UNIVERSIDAD DE MÁLAGA
ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA:

DESARROLLO DE UN SISTEMA DE CONTROL WEB DE ACCESO A LOS LABORATORIOS DEL DTE

INGENIERÍA TELECOMUNICACIÓN

Málaga, 2007 José Luis Guerrero Marín

UNIVERSIDAD DE MÁLAGA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Titulación: Ingeniería de Telecomunicación

Reunido el tribunal examinador en el día de la fecha, constituido por:

D. _____

D. _____

D. _____

para juzgar el Proyecto Fin de Carrera titulado:

DESARROLLO DE UN SISTEMA DE CONTROL WEB DE ACCESO A LOS LABORATORIOS DEL DTE.

del alumno D. José Luis Guerrero Marín

dirigido por D. Francisco Javier González Cañete

ACORDÓ POR: _____ OTORGAR LA CALIFICACIÓN DE

_____

y, para que conste, se extiende firmada por los componentes del Tribunal, la presente diligencia.

Málaga, a ____de_____de Año

El Presidente: El Secretario: El Vocal:

Fdo. _____ Fdo. _____ Fdo. _____

UNIVERSIDAD DE MÁLAGA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DESARROLLO DE UN SISTEMA DE CONTROL WEB DE ACCESO A LOS LABORATORIOS DEL DTE

REALIZADO POR:

José Luis Guerrero Marín

DIRIGIDO POR:

Francisco Javier González Cañete

DEPARTAMENTO DE: Tecnología Electrónica.

TITULACIÓN: Ingeniería de Telecomunicación

Palabras claves: Java, JSP, servlets, JavaScritpt, HTML, CSS, PDF, abrepuertas, puerto serie.

RESUMEN:

El objetivo de este proyecto es dotar a los laboratorios del DTE o Departamento de Tecnología Electrónica de un renovado y completo sistema de gestión de sus accesos, tanto de la parte software como de la hardware. Por una parte se pretende posibilitar todas las labores de mantenimiento de forma remota a través de un navegador Web en cuanto a los usuarios que acceden, los momentos en que se puede acceder, y las características de cada laboratorio, permitiendo asimismo obtener informes personalizables de los datos más relevantes, y por otra parte renovar el obsoleto método de apertura de puertas que funciona actualmente.

Málaga, Febrero de 2007

Al “lelo”, el primer teleco de la familia,
a mis padres, a mi hermana y a Lorena.

Agradecimientos

En este apartado no quiero mencionar a nadie directamente, pues cada quien sabe lo que merece y el lugar que ocupa en mi recuerdo sin necesidad de tenerlo por escrito, pero sí me siento en la obligación de describir la cara y la cruz de lo que todos estos años de duro estudio me han hecho vivir.

Gracias por esas fugaces risas en la cafetería pasando cortos y geniales momentos en buena compañía. Gracias por estar ahí para echarme una mano sin que lo pida. Gracias por ser mi paño de lágrimas. Gracias por dejarme ese examen. Gracias por ayudarme en todo lo que pudiste. Gracias por apoyarme aún cuando fue difícil. Gracias por ser mi amigo. Gracias por ser mi amiga. Gracias por escucharme cuando soy insoportable. Gracias por acordarte de mí. Gracias por entenderme con una mirada. Gracias por una segunda oportunidad. Gracias por seguir ahí para mí. Gracias por dejarme esos apuntes sin que lo mereciera. Gracias por ser mi prisión del placer.

Tampoco puedo olvidarme de todos aquellos de mi entorno más cercano y ¿familiar? que me lo hicieron más difícil:

Te equivocaste cuando pensaste que no todos pueden ser ingenieros. Te equivocaste cuando pensaste que podías juzgarme con una mirada. Te equivocaste cuando me menospreciaste y te refugiaste en tu altivez, tu prepotencia y tu verborrea insostenible. Te equivocaste cuando pensaste que tengo algo que agradecerte. Te equivocaste cuando pensaste que estás en la cima del mundo. Te equivocaste cuando pensaste que podías pisotearme. Te equivocaste cuando pensaste que me podrías hundir, pero sobre todo te equivocaste cuando pensaste que eres una persona.

Y de ninguna manera puedo olvidar al que ha sido mi compañero de estudios incansable…Gracias Chuck Norris, por no dejar de hacer flexiones mientras yo intentaba entender circuitos sin sentido o ecuaciones que se salían del folio.

Sólo dos cosas me quedan por decir, una: Mónica, Dani dejó la mochila en casa porque la olvidó en clase y dos: si alguien es capaz de leer todo el proyecto entero… MUCHAS GRACIAS

Índice de Contenidos

Relación de Acrónimos

AC	Autoridades Certificadoras
API	*Application Programming Interface*
ASP.NET	*Active Server Pages* con tecnología *.NET*
CCITT	*Consultative Committee for International Telephony and Telegraphy*
CD	*Compact Disc*
CGI	*Common Gateway Interface*
CSS	*Cascade StyleSheet*
DBMS	*DataBase Manager System*
DHTML	*Dynamic HTML*
DOM	*Document Object Model*
DTE	Departamento de Tecnología Electrónica
EAI	*Enterprise Application Integration*
EDI	*Electronic Data Interchange*
EIA	Asociación de Industrias Electrónicas
FOP	*Formatting Object to* PDF
GPL	*General Public License*
HTML	*Hypertext Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IMAP	*Internet Message Access Protocol*
ISA	*Industry Standard Architecture*
J2EE	*Java 2 Enterprise Edition*
JCS	*Java Caching System*
JDT	*Java Development Toolkit*
JRE	*Java Runtime Environment*
JSP	*Java Server Pages*
LDAP	*Lightweight Directory Access Protocol*
MathML	*Mathematical Markup Language*
MS-DOS	*Microsoft Disk Operating System*
MSXML	*Microsoft XML parser*
NIF	Número de Identificación Fiscal

OASIS	*Organization for the Advancement of Structured Information Standards*
ODBC	*Open DataBase Connectivity*
OSI	*Open Systems Interconnection*
PDF	*Portable Document Format*
PERL	*Practical Extraction and Reporting Language*
PHP	*HyPertext Preprocessor*
PKI	*Public Key Infraestructura*
POP3	*Post Office Protocol* 3ª edición
PS/2	*IBM Personal System/2*
RPC	*Remote Procedure Call*
S/MIME	*Secure Multipurpose Internet Mail Extensions*
SGBD	*Sistemas Gestores de Bases de Datos*
SGML	*Standard Generalized Markup Language*
SMTP	*Simple Mail Transfer Protocol*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
SSL	*Secure Socket Layer*
STP	*Spanning Tree Protocol*
TCP	*Transfer Control Protocol*
TCP/IP	*Transfer Control Protocol/Internet Protocol*
TLS	*Transport Layer Security*
TXT	*TeXT*
UFO	*Unicorn Formatting Objects*
USB	*Universal Serie Bus*
UTF-8	*8-bit Unicode Transformation Format*
W3C	*World Wide Web Consortium*
WSDL	*Web Services Description Language*
WWW	*World Wide Web*
XHTML	*eXtensible, HyperText Markup Language*
XML	*eXtended Markup Language*
XPath	*XML Path language*
XSL	*eXtensible Stylesheet Language*
XSL-FO	XSL-*Formatting Objects*
XSLT	*XSL Transformations*

RFID *Radio Frequency IDentification*

Índice de Figuras

Índice de Tablas

CAPÍTULO 1: INTRODUCCIÓN

Actualmente el control de acceso a los laboratorios del departamento de Tecnología Electrónica se realiza de forma local, ofreciendo un servicio muy limitado y haciendo muy difícil las labores de mantenimiento por parte del técnico correspondiente.

Con el sistema vigente el usuario teclea un número de acceso en un teclado microprocesado en la entrada del laboratorio, y es el ordenador local al que está conectado el teclado el que decide si tiene acceso o no. El interfaz entre el teclado microprocesado y el ordenador local es una tarjeta de adquisición de datos insertada en una ranura ISA totalmente obsoleta que sólo existe en ordenadores muy antiguos y que por lo tanto sólo pueden usar sistemas operativos obsoletos en la actualidad, en este caso MS-DOS (*Microsoft Disk Operating System*). Dicho esto, quedan de manifiesto varios problemas asociados a este sistema:

- Es necesario usar ordenadores antiguos con el consiguiente riesgo de que un fallo hardware en el equipo sea difícil de solventar.
- Cualquier operación de mantenimiento se debe realizar de forma local, lo que sugiere un evidente desperdicio de recursos al existir una red de datos en la Universidad.
- No se pueden elaborar informes de uso de los laboratorios.

En este proyecto se pretende dar solución a todos esos problemas proporcionando además nuevos servicios respecto al sistema que existe hoy por hoy.

Se han implementado dos opciones que incluso pueden usarse simultáneamente. En la primera de ellas, el sistema consta de un teclado conectado al ordenador del laboratorio, en la segunda se dispone de un lector RFID (*RadioFrequency IDentification*) conectado al ordenador del laboratorio. Y es este ordenador el encargado de recoger las contraseñas de los usuarios y enviarlas al servidor para comprobar si con esa contraseña se puede acceder en ese momento a ese laboratorio. En caso de que el servidor no esté accesible, el ordenador del laboratorio buscará en su base de datos local, y si se accedió a ese laboratorio con esa contraseña en las últimas 24 horas se permite el acceso. También se ha implementado en este sistema el mecanismo físico que realiza la apertura usando uno de los puertos disponibles del ordenador.

Por otra parte, se ha implementado el servicio Web al que se conecta el ordenador local para comprobar si se da acceso o no a un usuario. Además, el sistema debe poner a disposición del técnico una herramienta Web que, previa identificación, permita introducir nuevos laboratorios y usuarios y realizar todas las modificaciones que se consideren oportunas en los ya existentes. En el caso de los usuarios, se deben poder modificar sus privilegios, fecha de caducidad de su clave privada, datos personales y otros datos que se detallarán más adelante en este documento. Al mismo tiempo, el sistema deberá estar dotado de un calendario, con la correspondiente representación gráfica, que permita al técnico introducir días festivos y de horario especial que restrinjan el uso de los laboratorios, almacenándose los cambios y permitiendo así la planificación de los cursos académicos en cuanto a disponibilidad de los laboratorios.

Junto a todo esto, el sistema debe ser capaz de generar informes en formato PDF (*Portable Document Format*) que se requieran: de los días festivos y de horario especial, listado de laboratorios, listado de usuarios con un determinado perfil y registro de accesos con un determinado patrón.

El sistema debe funcionar tanto en Windows como en Linux y se debe adoptar un mecanismo de apertura de puertas que reemplace al existente para evitar el uso obligado de equipos anticuados.

Paso a paso, el nuevo proceso sería el siguiente:

- El servidor arranca con los parámetros adecuados.
- En el laboratorio se ejecuta el programa encargado de recoger y enviar las claves que se meten vía teclado o vía lector RFID en la puerta. En ese momento recibe automáticamente vía servicio Web la lista de los laboratorios disponibles en la red y el técnico puede entonces configurar los parámetros correspondientes en cada caso según se use una modalidad u otra de entrada de contraseñas.
- El usuario mete su clave única de acceso al laboratorio (vía teclado o vía lector RFID). A partir de aquí pueden suceder dos cosas:
    - Si el servidor funciona correctamente se envía de forma segura la clave al servidor que responde devolviendo la clave introducida y si tiene acceso o no. El acceso queda registrado en una base de datos global y en otra local (caché), para eventuales caídas del servidor o inaccesibilidad del mismo.

- Si el servidor ha caído o no está accesible se consulta la base de datos local (caché) y en caso de haber accedido en las 24 horas anteriores al laboratorio con esa contraseña se le dará acceso.

➢ En cualquiera de los dos casos, si se da acceso, se acciona el mecanismo de apertura de puertas y en caso contrario no se acciona.

