1 Resumen del TOE.	Introducción Capítulo 3. Especificaciones y Requisitos-
ATE_IND.1 Pruebas independientes de conformidad.	Pendiente.
AVA_VAN.1 Vulnerabilidades.	Capítulo 5.4.3. Operaciones de error.

A continuación se detallan los requisitos que podemos encontrar en la norma Common Criteria para un nivel de seguridad EAL 1.

1.- ADV_FSP.1 Basic functional specification.

Dependencies:

- No dependencies.

Developer action elements:

- ADV_FSP.1.1D The developer shall provide a functional specification.
- ADV_FSP.1.2D The developer shall provide a tracing from the functional specification to the SFRs.

Content and presentation of evidence elements:

- ADV_FSP.1.1C The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.
- ADV_FSP.1.2C The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.
- ADV_FSP.1.3C The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.
- ADV_FSP.1.4C The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2.- AGD_OPE.1 Operational user guidance.

Dependencies:

- ADV_FSP.1 Basic functional specification

Developer action elements:

- AGD_OPE.1.1D The developer shall provide operational user guidance.

Content and presentation of evidence elements:

- AGD_OPE.1.1C The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.
- AGD_OPE.1.2C The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.
- AGD_OPE.1.3C The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.
- AGD_OPE.1.4C The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.
- AGD_OPE.1.5C The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.
- AGD_OPE.1.6C The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.
- AGD_OPE.1.7C The operational user guidance shall be clear and reasonable.

3.- AGD_PRE.1 Preparative procedures.

Dependencies:

- No dependencies.

Developer action elements:

- AGD_PRE.1.1D The developer shall provide the TOE including its preparative procedures.

Content and presentation of evidence elements:

- AGD_PRE.1.1C The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.
- AGD_PRE.1.2C The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

4.- ALC_CMC.1 Labeling of the TOE.

Dependencies:

- ALC_CMS.1 TOE CM coverage

Developer action elements:

- ALC_CMC.1.1D The developer shall provide the TOE and a reference for the TOE.

Content and presentation of evidence elements:

- ALC_CMC.1.1C The TOE shall be labeled with its unique reference.

5.- ALC_CMS.1 TOE CM coverage.

Dependencies:

- No dependencies.

Developer action elements:

- ALC_CMS.1.1D The developer shall provide a configuration list for the TOE.

Content and presentation of evidence elements:

- ALC_CMS.1.1C The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.
- ALC_CMS.1.2C The configuration list shall uniquely identify the configuration items.

6.- ASE_INT.1 ST introduction

Dependencies:

- No dependencies.

Developer action elements:

- ASE_INT.1.1D The developer shall provide an ST introduction.

Content and presentation of evidence elements:

- ASE_INT.1.1C The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.

- ASE_INT.1.2C The ST reference shall uniquely identify the ST.
- ASE_INT.1.3C The TOE reference shall identify the TOE.
- ASE_INT.1.4C The TOE overview shall summarize the usage and major security features of the TOE.
- ASE_INT.1.5C The TOE overview shall identify the TOE type.
- ASE_INT.1.6C The TOE overview shall identify any non-TOE hardware/software/firmware required by the TOE.
- ASE_INT.1.7C The TOE description shall describe the physical scope of the TOE.
- ASE_INT.1.8C The TOE description shall describe the logical scope of the TOE.

7.- ASE_CCL.1 Conformance claims.

Dependencies:

- ASE_INT.1 ST introduction
- ASE_ECD.1 Extended components definition
- ASE_REQ.1 Stated security requirements

Developer action elements:

- ASE_CCL.1.1D The developer shall provide a conformance claim.
- ASE_CCL.1.2D The developer shall provide a conformance claim rationale.

Content and presentation of evidence elements:

- ASE_CCL.1.1C The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.
- ASE_CCL.1.2C The CC conformance claim shall describe the conformance of the ST to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.
- ASE_CCL.1.3C The CC conformance claim shall describe the conformance of the ST to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.
- ASE_CCL.1.4C The CC conformance claim shall be consistent with the extended components definition.
- ASE_CCL.1.5C The conformance claim shall identify all PPs and security requirement packages to which the ST claims conformance.
- ASE_CCL.1.6C The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.
- ASE_CCL.1.7C The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.
- ASE_CCL.1.8C The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.
- ASE_CCL.1.9C The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PPs for which conformance is being claimed.
- ASE_CCL.1.10C The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.

8.- ASE_OBJ.1 Security objectives for the operational environment.

