Universitat Oberta de Catalunya

TFC Seguridad en Redes de Computadoras

José Mª Zaidín Villarrubia

1.- ÍNDICE

2.- INTRODUCCIÓN

Este texto forma parte de un Trabajo Final de Carrera dentro del ámbito de la Seguridad en las Redes de Computadores y se centra, específicamente, en la creación de un aplicativo web que pueda funcionar como una Autoridad de Certificación dentro del contexto de las Infraestructuras de Clave Pública (PKI), en concreto en el definido por el directorio X.500, que está publicado en la serie de recomendaciones ITU-T X.500, una de las cuales es la Recomendación X.509 que es la que seguiremos en este proyecto ya que, aunque hay muchos modelos de certificado, es el más usado internacionalmente.

Así pues, el trabajo a desarrollar consistirá en todas las tareas de investigación, aprendizaje, diseño y programación para configurar una aplicación que actúe como una autoridad certificadora (CA) PKI mediante una interfaz web que distinga entre funciones de administración y funciones de certificación para todos los usuarios.

Se contemplan también una serie de extensiones del enunciado para complementar el trabajo de investigación realizado.

3.- MOTIVACIÓN

El mundo de la Seguridad en Redes de Computadores está experimentando cambios a pasos agigantados. Las contraseñas de 56 ó 128 bits han dado paso a larguísimas contraseñas de 1.024, 2.048 y hasta 4.096 bits de clave. Los mecanismos para cifrar y proteger las comunicaciones por internet evolucionan día a día a lo que ahora se suman las transacciones electrónicas y las firmas y certificados digitales.

Personalmente, me he sentido atraído por este tema desde hace años, tanto a nivel lúdico (películas como “Juegos de Guerra” o éxitos recientes como la saga “Millenium” han configurado en mi interior una imagen romántica de la Seguridad Informática) como de interés científico (lecturas de libros especializados, artículos de revistas o, simplemente, el seguimiento de las noticias relacionadas con casos de violaciones de seguridad que aparecen en la prensa y la televisión), aunque sólo al entrar a estudiar en la UOC he podido empezar a profundizar un poco en los conocimientos implicados en este apasionante mundo.

Las dos asignaturas más vinculadas a esta rama del conocimiento (Seguridad en Redes de Computadores y Criptografía) las he vivido de forma desigual, ya que una me decepcionó profundamente mientras que de la otra disfruté cada minuto que le dedicaba. Así pues, aparte de la atracción vocacional por el tema en cuestión tengo una especial motivación para sanear los recuerdos y conseguir una reconciliación con la materia.

Por otro lado, los certificados digitales tienen cada vez un uso más difundido y generalizado (firma de contratos, presentación telemática de documentos oficiales, etc.) y están empezando a formar parte de nuestra vida cotidiana. La oportunidad de participar en un proyecto como el presente que involucra el desarrollo de una interfaz web, que trata a fondo el tema de la Seguridad y que profundiza en el tema de los certificados digitales me parece una oportunidad difícil de rechazar.

4.- OBJETIVOS

Los principales objetivos del presente Trabajo Final de Carrera son:

- Profundizar en los esquemas de PKI y tomar conciencia de su importancia en el mundo actual.
- Conocer y aprender a instalar, configurar y manejar un servidor web para servir las diferentes páginas de la interfaz web.
- Poner en práctica lo aprendido en Ingeniería del Software sobre Unified Modeling Language (UML) para desarrollar una aplicación orientada a objetos eficaz y versátil.
- Aprender a diseñar una interfaz web que permita interaccionar con el usuario y, a su vez, tenga una presentación agradable, que tenga un comportamiento dinámico y se relacione con una base de datos y disponga de una lógica interna diseñada en Java.
- Familiarizarme con las librerías que proporcionan recursos criptográficos en Java, en concreto las que crean CA, generan claves, emiten certificados y gestionan las listas de certificados revocados (CRL).
- Poner en práctica lo aprendido en las asignaturas de Bases de Datos sobre el Diseño y Creación de una base de datos que gestione los certificados.
- Aprender a dotar de la interfaz web de mecanismos de seguridad para su utilización en ámbito local y remoto.

5.- DESCRIPCIÓN DEL PROYECTO

El presente proyecto se estructura en cuatro entregas parciales que coincidirán con las 4 PACs de la asignatura y que irán cumplimentando los diferentes requisitos de que consta el proyecto.

En primer lugar, realizaré una labor de investigación para poder definir el programario a utilizar y cual es la configuración más adecuada para la realización del proyecto. En este sentido cabe destacar varias líneas de investigación: La realización de la aplicación web, la relación de la interfaz web con el programa de la aplicación, la base de datos, la librería especializada en aplicaciones criptográficas, etc. Toda esta labor se reflejará en el planning de trabajo que acompaña el presente documento y que es el documento que culmina la primera entrega.

La segunda entrega consistirá en un primer desarrollo de la interfaz web, apenas una página de bienvenida, que interrelacionará con la aplicación para crear la CA. Es por lo tanto necesario para culminar esta segunda entrega haber resuelto la forma de crear e iniciar una autoridad certificadora.

La tercera entrega consistirá en el desarrollo de la interfaz web para permitir al usuario interaccionar con la aplicación dándole varias opciones de crear certificados digitales. En concreto, se contemplan dos posibilidades distintas:

- La creación de certificados para un usuario que ya dispone de su correspondiente par de claves.
- La creación de certificados mediante los datos que introduzca el usuario en la aplicación, lo que implica la creación por parte del aplicativo de las claves del usuario.

La cuarta entrega añadirá a las anteriores la gestión de las CRL por parte de la CA (tanto a nivel de gestión mediante la base de datos: consultas, búsquedas, etc.) como de publicación en el aplicativo web de la CRL.

Es a partir de esta cuarta entrega que, o bien se realizará la depuración y mejora de la aplicación, o bien se incluirán las posibles extensiones del enunciado del proyecto, consistentes en:

- La autenticación del administrador a nivel débil (usuario y contraseña) y a nivel fuerte (mediante el uso de certificados) para su conexión en local y remoto.

Todas estas entregas parciales se articulan en tres documentos finales de entrega que serán los documentos definitivos del proyecto:

- La memoria. Consistirá en el resumen de todas las investigaciones y trabajos realizados (diseño del producto, decisiones tomadas, dificultades encontradas, soluciones a los problemas, conceptos aprendidos, etc.)

- El producto. Consiste en el aplicativo, la programación de toda la aplicación, así como el manual de instrucciones para su puesta en marcha.

- La presentación del producto. Consiste en una presentación virtual del trabajo realizado debidamente documentado, con todos los elementos necesarios para que su utilización y comprensión sean asequibles.

6.- REQUERIMIENTOS Y EXPERIENCIA PREVIA

Los requerimientos para este trabajo son:

- Configurar la aplicación en el ordenador de sobremesa de manera que cumpla las funciones de servidor web y servidor de la aplicación. Definir la seguridad de la aplicación a nivel débil en el ordenador de sobremesa para poder conectarse a la aplicación en local.
- Simular desde el ordenador portátil el usuario que se conecta al aplicativo. Dispondrá de acceso libre a la aplicación como usuario.
- Simular desde el ordenador portátil el administrador que se conecta al aplicativo. Se definirá la seguridad de la aplicación para poder conectarse en remoto.

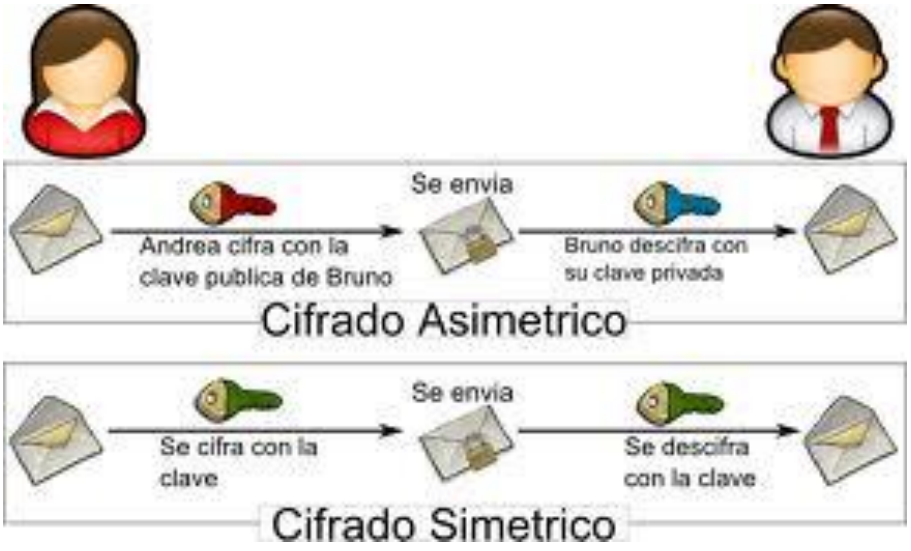
Mi experiencia previa en trabajos realizados, tanto de diseño y gestión de páginas web, como en temas de seguridad criptográfica o gestión de bases de datos son muy limitados, y se reducen prácticamente a los trabajos y prácticas desarrollados en la UOC.

7.- ESTADO DEL ARTE

La condición multidisciplinar de este trabajo, hace que la redacción de este apartado abarque diferentes materias, por lo que desglosaremos su redacción en diferentes apartados.

Criptografía PKI

En el mundo de la criptografía moderna se pueden diferenciar dos claras vertientes, la criptografía simétrica y la criptografía asimétrica o de clave pública. La primera se basa en algoritmos simétricos que usan una misma clave para encriptar y desencriptar mensajes, y la segunda se basa en algoritmos de clave pública que usan claves distintas para la encriptación y desencriptación. Una infraestructura de clave pública (PKI, Public Key Infrastructure) es el conjunto de hardware, software, personas, políticas y procedimientos que se necesitan para crear, manejar, almacenar, distribuir y revocar certificados digitales basados en la criptografía asimétrica. Así está definida en el RFC 2882 (Internet Security Glossary), a continuación se muestra una figura de estos dos tipos de criptografía.



En el proceso de firmado digital cobran vital importancias los algoritmos hash o hash codes. Son criptosistemas de resumen que aceptan de entrada un mensaje de longitud abierta y tras aplicar un algoritmo devuelven una cadena de bits de longitud fija, esta longitud varía según el tipo de algoritmo que se use. Los algoritmos hash más

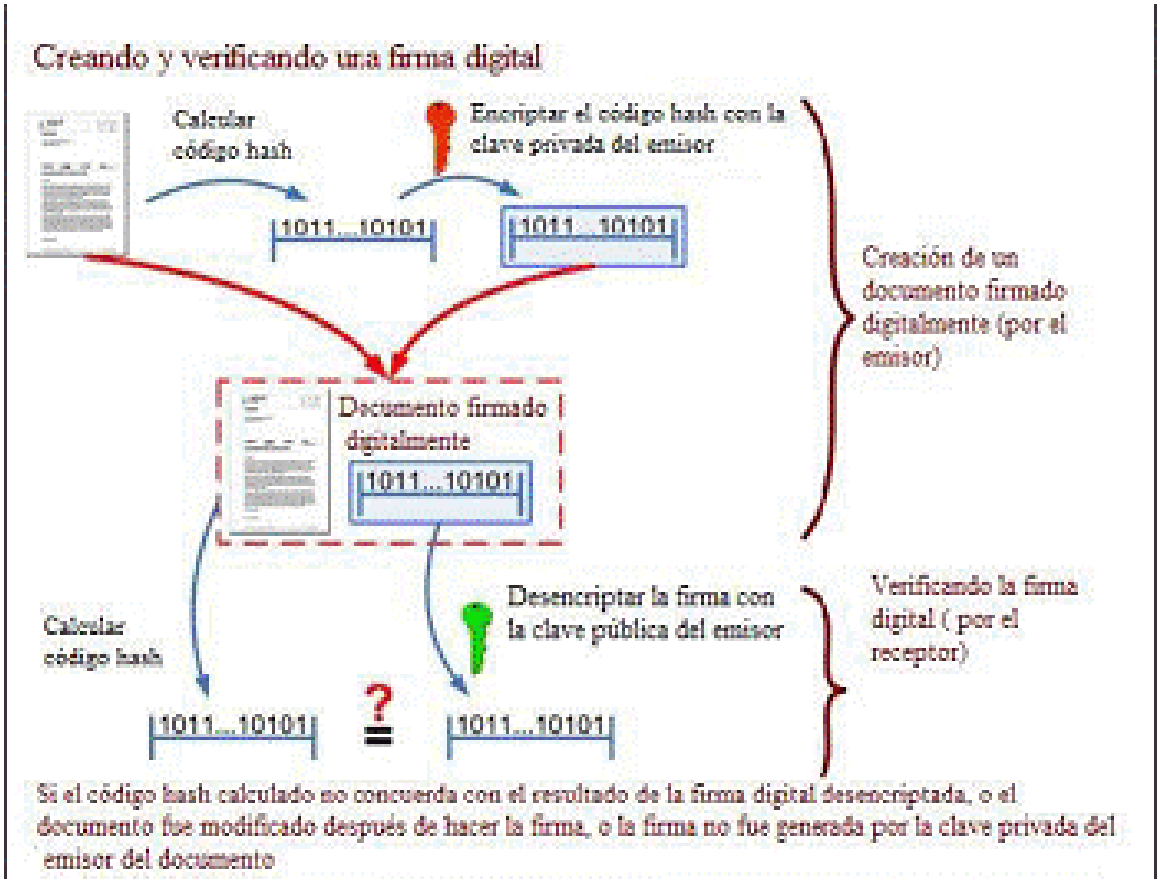
conocidos actualmente son los Message Digest (siendo el más usado el Message Digest 5 ó MD5) y los Secure Hash Algorithm (siendo los más usados el SHA-1 y el SHA-256). Al valor obtenido tras haber aplicado el algoritmo hash se le llama valor del hash, o simplemente hash, y tiene algunas características interesantes como por ejemplo que:

- El valor que produce cada mensaje es único.
- No puede haber otro mensaje que produzca el mismo valor
- Es imposible reconstruir el mensaje a partir de su hash.

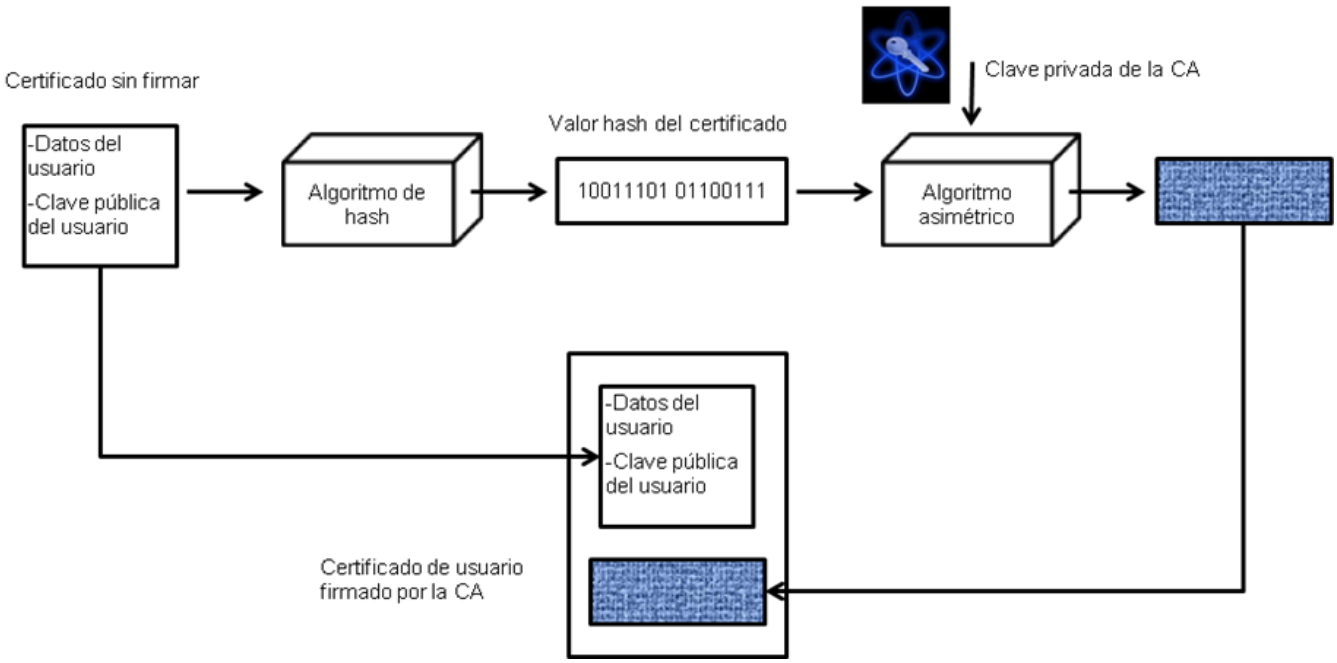
En el proceso de obtención de la firma digital de un mensaje, el usuario posee dos pares de claves distintas que se llaman clave privada y clave pública (estamos pues en un caso de criptografía asimétrica), la clave privada se usa para encriptar y debe permanecer secreta mientras que la otra se usa para desencriptar y es de carácter público (por ejemplo mediante un certificado digital). Además, se debe acordar un algoritmo de firma y un algoritmo hash que usarán tanto el firmante como la persona que verifique la firma.