La arquitectura del sistema se muestra en la figura 1.1:

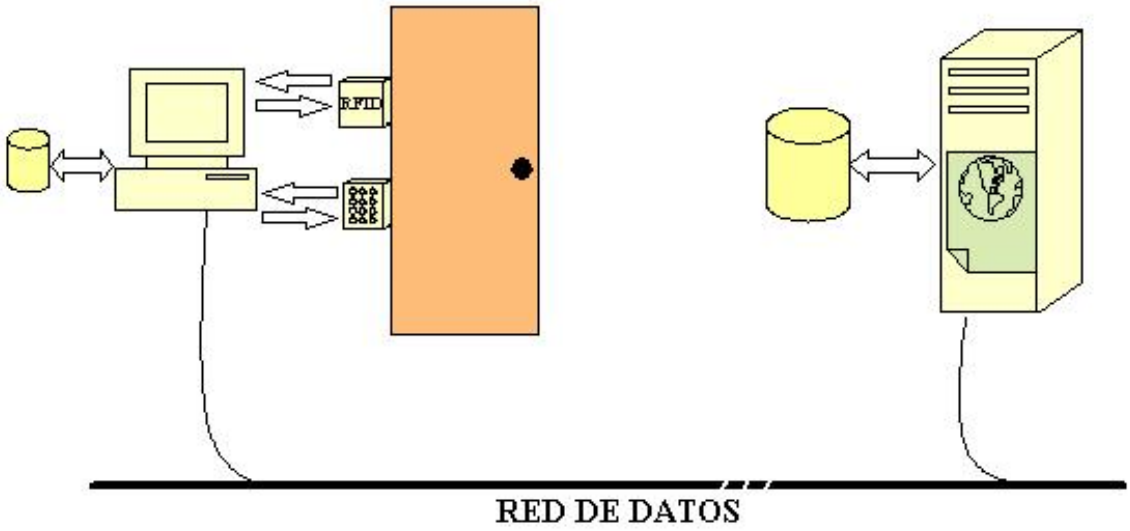

Figura 1.1. Esquema general del sistema de acceso

Con este proyecto fin de carrera se cumplen todos los citados objetivos a un coste mínimo, ya que todas las herramientas software empleadas son libres, y la parte hardware se ha procurado simplificar al máximo sin sacrificar en ningún momento la funcionalidad.

Respecto a la idea del acceso por lector RFID, cabe resaltar que no estaba planteada como objetivo a cumplir en el anteproyecto, pero se decidió incluir para conseguir un sistema más completo y versátil. La filosofía del sistema sigue siendo exactamente la misma, con la diferencia de que la entrada de datos se realiza a través del puerto serie mediante un lector adecuado. De hecho, y tal y como se ha diseñado, el programa es capaz de leer cualquier entrada de datos que vaya por el puerto serie, siempre y cuando los datos sigan el mismo protocolo que se ha programado para el puerto serie (mismos caracteres de inicio y final y número de caracteres de contraseña), ya que todos los parámetros de lectura de dicho puerto son configurables a través del programa.

Para explicar en su totalidad el proyecto este documento se encuentra estructurado en los siguientes capítulos:

1. Introducción: es el presente capítulo en el cual se hace una somera explicación y justificación del proyecto y la necesidad que cubre, así como introducir al resto de capítulos que conforman el documento.
2. Tecnologías y herramientas empleadas: en el cual se explican y justifican cada una de las herramientas software y hardware usadas, comparándolas a otras similares para ver sus ventajas e inconvenientes.
3. Diseño del sistema de apertura de puertas: en el cual se describe el software y hardware desarrollado y se expone la puesta en marcha y el uso en la parte local (laboratorio). También se dedica un apartado a describir las funciones y clases implementadas para conseguir el correcto funcionamiento del sistema.
4. Diseño de la aplicación Web de gestión de accesos: en el cual se relata el desarrollo del sitio Web desde la que se realizan todas las labores de mantenimiento del control de accesos a los laboratorios del DTE (*Departamento de Tecnología Electrónica*) y su uso. Al igual que en el capítulo anterior se dedica un apartado a describir las funciones y clases implementadas para conseguir el correcto funcionamiento del sistema.
5. Instalación y manual de usuario: en el cual se explica paso a paso cómo instalar todos los elementos que conforman el sistema para que funcione correctamente así como su uso.
6. Plan de pruebas: en el cual se detallan todas y cada una de las pruebas realizadas al sistema para comprobar su correcto funcionamiento, al margen de las realizadas en la fase de desarrollo.
7. Conclusiones y líneas futuras: en el que se hace un pequeño balance del proyecto, su utilidad, posibles ampliaciones y mejoras.

Anexo A. Presupuesto: en el que detalla el coste de poner en marcha realmente el sistema.

CAPÍTULO 2: TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS

2.1 INTRODUCCIÓN

En la realización de este proyecto se han puesto en conjunción multitud de herramientas tanto software (en su mayoría) como hardware, que a lo largo de este capítulo serán explicadas y comentadas, así como la justificación de su uso, en contraposición a otras con propósitos equivalentes. Se dará una somera explicación de cada una de las tecnologías disponibles y de la medida en que han participado en el presente proyecto. Cabe destacar que todas las herramientas software empleadas son de libre uso y distribución siguiendo unas premisas fundamentales a lo largo de todo el proyecto que son el bajo coste y la estandarización. Igualmente en la parte hardware se ha procurado conseguir los mejores resultados posibles al mínimo coste y máxima sencillez.

2.2 JAVA

Prácticamente desde que empezaron a surgir lenguajes de programación de alto nivel su ciclo de vida siempre ha sido el mismo, nacen, se extienden, y, pasado un cierto tiempo surge otro que lo supera y el primero cae en el olvido. Desde hace unos años parece que este paradigma ha cambiado, o al menos, se ha frenado bastante con la aparición de Java.

Y es que este lenguaje ofrece unas ventajas que no se pueden ignorar. Cabe destacar entre ellas fundamentalmente a dos, es seguro y es multiplataforma.

- Por una parte es seguro porque es un lenguaje que, por naturaleza, no permite acceder a la máquina físicamente más que a través de librerías que sí dependen del sistema operativo y que no son estándares de Java. Una de estas librerías ha sido necesaria para la ejecución de este proyecto, ya que hay una parte hardware (apertura física de la puerta) que había que solventar.
- Por otra parte es multiplataforma, ya que se ejecuta, a través de los *bytecodes*, sobre la máquina virtual de Java, que actúa como interfaz (esta sí es dependiente del sistema operativo) y que se puede instalar de forma gratuita en cualquier sistema operativo.

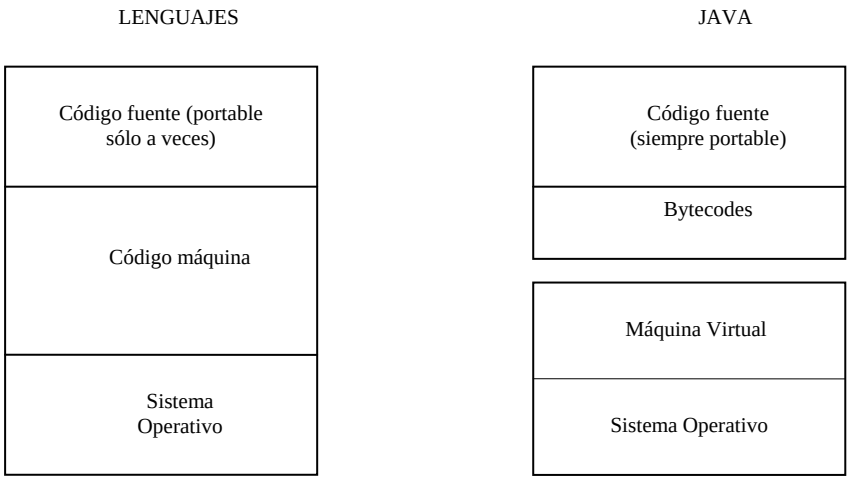

Figura 2.1. Diferencias entre Java y los lenguajes tradicionales

Son estas y otras ventajas las que lo hacen merecedor de estar presente como lo está en cada vez más aplicaciones de la WWW (*World Wide Web*) y en caso de este proyecto, de ser el eje fundamental sobre el cual se sustenta toda la aplicación.

En lo que respecta a este proyecto las claves para elegir Java han sido, por una parte, el hecho de ser multiplataforma, y por otra, la gratuidad del mismo. En ambos factores Java está aventajado frente a otros lenguajes como Visual Basic o Visual C, que bien podrían haber servido para los mismos propósitos.

2.3 SERVICIOS WEB

Un servicio Web es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Es una definición de las muchas posibles que se pueden encontrar, pero lo más interesante de los servicios Web es que distintas aplicaciones desarrolladas por separado y funcionando sobre diferentes sistemas operativos pueden interoperar e intercambiar datos a través de ellos. Esto es posible, por supuesto, gracias al uso de estándares abiertos y organizaciones como OASIS (*Organization for the Advancement of Structured Information Standards*) y W3C (*World Wide Web Consortium*) que se encargan de crear y velar por estos estándares.

Otro de los puntos fuertes de los servicios Web es que, al apoyarse en HTTP (*HyperText Transfer Protocol*) se pueden aprovechar de los sistemas de seguridad mediante cortafuegos sin necesidad de cambiar las reglas de filtrado.

La última cualidad que cabe resaltar es el débil acoplamiento entre la aplicación que se sirve del servicio Web y el mismo. Los cambios que se realicen en uno no afectan al otro mientras que los mensajes que se intercambien sigan siendo los mismos. Esta flexibilidad será más importante cuanto mayor sea la envergadura del proyecto que se trate de construir, ya que la modularidad es una propiedad que siempre se debe de perseguir.

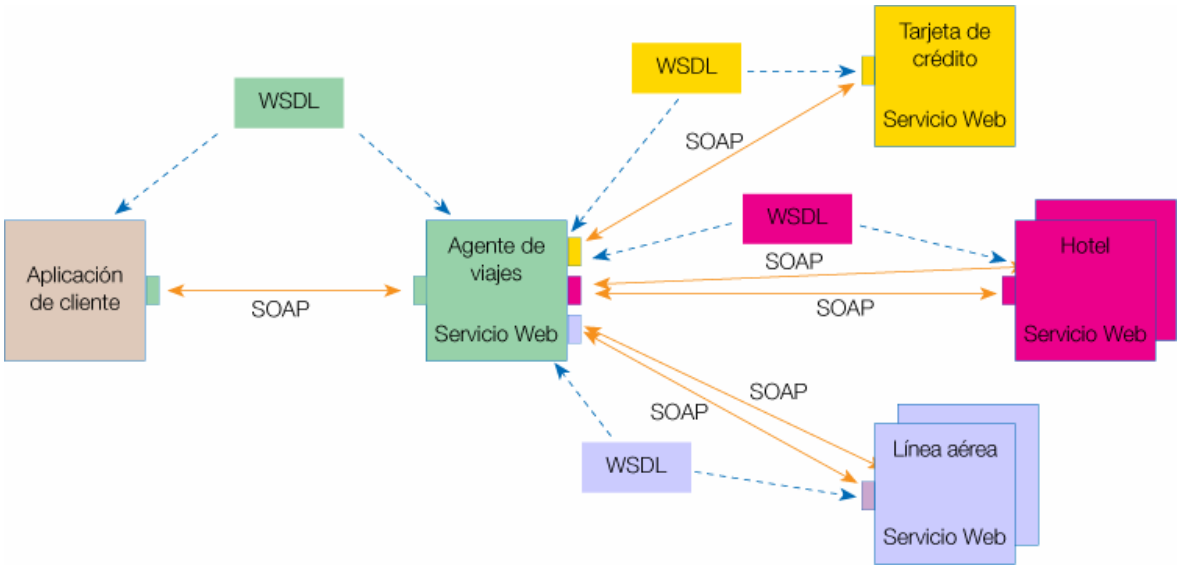

Figura 2.2. Ejemplo de servicio Web

En la figura 2.2 se puede ver un ejemplo de los mensajes que se intercambian en varios servicios Web relacionados con las transacciones llevadas a cabo para reservar un viaje. En todo este proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría SOAP (*Simple Object Access Protocol*). Se trata de un protocolo basado en XML (*eXtended Markup Language*), que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP (*Simple Mail Transfer Protocol*), etc. SOAP especifica el formato de los mensajes. El mensaje SOAP está compuesto por un sobre, cuya estructura está formada por los siguientes elementos: cabecera y cuerpo. Por otro lado, WSDL (*Web Services Description Language*), permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes

y su contenido, a través de un documento procesable por dispositivos. WSDL representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes.