Dependencies:

- No dependencies.

Developer action elements:

- ASE_OBJ.1.1D The developer shall provide a statement of security objectives.

Content and presentation of evidence elements:

- ASE_OBJ.1.1C The statement of security objectives shall describe the security objectives for the operational environment.

9.- ASE_ECD.1 Extended components definition.

Dependencies:

- No dependencies.

Developer action elements:

- ASE_ECD.1.1D The developer shall provide a statement of security requirements.
- ASE_ECD.1.2D The developer shall provide an extended components definition.

Content and presentation of evidence elements:

- ASE_ECD.1.1C The statement of security requirements shall identify all extended security requirements.
- ASE_ECD.1.2C The extended components definition shall define an extended component for each extended security requirement.
- ASE_ECD.1.3C The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.
- ASE_ECD.1.4C The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.
- ASE_ECD.1.5C The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements can be demonstrated.

10.- ASE_REQ.1 Stated security requirements.

Dependencies:

- ASE_ECD.1 Extended components definition

Developer action elements:

- ASE_REQ.1.1D The developer shall provide a statement of security requirements.
- ASE_REQ.1.2D The developer shall provide a security requirements rationale.

Content and presentation of evidence elements:

- ASE_REQ.1.1C The statement of security requirements shall describe the SFRs and the SARs.
- ASE_REQ.1.2C All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.

- ASE_REQ.1.3C The statement of security requirements shall identify all operations on the security requirements.
- ASE_REQ.1.4C All operations shall be performed correctly.
- ASE_REQ.1.5C Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
- ASE_REQ.1.6C The statement of security requirements shall be internally consistent.

11.- ASE_TSS.1 TOE summary specification.

Dependencies:

- ASE_INT.1 ST introduction
- ASE_REQ.1 Stated security requirements
- ADV_FSP.1 Basic functional specification

Developer action elements:

- ASE_TSS.1.1D The developer shall provide a TOE summary specification.

Content and presentation of evidence elements:

- ASE_TSS.1.1C The TOE summary specification shall describe how the TOE meets each SFR.

12.- ATE_IND.1 Independent testing – conformance.

Dependencies:

- ADV_FSP.1 Basic functional specification
- AGD_OPE.1 Operational user guidance
- AGD_PRE.1 Preparative procedures

Developer action elements:

- ATE_IND.1.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

- ATE_IND.1.1C The TOE shall be suitable for testing.

13.- AVA_VAN.1 Vulnerability survey.

Dependencies:

- ADV_FSP.1 Basic functional specification
- AGD_OPE.1 Operational user guidance
- AGD_PRE.1 Preparative procedures

Developer action elements:

- AVA_VAN.1.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

- AVA_VAN.1.1C The TOE shall be suitable for testing.

Anexo B. Requisitos funcionales de seguridad.

1.- Requisitos para garantizar la integridad de los datos por canal SSL.

- El autor de la declaración de seguridad dispone que el canal de comunicaciones hereda todas las garantías que tiene el canal SSL que establecemos con IIS 7 y los navegadores y el canal de comunicaciones con el dispositivo seguro gestionado con los drives de DNIe.

1.1.- FTP_ITC.1. Inter-TSF confidentiality during transmission.

FTP_ITC.1.1 The TSF shall provide a communication channel between itself and **the SSCD** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2 The TSF shall permit [selection: *la TSF*] to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted cannel.

1.2.- FTP_TRP.1 Trusted path.

FTP_TRP.1.1 The TSF shall provide a communication path between itself and [selection: remote, local] users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure.

FTP_TRP.1.2 The TSF shall permit [selection: the TSF, local users, remote users] to initiate communication via the trusted path.

FTP_TRP.1.3 The TSF shall require the use of the trusted path for [selection: initial user authentication, [assignment: other services for which trusted path is required]]. Page

2.- Requisitos para garantizar la validación del certificado de usuario.

- El autor de la declaración de seguridad define que solo son válidos para la identificación del usuario los certificados DNIe y que se comprobarán mediante conexión OCSP con ocsp.dnie.es es utilizando los certificados de Autoridad de Validación DNIe vigentes.

2.1.- FDP_DAU.2 Data Authentication with Identity of Guarantor.

FDP_DAU.2.1 The PP/ST author should specify the list of objects or information types for which the TSF shall be capable of generating data authentication evidence.

FDP_DAU.2.2. The PP/ST author should specify the list of subjects that will have the ability to verify data authentication evidence for the objects identified in the previous element as well as the identity of the user that created the data authentication evidence.