El primer paso es obtener el valor hash del mensaje y en el segundo paso el firmante debe utilizar su clave privada para encriptar este valor hash, a estos dos pasos se les conoce como firmar digitalmente un mensaje y al resultado obtenido como firma digital.

La verificación de la firma digital implica que otra persona desencripte la firma digital utilizando la clave pública del firmante. El proceso consiste en que el verificador, por un lado desencripte el resumen hash del texto con la clave pública del firmante y, por otro, calcule el resumen hash del texto original. Si los valores son iguales entonces puede confiar en que el mensaje que tiene es el mismo que el que recibió firmado. En la siguiente figura se muestra el proceso de obtención de la firma digital explicado.



Un certificado digital es emitido por una autoridad certificadora (CA, Certificate Authority), y garantiza que esta CA ha verificado y confía en la identidad de la persona a la que pertenece el certificado. La CA manifiesta esta conformidad firmando el certificado y adjuntando la firma al final del mismo certificado, esta firma es efectuada con la clave privada de la CA que solo ella conoce. Esto permite que cualquier receptor del certificado digital pueda verificar, utilizando la clave pública de la CA, que el certificado ha sido firmado por ella. Otro documento que suele firmar la CA es la lista de revocación de certificados. En la siguiente figura se muestra el proceso de generación de un certificado digital.



El certificado asocia una única clave pública a una persona, esta clave se adjunta con el contenido del certificado junto con otros parámetros de la clave. En el proceso de firmado digital se utiliza el certificado digital para constatar que la clave pública, necesaria para la verificación de la firma, pertenece al firmante. El certificado contiene además un conjunto de datos relevantes que son definidos en la recomendación ITU-T X.509 por medio de una estructura de certificado, la estructura actual está en su versión 3 y también se le llama formato de certificado X.509 v3.

Así pues la tecnología PKI permite a los usuarios autenticarse frente a otros usuarios y usar la información de los certificados de identidad (por ejemplo, las claves públicas de otros usuarios) para cifrar y descifrar mensajes, firmar digitalmente información, garantizar el no repudio de un envío, y otros usos.

En una operación criptográfica que use PKI, intervienen conceptualmente como mínimo las siguientes partes:

- Un usuario iniciador de la operación.
- Unos sistemas servidores que dan fe de la ocurrencia de la operación y garantizan la validez de los certificados implicados en la operación (autoridad de certificación, Autoridad de registro y sistema de Sellado de tiempo).
- Un destinatario de los datos cifrados/firmados/enviados garantizados por parte del usuario iniciador de la operación (puede ser él mismo).

Servidores web

Un servidor Web es un software de aplicación que nos brinda un servicio vinculado a la red, por ello un “Servidor Web” debe implementar el protocolo HTTP (protocolo de transferencia de hipertexto). Este protocolo esta diseñado para transferir paginas web, es decir, documentos en HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

El termino servidor es algo ambiguo, ya que un servidor es el software que presta un servicio, pero también se llama servidor a la maquina donde está instalado dicho software. En este trabajo vamos a hablar del software, mas no del hardware.

El funcionamiento del servidor es muy sencillo. Básicamente está siempre a la espera de peticiones Web. Dichas peticiones son hechas por un cliente http (un navegador web, que en nuestro caso serán los usuarios), que después de realizar la petición espera la respuesta del servidor. Por ejemplo, cuando un usuario digite en su navegador la dirección donde está ubicada nuestra aplicación, el usuario envía una petición HTTP a nuestro servidor, que responde al cliente enviando el código HTML de nuestra pagina. El cliente recibe el código fuente, lo interpreta y lo muestra en pantalla.

El servidor se limita a recibir las peticiones y responderlas adecuadamente, mientras el cliente se encarga del proceso de interpretación.

En cuanto a programación se refiere, existen dos tipos de aplicaciones web: del lado del cliente y del lado del servidor. Las aplicaciones del lado del cliente se ejecutan en el navegador web, entre ellas cabe destacar JavaScript, Visual Basic Script y los applets de Java. En cuanto a las aplicaciones del lado del servidor existen lenguajes de programación, que se ejecutan en el equipo servidor, generalmente formando documentos HTML dinámicos (basándose en operaciones y/o acceso a bases de datos, por ejemplo). Entre los lenguajes más destacables del lado del servidor están: PHP, JSP, ASP, Perl, CGI, entre otros.

En la mayoría de los casos se opta por utilizar tecnologías del lado del servidor, por varios motivos, por ejemplo: al ejecutarse en el servidor las respuestas son, por lo general, estándares XHTML por lo que cualquier navegador puede interpretarlas, cosa que no pasa con las tecnologías cliente (que en algunos casos necesitan plugins).

Otra ventaja es la seguridad: al ejecutarse el código fuente en el servidor, la programación es transparente al cliente, permitiendo ocultar así los detalles de implementación.

Algunos de los servidores más conocidos son Apache, Internet Information Server (IIS), Tomcat (un servidor-contenedor basado en Apache para estándares Java: JSP y Servlets), Lighttpd o Thttpd.

Por su especificidad para Java y su simplicidad de manejo e instalación nos decantaremos por el Tomcat como servidor web.

Programación de la aplicación web

Actualmente existen diferentes lenguajes de programación para desarrollar en la web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Desde los inicios de Internet, fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. A medida que pasó el tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas a dar solución. Esto dio lugar a desarrollar lenguajes de programación para la web, dinámicos, que permitieran interactuar con los usuarios y utilizaran sistemas de bases de datos. De entre los principales lenguajes queremos reseñar los siguientes:

LENGUAJE HTML: Desde el surgimiento de internet se han publicado sitios web gracias al lenguaje HTML. Es un lenguaje estático para el desarrollo de sitios web (acrónimo en inglés de HyperText Markup Language, en español Lenguaje de Marcas Hipertextuales). Desarrollado por el World Wide Web Consortium (W3C). Los archivos pueden tener las extensiones (htm, html).

Se trata de un lenguaje sencillo y de fácil aprendizaje que permite escribir hipertexto, que presenta el texto de forma estructurada y agradable y genera archivos pequeños, por lo que se despliega muy rápido y lo admiten todos los exploradores. Como inconvenientes, es un lenguaje estático que guarda muchas etiquetas que pueden convertirse en “basura” y que da lugar a un diseño muy lento y de corrección difícil.

LENGUAJE JAVASCRIPT: Se trata de un lenguaje interpretado que no requiere compilación. Fue creado por Brendan Eich y es utilizado principalmente en páginas web. Es similar a Java, aunque no es un lenguaje orientado a objetos ya que no dispone de herencias. La mayoría de los navegadores en sus últimas versiones interpretan código Javascript.

El código Javascript puede ser integrado dentro de las páginas web. Para evitar incompatibilidades el World Wide Web Consortium (W3C) diseño un estándar denominado DOM (en inglés Document Object Model, en su traducción al español Modelo de Objetos del Documento).

Sus principales ventajas son que se trata de un lenguaje de scripting seguro y fiable que se ejecuta en el cliente, pero tiene como desventajas que el código es visible por cualquier usuario y que debe descargarse completamente.

LENGUAJE PHP: Es un lenguaje de programación utilizado para la creación de sitios web. PHP es un acrónimo recursivo que significa “PHP Hypertext Pre-processor”, (inicialmente se llamó Personal Home Page). Surgió en 1995, desarrollado por PHP Group.

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas.

Sus principales ventajas son que es un lenguaje libre y multiplataforma, muy rápido y fácil de aprender, con capacidad de conexión con los principales manejadores de bases de datos y multitud de funciones. Sus desventajas son que necesita instalar un servidor web que realiza todo el trabajo sin delegar en el cliente y que dificulta la modulación.

LENGUAJE ASP: Es una tecnología del lado de servidor desarrollada por Microsoft para el desarrollo de sitios web dinámicos. ASP significa en inglés (Active Server Pages), fue liberado por Microsoft en 1996. Las páginas web desarrolladas bajo este lenguaje necesitan tener instalado Internet Information Server (IIS).

ASP no necesita ser compilado para ejecutarse. Existen varios lenguajes que se pueden utilizar para crear páginas ASP. El más utilizado es VBScript, nativo de Microsoft. ASP se puede hacer también en Perl and Jscript (no JavaScript). El código ASP puede ser insertado junto con el código HTML

Sus principales ventajas son que usa Visual Basic Script, que es muy fácil para los usuarios, que tiene una comunicación óptima con SQL Server pero el código suele quedar muy desorganizado porque necesita muchas líneas para realizar funciones

sencillas, se trata de una tecnología propietaria y que conduce a hospedajes de sitios web costosos.

LENGUAJE JSP: Es un lenguaje para la creación de sitios web dinámicos, acrónimo de Java Server Pages y que está orientado a desarrollar páginas web en Java. JSP es un lenguaje multiplataforma creado para ejecutarse del lado del servidor.

JSP fue desarrollado por Sun Microsystems para la creación de aplicaciones web potentes que posee un motor de páginas basado en los servlets de Java. Para su funcionamiento se necesita tener instalado un servidor Tomcat.

Sus características principales son que las páginas son compiladas en la primera petición (que suele hacerla el propio servidor) y que mantiene el código separado de la lógica del programa y permite separar la parte dinámica de la estática en las páginas web.

Sus grandes ventajas son que crea las páginas del lado del servidor con una tecnología multiplataforma que permite (y ejecuta muy rápido) los servlets. Permite mantener el código muy bien estructurado confinando la parte dinámica con la lógica compleja al Java, facilitando así su tratamiento. Su principal desventaja es la complejidad de aprendizaje.

Hay algún lenguaje más de programación web, pero no es el objetivo de este trabajo realizar una enumeración exhaustiva de los mismos. Baste citar de pasada ASP.NET, Python, Ruby….

Base de datos

El mercado dispone de multitud de lenguajes SQL para bases de datos disponibles para configurar el sistema de gestión de la base de datos que necesita nuestra aplicación. A título informativo citaremos los más conocidos como pueden ser MySQL, Informix, postgreSQL, Oracle, MS SQL Server, etc.

Si bien cada uno de estos posibles candidatos tiene sus ventajas e inconvenientes, la elección del entorno de trabajo postgreSQL obedece básicamente a su utilización en la asignatura Bases de Satos 2 que estoy cursando actualmente en la UOC, por lo que simplemente me limito a la ya realizada enumeración de posibilidades.

8.- ORGANIZACIÓN DEL PROYECTO

En este apartado se detallan los recursos destinados a la realización del proyecto.

Recursos humanos

Los recursos humanos destinados a este proyecto consisten básicamente en el estudiante autor del trabajo, que asumirá para sí diversos roles, como son el de Gerente del Proyecto, Arquitecto de Maquinario, Programador de Sistemas i Gestor de la Calidad.

Cabría mencionar la figura del Consultor de la Asignatura como Asesor Técnico.

Recursos temporales

Teniendo en cuenta que se trata de un trabajo desarrollado en ambiente docente y al que le corresponden un total de 7,5 créditos cuatrimestrales universitarios, el cómputo global de horas invertidas (calculando a razón de 30 horas por crédito) no debería superar las 225 horas.

Recursos materiales

Documentales: Acceso a las diferentes bibliotecas y fuentes de Internet necesarias para realizar los estudios teóricos, así como la implementación de la aplicación. Este recurso será la fuente bibliográfica del proyecto que, de momento no figura en esta entrega sino que se incluirá en la memoria final del Trabajo Final de Carrera ya que así contendrá todas las referencias utilizadas y no sólo una parte..

Tecnológicos: El programario necesario para la aplicación debería ser de software libre y figurar disponible en la red. Por otro lado, dispondremos de dos ordenadores, uno de sobremesa y otro portátil para simular las conexiones a la aplicación.

Las características de cada uno de ellos son:

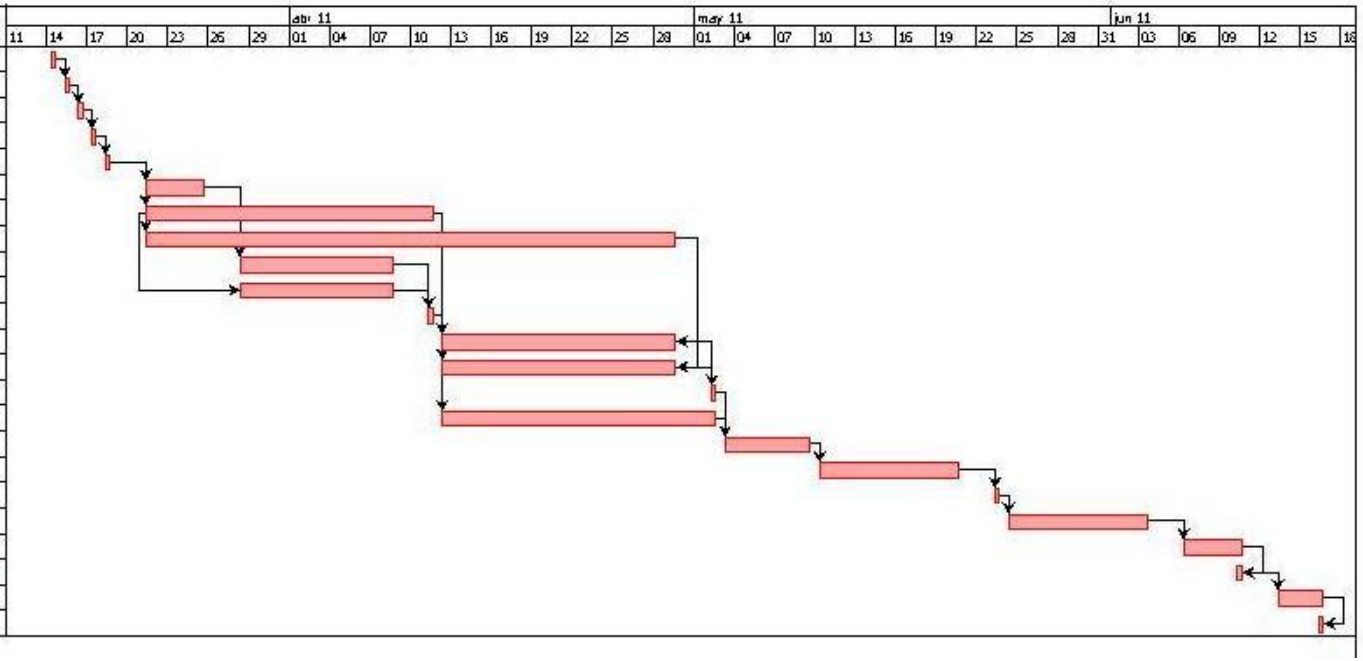
- Sobremesa: Procesador Intel Pentium(R) D CPU 2,80 GHz con 3,00 GB de memoria RAM funcionando con un sistema operativo Windows Vista Home Basic de 32 bits.
- Portátil: Procesador Intel Pentium(R) D CPU T3400 a 2,17 GHz con 4,00 GB de memoria RAM funcionando con un sistema operativo Windows Vista Home Basic de 32 bits.

9.- PLAN DE TRABAJO

Se adjunta a continuación el documento Plan de trabajo con la descomposición en tareas del proyecto y la estimación temporal para cada una de ellas, así como la tabla resumen.