En este proyecto se va aprovechar al máximo una de las cualidades aquí descritas, que es la de aprovecharse de las medidas de seguridad con las que cuenta HTTP. Se conseguirán de esta manera conexiones seguras usando las directrices que el hipertexto permite. Como ya se ha comentado, los servicios Web se transmiten vía HTTP sobre TCP (*Transfer Control Protocol*), y todo ello a través del puerto 80. Dado que las organizaciones (tales como la Universidad en este caso) protegen sus redes mediante cortafuegos (que filtran y bloquean gran parte del tráfico de Internet), cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web usan este puerto, por la simple razón de que no resultan bloqueados.

Los interfaces alternativos a SOAP para acceder a las funcionalidades de otros ordenadores en red no son demasiado buenos. Los que hay son específicos para un determinado sistema y poco conocidos, tales como EDI (*Electronic Data Interchange*), RPC (*Remote Procedure Call*), u otras API (*Application Programming Interface*). Las tecnologías que usan estos interfaces son las llamadas EAI (*Enterprise Application Integration*). Sin embargo, las soluciones de EAI disponibles hoy en día son propietarias y presentan limitaciones en cuanto a interoperabilidad. EAI se caracteriza por ser menos flexible y por tener una implementación más cara que los servicios Web, mediante el uso de estándares y tecnologías. En resumen los servicios Web están indicados para una escala menos específica pero de amplia cobertura ya que permite la integración de sistemas de información tanto dentro de la propia organización como entre distintas organizaciones.

2.4 JCS

JCS (*Java Caching System*) es un sistema de caché en Java que permite manejar de manera sencilla el almacenamiento en caché de datos en servidores de aplicaciones, gestionando sus tiempos de refresco y mejorando sustancialmente el rendimiento global. Es parte del proyecto Apache Jakarta y posee numerosas cualidades como manejo de memoria, mínimas dependencias, es totalmente configurable mediante parámetros sencillos, se puede hacer recuperación remota…

En este proyecto se usa la caché de forma inversa a como se usa habitualmente. Es decir, lo tradicional en una aplicación en que se debe consultar frecuentemente en memoria

es, primero consultar la caché, que es más pequeña y más rápida que la memoria convencional, y en caso de no encontrar lo que se desea, se busca en la memoria principal. En este caso se usa la caché JCS como sistema de emergencia, esto es, primero se intenta acceder al servidor (en el cual reside la base de datos), y si no está disponible se busca en la caché (local) que guarda una copia en memoria principal y otra en el disco duro del ordenador local.

De esta forma es como se puede crear una base de datos local de una forma mucho más sencilla que creando una base de datos propiamente dicha en cualquier lenguaje, manteniendo además mayor homogeneidad en la aplicación software, ya que se sigue programando en Java.

2.5 PUERTO SERIE

El puerto serie RS-232C, presente en casi todos los ordenadores de sobremesa actuales, es una forma comúnmente utilizada para transmitir datos entre ordenadores o para comunicación de este con otros periféricos. El RS-232C es un estándar que constituye la tercera revisión de la antigua norma RS-232, propuesta por la EIA (Asociación de Industrias Electrónicas), realizándose posteriormente un versión internacional por el CCITT (*Consultative Committee for International Telephony and Telegraphy*), conocida como V.24. Las diferencias entre ambas son mínimas, por lo que a veces se habla indistintamente de V.24 y de RS-232C (incluso sin el sufijo "C"), refiriéndose siempre al mismo estándar.

Número de pin	**Señal**
1	DCD (*Data Carrier Detect*)
2	RX (*Recepción*)
3	TX (*Transmisión*)
4	DTR (*Data Terminal Ready*)
5	GND (*Ground*)
6	DSR (*Data Sheet Ready*)
7	RTS (*Request To Send*)
8	CTS (*Clear To Send*)
9	RI (*Ring Indicator*)

Tabla 2.1. Pines del puerto serie y su significado

Dicho estándar contempla dos tipos de conectores, de 25 y 9 pines respectivamente (este último es el que se ha usado en el proyecto). En ambos casos se trabaja con señales eléctricas de +12 V (cero lógico) y -12 V (uno lógico) y dependiendo de la velocidad de transmisión de los datos se pueden usar cables de hasta 15m. Cada pin puede ser una entrada o una salida, teniendo una función específica cada uno de ellos. La información asociada a cada uno de los pines en el caso del conector de 9 pines es la que se muestra en la tabla 2.1.

Las señales TX, DTR y RTS son de salida, mientras que RX, DSR, CTS y DCD son de entrada. La masa de referencia para todas las señales es GND (señal de tierra).

Todas estas señales se corresponden con unos ciertos pines tanto en el conector de 9 pines como en el de 25 (en el que muchos pines están desaprovechados, ya que el de 9 pines realiza lo mismo que el de 25). En el caso de este proyecto sólo es necesario saber que se puede obtener una señal de +12 V manejable a través de software por un cierto programa. Los conectores que se usan en el interfaz implementado son los de 9 pines, numerados tal y como aparece en la figura 2.3.

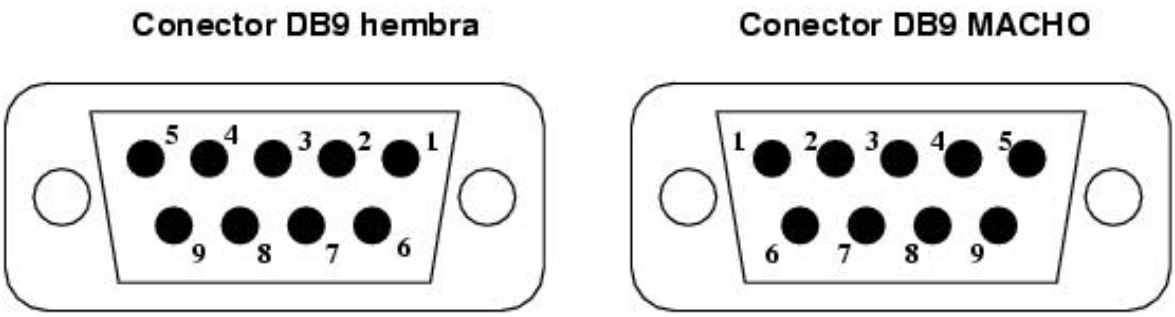

Figura 2.3. Conectores del puerto serie de 9 pines

Igualmente se podía haber utilizado el puerto paralelo, pero dado que es un puerto del que sólo disponemos uno y que tiene muchos más pines, sería, por una parte perder una forma de comunicación con el ordenador y por otra parte, desaprovechar más pines. Es por esta última razón que en muchas aplicaciones de domótica se acostumbra a utilizar el puerto paralelo, por su capacidad para manejar muchas salidas para accionar determinados elementos.

Otra posibilidad hubiera sido usar un puerto USB (*Universal Serie Bus*), pero nuevamente estaríamos desaprovechando un puerto que seguramente es más útil en otros menesteres.

2.6 SERVLETS

Se podría definir un *servlet* como un programa escrito en Java que se ejecuta en el marco de un servicio de red (un servidor HTTP, por ejemplo), y que recibe y responde a las peticiones de uno o más clientes [García de Jalón'99]. El uso más común de los *servlet* es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Cabe destacar como principales características la portabilidad (siguiendo el famoso lema en inglés "*write once, run anywhere*") y la seguridad, ya que son dos aspectos característicos del lenguaje Java, y un *servlet* no es más que una clase Java como cualquier otra. Otra característica de los *servlet* es que una vez que se llaman por primera vez quedan residentes en memoria listos para la próxima vez que se soliciten hasta que el programa que controla el servidor los desactive, minimizando así el tiempo de respuesta.

Sin embargo, la mayor parte del proyecto se ha realizado con la tecnología que se describe en el apartado 2.7, JSP (*Java Server Pages*). Entonces, ¿por qué son necesarios los *servlet* en este proyecto? La razón es muy sencilla, no se puede hacer cualquier cosa con JSP. Las JSP son muy buenas para trabajar con HTML (*Hypertext Markup Language*) o TXT (*TeXT*), pero no así con otros formatos como PDF, que son necesarios en esta aplicación. Sí que se puede trabajar con estos formatos con los *servlet*, por lo que la elección es obvia.

2.7 JSP

JSP es una tecnología orientada a crear páginas Web con tecnología Java. Sólo con decir esto ya se hacen evidentes dos características básicas. Por una parte es multiplataforma, y por tanto se pueden crear aplicaciones que se ejecuten en diferentes servidores Web que funcionen en diferentes sistemas operativos, ya que las JSP se basan en tecnología Java. Por otra parte, siguen siendo páginas Web, por lo que, aunque están mezcladas con etiquetas especiales para programar scripts escritos en Java, siguen estando compuestas por código HTML/XML. De esta forma, se pueden componer en cualquier editor HTML/XML o, simplemente, en un editor de texto como el bloc de notas en Windows.

Internamente las JPS son *servlet*, el servidor se encarga de compilar la página JSP y a partir de ese momento son un *servlet* como cualquier otro. Para entender el proceso por el que se transforman en *servlet* se puede ver un esquema como el de la figura 2.4.

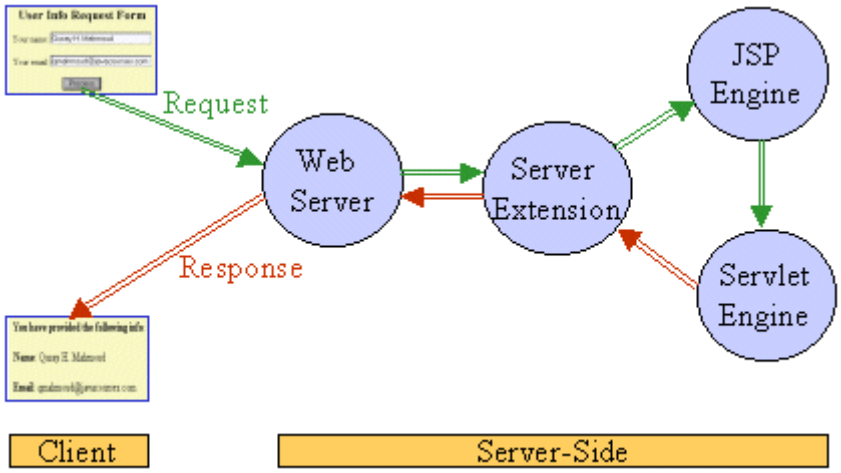

Figura 2.4. Flujo petición/respuesta cuando se llama a una página JSP

Esta transformación en *servlet* ocurre la primera vez que se accede a la página. La próxima vez, el motor *servlet* se encarga de dar la respuesta con el *servlet* ya compilado, a menos que la página haya cambiado, en cuyo caso se recompila y se vuelve a poner a disposición del servidor.