3.- Requisitos para garantizar la autenticación del usuario.

El autor de la declaración de seguridad define que la gestión de la base de datos es totalmente autónoma del portal de acceso y que a través de un canal seguro, esa base de datos validará la información suministrada por el portal en forma resumen criptográfico SHA 1 de un parámetro de control.

3.1.- FDP_DAU.1 Basic Data Authentication.

FDP_DAU.1.1. The PP/ST author should specify the list of objects or information types for which the TSF shall be capable of generating data authentication evidence.

FDP_DAU.1.2. The PP/ST author should specify the list of subjects that will have the ability to verify data authentication evidence for the objects identified in the previous element. The list of subjects could be very specific, if the subjects are known, or it could be more generic and refer to a "type" of subject such as an identified role.

3.2.- FCS_COP.1 Cryptographic operation

FCS_COP.1.1 The TSF shall perform [assignment: *relación de operaciones criptográficas (a_1)*] in accordance with a specified cryptographic algorithm [assignment: *algoritmos criptográficos (a_2)*] and cryptographic key sizes [assignment: *tamaños de clave (a_3)*] that meet the following: [assignment: *relación de normas (a_4)*].

- (a_1) el autor de la declaración de seguridad especificará las operaciones criptográficas que realiza la aplicación, sobre los datos de usuario.
- (a_2) el autor de la declaración de seguridad especificará los algoritmos criptográficos que implementa la aplicación para que el resultado sea la creación de un resumen reconocido conforme con las especificaciones de la base de datos.
- (a_3) el autor de la declaración de seguridad especificará los tamaños de las claves a utilizar, que deben ser apropiados para cada algoritmo y su uso esperado.
- (a_4) el autor de la declaración de seguridad especificará la relación de normas o estándares que satisface la implementación de los algoritmos criptográficos definidos.

4.- Requisitos para dar acceso a zona segura.

4.1.- FDP_ACC.2 Complete access control.

FDP_ACC.2.1. The TSF shall enforce the [assignment: access control SFP] on [assignment: list of subjects and objects] and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2 The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

-El autor de la declaración de seguridad ha creado un único usuario con acceso a todos los recursos desde el portal. La configuración y cambios en las bases de dato deberán realizarse en modo local con una aplicación específica fuera del ámbito de este TOE.

4.2.- FDP_ITC.1 Import of user data without security attributes

FDP_ITC.1.1 The TSF shall enforce the [assignment: *ninguna*] when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.1.2 The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

FDP_ITC.1.3 The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [assignment: *reglas adicionales de control de la importación (a_1)*].

- (a_1) el autor de la declaración de seguridad especificará las reglas de importación de los datos de usuario, que se aplicarán en la importación de la política de certificación, y otros datos de usuario necesarios para la creación o verificación de firmas.

4.3.- FIA_UAU.2 User authentication before any action.

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

4.4.- FIA_UAU.4 Single-use authentication mechanisms.

FIA_UAU.4.1 The TSF shall prevent reuse of authentication data related to [assignment: eliminando todos los datos de sesión y cerrando la aplicación al terminar el acceso al portal].

4.5.- FIA_UID.2 User identification before any action.

FIA_UID.2.1 The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

5.- Requisitos en la validación del entorno mediante auditoria de eventos.

- El autor de la declaración de seguridad indica que el TOE sólo podrá generar los registros de eventos. Estos registros serán en el formato concreto definido por el TOE para satisfacer todos los requisitos de seguridad, y llevarán un timestamp independiente asociado. Cualquier revisión y auditoria del registro se llevará a cabo en forma local por un usuario autorizado quedando de esta manera la seguridad limitada en el ámbito del TOE.

5.1.- FAU_ARP.1 Security alarms.

FAU_ARP.1.1. The TSF shall take [assignment: terminar el proceso de acceso al portal, generar inscripción en archivo log, e informar al usuario que intentaba acceder] upon detection of a potential security violation.

5.2.- FAU_GEN.1 Audit data generation.

FAU_GEN.1.1. The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;(inicio y fin de la conexión)

b) All auditable events for the [selection, choose one of: solo se definirá un nivel básico de registro.] level of audit.

c) [assignment: cualquier error en el intento de acceso será objeto de registro en archivo log].

FAU_GEN.1.2. The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [assignment: other audit relevant information].

5.3.- FAU_GEN.2 User identity association

FAU_GEN.2.1. The TSF shall be able to associate each auditable event with the identity of the user that caused the event.