Actividad	Días	Inicio	Final
Redacción planning V0	1	14/03	14/03
Redacción planning V1	1	15/03	15/03
Elección de entorno y herramientas de trabajo	1	16/03	16/03
Configuración del entorno de trabajo	1	17/03	17/03
Entrega PAC 1	1	18/03	18/03
Aprendizaje aplicativo web	5	21/03	25/03
Aprendizaje librería criptográfica	16	21/03	11/04
Programación UML	30	21/03	29/04
Codificación entorno web inicial: Bienvenida	10	28/03	08/04
Codificación criptográfica: Inicialización CA	10	28/03	08/04
Entrega PAC 2	1	11/04	11/04
Mejora codificación web: Elección funcionalidades	14	12/04	29/04
Codificación criptográfica: Emisión de certificados	14	12/04	29/04
Entrega PAC 3	1	02/05	02/05
Aprendizaje vinculación base de datos	15	12/04	02/05
Modificación codificación web: Incluir CRL	5	03/05	09/05
Codificación criptográfica y base de datos: CRL	9	10/05	20/05
Entrega PAC 4	1	23/05	23/05
Depuración y codificación extensiones	9	24/05	03/06
Redacción memoria	5	06/06	10/06
Entrega memoria y producto	1	10/06	10/06
Preparación presentación	4	13/06	16/06
Entrega presentación	1	16/06	16/06

	Nombre	Duración	Inicio	Terminado
1	Redacció del planning V.0	1 day	14/03/11 8:00	14/03/11 17:00
2	Redacció del planning V.1	1 day	15/03/11 8:00	15/03/11 17:00
3	Elecció d'entorn i eines de treball	1 day	16/03/11 8:00	16/03/11 17:00
4	Configuració entorn de treball	1 day	17/03/11 8:00	17/03/11 17:00
5	Lliurament PAC 1	1 day	18/03/11 8:00	18/03/11 17:00
6	Aprenentatge aplicatiu web	5 days	21/03/11 8:00	25/03/11 17:00
7	Aprenentatge llibreria criptogràfica	16 days	21/03/11 8:00	11/04/11 17:00
8	Programació UML	30 days	21/03/11 8:00	29/04/11 17:00
9	Codificació entorn web inicial - Benvinguda	10 days	28/03/11 8:00	8/04/11 17:00
10	Codificació criptogràfica: inicialització CA	10 days	28/03/11 8:00	8/04/11 17:00
11	Lliurament PAC 2	1 day	11/04/11 8:00	11/04/11 17:00
12	Millora codificació web - Elecció funcionalitats	14 days	12/04/11 8:00	29/04/11 17:00
13	Codificació criptogràfica: Emissió certificats	14 days	12/04/11 8:00	29/04/11 17:00
14	Lliurament PAC 3	1 day	2/05/11 8:00	2/05/11 17:00
15	Aprenentatge vinculació base de dades	15 days	12/04/11 8:00	2/05/11 17:00
16	Modificació codificació web - Incloure CRL	5 days	3/05/11 8:00	9/05/11 17:00
17	Codificació criptogràfica i base de dades CRL	9 days	10/05/11 8:00	20/05/11 17:00
18	Lliurament PAC 4	1 day	23/05/11 8:00	23/05/11 17:00
19	Depuració i codificació extensions	9 days	24/05/11 8:00	3/06/11 17:00
20	Redacció memòria	5 days	6/06/11 8:00	10/06/11 17:00
21	Lliurament memòria i producte	1 day	10/06/11 8:00	10/06/11 17:00
22	Preparació presentació	4 days	13/06/11 8:00	16/06/11 17:00
23	Lliurament presentació	1 day	16/06/11 8:00	16/06/11 17:00



10.- PRIMEROS PASOS (ELECCIÓN DE PROGRAMARIO)

Fruto de la investigación realizada hasta el momento, las decisiones que se han tomado han sido las siguientes:

- Como lenguaje de programación de la aplicación se ha escogido el Java, debido a su portabilidad y facilidad de trabajo en interacciones con otros entornos (servidores web, bases de datos, herramientas criptográficas).
- El IDE de trabajo será el Eclipse, ya que es el que vengo utilizando en todas las asignaturas de la UOC en las que he tenido que programar Java (POO, Estructura de la información, TALF, etc.)
- Como servidor (contenedor) web he escogido el Apache Tomcat por su amplia difusión, disponibilidad y por la cantidad de recursos relacionados que hay en la red.
- Para programar la aplicación web utilizaré JSP ya que su utilización es más sencilla que la de los Servlets y así la programación en Java se centrará en los aspectos criptográficos y la lógica interna de la aplicación.
- Como base de datos utilizaré el postgrSQL, básicamente porque estoy cursando Bases de Datos 2 y utilizaremos este programa para la realización de las prácticas. Así pues, ya tengo configurado el postgreSQL para que interactúe con Java.
- Para todo lo relacionado con los aspectos criptográficos de la aplicación utilizaré la biblioteca de Bouncy Castle, recomendada por el consultor por su sencillez y completitud.
- Para la realización del planning he utilizado openproj. Si bien estoy mucho más habituado al Microsoft Project por motivos de utilización en mi ambiente laboral, el hecho de no disponer de una licencia a nivel particular me ha hecho decidir por un programario libre como el openproj. Es más limitado que el Project (bastante más limitado), pero la escasa complejidad del proyecto hace que baste para los usos a que está destinado.

- Para el diseño de la aplicación y la creación de gráficos UML utilizaré el MagicDraw, también por una razón de disponibilidad. Se trata del software que la UOC pone a disposición de sus alumnos para la asignatura Ingeniería del Programario y al ya estar familiarizado con él ha sido relativamente fácil realizar esta elección.

11.- PRIMEROS PROBLEMAS Y REVISIONES DEL PROGRAMARIO

Una vez escogidas las principales herramientas para el desarrollo de la aplicación, se me han generado algunos problemas inesperados que han llevado a ligeras modificaciones.

En el proceso de realización de la PAC2, en la que se debía entregar un primer esbozo de la aplicación con la realización de la Inicialización de la Autoridad Certificadora, se puso de manifiesto que el programa Eclipse que estaba utilizando no era el adecuado.

Al intentar desligar la aplicación web de la operativa de la aplicación (es decir, cuando empecé a trabajar con páginas jsp para presentar la aplicación y con Servlets java para la realización de las operaciones criptográficas) me encontré con serios problemas debido a que la versión del Eclipse que estaba utilizando era muy antigua (por no decir obsoleta).

La versión de Eclipse que utilizaba era la que en su día suministró la UOC, la versión Eclipse 3.1.2, pero a la hora de vincular Eclipse con Tomcat y desarrollar una aplicación web como la que nos ocupa, puso de manifiesto sus carencias. La sugerencia del consultor fue cambiar a Eclipse Helios 2, con lo que la programación de los Servlets se hizo mucho más sencilla.

Otro de los problemas que se solucionó durante la realización de la segunda PAC fue la instalación de la librería criptográfica de Bouncy Castle. El primer intento fue siguiendo las orientaciones de la propia Bouncy Castle que me orientaban a instalarlas en el java build path del Proyecto mediante la instrucción “Add External Jars”.

Sin embargo, cuando empecé a trabajar con la librería obtenía constantemente un error en el que me decía que la aplicación no encontraba el proveedor “import org/bouncycastle/jce/provider/BouncyCastleProvider;". El consultor me recomendó copiar el jar directamente desde el explorador de archivos y pegarlo en Eclipse dentro de la carpeta WEB-INF/lib. Desde ese momento la aplicación encontraba la librería de Bouncy Castle perfectamente.

Al abordar el problema de la generación de un certificado X509 me encontré con que la librería Bouncy Castle ha cambiado su organización interna, ha modificado algunas

clases y muchos de los métodos que estaba utilizando se habían “deprecado”. La solución fue bajar un complemento de la librería principal, la bcmail-jdk16-146.jar, que instalé junto a la principal con el método descrito anteriormente.

Al preparar esta entrega tuve que cambiar algunos datos de partida, como por ejemplo la longitud de las claves utilizadas, la forma de archivar el certificado de la CA en un Keystore (utilizando un array con un único certificado) o el fichero wb.xml para que la aplicación funcionara correctamente. Al tratarse de cambios menores, quedan documentados en las entregas realizadas y simplemente los menciono para dejar constancia de las dificultades superadas.

Por último, comentar que al instalar el Apache Tomcat, tuve que configurar el puerto 8081 ya que el 8080 me daba problemas.

12.- DESCRIPCIÓN DE LA APLICACIÓN

La aplicación web desarrollada simula una Certification Authority (CA) a la que se pueden conectar los usuarios para solicitar certificados y los administradores (remotos) que quieran realizar alguna operación de mantenimiento.

La CA genera un certificado autofirmado así como un par de claves (pública y privada) para la firma de los certificados de los usuarios. También lleva una base de datos con los certificados emitidos y los que están vigentes o revocados. Todas estas operaciones se pueden realizar a través de la red.

También acepta las peticiones de los diferentes usuarios para generar certificados digitales que son enviados a sus peticionarios a través de la red.

La página de inicio de la aplicación es muy sencilla, y consiste en una bienvenida y la posibilidad de escoger entre los dos tipos de usuarios que acepta la aplicación:

Los Usuarios son los potenciales clientes de la aplicación, que solicitarán los certificados digitales a la CA, mientras que el Administrador será uno de los administradores del sistema que tiene la posibilidad de conectarse en remoto para la realización de ciertas tareas administrativas.

En toda la aplicación se están utilizando claves RSA de 2048 bits, certificados X509V3, formatos PKCS12 para enviar los certificados y formato csr para recibir las peticiones de certificados.

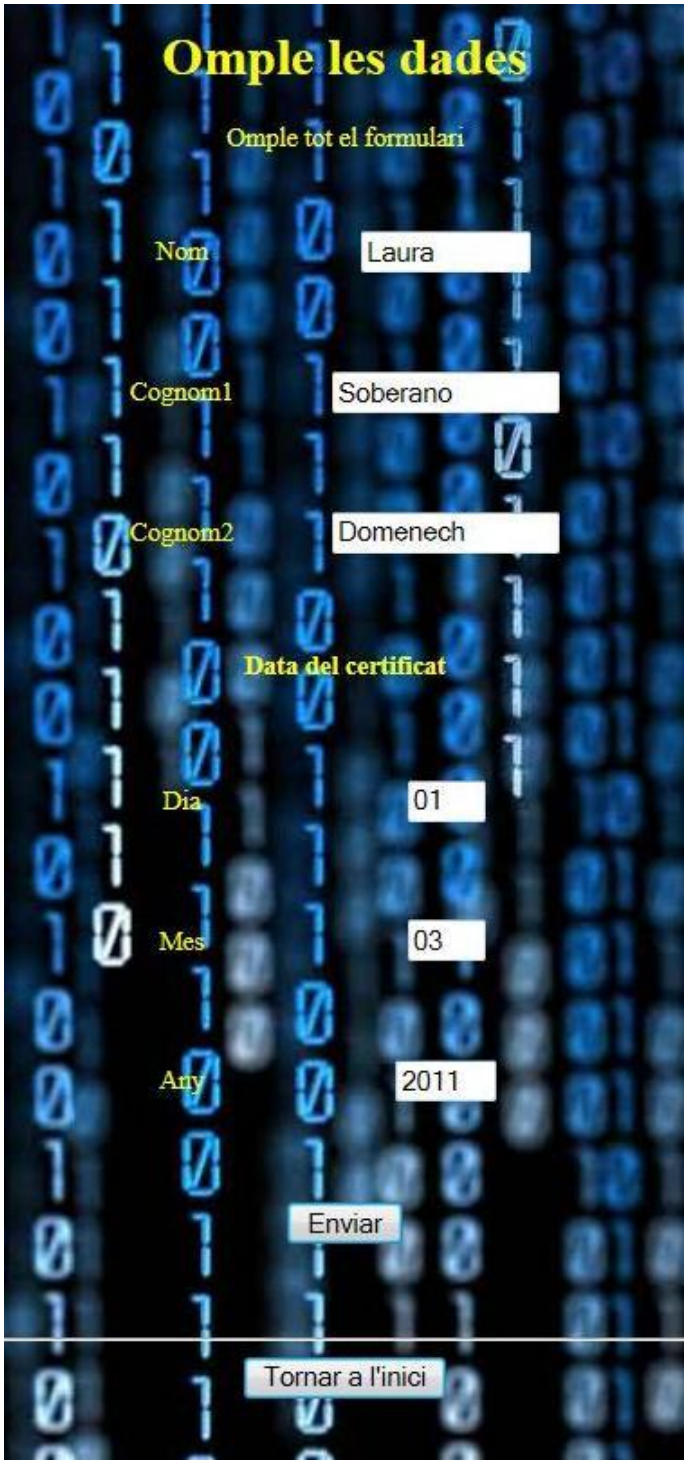
12.1.- FUNCIONALIDADES DEL USUARIO

El posible cliente de la aplicación se encontrará con un menú que le brindará tres opciones distintas dependiendo de si ya tiene sus propias claves o no o si lo que quiere realizar es la consulta de la CRL:

Como se puede comprobar en la captura anterior, la pantalla (tan sencilla como la anterior) se complementa con un botón que da acceso a la pantalla inicial de la aplicación.

12.1.1.- PEDIR UN CERTIFICADO COMPLETO

En el caso de que el usuario no disponga de sus propias claves pública y privada, el usuario debe elegir la primera opción y la aplicación web le redirigirá a un formulario que el usuario debe rellenar:



Cuando el usuario envía la información la aplicación web realiza los siguientes pasos:

1.- Inserta los datos en la base de datos.

2.- Genera las claves pública y privada del usuario y las guarda.

3.- Genera el certificado firmado por la CA (incluyendo una extensión con la URL de la CRL) y lo guarda.

4.- Realiza el correspondiente commit (si todo ha ido bien) o rollback (si algo ha fallado) en la base de datos.

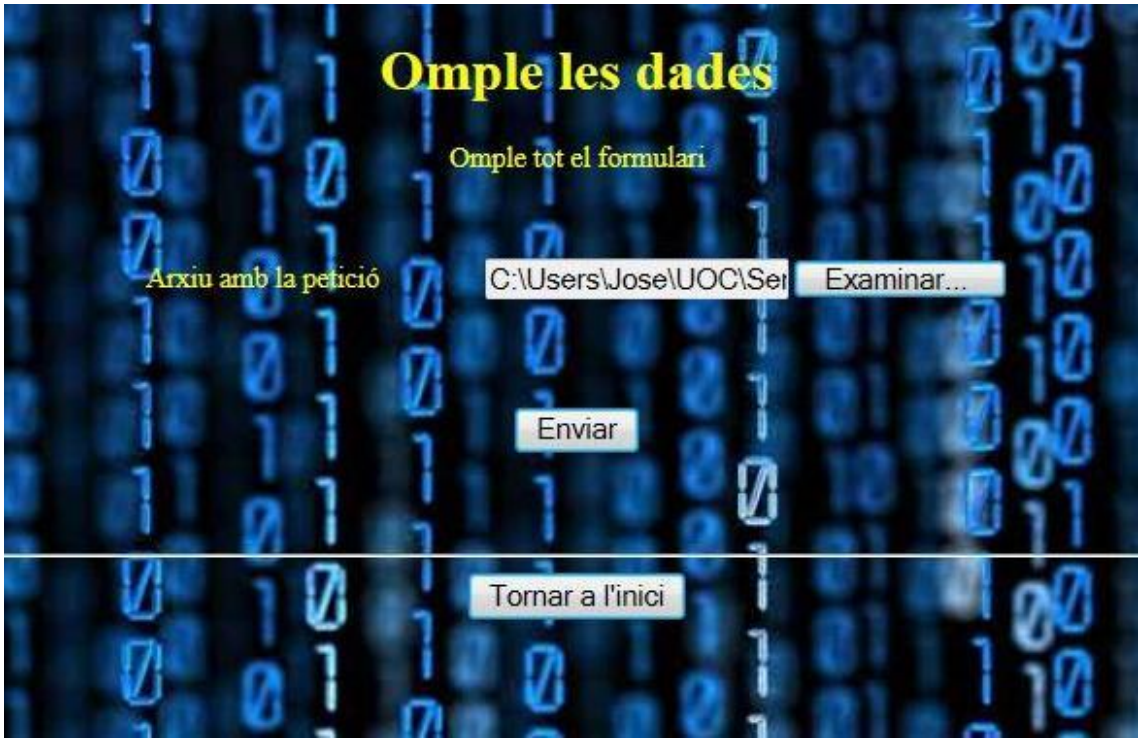
5.- Envía por la red el certificado en un fichero PKCS12

Una vez el usuario ha guardado el certificado en su ordenador, el usuario tiene la opción de volver a la pantalla de inicio y continuar navegando.