La primera tecnología que se puede ocurrir con propósitos parecidos a las JSP es CGI (*Common Gateway Interface*), que constituyen el sistema más antiguo que existe para la programación de las páginas dinámicas de servidor. Actualmente se encuentra un poco desfasado por diversas razones entre las que destaca la dificultad con la que se desarrollan los programas y la pesada carga que supone para el servidor que los ejecuta, ya que habitualmente es interpretado en el servidor en PERL (*Practical Extraction and Reporting Language*), o ya compilado en C.

Otra posibilidad es usar *servlet*, que sirven para el mismo propósito, y de hecho se usan en este proyecto para un objetivo concreto, pero que también carecen de la división entre la presentación y el código en sí, cosa que en las JSP se soluciona con el uso de JavaBeans. Las JavaBeans son clases de Java de las que hacen uso las JSP para establecer, modificar, y usar sus propiedades, y en función de ellas mostrar lo que interese por pantalla.

La tecnología PHP (*Hypertext Preprocessor*) habría sido otra buena opción a considerar. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Al igual que JSP se escribe dentro del código HTML, por lo que comparte en este sentido también sus ventajas. Además es compatible con las bases de datos más comunes, como MySQL, Oracle, y ODBC (*Open DataBase Connectivity*), por ejemplo. Así, la única razón de usar

JSP frente a PHP es por homogeneidad del proyecto, que emplea Java como base, ya que básicamente son iguales.

El último contendiente de JSP son las ASP.NET (*Active Server Pages* con tecnología *.NET*), que tienen dos diferencias fundamentales respecto a las JSP:

1. Las páginas ASP.NET están escritas en cualquier lenguaje soportado por el marco de trabajo .NET de Microsoft como *Visual Basic.NET*, C# o *JScript.NET*, mientras que las JSP están en lenguaje Java. Por lo tanto, las páginas JSP son independientes de la plataforma y las páginas ASP.NET no lo son.
2. Las páginas JSP usan tecnología JavaBeans como arquitectura de componentes y las páginas ASP.NET usan componentes .NET.

Como diferencias más importantes a la hora de elegir JSP frente a ASP.NET tenemos las siguientes:

1. Velocidad y Escalabilidad: Aunque las páginas ASP.NET son cacheadas, siempre son interpretadas, en cambio las páginas JSP son compiladas en *servlet* Java y cargadas en memoria la primera vez que se las llama, y son ejecutadas para todas las llamadas siguientes. Esto le da a las páginas JSP la ventaja de la velocidad y escalabilidad sobre las páginas ASP.NET.
2. Etiquetas Extensibles: Las páginas JSP tienen una característica avanzada conocida como etiquetas extensibles. Este mecanismo permite a los desarrolladores crear etiquetas personalizadas. En otras palabras, las etiquetas extensibles permiten extender la sintaxis de las etiquetas de las páginas JSP. Esto no tiene su equivalente en ASP.NET.
3. Libertad de Elección: Las páginas ASP.NET sólo funcionan con Microsoft IIS. El uso de páginas ASP.NET requiere un compromiso con los productos de Microsoft, mientras que las páginas JSP no imponen ningún servidor Web ni sistema operativo. Esta es quizá la razón más importante para interesarse por las JSP en detrimento de las ASP.NET.

2.8 HTML

2.8.1 HTML A GRANDES RASGOS

HTML es, en una primera definición, un lenguaje de marcado diseñado para estructurar datos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

Más ampliamente, el HTML no es más que una aplicación del SGML (*Standard Generalized Markup Language*), un sistema para definir tipos de documentos estructurados y lenguajes de marcas para representar esos mismos documentos. El término HTML se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de marcas.

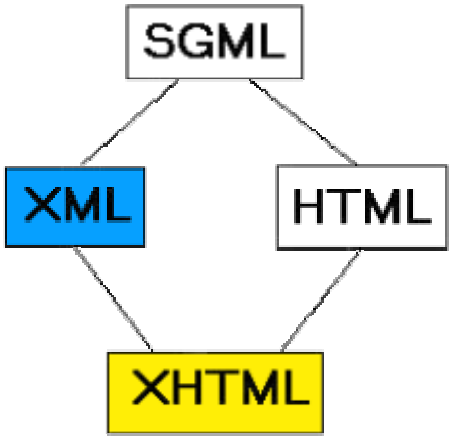

Figura 2.5. Relación de los lenguajes de marcas

Los lenguajes de marcas, también denominados lenguajes de marcado o lenguajes de descripción de documentos, construyen un conjunto de reglas que definen todo aquello que es parte de un documento digital, pero que no pertenece al texto del mismo. En la figura 2.5 se puede ver la relación entre los lenguajes de marcas. En realidad, más que de lenguajes, se podría hablar de metalenguajes o sistemas formales mediante los cuales se añade información o codificación a la forma digital de un documento bien para controlar su procesamiento, bien para representar su significado.

HTML no permite definir de forma estricta la apariencia de una página, aunque una utilización algo desviada hace que se utilice en ocasiones como un lenguaje de presentación. Además, la presentación de la página depende mucho del navegador que se use; un mismo documento no produce el mismo resultado en pantalla si se visualiza con un navegador en modo texto, Mosaic o Netscape, es decir, HTML se limita a describir la estructura y el contenido de un documento, y no el formato de la página y su apariencia.

Una de las claves del éxito de WWW, aparte de lo atractivo de su presentación es sin duda, su organización y coherencia. Todos los documentos WWW comparten un mismo aspecto y una única interfaz, lo que facilita enormemente su manejo por parte de cualquier persona. Esto es posible porque el lenguaje HTML, en que están escritos los documentos, no solo permite establecer hiperenlaces entre diferentes documentos, sino que es un "lenguaje de descripción de página" independiente de la plataforma en que se utilice. Es

decir un documento HTML contiene toda la información necesaria sobre su aspecto y su interacción con el usuario, y es luego el navegador que utilicemos el responsable de asegurar que el documento tenga un aspecto coherente, independientemente del tipo de estación de trabajo desde donde estemos efectuando la consulta.

Se pueden crear documentos HTML con cualquier editor de texto, tan simples como el EDIT de MS-DOS o el bloc de notas de Windows, aunque existen otros editores más elaborados que son de buena ayuda a la hora de evitar y depurar errores. Las marcas o *tags* que controlan el comportamiento del documento son fragmentos de texto encerrados entre los signos "mayor que" y "menor que" (<marca>). Existen diferentes tipos de marcas: algunas controlan simplemente la presentación del texto del documento; otras, la forma en que se incluirán en él imágenes; otras, finalmente, los hiperenlaces con documentos o con diferentes partes del mismo documento.

Como ya se ha dicho este lenguaje es el que permite la generación de hipertextos en la World Wide Web de una forma sencilla y eficaz. Por lo tanto es evidente que su uso es obligado a la hora de construir una aplicación accesible con un navegador Web.

2.8.2 DHTML

El DHTML (*Dynamic HTML*) o HTML dinámico no es un estándar definido por el W3C, sino que es un término de marketing que utilizan Netscape y Microsoft para referirse al conjunto de nuevas tecnologías de Web. Dicho conjunto comprende:

- HTML, especialmente HTML 4.0
- Hojas de estilo (CSS)
- JavaScript

Se suele calificar como DHTML a este conjunto de tecnologías, especialmente en aquellos casos en que operan conjuntamente para enriquecer la experiencia web del usuario. Esta conjunción de tecnologías permite, entre otras cosas, ofrecer al usuario interfaces gráficas mucho más ricas y complejas, controlar formularios de forma más eficiente (el código JavaScript se ejecuta en el cliente, con el consiguiente incremento de rendimiento), etc.

Uno de los puntos claves de DHTML es DOM (*Document Object Model*), que define una jerarquía de objetos accesibles mediante JavaScript (un árbol de hecho) que representan todos y cada uno de los elementos del documento HTML. El árbol usado en DOM se muestra en la figura 2.6.

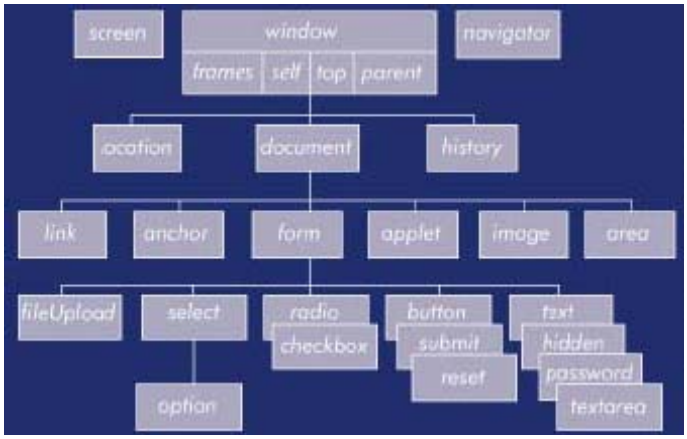

Figura 2.6. DOM del documento HTML

En este caso, y según las características explicadas de DHTML, en este proyecto se ha dado ese paso de más para conseguir una aplicación más amigable.

2.9 XML

El lenguaje extensible de marcas, abreviado XML, describe una clase de objetos de datos llamados documentos XML y parcialmente describe el comportamiento de programas de computador que pueden procesarlos. XML es un perfil de aplicación o forma restringida de SGML [ISO8879'86], al igual que HTML (tal y como se veía en el punto anterior).

La diferencia fundamental del XML con el lenguaje HTML es que, aún usando etiquetas como HTML, las etiquetas XML describen el contenido en vez de presentarlo. Puesto que se trata de describir los datos y éstos pueden ser de muy diversa índole, el lenguaje XML permite crear etiquetas propias así como los atributos de las mismas. La única restricción a la que se debe atender es un conjunto de normas de sintaxis que están hechas con el fin de garantizar la consistencia de los datos representados. En teoría HTML es un subconjunto de XML especializado en presentación de documentos para la Web, mientras que XML es un subconjunto de SGML especializado en la gestión de información para la Web. En la práctica XML contiene a HTML aunque no en su totalidad. La definición de HTML contenido totalmente dentro de XML y por lo tanto que cumple a rajatabla la especificación SGML es XHTML (*eXtensible, HyperText Markup Language*).

Por todo lo dicho es obvio que con tener un documento XML no es suficiente para conseguir una salida por pantalla adecuada, y para ello se usan las hojas de estilo, de las

que se hablará más profusamente en el punto 2.11 de este capítulo. Como anticipo cabe decir que las hojas de estilo sirven para dar formato a la información, es la forma de poder visualizar el contenido de los documentos XML en distintos formatos. A un mismo documento XML se le pueden aplicar las hojas de estilo XSL (*eXtensible Stylesheet Language*) que queramos e incluso podemos transformar un documento XML en otro distinto mediante hojas XSLT (*XSL Transformations*). También se pueden aplicar filtros a los datos para visualizar únicamente los que nos interesen.

Estas últimas características serán las que más interesen de cara a este proyecto, ya que para obtener informes en PDF es necesario partir de un documento que será el que contenga unos ciertos datos y pasarles una hoja de estilo que los convierta en un documento PDF. Este documento PDF será lo que muestre el navegador directamente.

2.10 JAVASCRIPT

JavaScript es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML. Al ser la más sencilla, es por el momento la más extendida. Es conveniente tener claras dos cosas sobre este lenguaje:

1. JavaScript no es un lenguaje de programación propiamente dicho. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto. No está orientado a hacer un programa con JavaScript, sino tan sólo mejorar una página Web con algunas acciones sencillas (revisión de formularios, efectos en la barra de estado...) y, ahora, no tan sencillas (animaciones usando HTML dinámico, por ejemplo).
2. JavaScript y Java son dos cosas distintas. Principalmente porque Java sí que es un lenguaje de programación completo. Lo único que comparten es la misma sintaxis, debido a que cuando Netscape lo diseñó se inspiró en la sintaxis de Java.

Es un lenguaje interpretado, lo que quiere decir que no se compila una vez y se usa la versión compilada siempre, sino que las instrucciones en JavaScript se compilan en tiempo de ejecución.

De cara al proyecto, su uso ha estado restringido al campo de los formularios, ya que las piezas fundamentales de la aplicación Web desarrollada son justamente ellos. Se usan pues, para comprobar la validez de los datos introducidos, que no se dejan campos en blanco, que no llegarán datos al servidor que puedan generar errores…

Por último, es de resaltar que son características ampliamente reconocidas de JavaScript el hecho de ser un lenguaje flexible, versátil, potente de cara a la Web, y al mismo tiempo sencillo. Por todo esto el uso de JavaScript para mejorar el rendimiento de la aplicación y asegurar su corrección se hace absolutamente necesario.

Frente a otros lenguajes que pudieran valer para realizar las mismas funciones, como VBScript, se impone la principal premisa que se ha seguido en todo el proyecto, que es la gratuidad de las herramientas utilizadas. VBScript es software propietario de Microsoft y por tanto de pago, por lo que la elección de JavaScript es la mejor.

2.11 HOJAS DE ESTILO CSS Y XSL

2.11.1 INTRODUCCIÓN

Las hojas de estilo son conjuntos de instrucciones, a veces en forma de archivo anexo, que se asocian a los archivos de texto y se ocupan de los aspectos de formato y de presentación de los contenidos: tipo, fuente y tamaño de letras, justificación del texto, colores y fondos, etc. Las hojas de estilo permiten liberar la composición del texto de los aspectos visuales y favorecen que se estructure y anote mediante códigos que permiten un tratamiento más eficaz de los contenidos. Eso se consigue asociando atributos de presentación a cada una de las etiquetas de HTML o a subclases de éstas.

2.11.2 HOJAS DE ESTILO CSS

Las hojas de estilo en cascada o CSS (*Cascade StyleSheet*) son, en esencia, documentos en los cuales se define, de forma separada a la página a la que estén ligados, las características con las que ésta se va a mostrar; características tales como el tamaño de la fuente, el tipo de letra, el color, adornos en las letras… Otra forma de crear las hojas de estilo es incrustándolas en el propio documento al cual queremos dotar de un estilo, pero esta es la forma menos recomendable, ya que así se pierde una gran ventaja que es la separación entre presentación y contenido.

Son documentos tan sencillos que no tienen ningún editor especializado en este lenguaje, ya que sólo definen unos atributos para un determinado estilo. En una misma hoja CSS se pueden definir tantos estilos como se quiera para, una vez ligada con cierta página, poder usarlos libremente desde ella tan sólo usando el nombre del estilo. En la figura 2.7 se puede ver un fragmento de la hoja de estilos usada en el proyecto donde se

detallan dos de los estilos empleados. Es evidente por tanto, que con esta separación de presentación y contenido es muy fácil cambiar completamente el aspecto de una página Web con unos pocos cambios en su hoja de estilos. Justamente esta cualidad es la que posibilita ventajas como las siguientes:

- Fácil mantenimiento y actualización de la presentación de la página.
- Posibilidad de tener varias hojas de estilo para una misma página, según las necesidades del usuario que vaya a verla (en un dispositivo móvil, para impresión…).
- Claridad en el documento HTML.
- Accesibilidad. Usuarios con deficiencias visuales, por ejemplo pueden usar fuentes o enlaces más grandes.

Es por ello que su uso es obligado en este proyecto si se pretende hacer una página medianamente vistosa, y con posibilidades de realizar un mantenimiento adecuado en ella si fuera necesario.

```
.TituloPrincipal {
        font-family: Georgia, "Times New Roman",
Times, serif;
        font-size: xx-large;
        font-weight: bold;
        color: #009900;
}
.LeyendaDelPrincipal {
        font-family: "Times New Roman", Times, serif;
        font-size: large;
        font-style: italic;
        color: #000000;
}
```

Figura 2.7. Fragmento de la hoja de estilos aplicada en las páginas Web del proyecto

2.11.3 HOJAS DE ESTILO XSL

2.11.3.1 Introducción

Además de las CSS existen las hojas de estilo XSL, de las que ya se hizo mención en el punto 2.9 de este capítulo y que ofrecen muchas más posibilidades que las hojas de estilo en cascada. Si bien las hojas XSL son documentos XML y ese sentido podrían haberse

explicado en el punto 2.9 de este capítulo, aquí se ha marcado una diferenciación para compararlas con las CSS. En general, para documentos en el que los datos tengan una estructura sencilla y regular, puede que las hojas de estilo CSS sean más que suficientes. El lenguaje XSL es algo más complejo, pero pone a nuestra disposición toda la flexibilidad y potencia de un verdadero mecanismo de consulta de datos. Consiste en tres componentes principales: XSLT, XPath (*XML Path language*) y XSL-FO (XSL-*Formatting Objects*).

2.11.3.2 XSLT

XSLT es la parte del estándar XML que sirve para transformar documentos XML en otros documentos XML, como por ejemplo XHTML.

Normalmente XSLT lo hace transformando cada elemento XML en otro elemento XML. XSLT también puede añadir otros elementos XML a la salida, o bien puede eliminar elementos. Asimismo puede reordenar o recolocar elementos y hacer comprobaciones y decisiones sobre qué elementos mostrar.

Durante la transformación, XSLT utiliza XPath para especificar o referenciar a partes del documento que cumplen uno o más patrones definidos. Cuando encuentra una coincidencia de patrones, XSLT transformará la parte coincidente del documento origen en el documento destino. Las partes no coincidentes no se transforman, sino que quedan en el documento destino sin ningún cambio.

2.11.3.3 XPath

XPath es un lenguaje con una sintaxis fija que permite seleccionar subconjuntos de un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto sin atributos. XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML.

2.11.3.4 XSL-FO

Un documento XSL-FO es un documento XML en el que se especifica cómo se van a formatear unos datos para presentarlos en pantalla, papel u otros medios. Hay que destacar que en el documento XSL-FO figuran tanto los datos como el formato que se les va a aplicar.

La unidad básica de trabajo en un documento XSL-FO es el "*Formatting Object*", unidad básica para presentar (formatear) la información. Estos objetos de formato se refieren a páginas, párrafos, tablas, etc. En la figura 2.8 se muestra un breve ejemplo de documento XSL-FO donde se detallan todos los márgenes de la página (superior, inferior, derecho e izquierdo), las dimensiones de la misma (con lo cual ya se puede saber por ejemplo si se va a seguir un formato apaisado), y un bloque de texto simple:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="hola"
                  page-height="29.7cm"
                  page-width="21cm"
                  margin-top="5mm"
                  margin-bottom="10mm"
                  margin-left="20mm"
                  margin-right="20mm">
      <fo:region-body margin-top="10mm" margin-bottom="10mm" />
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="hola">
    <fo:flow>
      <fo:block>Hola, mundo</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Figura 2.8. Ejemplo de documento XSL-FO

Para obtener el documento XSL-FO pueden seguirse dos vías:

1. Generarlo directamente a partir de los datos. El documento XSL-FO contiene las especificaciones de formato y los propios datos.
2. Transformar un documento XML que contenga los datos a presentar con una hoja de estilos XSLT. De esta forma los datos (XML) se independizan del formato que proporcionará la hoja de transformación XSLT. Esta transformación se puede ver gráficamente en la figura 2.9.

Cuando se tiene el documento XSL-FO, puede ser procesado por un programa llamado "procesador de XSL-FO" para obtener el documento final en distintos formatos. El formato final más utilizado es el PDF.

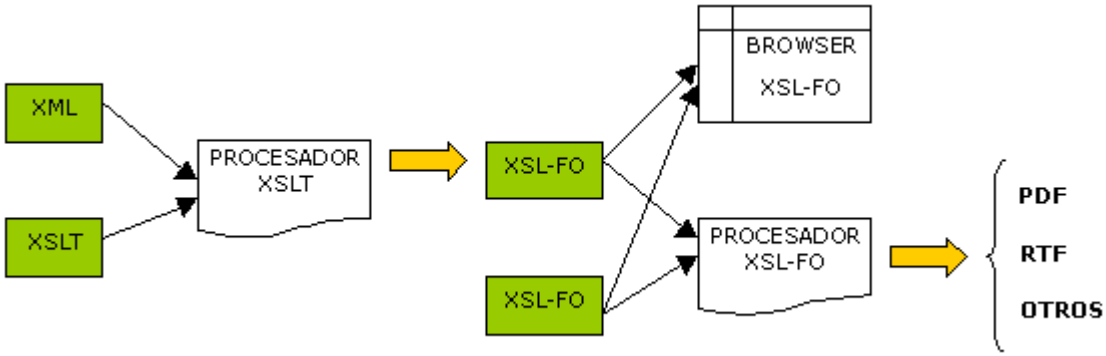

Figura 2.9. Formateo de los datos hasta el documento final

2.12 FOP

El FOP (*Formatting Object to* PDF) es el primer procesador de objetos de formateo XSL que apareció. Empezó a ser desarrollada en solitario por James Tauber pero posteriormente se incorporo al "*Apache XML Project*" lo que está acelerando su desarrollo. Entre sus características cabe resaltar que está desarrollado en Java, es totalmente gratuito, y es, sin duda alguna, el más utilizado en la actualidad.

Otras alternativas a este procesador son las siguientes:

- XEP, desarrollado en Java por RenderX. Es sin duda alguna el más avanzado. El único inconveniente es que se trata de una aplicación comercial aunque es posible bajar de Internet una versión de demostración evidentemente limitada en sus funcionalidades.
- PassiveTex. Es una librería de macros que pueden ser usadas para procesar documentos XML formados por XSL-FO. Ejecutando PassiveTex con pdfTex se pueden generar fácilmente documentos PDF. Lo mejor de estas librerías es el buen soporte para MathML (*Mathematical Markup Language*) que poseen.
- XSL Formatter, de Antenna House Inc. Consiste en un procesador de XSL-FO acompañado de un interfaz de usuario. Necesita la MSXML3.0 (*Microsoft XML parser*) como procesador de XSLT y funciona sólo sobre plataforma Windows.
- *Unicorn Formatting Objects* (UFO), que es un procesador de XSL-FO implementado en C++. La salida de esta herramienta es TeX y a partir de este formato podemos generar PostScript, PDF, etc.
- iText. Es una librería que permite generar PDF al vuelo. También es una herramienta gratuita y muy usada en combinación con *servlet* basados en

tecnología Java, por lo que no es una mala opción frente a la que se ha usado en este proyecto (FOP).

- Gnujpdf. Es un paquete de Java bajo la licencia GPL (*General Public License*). Provee un API para crear archivos PDF e imprimirlos usando subclases de java.awt.Graphics y java.awt.PrintJob. Las clases PDF escriben en un flujo de salida (OutputStream) en formato PDF en lugar de los objetos de gráficos, pero los métodos a los que se llaman son los mismos que si se tratara de una aplicación Java ejecutándose en un navegador, por ejemplo. Tampoco es una mala opción frente a FOP.

Aún existen algunos más, pero estos son los más importantes. En cualquier caso, su uso en este proyecto se basa principalmente en tres pilares: es gratuito, desarrollado en Java y es el más usado, lo que implica que está muy probado.