12.1.2.- ENVIAR UNA PETICIÓN

En caso de que el usuario ya disponga de sus propias claves pública y privada, la opción que contempla la aplicación es la de recibir una petición de certificado generada por el usuario.

Para ello, la aplicación nos lleva a una pantalla donde el usuario puede escoger el archivo que contiene la petición del certificado (en formato csr).



Una vez escogido el archivo, el usuario lo envía por la red y, si todo va bien, la aplicación muestra una pantalla indicando que se ha recibido el archivo y preguntando si se quiere continuar.

En caso de que el usuario opte por continuar, la aplicación realiza los pasos descritos anteriormente:

1.- Inserta los datos en la base de datos.

2.- Genera el certificado firmado por la CA (incluyendo una extensión con la URL de la CRL) y lo guarda.

3.- Realiza el correspondiente commit (si todo ha ido bien) o rollback (si algo ha fallado) en la base de datos.

4.- Envía por la red el certificado en un fichero PKCS12

Una vez el usuario ha guardado el certificado en su ordenador, el usuario tiene la opción de volver a la pantalla de inicio y continuar navegando.

12.1.3.- DESCARGAR LA CRL

Al seleccionar esta opción, la aplicación dirige al usuario a una pantalla que nos brinda la opción de descargar la CRL o de volver al inicio.

Si se elige descargar la CRL, la aplicación nos envía por la red el archivo con la CRL.

12.2.- FUNCIONALIDADES DEL ADMINISTRADOR

Si al inicio de la aplicación nos identificamos como Administrador, la aplicación nos redirige a una pantalla donde se implementa la seguridad de acceso mediante un simple “reto-respuesta”. La aplicación nos pide un password (que hemos definido con el transparente “admin”) y, si introducimos la contraseña correctamente nos da libre acceso a las funcionalidades del administrador y si la introducimos incorrectamente, nos devuelve a la misma pantalla.

Al acceder a las funcionalidades del Administrador, se nos plantean tres opciones, inicializar la CA, realizar tareas relacionadas con la CRL o realizar consultas a la base de datos.

12.2.1.- INICIALIZAR LA CA

Al escoger esta opción, la CA da paso a un proceso de Inicialización de toda su estructura. Las acciones que desarrolla la CA son:

1.- Inicializar la base de datos (es decir, borrar todos los registros existentes en su tabla).

2.- Generar de nuevo la clave pública y la clave privada de la CA.

3.- Generar de nuevo el certificado autofirmado de la CA.

Al terminar, nos deja de nuevo en la pantalla anterior.

12.2.2.- OPCIONES DE CRL

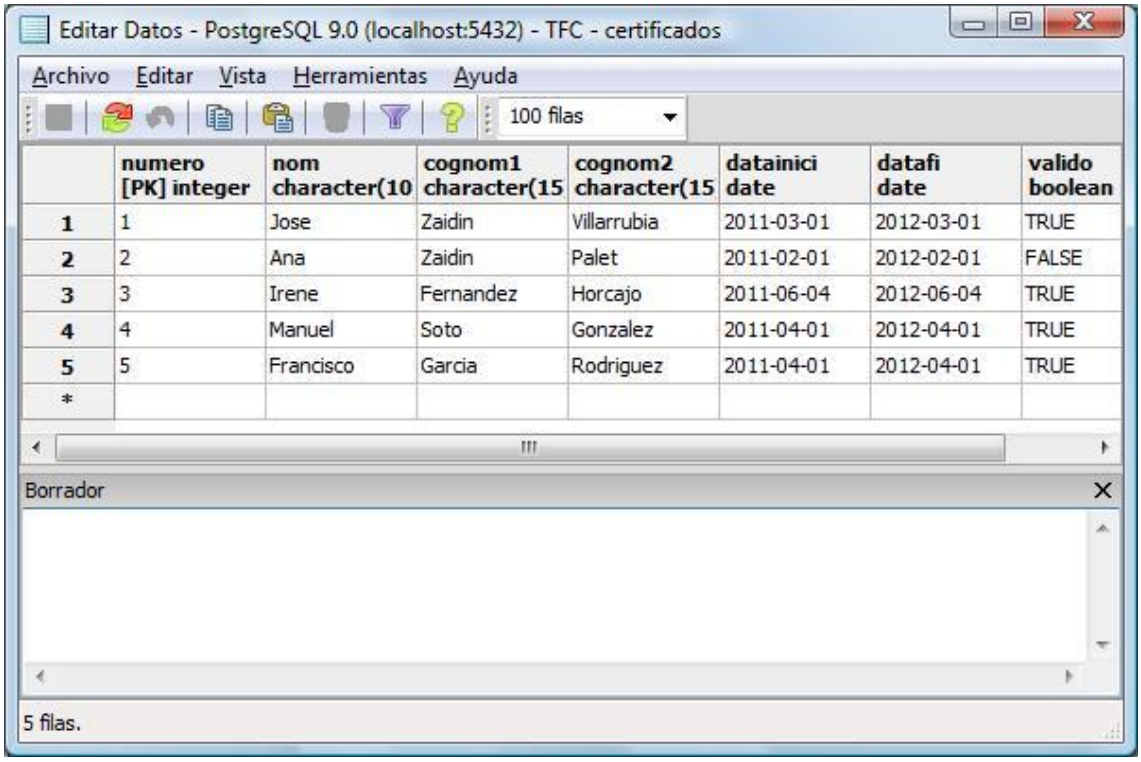
Si el administrador escoge esta opción, la aplicación nos redirige a una nueva pantalla donde el administrador puede escoger entre revocar un certificado o actualizar la CRL.

Como siempre, aparece una opción de vuelta al inicio pero, como que ya nos hemos autenticado como administrador, esta vez es a las opciones de administrador.

12.2.2.1.- REVOCAR UN CERTIFICADO

Al elegir esta opción, la aplicación nos lleva a una pantalla donde se nos piden los datos del usuario propietario del certificado que queremos revocar.

Una vez se envían los datos del usuario, la aplicación se conecta con la base de datos y cambia el valor del campo “valido” de true a false.



	numero [PK] integer	nom character(10	cognom1 character(15	cognom2 character(15	datainici date	datafi date	valido boolean
1	1	Jose	Zaidin	Villarrubia	2011-03-01	2012-03-01	TRUE
2	2	Ana	Zaidin	Palet	2011-02-01	2012-02-01	FALSE
3	3	Irene	Fernandez	Horcajo	2011-06-04	2012-06-04	TRUE
4	4	Manuel	Soto	Gonzalez	2011-04-01	2012-04-01	TRUE
5	5	Francisco	Garcia	Rodriguez	2011-04-01	2012-04-01	TRUE
*							

Una vez realizado el cambio en la base de datos, nos devuelve a la pantalla de opciones del administrador relacionadas con la CRL (para que tenga la opción de actualizar la CRL o de volver al inicio).

12.2.2.2.- ACTUALIZAR LA CRL

En caso de seleccionar esta opción, la aplicación realiza los pasos siguientes:

1.- Realiza una consulta con la base de datos para obtener una lista de los certificados revocados.

2.- Escribe esa lista en una CRL.

3.- Escribe esa CRL en un fichero y lo guarda.

4.- Nos redirige a la pantalla de descarga de la CRL (común con la opción ya vista de los usuarios).

12.2.2.3.- DESCARGAR LA CRL

Esta pantalla es común con la que ya hemos visto al describir el área de los usuarios. Es por ello, que al acabar la descarga de la CRL, la aplicación sólo nos ofrece la posibilidad de volver al Inicio, perdiendo así nuestros privilegios como administrador (simplemente hay que identificarse de nuevo).

12.2.3.- CONSULTAR LA BASE DE DATOS

En esta pantalla se nos ofrece simplemente la posibilidad de escoger qué consulta queremos realizar. Disponemos de tres opciones que son las que estudiaremos a continuación.

Como ya nos hemos autenticado como Administrador, la opción que nos ofrece es volver a la pantalla de Opciones del Administrador.

12.2.3.1.- CONSULTA POR NOMBRE

La pantalla actual nos presenta un formulario donde se nos pide el nombre a buscar. Se trata de una consulta sobre el nombre del usuario y nos pide una única cadena que la base de datos buscará en los tres campos correspondientes al nombre del usuario (nom, cognom1 y cognom2).

Al enviar el formulario, se realizan los siguientes pasos:

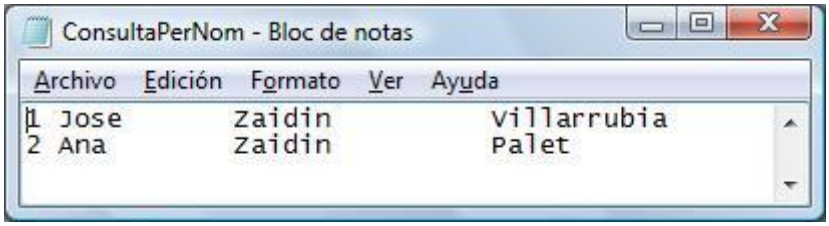
1.- Realiza una consulta con la base de datos para obtener una lista de los certificados emitidos que contienen ese nombre (válidos en ese momento o no).

2.- Escribe el resultado en una lista.

3.- Escribe esa lista en un fichero y lo guarda.

4.- Nos ofrece la oportunidad de guardar esa lista.

Un ejemplo de resultado de esa consulta (hay dos usuarios con ese nombre) nos daría la siguiente lista como resultado:



12.2.3.2.- LISTADO DE CERTIFICADOS VÁLIDOS

Al seleccionar esta opción, y dado que no hace ningún formulario intermedio, la aplicación web realiza automáticamente la consulta de los certificados emitidos y válidos. Se trata de la consulta no-CRL, es decir, la opuesta y nos proporciona un listado de los certificados válidos en ese momento.

Un ejemplo de esta lista sería el siguiente, donde se puede observar que “falta” el certificado número 2, que es el que anteriormente habíamos revocado.

Así pues, los pasos que se realizan al seleccionar esta opción son:

1.- Realiza una consulta con la base de datos para obtener una lista de los certificados validos.

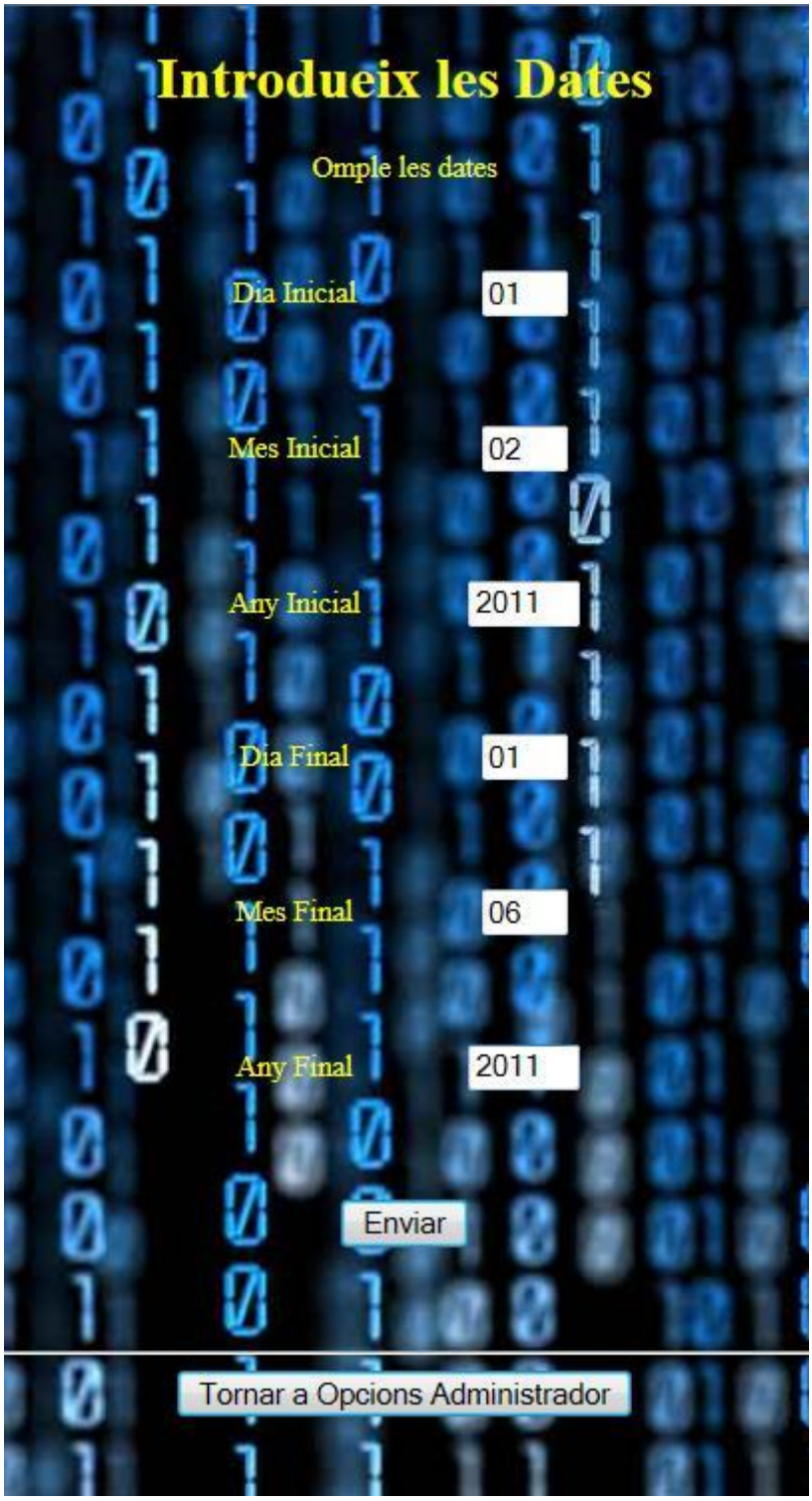
2.- Escribe el resultado en una lista.

3.- Escribe esa lista en un fichero y lo guarda.

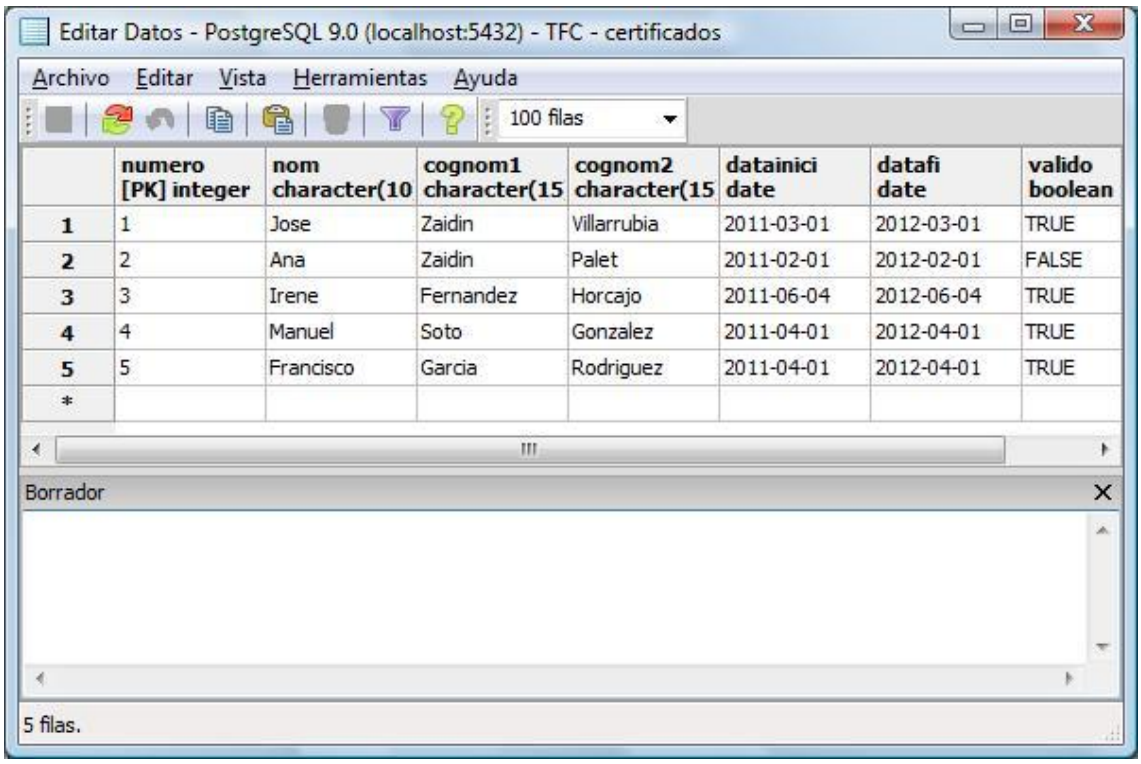
4.- Nos ofrece la posibilidad de guardar esa lista.