2.13 PDF

Es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems. Está especialmente ideado para documentos susceptibles de ser imprimidos, ya que especifica toda la información necesaria para la presentación final del documento, determinando todos los detalles de cómo va a quedar, no requiriéndose procesos ulteriores de ajuste ni de maquetación. Cada vez se utiliza más también como especificación de visualización, gracias a la gran calidad de las fuentes utilizadas y a las facilidades que ofrece para el manejo del documento, como búsquedas, hiperenlaces, etc. Como características hay que resaltar:

- Es multiplataforma, es decir, puede ser presentado por los principales sistemas operativos (Windows, Unix o Mac), sin que se modifiquen ni el aspecto ni la estructura del documento original.
- Puede integrar cualquier combinación de texto, gráficos, imágenes e incluso música.
- Es uno de los formatos más extendidos en Internet para el intercambio de documentos. Por ello es muy utilizado por empresas, gobiernos e instituciones educativas.
- Es una especificación abierta, para la que se han generado herramientas de Software Libre que permiten crear, visualizar o modificar documentos en formato PDF. Un ejemplo es la suite ofimática OpenOffice.org.

Por todo ello es evidente que es el formato idóneo para, entre otras cosas, almacenar informes, que es justamente su uso en el marco de este proyecto.

2.14 SQL

El SQL (*Structured Query Language*) o Lenguaje de Consulta Estructurado es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

Como características generales hay que decir que SQL explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro.

Es el lenguaje más habitual para construir las consultas a las bases de datos relacionales y se ha convertido en un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

2.15 BASE DE DATOS MySQL

Una base de datos no es más que un conjunto de datos de varios tipos, organizados e interrelacionados. Estos datos deben estar libres de redundancias innecesarias y ser independientes de los programas que los usan. Para manejar bases de datos se usan los SGBD (*Sistemas Gestores de Bases de Datos*) o en inglés DBMS (*DataBase Manager System*). En este caso, MySQL es un SGBD. Usar estos SGBD aporta varias ventajas:

1. Acceso a las bases de datos de forma simultánea por varios usuarios y/o aplicaciones.
2. Seguridad, en forma de permisos y privilegios; determinados usuarios tendrán permiso para consulta o modificación de determinadas tablas. Esto permite

compartir datos sin que peligre la integridad de la base de datos o protegiendo determinados contenidos.

3. Potencia: SQL es un lenguaje muy potente para consulta de bases de datos, usar un motor nos ahorra una enorme cantidad de trabajo.
4. Portabilidad: SQL es también un lenguaje estandarizado, de modo que las consultas hechas usando SQL son fácilmente portables a otros sistemas y plataformas. Esto, unido al uso de C/C++ proporciona una portabilidad enorme.

En concreto, usar MySQL tiene ventajas adicionales:

1. Escalabilidad: es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.
2. MySQL está escrito en C y C++ y probado con multitud de compiladores y dispone de API para muchas plataformas diferentes.
3. Conectividad: es decir, permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.
4. Es multihilo, con lo que puede beneficiarse de sistemas multiprocesador.
5. Permite manejar multitud de tipos para columnas.
6. Permite manejar registros de longitud fija o variable.

Todo ello unido al hecho de ser software gratuito con licencia GPL, ampliamente extendido, y por tanto probado, lo hacen el candidato ideal a la hora de elegir un SGBD.

2.16 JDBC

JDBC es el estándar que proporciona un mecanismo para el acceso a bases de datos relacionales desde aplicaciones desarrolladas en Java. Como curiosidad cabe señalar que Sun niega que JBDC sea un acrónimo. De esta forma, y de acuerdo a la versión oficial, no sería propio hablar de Java DataBase Connectivity, aunque todo el mundo lo entienda así.

En realidad, la idea de JDBC no es nueva, y está basada en el modelo ODBC. La necesidad de trabajar con un nivel de abstracción superior, que proporcione una interfaz común para acceder a bases de datos es innegable. Aunque todos los gestores de bases de datos relacionales, como Oracle, SQL Server, Sybase, Informix, etc. trabajen con el

lenguaje de consultas SQL, cada solución comercial ha ido introduciendo desviaciones en la nomenclatura y definición de los tipos de datos, así como en la propia sintaxis de SQL. A modo de ejemplo, cabe decir que el lenguaje SQL de algunas bases de datos considera separador de línea (;), mientras que en otras bases de datos no es necesario. Esta falta de homogeneidad entre los diferentes gestores de base de datos existentes en el mercado puede producir verdaderos quebraderos de cabeza a la hora de migrar aplicaciones de una base de datos a otra. Es más, en un mismo entorno empresarial, durante periodos de transición, se puede trabajar simultáneamente con más de un gestor de base de datos. En este escenario, habría que rescribir completamente el código de la aplicación encargado del acceso a datos en función de la base de datos empleada.

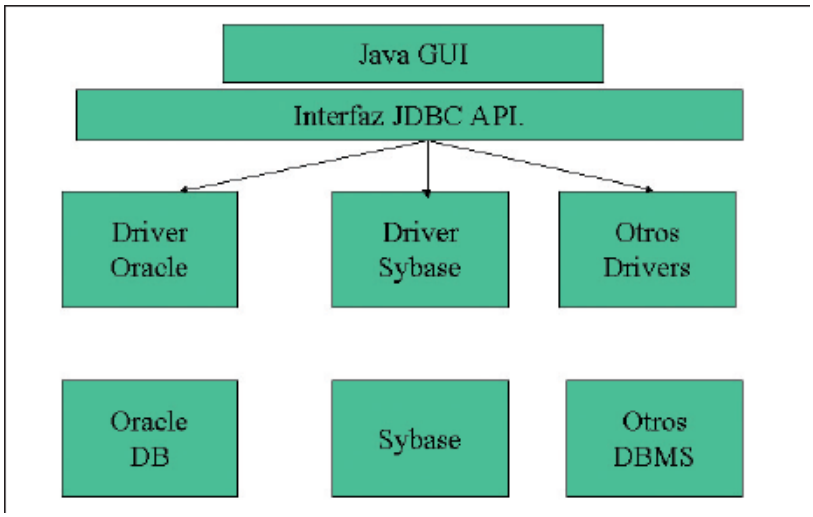

Figura 2.10. Diagrama de arquitectura de JDBC

Tanto ODBC, que fue el primero en aparecer, como JDBC solventan el problema de la falta de homogeneidad entre los diferentes vendedores de bases de datos ofreciendo un interfaz común de programación. ODBC, sin embargo, sólo permite trabajar con bases de datos específicas para el sistema operativo Windows, mientras que JDBC, al formar parte de la órbita de tecnologías de la plataforma Java, no adolece de esta limitación.

La arquitectura del paradigma de JDBC se muestra en la figura 2.10, en la que se ve la versatilidad de la tecnología JDBC y cómo a través de un mismo interfaz se puede relacionar con diferentes tipos de gestores de bases de datos.

Gracias a estas características JDBC se convierte en el estándar perfecto a usar en este proyecto para asegurar una buena conectividad a la base de datos MySQL.

2.17 SERVIDOR DE APLICACIONES TOMCAT

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de *servlet* desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los *servlet* y de JavaServer Pages (JSP) de Sun Microsystems. Es un servidor web con soporte de *servlet* y JSP e incluye el compilador Jasper, que compila JSP (que son justamente las piezas fundamentales de la Web de este proyecto) convirtiéndolas en *servlet*. El motor de *servlet* de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios, existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción, y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual. Además, se da la circunstancia de que lo desarrollan y mantienen miembros de la *Apache Software Foundation* y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la *Apache Software License*. Estas características son sin duda la mejor baza para formar parte del presente proyecto.

2.18 SEGURIDAD Y AUTENTICACIÓN

Para poder afirmar que una comunicación entre dos entidades es segura se deben cumplir cuatro requisitos principales:

1. Autenticidad: todas las entidades participantes en la transacción deben estar perfecta y debidamente identificadas antes de comenzar la misma. Se debe estar seguro de que la persona con la que nos comunicamos es realmente quién dice ser, ya que si no se puede estar facilitando datos íntimos y/o sensibles a una persona o entidad no deseada, que puede hacer con ellos luego lo que le venga en gana.
2. Confidencialidad: se debe estar seguro de que los datos que enviamos no pueden ser leídos por otra persona distinta del destinatario final deseado, o que si ocurre esto, el espía no pueda conocer el mensaje enviado. O en su defecto, que cuando

consiga obtener los datos, éstos ya no le sirvan para nada. Es decir, se debe estar seguro de que ninguna persona ajena a la transacción puede tener acceso a los datos de la misma.

3. Integridad: es necesario estar seguro de que los datos que enviamos llegan íntegros, sin modificaciones, a su destino final.
4. No repudio: se debe estar seguro de que una vez enviado un mensaje con datos importantes o sensibles el destinatario de los mismos no pueda negar el haberlos recibido. Y en una compra on-line debe garantizarse que una vez finalizada la misma ninguna de las partes que intervienen pueda negar haber participado en ella.

En el caso de este proyecto, las características que más interesan son las dos primeras.

La solución al problema de la autenticación la trajo la aparición de los Certificados Digitales o Certificados Electrónicos, documentos electrónicos basados en la criptografía de clave pública y en el sistema de firmas digitales. La misión principal de un Certificado Digital es garantizar con toda confianza el vínculo existente entre una persona, entidad o servidor web con una pareja de claves correspondientes a un sistema criptográfico de clave pública.

Un Certificado Digital es un documento electrónico que contiene datos identificativos de una persona o entidad (empresa, servidor web, etc.) y la clave pública de la misma, haciéndose responsable de la autenticidad de los datos que figuran en el certificado otra persona o entidad de confianza, denominada Autoridad Certificadora. Las principales Autoridades Certificadoras actuales son Verisign (filial de RSA Data Security Inc.) y Thawte.

Desde el punto de vista de la finalidad, los certificados electrónicos se dividen en:

1. Certificados SSL (*Secure Socket Layer*) para cliente: usados para identificar y autenticar a clientes ante servidores en comunicaciones mediante el protocolo SSL, y se expiden normalmente a una persona física, bien un particular, bien un empleado de una empresa.
2. Certificados SSL para servidor: usados para identificar a un servidor ante un cliente en comunicaciones mediante el protocolo *Secure Socket Layer*, y se expiden generalmente a nombre de la empresa propietaria del servidor seguro o del servicio que éste va a ofrecer, vinculando también el dominio por el que se

debe acceder al servidor. La presencia de éste certificado es condición imprescindible para establecer comunicaciones seguras SSL.

3. Certificados S/MIME (*Secure Multipurpose Internet Mail Extensions*): usados para servicios de correo electrónico firmado y cifrado, que se expiden generalmente a una persona física. El mensaje lo firma digitalmente el remitente, lo que proporciona Autenticación, Integridad y No Rechazo. También se puede cifrar el mensaje con la llave pública del destinatario, lo que proporciona Confidencialidad al envío.
4. Certificados de firma de objetos: usados para identificar al autor de ficheros o porciones de código en cualquier lenguaje de programación que se deba ejecutar en red (Java, JavaScript, CGI, etc.). Cuando un código de este tipo puede resultar peligroso para el sistema del usuario, el navegador lanza un aviso de alerta, en el que figurará si existe certificado que avale al código, con lo que el usuario puede elegir si confía en el autor, dejando que se ejecute el código, o si por el contrario no confía en él, con lo que el código será rechazado.
5. Certificados para AC (Autoridades Certificadoras): que identifican a las propias Autoridades Certificadoras, y es usado por el software cliente para determinar si pueden confiar en un certificado cualquiera, accediendo al certificado de la AC y comprobando que ésta es de confianza.

Toda persona o entidad que desee obtener un certificado debe pagar una cuota a las Autoridades de Certificación, cuota que irá en función de la clase del certificado y del uso que se le vaya a dar al mismo (ambas están relacionadas). A mayor nivel de comprobación de datos (clase mayor), más costará el certificado. En el caso de este proyecto se harán los certificados de cliente, servidor y AC usando OpenSSL para evitar el tener que pagar a nadie.

Son varios los sistemas y tecnologías que se han desarrollado para intentar implementar autenticidad y confidencialidad en las transacciones electrónicas, siendo sin duda SSL el más conocido y usado en la actualidad. SSL permite la Confidencialidad y la Autenticación en las transacciones por Internet, siendo usado principalmente en aquellas transacciones en la que se intercambian datos sensibles, como números de tarjetas de crédito o contraseñas de acceso a sistemas privados. SSL es una de las formas base para la implementación de soluciones PKI (*Public Key Infraestructure*).

Secure Socket Layer es un sistema de protocolos de carácter general diseñado en 1994 por la empresa Netscape Communications Corporation, y está basado en la aplicación conjunta de Criptografía Simétrica, Criptografía Asimétrica (de clave pública), certificados digitales y firmas digitales para conseguir un canal o medio seguro de comunicación a través de Internet. De los sistemas criptográficos simétricos, motor principal de la encriptación de datos transferidos en la comunicación, se aprovecha la rapidez de operación, mientras que los sistemas asimétricos se usan para el intercambio seguro de las claves simétricas, consiguiendo con ello resolver el problema de la Confidencialidad en la transmisión de datos.

SSL implementa un protocolo de negociación para establecer una comunicación segura a nivel de *socket* (nombre de máquina más puerto), de forma transparente al usuario y a las aplicaciones que lo usan. Por lo tanto, si se habilita SSL el sitio Web implementado en este proyecto será seguro.

Desde el punto de vista de su implementación en los modelos de referencia OSI (*Open Systems Interconnection*) y TCP/IP (*Transfer Control Protocol/Internet Protocol*), SSL se introduce como una especie de nivel o capa adicional, situada entre la capa de Aplicación y la capa de Transporte (figura 2.11), sustituyendo los *socket* del sistema operativo, lo que hace que sea independiente de la aplicación que lo utilice, y se implementa generalmente en el puerto 443.

Figura 2.11. Situación de la capa de seguridad en el modelo OSI

2.19 OPENSSL

El software OpenSSL es un proyecto de software desarrollado por los miembros de la comunidad de código libre. Es un robusto juego de herramientas que ayudan a un sistema a implementar el *Secure Sockets Layer*, así como otros protocolos relacionados con la

seguridad, tales como el TLS (*Transport Layer Security*). También incluye una librería de criptografía. Este paquete de software es importante para cualquiera que esté planeando usar cierto nivel de seguridad en su máquina. Debido a que se trata de software libre, puede descargarse libremente, a diferencia de los paquetes comerciales SSL que requieren la licencia/pago del software.

SSL y TLS son comúnmente usadas para proporcionar servicios de autenticación, encriptación, integridad y no repudio para aplicaciones de red como HTTP, IMAP (*Internet Message Access Protocol*), POP3 (*Post Office Protocol* 3ª edición), STP (*Spanning Tree Protocol*) y LDAP (*Lightweight Directory Access Protocol*). OpenSSL es ampliamente utilizado entre una diversidad de plataformas y sistemas. En particular, muchos routers y otros tipos de equipo de red usan OpenSSL.

Por todo esto parece una herramienta ideal para ayudar a dotar a este proyecto de la seguridad necesaria.

2.20 ECLIPSE

Eclipse es un IDE (*Integrated Development Environment*) multiplataforma libre para crear aplicaciones clientes de cualquier tipo. La primera y más importante aplicación que ha sido realizada con este entorno es el afamado IDE Java llamado JDT (*Java Development Toolkit*) y el compilador incluido en Eclipse, que se usaron para desarrollar el propio Eclipse. Se puede ver una captura de pantalla en la figura 2.12, en la que se puede apreciar su interfaz gráfica.

Eclipse fue creado originalmente por IBM. Ahora lo desarrolla 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.

El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, existe un módulo para dar soporte a C/C++. Existen módulos para añadir un poco de todo, desde Telnet hasta soporte a bases de datos. La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

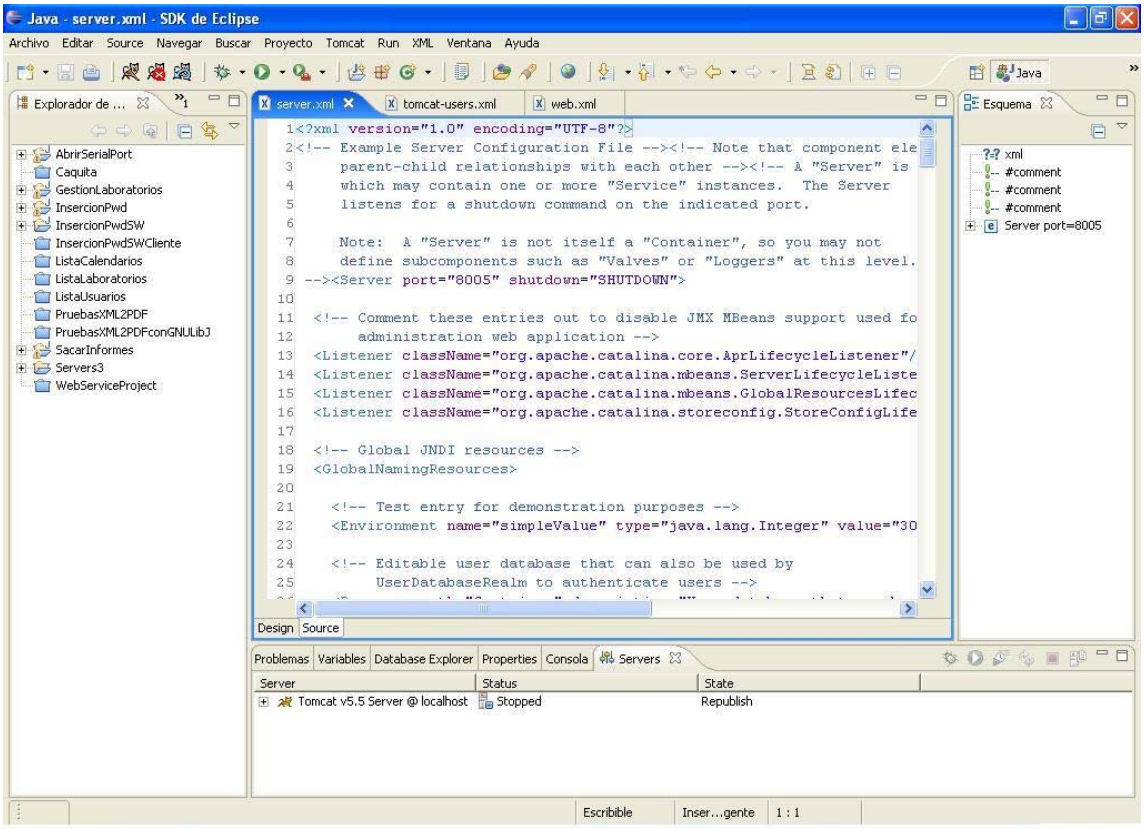

Figura 2.12. Vista del entorno de desarrollo Eclipse

Vistas las cualidades de Eclipse como entorno de desarrollo, y que se adaptan muy bien a las necesidades de este proyecto, queda claro que es una opción muy buena. Otras alternativas parecidas podrían haber sido:

- NetBeans. Es un IDE escrito completamente en Java usando la plataforma NetBeans. La versión actual es NetBeans IDE 5.5, la cual fue lanzada en Noviembre de 2006. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.
- JBuilder, un IDE Java de Borland. La versión 2006 (Borland JBuilder 2006) tiene 3 ediciones: Enterprise (para aplicaciones J2EE o *Java 2 Enterprise Edition*, Web Services y Struts), Developer (para el completo desarrollo de aplicaciones Java) y Foundation (con capacidades básicas para iniciarse en el desarrollo de aplicaciones java y de momento es de libre uso).

La razón de usar Eclipse frente a NetBeans es por la necesidad de decantarse por uno, ya que ambos son libres y, prácticamente, permiten hacer las mismas cosas de una forma parecida en lo que respecta a este proyecto. Respecto al JBuilder, la elección es más sencilla, ya que la versión libre tiene capacidades reducidas respecto a las otras y ya se ha comentado varias veces a lo largo de este documento que el uso de herramientas libres se ha cuidado especialmente.

2.21 RFID

Son las siglas en inglés de identificación por radiofrecuencia y corresponden a un método de almacenamiento y recuperación remota de datos que usa dispositivos denominados etiquetas o *tags* RFID. Una etiqueta RFID es un dispositivo pequeño, como una pegatina como las que se pueden ver en la figura 2.13, que puede ser adherida o incorporada a un producto, animal o persona. Las etiquetas RFID contienen antenas para permitirles recibir y responder a peticiones por radiofrecuencia desde un emisor-receptor RFID y pueden ser pasivas, semipasivas o activas.

Las etiquetas RFID pasivas no tienen fuente de alimentación propia. La mínima corriente eléctrica inducida en la antena por la señal de escaneo de radiofrecuencia proporciona suficiente energía al circuito integrado CMOS de la etiqueta para poder transmitir una respuesta. Debido a las preocupaciones por la energía y el coste, la respuesta de una etiqueta pasiva RFID es necesariamente breve, normalmente apenas un número de identificación. La falta de una fuente de alimentación propia hace que el dispositivo pueda ser bastante pequeño: existen productos disponibles de forma comercial que pueden ser insertados bajo la piel. Las etiquetas pasivas, en la práctica tienen distancias de lectura que varían entre unos 10 milímetros hasta cerca de 6 metros dependiendo del tamaño de la antena de la etiqueta y de la potencia y frecuencia en la que opera el lector.

Las etiquetas RFID semipasivas son muy similares a las pasivas, salvo que incorporan además una pequeña batería. Esta batería permite al circuito integrado de la etiqueta estar constantemente alimentado. Además, elimina la necesidad de diseñar una antena para recoger potencia de una señal entrante.

Las etiquetas RFID activas, por otra parte, deben tener una fuente de energía, y pueden tener rangos mayores y memorias más grandes que las etiquetas pasivas, así como la capacidad de poder almacenar información adicional enviada por el transmisor-receptor.

Actualmente, las etiquetas activas más pequeñas tienen un tamaño aproximado de una moneda. Muchas etiquetas activas tienen rangos prácticos de diez metros, y una duración de batería de hasta varios años.

Como las etiquetas pasivas son mucho más baratas de fabricar y no necesitan batería, la gran mayoría de las etiquetas RFID existentes son del tipo pasivo y son las que interesan en este proyecto. En cuanto a la tecnología RFID en sí, es interesante su posible aplicación, ya que permite tener alternativas al acceso al laboratorio por medio de teclado proporcionando al usuario final una interfaz más cómoda de usar, sin tener que recordar ningún número.