12.2.3.3.- CONSULTA POR FECHAS

En esta tercera opción de consulta a la base de datos nos aparece un formulario en el que se nos piden las dos fechas entre las cuales tienen que ser válidos los certificados. El formulario tiene el siguiente aspecto:

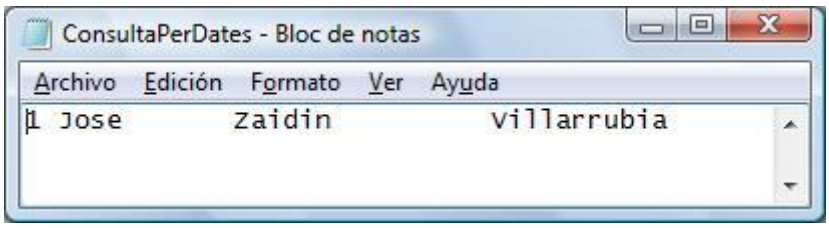


Se trata de que los certificados sean válidos durante todo ese periodo. Además, no se incluyen certificados que hayan sido revocados. Si en el momento de realizar la consulta la tabla general de los certificados tiene el siguiente aspecto:



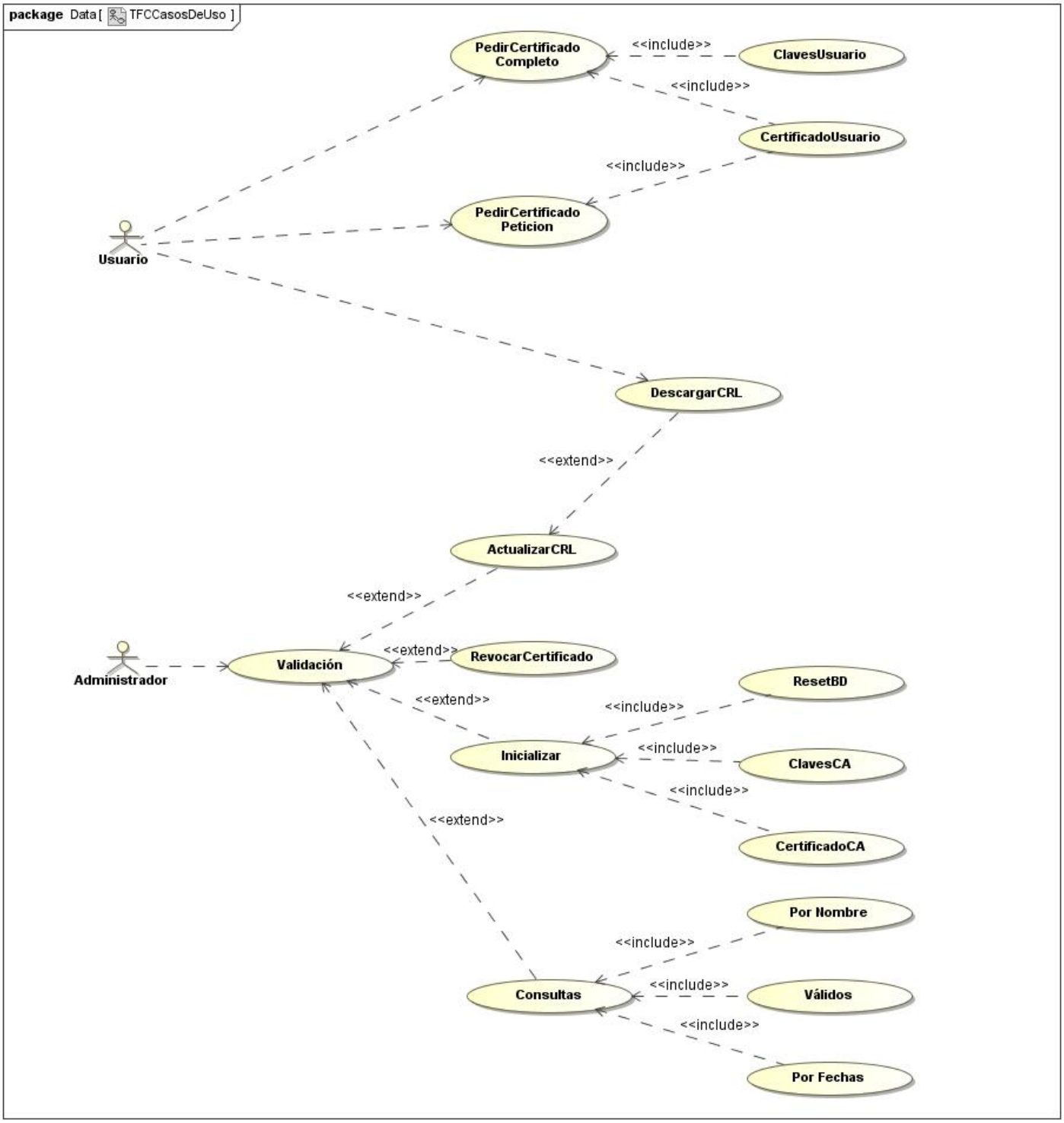
	numero [PK] integer	nom character(10	cognom1 character(15	cognom2 character(15	datainici date	datafi date	valido boolean
1	1	Jose	Zaidin	Villarrubia	2011-03-01	2012-03-01	TRUE
2	2	Ana	Zaidin	Palet	2011-02-01	2012-02-01	FALSE
3	3	Irene	Fernandez	Horcajo	2011-06-04	2012-06-04	TRUE
4	4	Manuel	Soto	Gonzalez	2011-04-01	2012-04-01	TRUE
5	5	Francisco	Garcia	Rodriguez	2011-04-01	2012-04-01	TRUE
*							

Entonces con el formulario anterior deberían aparecer, por fechas, los certificados 1 y 2, pero al estar el 2 revocado el resultado es:



12.3.- DIAGRAMA DE CASOS DE USO

Incluimos a continuación el diagrama de casos de uso de la aplicación:



13.- ORGANIZACIÓN DEL PROGRAMA DE LA APLICACIÓN

El programa de la aplicación tiene cinco elementos claramente diferenciados, cada uno con sus funcionalidades claramente definidas:

1.- Las páginas JSP.

2.- Los servlets.

3.- Las clases Java.

4.- La base de datos.

5.- Los ficheros properties.

Cada uno de ellos cumple con sus tareas específicas y diferenciadas.

13.1.- LAS PÁGINAS JSP

Las páginas JSP son las responsables de la presentación de las diferentes pantallas de la aplicación y son, por lo tanto, las responsables del aspecto externo del programa. Suyas son la responsabilidad sobre los fondos, tipos, colores y tamaños de letra, aspecto de los formularios y diseño de los botones.

También ayudan en la navegación de unas páginas a otras, aunque no se trata de una labor que realicen de manera exclusiva.

Las páginas JSP que he utilizado son (por orden alfabético):

- CertComplert.jsp: Es la pantalla con el formulario pidiendo todos los datos del usuario que no dispone de claves para la generación del certificado.
- Consulta1: Esta pantalla es la que contiene el formulario para realizar la consulta a la base de datos por nombre. Contiene una única casilla donde hay que introducir el nombre a buscar.
- Consulta3: Esta pantalla es parecida a la anterior, sólo que más compleja. Hay que introducir los datos para realizar la búsqueda entre dos fechas, por lo que hay que introducir seis valores.
- Consultes: Se trata de la pantalla que nos muestra las opciones de consulta a la base de datos. Las opciones primera y tercera nos llevan a las dos pantallas anteriores, mientras que la opción segunda nos generará la lista de certificados válidos en el momento de la consulta.
- DescargarCRL.jsp: Es la pantalla que da opción a descargar la CRL. Es una pantalla común a los usuarios y los administradores.
- index.jsp: Es la pantalla de inicio de la aplicación y a la que recurrentemente volvemos durante la navegación.
- OpcionsAdm.jsp: Es la pantalla a la que acceden los administradores y pueden elegir entre inicializar la CA y los trabajos relacionados con la CRL.
- OpcionsCRL.jsp: Es la pantalla que da la opción entre revocar un certificado y actualizar la CRL.

- Password.jsp: Es la pantalla que implementa la seguridad de acceso de los administradores.
- PeticioCert.jsp: Es la pantalla que da acceso al segundo tipo de petición de certificado. El usuario ya tiene sus claves y envía una petición de certificado. Da la opción de seleccionar el fichero con la petición.
- Pujat.jsp: Se trata de una pantalla absolutamente de transición. Una vez el usuario ha enviado su petición de certificado, esta pantalla informa del éxito en subir el archivo y ofrece la posibilidad de continuar.
- Revocar.jsp: Es la pantalla que presenta al administrador un formulario para que éste lo rellene con los datos del usuario al que se le va a revocar el certificado.
- Usuari.jsp: Es la pantalla que da al usuario las tres opciones de que dispone (solicitar un certificado completo, solicitar un certificado mediante una petición, consultar la lista de certificados revocados).

13.2.- LOS SERVLETS

Los servlets son los que, de alguna manera, conectan las páginas jsp encargadas de presentar las pantallas web de una manera estética y funcional, con las clases Java encargadas de realizar los trabajos criptográficos, de conectar con la base de datos, etc. Es decir, conectan la interfaz del usuario con el núcleo duro del programa.

Son los que se encargan de recoger los datos de los formularios y llamar a los diferentes métodos de las clases Java, realizando así las distintas transiciones entre unos y otros.

Los servlets de la aplicación son (por orden alfabético):

- ConsultaDates: Es el servlet que recibe las fechas entre las que deben ser válidos los certificados al realizar la consulta. Una vez recibidos los datos, llama a la clase Java que implementa la consulta y da la opción de guardar el resultado.
- ConsultaNom: Este servlet recibe el nombre que se ha de buscar en la base de datos y llama a la clase Java que realiza la consulta para luego dar la opción de guardar el resultado.
- ConsultaValids: Este servlet simplemente redirige la acción a la clase Java que realiza la consulta de los certificados válidos y luego da la opción de guardar el resultado.
- demanaCRL: Es el servlet que recibe la orden de crear la CRL. Para ello, realiza una consulta a la base de datos y con la lista de certificados (Cert, ver clases Java) que obtiene como resultado crea una CRL llamando a la clase Java correspondiente (CRLBC). El resultado lo graba en un fichero y nos acaba redirigiendo a la página JSP DescargarCRL.jsp ya vista.
- EnviarCRL: Es el servlet que da al usuario o administrador correspondiente, la posibilidad de descargar la CRL. Busca el fichero y lo manda por la red.
- Inicialitzar: Es el servlet que llama una a una a todas las acciones implicadas en la inicialización. Crea un resetBD, crea un par de claves y crea un certificado de la CA. Va llamando a los métodos correspondientes de las clases Java y acaba dirigiéndonos a la página JSP OpcionsAdm ya vista.

- Password: Es el servlet que gestiona el acceso del administrador remoto al área restringida a los administradores. Consiste en leer el formulario con la contraseña, compararla con la guardada en el disco y dar paso o rechazar el acceso.

- PujarArxiu: Es el servlet que recibe el nombre del archivo con la petición de certificado del usuario y lo sube para gestionarlo. Al acabar nos lleva a la página de transición Pujat.jsp.

- Revocar: Es el servlet que recoge los datos del usuario al que se le va a revocar el certificado y realiza las llamadas necesarias para que la acción se realice. Al acabar nos redirige a OpcionsAdm.jsp.

- Usuari1: Es el servlet que recoge los datos del formulario del usuario que solicita un certificado completo. Una vez recogidos, va llamando a los métodos de las clases Java necesarios para la creación de las claves y para la emisión del certificado. Termina enviando por la red el certificado generado.

- Usuari2: Es un servlet muy parecido al anterior, sólo que esta vez lee los datos de la petición de certificado que ha enviado el usuario. Realiza los mismos pasos que el anterior, si bien aquí no hay que generar las claves.

13.3.- LAS CLASES JAVA

Las clases Java son las que encierran los métodos de acceso y consulta con la base de datos, los métodos criptográficos que generan y gestionan las claves y certificados y constituyen el "objetivo último" de este trabajo.

Si bien hay algunas clases con contenidos absolutamente sencillos y transparentes, las hay con un grado de complejidad elevado y son las que verdaderamente constituyen el meollo de la aplicación:

Las clases Java utilizadas son (también por orden alfabético):

- AccesoBD: Se trata de la clase que establece la conexión jdbc entre el aplicativo y la base de datos en PostgreSQL. Las posibilidades que nos ofrece son:
  - Leer un fichero properties.
  - Establecer una conexión.
  - Cerrar esa conexión.
  - Identificar los parámetros de la conexión de entre lo leído del fichero properties.
- Cert: Sencilla clase Java que se identifica con los certificados. Tiene sus constructores y los correspondientes getters y setters.
- certificatCA: Es la clase Java que se encarga de generar el certificado autofirmado de la CA. Tiene dos métodos, uno para leer un fichero properties y el fundamental que engloba todas las operaciones criptográficas para generar el certificado.
- certificatUsuari: Clase similar a la anterior, sólo que genera los certificados del usuario. Incluye un método más, ya que se pueden generar dos tipos distintos de certificado de usuario y hemos utilizado una única clase para llamar a ambos métodos.
- CRL: Es la clase que conecta con la base de datos y pide el listado de los certificados revocados. Devuelve una lista que es la que utiliza la clase JSP demanaCRL.jsp para generar la CRL.

- CRLBC: Esta clase Java tiene (principalmente) dos métodos. El primero crea una CRL mientras que el segundo añade un certificado (revocado) a la CRL.

- InsertaCerti: Es la clase que inserta un certificado nuevo en la base de datos. Primero averigua que número de certificado le corresponde y luego inserta los datos en la base de datos. Antes de realizar el commit, hace la llamada para crear el certificado y sólo realiza el commit si todo va bien, en caso contrario ejecuta un rollback.

- parellClausCA: Es la clase que genera el par de claves de la CA y las guarda en el disco. Tiene dos métodos, uno para cargar los datos de un fichero properties y el otro, el principal, que es el que gestiona la creación de las claves.

- parellClausUsuari: Es una clase paralela a la anterior, sólo que esta vez genera las claves del usuario. Igualmente las guarda en disco y tiene un método para leer datos de un fichero properties.

- PerDates: Esta clase es la encargada de realizar la consulta a la base de datos para obtener el resultado de qué certificados son válidos entre las dos fechas solicitadas.

- PerNom: Esta clase, parecida a la anterior, es la encargada de realizar la consulta de qué certificados incluyen determinada cadena en el nombre de usuario.

- PerValids: Esta clase implementa la tercera consulta a la base de datos, dando como resultado la lista de certificados actualmente vigentes.

- ResetBD: Como su nombre indica, lo que hace esta clase es conectarse a la base de datos y eliminar todos los registros de la tabla “certificados”.

- RevocaCerti: Clase muy parecida a la anterior, sólo que en este caso lo que hace es cambiar el campo “valido” de un determinado certificado a false, es decir, deja el certificado sin validez (revocado).

13.4.- BASE DE DATOS

La base de datos que mantiene actualizada la relación de certificados emitidos y revocados la he generado con PostgreSQL. Las razones ya las he explicado al inicio de esta memoria, pero se banas principalmente en que es el programario que he utilizado durante este semestre en la asignatura Bases de Datos 2. La base de datos se llama TFC y dispone de una única tabla, la tabla certificados con los siguientes campos:

- numero: integer. Es la clave primaria.
- nom: character(10)
- cognom1: character(15)
- cognom2: character(15)
- datainici: date
- datafi: date
- valido: boolean

Las operaciones que se realizan sobre la base de datos son (aparte de la conexión-desconexión, commit-rollback) las siguientes:

Insertar un certificado: Se establece la conexión y se realiza una consulta para saber cuál es el número de certificado más alto. La instrucción SQL es: ""SELECT COUNT(*)AS nombre FROM certificados". A partir de ahí, se insertan los valores del nuevo certificado mediante la instrucción: “INSERT INTO certificados(numero,nom,cognom1,cognom2,dataInici,dataFi,valido) VALUES (?,?,?,?,?,?,?);”. Una vez generado el certificado, se realiza el commit.

Revocar un certificado: Aparte de las labores de conexión-desconexión, se revoca el certificado perteneciente a un determinado usuario, por lo tanto, partiendo de esos datos como base, se cambia el valor del atributo “valido” de true a false. La instrucción es: “UPDATE certificados SET valido=false WHERE nom=? AND cognom1=? AND cognom2=?”.

Generar la lista previa de la CRL: Consiste en realizar una consulta de los certificados que tienen el atributo “valido” como false. La instrucción SQL es: “SELECT * FROM certificados WHERE valido=false”.

Inicializar la base de datos: Esta operación consiste en borrar todas las entradas de la tabla certificados mediante una instrucción DELETE. La instrucción es: “DELETE FROM certificados”.

Consultar por Nombre: Se trata de una operación de consulta que busca una determinada cadena de texto en los tres campos vinculados al nombre del usuario (nom, cognom1, cognom2). La instrucción SQL que la genera es: “SELECT * FROM certificados WHERE (nom=? OR cognom1=? OR cognom2=?)”.

Consultar por Fechas: Operación parecida a la anterior, sólo que busca certificados que sean válidos entre dos fechas (todo el período) y que no estén revocados. La instrucción que la genera es: “SELECT * FROM certificados WHERE (datainici <= ? AND datafi >= ? AND valido=true)”.

Consultar los certificados válidos: Tercera operación de consulta. Lanza un listado con todos los certificados que tienen el atributo valido como TRUE. La instrucción SQL que la genera es: “SELECT * FROM certificados WHERE (valido = true)”

13.5.- LOS FICHEROS PROPERTIES

Para evitar escribir las direcciones absolutas entre las líneas de código, así como los parámetros de conexión con la base de datos se han confeccionado dos ficheros properties cuyo contenido es el siguiente:

bd.properties

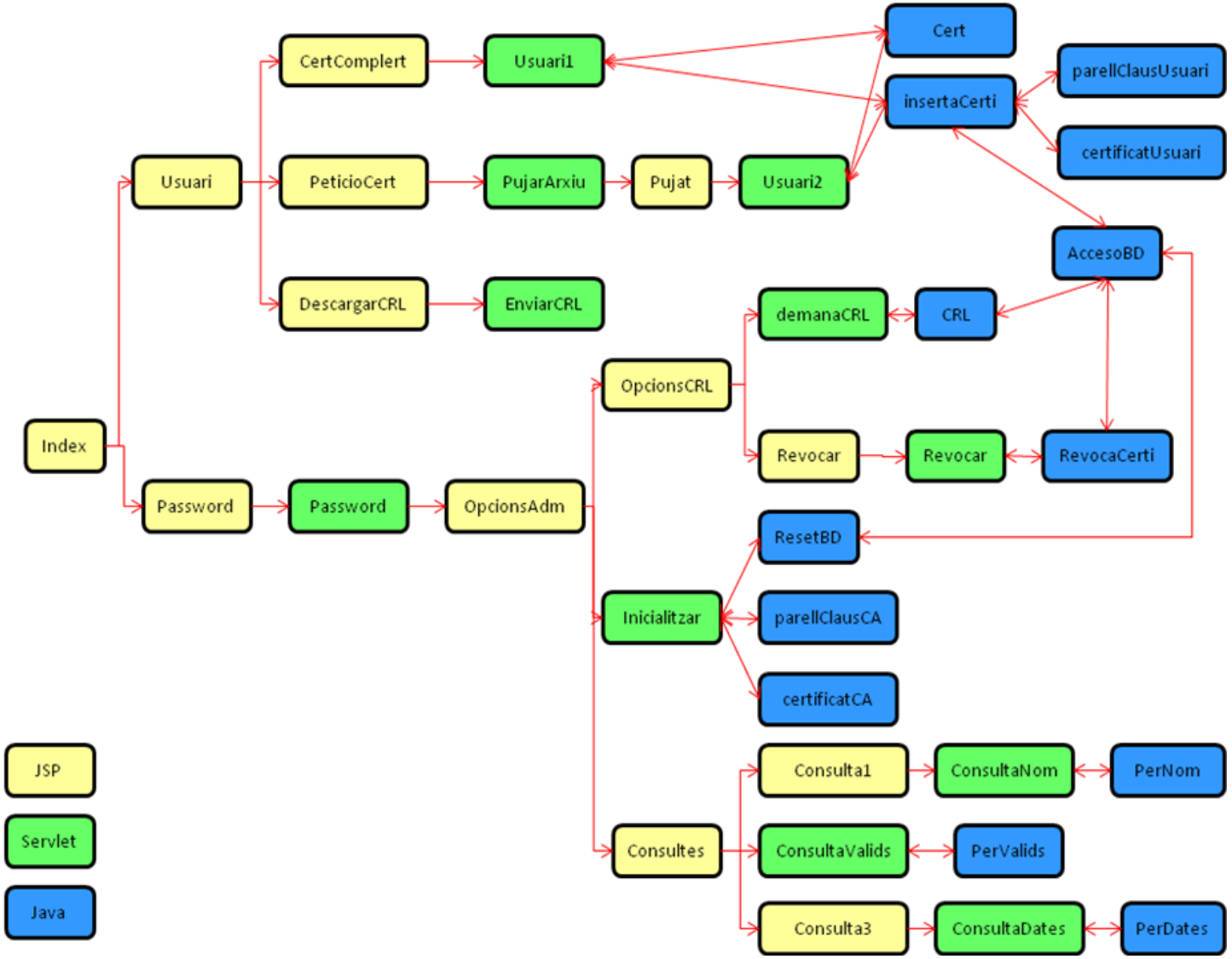
host	localhost
port	5432
dbname	TFC
params	CLIENT_LOCALE=es_ES.8859-1
user	postgres
passwd	Sirena

rutas.properties

clauCApriv	C:/TFC/CAclaves/clauCA.privada
clauCApub	C:/TFC/CAclaves/clauCA.publica
certiCA	C:/TFC/Keystore/certificatCA
magatzem	C:/TFC/Keystore/ksMagatzem
rutaUS	C:/TFC/CAclaves/clausUS
magatzemUS	C:/TFC/Keystore/ksMagatzemUS
pswd1	Criptotal
pswd2	CApass
rutaCRL	C:/TFC/tmp/CRL.tmp
pswd3	admin
peticion	C:/TFC/Peticion
URLCRL	http://Fijo:8081/TFC/DescargarCRL.jsp
rutaC1	C:/TFC/tmp/C1.tmp
rutaC2	C:/TFC/tmp/C2.tmp
rutaC3	C:/TFC/tmp/C3.tmp

13.6.- ESQUEMA DE LA APLICACIÓN

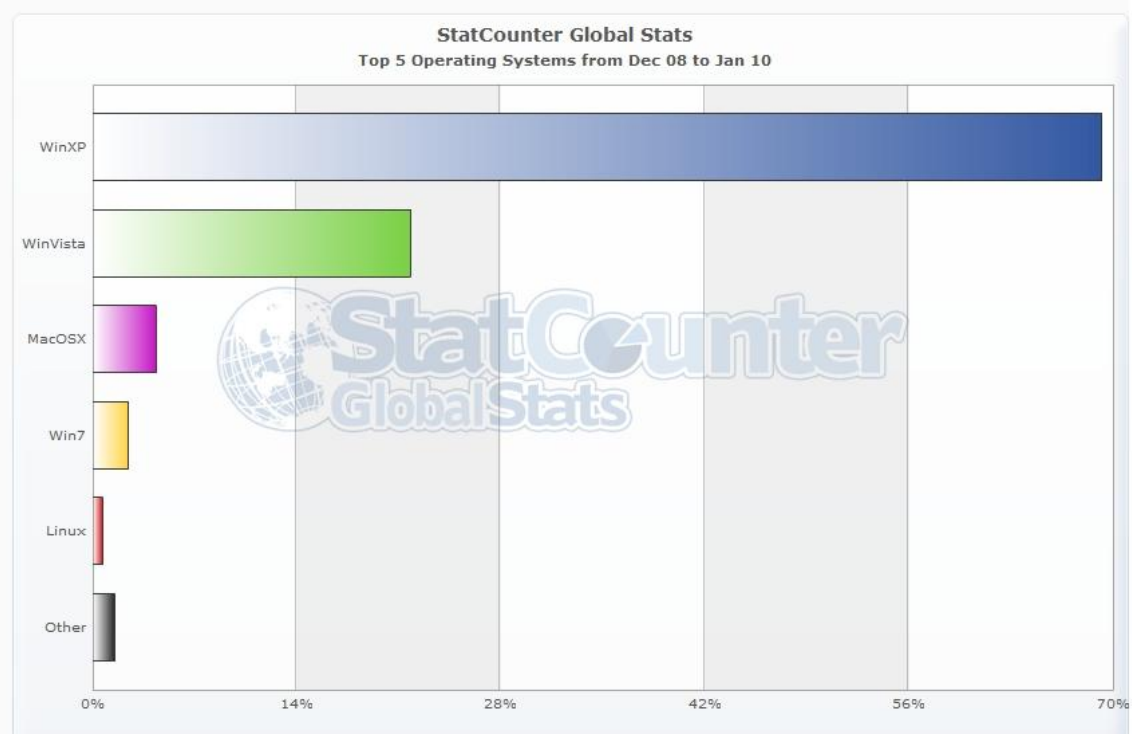
El esquema de la aplicación puede verse en la página siguiente:


Cert
CertComplert
Usuari1
parellClausUsuari
insertaCerti
certificatUsuari
Usuari
PeticioCert
PujarArxiu
Pujat
Usuari2
AccesoBD
DescargarCRL
EnviarCRL
demanaCRL
CRL
OpcionsCRL
Index
Revocar
Revocar
RevocaCerti
Password
Password
OpcionsAdm
ResetBD
Inicialitzar
parellClausCA
certificatCA
JSP
Consulta1
ConsultaNom
PerNom
Servlet
Consultes
ConsultaValids
PerValids
Java
Consulta3
ConsultaDates
PerDates

14.- INSTALACIÓN DEL APLICATIVO

En primer lugar, quiero dejar constancia de que todo el desarrollo del TFC lo he realizado desde un entorno equipado con Windows Vista, por lo que las instrucciones son para ese entorno. Este hecho no debería ser un problema, ya que el uso del entorno Windows está ampliamente distribuido llegando a casi el 95% de los usuarios. Este dato, que me costó creer al principio, viene refrendado por un informe de Stat-Counter / GlobalStats que establece los siguientes porcentajes para cada sistema operativo a fecha Enero de 2.010:

- Windows XP: 63,47%
- Windows Vista: 21,13%
- Windows 7: 8,37%
- Mac OS: 5,16%
- Linux: 0,7%
- Otros: 1,17%



De todos modos, aunque las explicaciones que vienen a continuación son para un entorno Windows, todo el programa está realizado en Java, por lo que debería ser multiplataforma y exportable a cualquier sistema que implemente la Java Virtual Machine sin más que cambiar algún dato de configuración inicial de los que figuran en los archivos properties ya mencionados.

Para la instalación del aplicativo en Windows, aparte de disponer de los diferentes programas y configuraciones ya mencionados, hay que seguir los siguientes pasos:

1.- Abrir la siguiente carpeta en nuestro explorador:

C:/TFC

y dentro de ella las subcarpetas:

C:/TFC/CAClaves

C:/TFC/Keystore

C:/TFC/tmp

ya que en ellas es donde se van a guardar los diferentes archivos que genere la aplicación.

2.- En el Eclipse Helios descargar todos los archivos del archivo comprimido producto.rar, que nos generará la carpeta TFC y todas sus subcarpetas.

3.- En el PostgreSQL crear la base de datos TFC con la instrucción:

```
CREATE DATABASE "TFC"
        WITH OWNER = postgres
                ENCODING = 'UTF8'
                TABLESPACE = pg_default
                LC_COLLATE = 'Catalan, Spain'
                LC_CTYPE = 'Catalan, Spain'
                CONNECTION LIMIT = -1;
```

4.- En la base de datos, crear una única tabla llamada certificados mediante la siguiente instrucción:

```
 CREATE TABLE certificados
(
        numero integer NOT NULL,
```

```
        nom character(10),
        cognom1 character(15),
        cognom2 character(15),
        datainici date,
        datafi date,
        valido boolean,
        CONSTRAINT pk_certificados PRIMARY KEY (numero)
    ) WITH (OIDS=FALSE);
    ALTER TABLE certificados OWNER TO postgres;
```

5.- Verificar que en los archivos properties tenemos introducidos los valores correspondientes a nuestro puente JDBC y las contraseñas correspondientes.

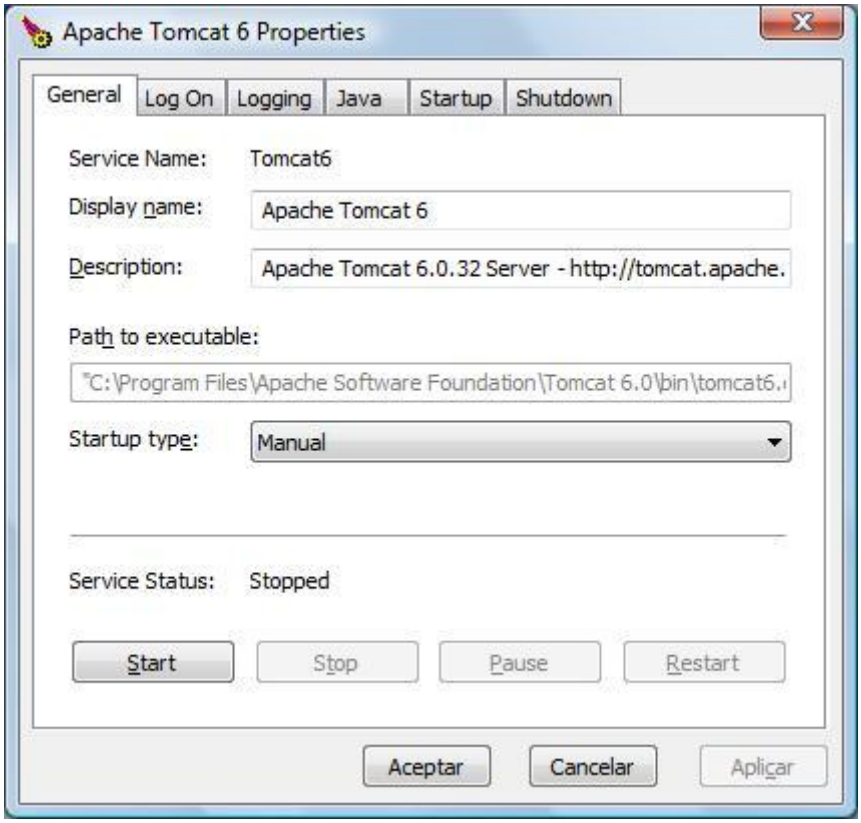
6.- Desde el Eclipse Helios, correr el aplicativo desde index.jsp.

7.- En nuestro navegador de internet escribir la URL: “fijo:8081/TFC/index.jsp” (esta es la que he utilizado yo en mi red casera, cada uno deberá introducir el nombre del ordenador donde haya instalado la aplicación. Una vez más, comprobar el properties).

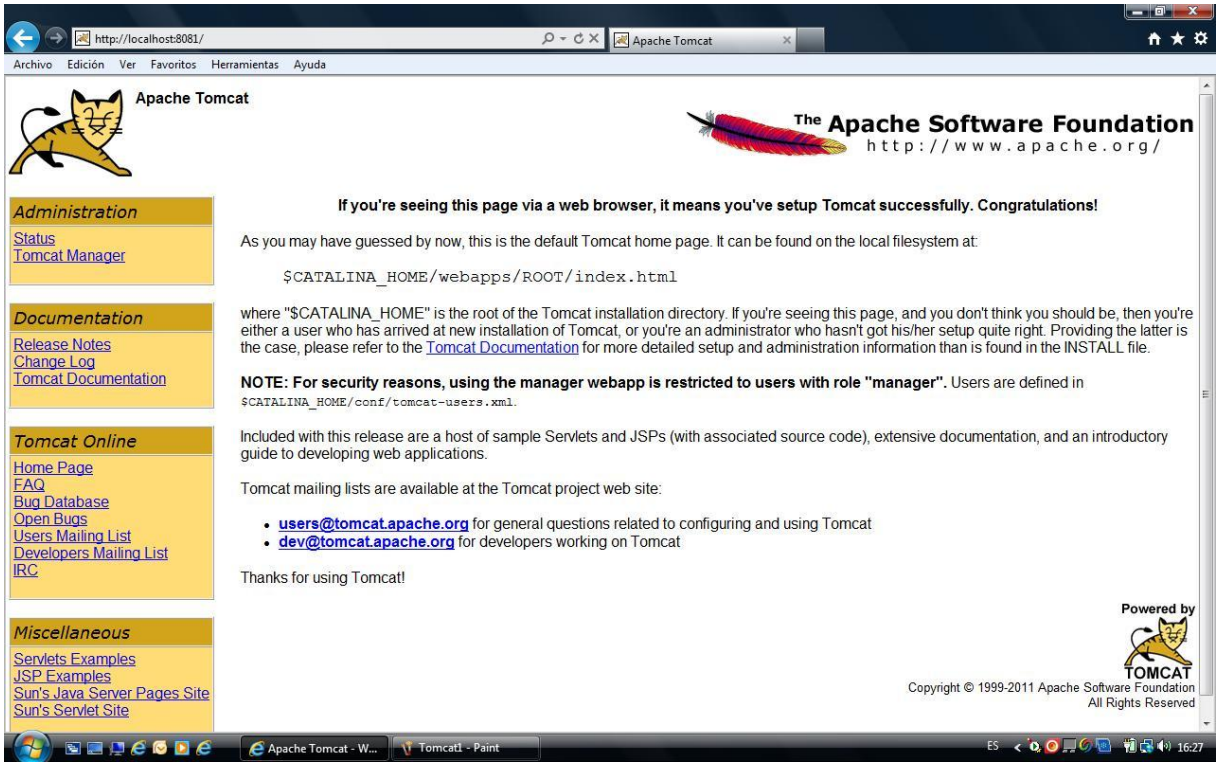
Así mismo, toda la aplicación, que estoy corriendo desde el Eclipse Helios y por lo tanto la estoy desplegando en un entorno de desarrollo, la he grabado en un fichero war exportable, por lo que podría desplegarse también en un entorno de producción.

Para ello, hay que grabar el fichero war en la carpeta webapps del Tomcat y, a partir de ahí, desde la aplicación web de administración del Tomcat, poner en marcha la aplicación. Eso sí, es necesario tener todos los directorios, ficheros y programas auxiliares correctamente instalados para que la aplicación funcione.

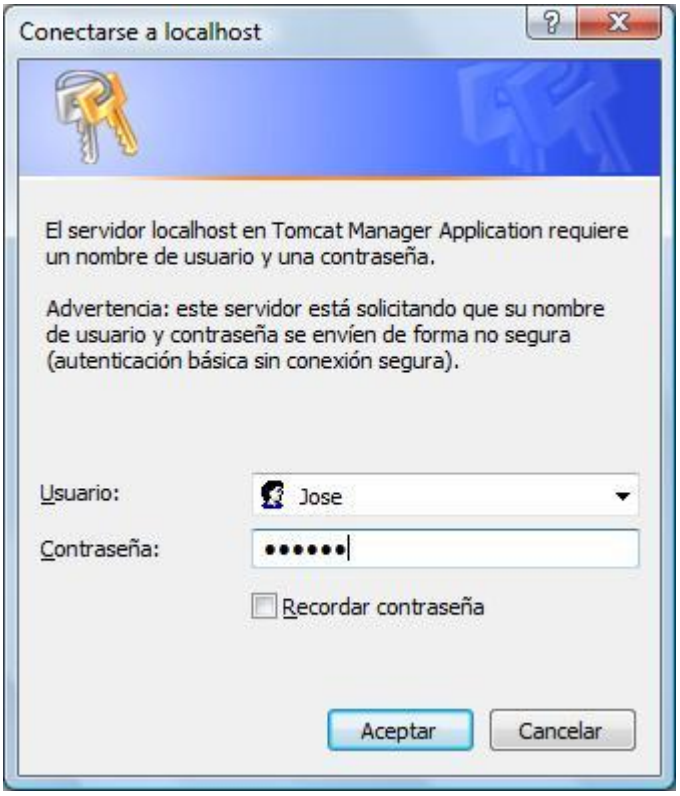
Para lanzar la aplicación, hay que arrancar el monitor del Tomcat.



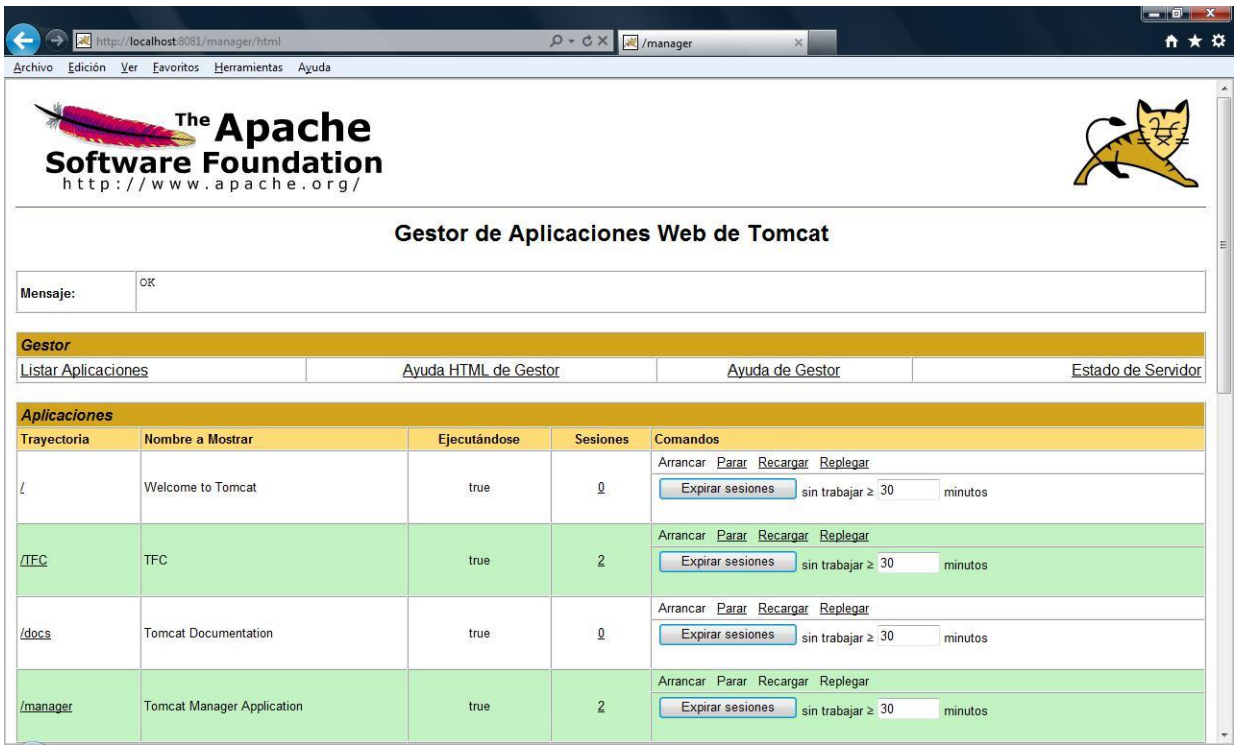
Una vez arrancado, hay ir a la página de administración del mismo (en mi caso http://localhost:8081) y, una vez nos sale la página de inicio del Tomcat,



clickar en el link “Tomcat Manager”, que nos redirigirá a la ventana de autenticación de administrador (de cuando instalamos el Tomcat) y nos pedirá nuestro nombre de usuario y contraseña



para que, una vez autenticados, nos permita lanzar la aplicación (de nombre TFC).



15.- RECURSOS CRIPTOGRÁFICOS DE BOUNCY CASTLE

Dado que la parte más específica de la aplicación web y la única relacionada con las técnicas de criptografía se ha realizado utilizando la librería de Bouncy Castle, procederemos a continuación a una descripción más detallada de los diferentes métodos utilizados.

Proveedor de seguridad

En primer lugar, cada vez que invocamos cualquier método de la librería hemos de establecer el proveedor de seguridad. Esto se realiza mediante la instrucción:

```
Security.addProvider(new BouncyCastleProvider());
```

Generación de claves

Para la generación de las claves (tanto las de la CA como las de los usuarios cuando se han necesitado), el proceso ha sido el mismo (transcribo el caso de la CA, pero son los dos iguales):

```
keyGen = KeyPairGenerator.getInstance("RSA", "BC");
keyGen.initialize(2048);
clausRSA = keyGen.generateKeyPair();
PrivateKey clauPrivada = clausRSA.getPrivate();
PublicKey clauPublica = clausRSA.getPublic();
```

Generación de los certificados

Para el caso de generación de los certificados (tanto de la CA como las dos versiones de usuarios) el sistema seguido ha sido el mismo. Primero se “consiguen” los diferentes parámetros que caracterizarán al certificado y, una vez conseguidos éstos, se genera el certificado:

```
X509V3CertificateGenerator certGen = new X509V3CertificateGenerator();
X500Principal dnName = new X500Principal("CN=Criptotal");
certGen.setSerialNumber(serialNumber);
certGen.setIssuerDN(dnName);
certGen.setNotBefore(startDate);
certGen.setNotAfter(expiryDate);
certGen.setSubjectDN(dnName);
```

```
certGen.setPublicKey(parellDeClaus.getPublic());
certGen.setSignatureAlgorithm("SHA256WithRSAEncryption");
certCA = certGen.generate(parellDeClaus.getPrivate(), "BC");
```

Guardar la clave privada

Para guardar la clave privada (tanto de la CA como del usuario), el paso previa es pasarla al formato PKCS8, para luego utilizar las funcionalidades Java para guardarla en un fichero.

```
PKCS8EncodedKeySpec pkcs8Spec = new
                PKCS8EncodedKeySpec(clauPrivada.getEncoded());
FileOutputStream out = new FileOutputStream(r1);
out.write(pkcs8Spec.getEncoded());
out.close();
```

Recuperar la clave privada

Se trata del proceso inverso al anterior.

```
PKCS8EncodedKeySpec clauPrivadaSpec = new
                PKCS8EncodedKeySpec(bufferPriv);
PrivateKey clauPrivada2;
clauPrivada2 = keyFactoryRSA.generatePrivate(clauPrivadaSpec);
```

Guardar la clave pública

La clave pública se guarda utilizando el formato X509Spec.

```
X509EncodedKeySpec x509Spec = new
                X509EncodedKeySpec(clauPublica.getEncoded());
out = new FileOutputStream(r2);
out.write(x509Spec.getEncoded());
out.close();
```

Recuperar la clave pública

Como anteriormente, se trata del proceso inverso al anterior.

```
X509EncodedKeySpec clauPublicaSpec = new X509EncodedKeySpec(bufferPub);
PublicKey clauPublica2;
clauPublica2 = keyFactoryRSA.generatePublic(clauPublicaSpec);
```

Guardar el certificado en un Keystore (y guardar el keystore)

El certificado de la CA lo guardamos en un KeyStore y éste lo guardamos en un fichero en el disco duro.

```
KeyStore magatzem = KeyStore.getInstance("PKCS12", "BC");

// Inicialitzem
magatzem.load(null, null);
char[] contrasenya = r5.toCharArray();
String aliasCA = r5;
magatzem.setCertificateEntry(aliasCA, certCA);
Certificate[] cadena = new Certificate[1];
cadena[0] = certCA;
magatzem.setKeyEntry(aliasCA, clauPrivada, contrasenya, cadena);

// Guardem el certificat
FileOutputStream out = new FileOutputStream(r3);
out.write(certCA.getEncoded());
out.close();

//Guardem el Keystore
out = new FileOutputStream(r4);
char[] password = r6.toCharArray();
magatzem.store(out, password);
out.close();
```

Recuperar el certificado de la CA

Estas son las instrucciones para recuperar el certificado de la CA del KeyStore.

```
String nom4=r3;
char[] password = r4.toCharArray();
KeyStore magatzem = KeyStore.getInstance("PKCS12", "BC");
in = new FileInputStream(nom4);
magatzem.load(in, password);
in.close();
CAcert = (X509Certificate) magatzem.getCertificate("Criptotal");
```

Añadir extensiones

Al crear los certificados del usuario, justo antes de la instrucción que crea el certificado, hemos añadido “Extensiones” para incluir la dirección URL donde se ubica la CRL. Este proceso se realiza mediante las instrucciones:

```
GeneralName gn = new GeneralName(GeneralName.uniformResourceIdentifier,new
                DERIA5String(r7));

GeneralNames gns = new GeneralNames(new DERSequence(gn));

DistributionPointName dpn = new DistributionPointName(0, gns);

DistributionPoint distp = new DistributionPoint(dpn, null, null);

certGen.addExtension(X509Extensions.CRLDistributionPoints, false, new
                DERSequence(distp));
```

Crear una CRL

Una vez que hemos hecho la consulta a la base de datos para saber qué certificados están revocados, hay que crear la CRL, para luego irlos añadiendo. Las instrucciones que generan la CRL, una vez hemos recuperado la clave privada de la CA, el certificado de la CA y el momento actual, son:

```
X509V2CRLGenerator crlGen = new X509V2CRLGenerator();

crlGen.setIssuerDN(new X500Principal("CN=Criptotal"));

crlGen.setThisUpdate(now);

crlGen.setSignatureAlgorithm(CAcert.getSigAlgName());

crlGen.addExtension(X509Extensions.AuthorityKeyIdentifier, false, new
                AuthorityKeyIdentifierStructure(CAcert));

crlBC = crlGen.generateX509CRL(CAKey, "BC");
```

Añadir un certificado a la CRL

Para añadir un certificado a la CRL se utilizan unas instrucciones muy parecidas:

```
BigInteger serialNumber = BigInteger.valueOf(serial);

Date now = new Date();

X509V2CRLGenerator crlGen = new X509V2CRLGenerator();
```

```
crlGen.setIssuerDN(new X500Principal("CN=Criptotal"));

crlGen.setThisUpdate(now);

crlGen.setSignatureAlgorithm(CAcert.getSigAlgName());

crlGen.addExtension(X509Extensions.AuthorityKeyIdentifier, false, new
                    AuthorityKeyIdentifierStructure(CAcert));

crlGen.addCRL(crlBC);

crlGen.addCRLEntry(serialNumber, now, CRLReason.privilegeWithdrawn);

nuevaLista = crlGen.generateX509CRL(CAKey, "BC");
```

16.- CONCLUSIONES

Una vez finalizado el TFC, llega la hora de hacer valoración del mismo y de hacer un análisis del cumplimiento de les expectativas primeras.

En el momento de iniciar el TFC y después de que el consultor confirmara lo escueto del enunciado y, sobre todo, que no se nos iba a dar ningún tipo de “esqueleto de clases” ni material introductorio, sobrevino un momento de pánico. En el TFC se espera del alumno la capacidad de investigar y resolver estos pequeños problemas, pero la indefinición me pareció abrumadora. Fue entonces cuando inicié la investigación (guiado por los comentarios del consultor) sobre los distintos elementos de programario a utilizar.

Esta investigación me llevó a familiarizarme con el contenedor Tomcat, las páginas JSP, los servlets de Java, el Eclipse Helios, el puente JDBC y tantas otras herramientas que he utilizado durante este trabajo.

Lo que en un principio parecía un mar sin riberas, fue poco a poco tomando forma en la aplicación y, lo que es más importante, limitando las fronteras de lo que hacía falta investigar o no .En este sentido, los elementos de la bibliografía que se agrupan bajo los apéndices “Configuración” y “JSP” fueron de gran utilidad.

De este primer gran bloque de trabajo quiero resaltar especialmente la satisfacción de haber dado forma “desde cero” a la aplicación y los conocimientos de XHTML que he adquirido en el proceso.

Empezaba entonces lo “realmente relacionado” con la Criptografía, con un primer estadio en el que simplemente se trataba de generar las claves y el certificado de la CA, para luego dar paso al resto de las funcionalidades de la aplicación.

Como nunca había trabajado con funcionalidades de Criptografía desde Java, ésta ha sido la parte que ha significado un mayor aprendizaje (con la dificultad que ello conlleva). Sin embargo, el resultado final me hace estar bastante satisfecho de los avances realizados debido a la amplitud de conceptos que he tenido que manejar (claves y certificados, KeyStores y Extensiones, manejo de ficheros en Java, recepción y envío de ficheros en una aplicación web, servlets y bases de datos, revocaciones y listas CRL…)

Cada vez que el enunciado del TFC o algún comentario del consultor me llevaban a un nuevo campo (lo que ocurría muy a menudo) se repetía el proceso de angustia, búsqueda de material, prueba y error, solución. Todo ello se ha traducido en un proceso largo y laborioso pero al mismo tiempo, muy gratificante una vez se ha podido resolver.

Soy consciente de que la aplicación puede evolucionar (y es posible que lo haga en un futuro, aunque ya fuera del ámbito del TFC) para incluir algunas funcionalidades que, si bien no forman parte del enunciado básico, sí lo forman de las posibles extensiones propuestas por el consultor. En este sentido, la autenticación “fuerte” del administrador remoto o la renovación de certificados son trabajos que podrían incluirse sin demasiada dificultad.

Por último, y ya en el plano de la valoración personal, he disfrutado mucho en la realización del TFC, aunque el, a veces, demasiado largo proceso de aprendizaje y prueba-error, han conseguido hacerme caer en el desánimo más de una vez.

Quiero resaltar la paciencia y disponibilidad del consultor durante todo el proceso, dos facultades sin las cuales me temo que yo no habría sido capaz de llevar a cabo toda la implantación.

17.- GLOSARIO

Incluyo a continuación un breve glosario de términos utilizados en el presente TFC con la finalidad de facilitar la comprensión del texto. Si bien la mayoría de las definiciones aquí plasmadas, por comodidad, han sido extraídas de la wikipedia (es.wikipedia.org), se han confrontado todas con los apuntes de las asignaturas “Seguridad en redes de computadores” y “Criptografía”.

- **Algoritmo**: Un algoritmo (del griego y latín, dixit algorithmus y éste a su vez del matemático persa Al Juarismi) es un conjunto preescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución

- **Apache Tomcat**: Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) es un servidor web que funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

- **Autoridad Certificadora**: En criptografía una Autoridad de certificación, certificadora o certificante (AC o CA por sus siglas en inglés Certification Authority) es una entidad de confianza, responsable de emitir y revocar los certificados digitales o certificados, utilizados en la firma electrónica, para lo cual se emplea la criptografía de clave pública o asimétrica.

- **Base de datos**: Una base de datos o banco de datos (en ocasiones abreviada con la sigla BD) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos. Existen programas denominados sistemas gestores de bases de datos, abreviado SGBD, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

- **Bouncy Castle**: Bouncy Castle es una colección de es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) para ser utilizado por otro software como una capa de abstracción utilizados en criptografía. Tiene versiones para los lenguajes Java y C#.

- **Certificado autofirmado**: Se trata del certificado de una CA que tiene la particularidad de estar firmado por la propia CA. Es decir, el Sujeto y el Expedidor del certificado son la propia CA.

- **Certificado digital**: Un certificado digital (también conocido como certificado de clave pública o certificado de identidad) es un documento digital mediante el cual un tercero confiable (una autoridad de certificación) garantiza la vinculación entre la identidad de un sujeto o entidad (por ejemplo: nombre, dirección y otros aspectos de identificación) y una clave pública. Este tipo de certificados se emplea para comprobar que una clave pública pertenece a un individuo o entidad. La existencia de firmas en los certificados aseguran por parte del firmante del certificado (una autoridad de certificación, por ejemplo) que la información de identidad y la clave pública perteneciente al usuario o entidad referida en el certificado digital están vinculadas.

- **Certificado revocado**: Un certificado revocado es un certificado que no es válido aunque se emplee dentro de su período de vigencia. Un certificado revocado tiene la condición de suspendido si su vigencia puede restablecerse en determinadas condiciones.

- **Clave privada y pública**: En la 'criptografía asimétrica' se usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves. Si el remitente usa la clave pública del destinatario para cifrar el mensaje, una vez cifrado, sólo la clave privada del destinatario podrá descifrar este mensaje, ya que es el único que la conoce. Por tanto se logra la confidencialidad del envío del mensaje, nadie salvo el destinatario puede descifrarlo. Si el propietario del par de claves usa su clave privada para cifrar el mensaje, cualquiera puede descifrarlo utilizando su clave pública. En este caso se consigue por tanto la identificación y autentificación del remitente, ya que se sabe que sólo pudo

haber sido él quien empleó su clave privada (salvo que alguien se la hubiese podido robar). Esta idea es el fundamento de la firma electrónica.

- **Contenedor web**: En la plataforma Java 2 Enterprise Edition, un contenedor web es la implementación que hace cumplimiento del contrato de componentes web de la arquitectura J2EE. Por lo tanto, los Contenedores Web son unos entornos de ejecución que encapsulan los protocolos HTTP y TCP/IP, que contienen Servlets y Paginas Web dinámicas y que pueden contener Páginas Web e imágenes

- **Criptografía asimétrica**: La criptografía asimétrica es el método criptográfico que usa una clave para el cifrado del mensaje y otra diferente (pero vinculada a la anterior) para el descifrado. Los sistemas de cifrado de clave pública o sistemas de cifrado asimétricos se inventaron con el fin de evitar por completo el problema del intercambio de claves de los sistemas de cifrado simétricos. Con las claves públicas no es necesario que el remitente y el destinatario se pongan de acuerdo en la clave a emplear. Todo lo que se requiere es que, antes de iniciar la comunicación secreta, el remitente consiga una copia de la clave pública del destinatario. Es más, esa misma clave pública puede ser usada por cualquiera que desee comunicarse con su propietario. Por tanto, se necesitarán sólo n pares de claves por cada n personas que deseen comunicarse entre sí.

- **CRL**: CRL es la sigla de "Certificate Revocation List", que significa "lista de certificados revocados". En la operación de algunos sistemas criptográficos, usualmente los de infraestructura de clave pública (PKI), una CRL es una lista de certificados (más concretamente sus números de serie) que han sido revocados, ya no son válidos y en los que no debe confiar ningún usuario del sistema.

- **CSR**: CSR son las siglas de Certificate Signing Request, que traducimos por Petición de Firma de un Certificado, y que se aplican a las peticiones de certificado en formato PKCS10 que un usuario envía a la CA.

- **Deprecado**: Dícese del método (función o procedimiento en la programación orientada a objetos) que, pese a estar todavía operativo no es recomendable usar por existir otro más moderno que realice sus funcionalidades.

- **Eclipse**: Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de

Cliente Enriquecido". Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Eclipse dispone de un Editor de texto con resaltado de sintaxis y realiza la compilación en tiempo real.

- **Extensión de un certificado**: La recomendación X.509 de la UIT-T la describe como: “Campo adicional de un certificado formado por un identificador, una bandera de condición crítica y una codificación canónica de un valor de datos de un tipo ASN.1 asociado a dicha extensión”.

- **PKCS10**: En criptografía, PKCS (Public-Key Cryptography Standards) se refiere a un grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA en California. En concreto el PKCS 10 define un formato de los mensajes enviados a una Autoridad de certificación para solicitar la certificación de una clave pública.

- **PKCS12**: Dentro de la familia PKCS, el PKCS 12 consiste en un estándar de sintaxis de intercambio de información personal. Define un formato de fichero usado comúnmente para almacenar claves privadas con su certificado de clave pública protegido mediante clave simétrica.

- **PKI**: Una infraestructura de clave pública (PKI, Public Key Infrastructure) es el conjunto de hardware, software, personas, políticas y procedimientos que se necesitan para crear, manejar, almacenar, distribuir y revocar certificados digitales basados en la criptografía asimétrica.

- **Servlet**: Los servlets, son objetos que corren dentro del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad. La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor. El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

- **SQL**: El lenguaje de consulta estructurado o SQL (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre ella.

18.- BIBLIOGRAFÍA

Para la realización de este trabajo he consultado una gran cantidad de bibliografía para irme familiarizando con los diferentes problemas a los que me iba enfrentando. He clasificado la bibliografía en subcapítulos para favorecer la localización de los recursos.

Todos los recursos de Internet han sido consultados entre febrero y mayo de 2.011, por lo que omitiré mencionar la fecha concreta de la consulta.

Libros

- Chopra [et al.]. *Beginning Java Server Pages*. Indianapolis: Editorial Wiley Publishing, Inc. 2005.
- Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and source code in C*, Second Edition. Minneapolis: Editorial John Wiley & Sons, Inc. 1996.
- García, Rodríguez e Imaz. *Aprenda Servlets de Java como si estuviera en primero*. San Sebastián: Editorial del Campus Tecnológico de la Universidad de Navarra. 1999.
- Belén Albeza. *XHTML+CSS de una maldita vez!*. Stanford: Editorial Creative Commons Attribution NonCommercial NoDerivs License. 2004.
- David Hook. *Beginning Cryptography with Java*. Indianapolis: Editorial Wrox Press. 2005.
- Jonathan Knudsen. *Java Cryptography*. Sebastopol: O'Reilly & Associates Inc. 1998.
- S. Allamaraju [et al.]. *Programación Java Server con J2EE,* Edición 1.3. Madrid: Editorial Anaya Multimedia. 2002.
- Rolf Oppliger. *Security Technologies for the World Wide Web*, Second Edition. Norwood: Editorial Artech House. 2003
- Elisabeth and Eric Freeman. *Head first HTML with CCS & XHTML*. Sebastopol: Editorial O'Reilly Media Inc. 2005.

- Bates, Sierra y Basham. *Head first Servlets and JSP*. Sebastopol: Editorial O'Reilly Media Inc. 2004.
- Bruce Eckel. *Piensa en Java*. Madrid: Editorial Prentice Hall. 2002.

Recursos de Internet

CONFIGURACION

http://www.datanoia.com/tutorial-instalacion-y-configuracion-apache-tomcat.html

http://www.cristalab.com/tutoriales/instalar-eclipse-y-tomcat-para-desarrollo-con-java-c64596l/

http://www.arcoe.es/2009/01/27/comenzar-un-proyecto-web-con-eclipse-y-servidor/

http://www.mygnet.net/articulos/java/aplicaciones_web_en_eclipse.708

http://www.programacion.com/articulo/crear_tu_primera_aplicacion_web_real_con_tomcat_4_y_mysql_215

http://javacafesv.blogspot.com/2009/04/emision-de-certificados-ssl-con-apache.html

http://coyotevil.blogspot.com/2006/11/configurar-el-tomcat-con-certificado.html

http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/

http://chuwiki.chuidiang.org/index.php?title=A%C3%B1adir_un_jar_a_nuestro_proyecto

http://www.bouncycastle.org/specifications.html

http://code.google.com/p/xmind3/wiki/How_to_build_XMind_from_source

http://foro.chuidiang.com/ides/incorporar-librerias-externas-a-proyectos-de-eclipse/

http://tirl.org/blogs/media-lab-blog/46/

http://osl2.uca.es/wikiCE/index.php/Apache_Struts

http://www.coreservlets.com/Apache-Tomcat-Tutorial/eclipse.html

JSP

http://www.desarrolloweb.com/articulos/2180.php

http://www.desarrolloweb.com/articulos/2276.php

http://www.desarrolloweb.com/articulos/2339.php

http://www.cristalab.com/tutoriales/crear-sitios-y-aplicaciones-web-en-jsp-con-java-c74730l/

http://www.proactiva-calidad.com/java/principal.html

http://www.hotscripts.com/category/java/jsp-servlets/

http://www.miwebjava.com/?p=28

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=miPrimeraWebStruts

http://www.proactiva-calidad.com/java/servlets/introduccion.html

http://www.programacion.com/articulo/servlets_y_jsp_82

http://jj.merelo.net/~jmerelo/JSP/

http://www.todoroms.com/web-development-with-java-server-pages

http://www.uv.es/grimo/teaching/SpringMVCv3PasoAPaso/part1.html

http://kikev.wordpress.com/2009/08/04/un-acercamiento-a-jsp/

http://www.programacionfacil.com/programacion:manual_java_jsp

BOUNCY CASTLE

http://www.bouncycastle.org/java.html

http://www.mobilefish.com/developer/bouncycastle/bouncycastle.html

http://ccia.ei.uvigo.es/docencia/SSI/practicas/jce.html

http://nachxs.wordpress.com/2010/02/11/tripledes-con-java-ii-bouncycastle/

http://www.opencertiac.org/recursos/OPENCERTIAC.Documentacion.Programador.Soluciones%20JS2E%20de%20Firma%20Digital.pdf

http://edwinmartinezrueda.blogspot.com/2010/07/firma-de-documentos-con-archivos-p12.html

http://bouncy-castle.1462172.n4.nabble.com/Problem-in-generation-certificates-td1462449.html

http://www.ernestojpg.com/

http://download.oracle.com/javase/6/docs/technotes/guides/security/

http://download.oracle.com/javase/6/docs/technotes/guides/security/cert3.html

http://download.oracle.com/javase/6/docs/technotes/guides/security/certpath/CertPathProgGuide.html

http://bouncy-castle.1462172.n4.nabble.com/CRLDistPoint-How-to-add-this-extension-td1466589.html

http://www.bouncycastle.org/wiki/display/JA1/X.509+Certificate+Revocation+Lists

SERVLETS

http://sanchez-soft.blogspot.com/2006/07/java-cear-un-servlet-en-eclipse.html

http://www.manualweb.net/java-ee/introduccion-a-los-servlets/

http://foro.chuidiang.com/javascript/llamar-a-clases-java-desde-javascript/

http://mediawiki.uca.es/index.php/Servlets

RESOURCEBOUNDLE

http://www.programacion.com/foros/java-basico/resourcebundle_193851

GESTIÓN DE ARCHIVOS

http://www.forosdelweb.com/f67/problema-guardar-archivo-ruta-apliacion-struts2-835220/

http://knol.google.com/k/juan-eduardo-ling%C3%A1n-cubas/leer-archivos-en-una-aplicaci%C3%B3n-web/4kj9c60cymla/5#

http://txos.blogspot.com/2007/01/file-download-servlet.html

http://codigos.zonalibre.org/archives/2009/08/como-subir-archivos-al-servidor-con-java-servlet.html

http://www.mkyong.com/java/how-to-download-file-from-website-java-jsp/

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=fileupload

http://fisiwikipedia.wikispaces.com/file/view/Manejo+de+Archivos+en+Java.pdf

http://lineadecodigo.com/java/numero-de-lineas-de-un-fichero/

http://www.chuidiang.com/java/novatos/entrada_standard_java.php

VARIOS

http://publikaccion.blogspot.com/2007/05/tipos-de-formatos-de-certificados.html

http://es.wikipedia.org/wiki/Base64

http://fisiwikipedia.wikispaces.com/file/view/Manejo+de+Archivos+en+Java.pdf

http://lineadecodigo.com/java/numero-de-lineas-de-un-fichero/

http://translate.google.es/translate?hl=es&langpair=en%7Ces&u=http://www.javadb.com/remove-a-line-from-a-text-file

http://www.desarrolloweb.com/manuales/21/

https://ws024.juntadeandalucia.es/plutonDescargas/20_395_gu-2005-01_generacion_de_certificado_de_servidor.pdf

http://es.wikipedia.org/wiki/Base64

http://publikaccion.blogspot.com/2007/05/tipos-de-formatos-de-certificados.html

http://www.bibigeek.com/2010/02/01/java-y-los-ficheros-properties/

httphttp://gs.statcounter.com/#os-ww-monthly-200812-

201001://tools.ietf.org/html/rfc5280