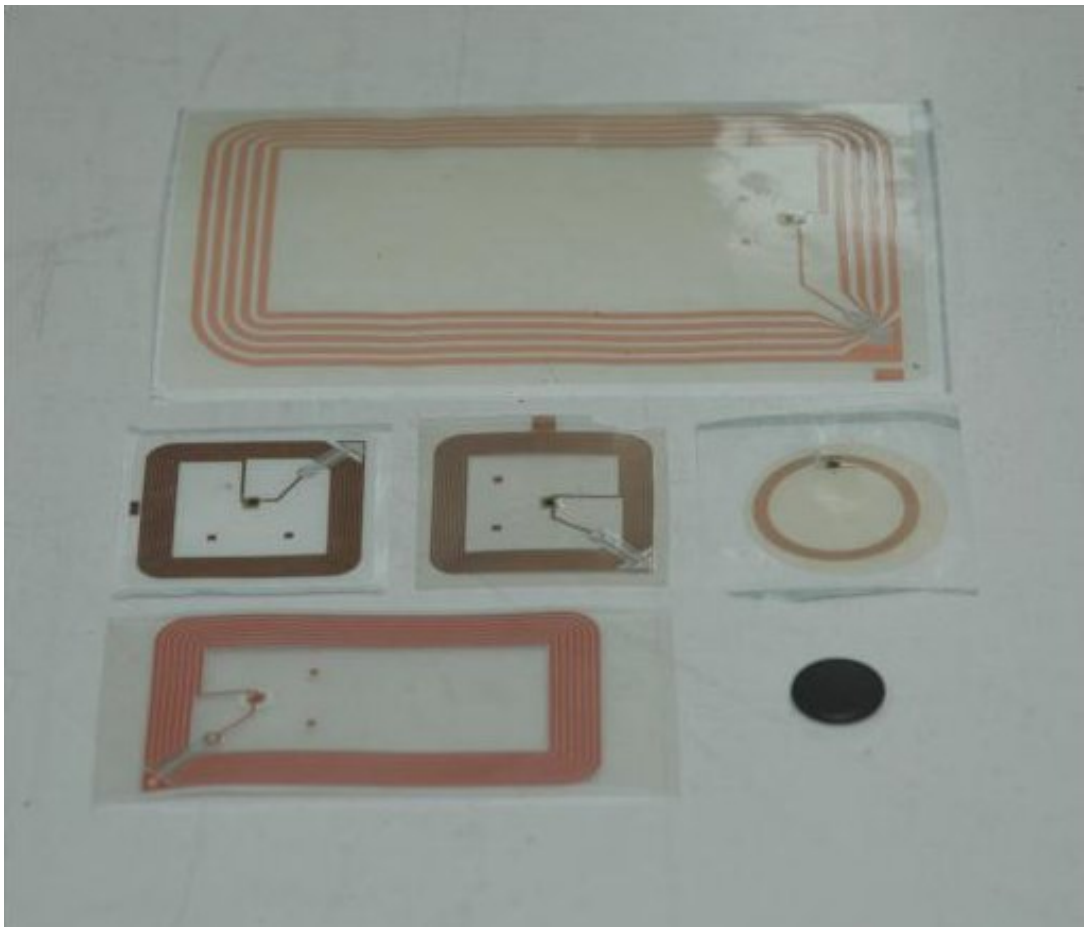

Figura 2.13. Etiquetas RFID pasivas

CAPÍTULO 3: DISEÑO DEL SISTEMA DE APERTURA DE PUERTAS

3.1 INTRODUCCIÓN

A lo largo de este capítulo se explicará cómo se ha realizado la parte *software* y *hardware* referentes a la entrada física del usuario en los laboratorios del Departamento de Tecnología Electrónica. Se hará una descripción del funcionamiento que se desea obtener y de cómo se ha obrado para cumplir dicho funcionamiento de la forma más estructurada y clara posible.

3.2 FUNCIONAMIENTO GENERAL

La mecánica del sistema implementado es la siguiente:

1. El usuario introduce su clave personal, que en el caso de ser a través del teclado a la entrada del laboratorio deberá seguir el siguiente formato: comenzar con "/" y acabar con "*". En el caso de usar el lector RFID, simplemente consistirá en acercar la tarjeta al lector.

2. Este número va directamente al programa que está ejecutándose en el ordenador al que está conectado el teclado, el cual lo envía de forma segura a un servidor vía servicio Web.

3. El servidor responde si esa clave es válida en ese momento concreto y para ese laboratorio, también de forma segura, accionándose (en caso afirmativo) el mecanismo de apertura. En caso de no estar operativo el servidor se consulta en la caché local, y si alguien ha accedido con esa clave en las 24 horas anteriores, se acciona el mecanismo de apertura.

4. Si ha podido acceder se almacena en la caché local la contraseña para eventuales caídas del servidor o imposibilidad de acceder al mismo.

3.3 CLIENTE VÍA TECLADO DEL SERVICIO WEB

3.3.1 DESCRIPCIÓN

Esta parte del diseño está dotada con un interfaz gráfico que de cara al usuario es muy sencillo en todo a la hora de configurar sus parámetros, tal y como se puede apreciar en las figuras 3.1 y 3.2.

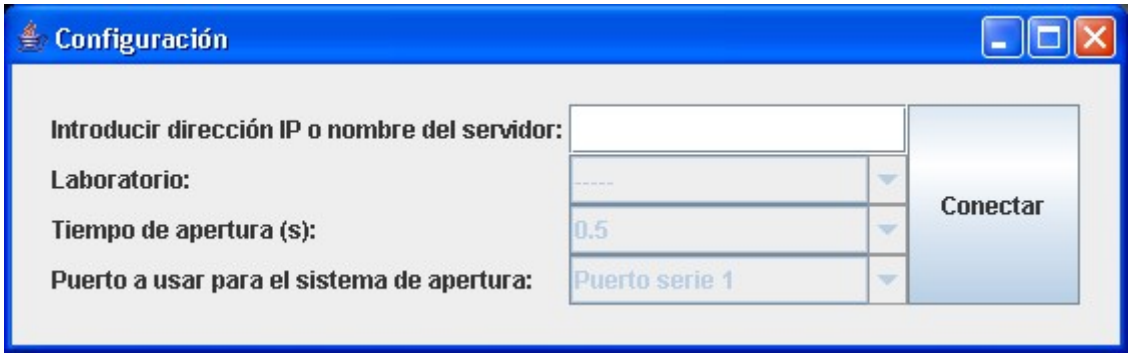

Figura 3.1. Ventana inicial del cliente vía teclado

Al arrancar este programa por primera vez, aparece la ventana que se muestra en la figura 3.1, donde se pide la dirección IP o el nombre del servidor al cual se debe conectar, mientras que el resto de parámetros aparecen deshabilitados hasta que no se confirme la conexión con el servidor. Así se asegura tener una lista de laboratorios existentes en la base de datos. Una vez localizado el servidor, la ventana queda como se muestra en la figura 3.2, con el campo correspondiente al servidor deshabilitada y el resto de campos ya habilitados. En esta ventana se puede proceder a configurar tres parámetros para el correcto funcionamiento del sistema. Los campos de los que consta esta ventana se detallan a continuación:

- **Laboratorio**: aparecería el primer laboratorio que se cargue del servicio Web. Para ello, al arrancar el programa hace una petición al servicio Web para obtener una lista de todos los laboratorios inscritos en la base de datos, y que por lo tanto pertenecen al Departamento de Tecnología Electrónica.

- **Tiempo de apertura (s)**: aquí se puede especificar durante cuánto tiempo se va a actuar en el abrepuertas para que la apertura sea efectiva. En principio no tendría porqué cambiarse este valor, pero para tener un margen de libertad sobre el mecanismo hardware es interesante contar con esta posibilidad. Se optó para ello dejar por tener un rango de variación desde 0.5 segundos a 5 segundos.

- **Puerto a usar para el sistema de apertura**: permite especificar el puerto al que se va a conectar el sistema de apertura de puertas, dando un grado de libertad al usuario.

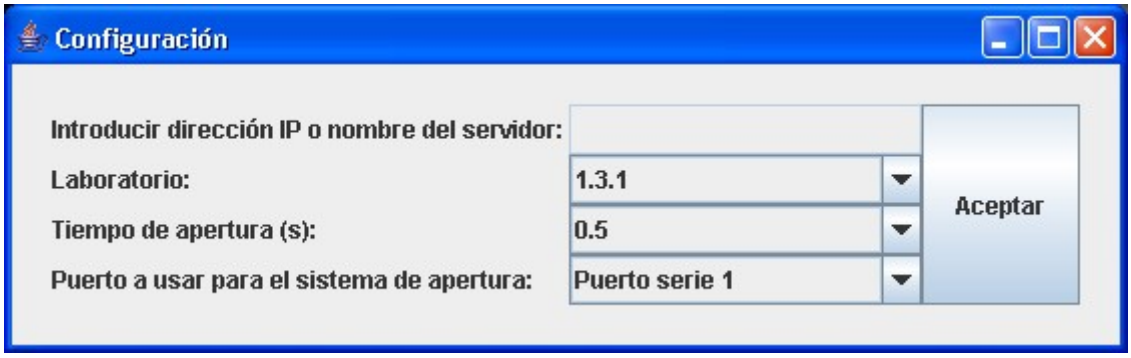

Figura 3.2. Ventana de configuración de parámetros del cliente vía teclado

Cuando ya están configurados todos los parámetros adecuadamente y se selecciona "Aceptar", se pasa a la ventana de la figura 3.3, dónde ya el programa se queda en espera de recibir datos a través del teclado.

Figura 3.3. Interfaz del cliente que recoge los datos de teclado

Por orden cronológico, lo primero que hace el programa automáticamente después de introducir la dirección o el nombre del servidor, es conectarse al servicio Web y obtener una lista de los laboratorios pertenecientes al Departamento de Tecnología Electrónica. Dicha lista se encuentra almacenada en una base de datos gestionada con MySQL llamada "accesolabdte", más concretamente en la tabla "laboratoriosdte". Esta tabla contiene una lista de los laboratorios con una pequeña descripción de cada uno y son los identificadores de los laboratorios los que aparecen posteriormente en el campo "Laboratorio" de la ventana de configuración de parámetros para conseguir varios objetivos:

1. Extrema sencillez a la hora de manejar el programa

2. Siempre se trabaja con identificadores válidos, ya que se toman directamente de la base de datos

3. El mismo programa es válido para instalarlo en cualquier laboratorio, sólo hay que configurarlo

4. El programa se actualiza en cuanto hay un cambio en la base de datos al arrancarlo de nuevo

Para conseguir la conexión del programa cliente a la base de datos que recupera la lista de laboratorios se emplea JDBC. Mediante este interfaz se puede comunicar Java con multitud de bases de datos; en este caso será MySQL, como ya se ha dicho.

Una vez arrancado con la lista de laboratorios cargada, se puede proceder a configurar todos los parámetros anteriormente comentados para así dejar la aplicación a la espera de datos.

3.3.2 LA APLICACIÓN EN DETALLE

3.3.2.1 Sistema operativo

En una primera versión de esta aplicación era necesario seleccionar el sistema operativo en el que se trabaja, pero finalmente se ha implementado de forma que lo reconozca automáticamente entre sistemas de tipo Unix y Windows. Es necesario conocer esto, ya que el programa debe acceder a los recursos físicos de la máquina, que se direccionan de forma diferente según se esté en uno o en otro. Es justamente en este punto donde hay que recurrir a las librerías externas, ya que Java, en principio no permite acceder a los puertos del ordenador; en este caso será al puerto serie. La librería usada es "*rxtx*", la cual permite acceder tanto al puerto serie como al paralelo.

3.3.2.2 Indicación del laboratorio actual ("Laboratorio:")

En este campo de la ventana de configuración se indica al programa en cual de los laboratorios cargados desde el servicio Web nos encontramos actualmente. Este campo es muy importante, ya que es un dato que se envía al servicio Web junto con la contraseña introducida y, obviamente, no todos los usuarios tienen acceso a todos los laboratorios y a todas horas.

3.3.2.3 Indicación del tiempo que se debe accionar el mecanismo de apertura (“Tiempo de apertura (s) :”)

En este campo de la ventana de configuración es donde se debe seleccionar uno de los valores aproximados posibles en el margen que va desde los 0.5 segundos a los 5 segundos en saltos de 0.5 segundos.

Para entender cómo se ha realizado esto hay que comenzar por la parte física y llegar hasta el punto en que se determina la duración de la acción de apertura:

1. Se consigue accionar el mecanismo abrepuertas a través de un relé que se encuentra dispuesto en una placa alimentada por 12V de tensión continua.

2. Este relé está controlado por un chip integrado que consiste en un optoacoplador integrado por un diodo de infrarrojos y un receptor, todo ello encapsulado. De esta forma se consigue aislar perfectamente la entrada y la salida.

3. El diodo a su vez está controlado directamente por el puerto serie del ordenador, en concreto por dos cables (hay que resaltar que a pesar de tener 9 pines el conector del puerto serie que se usa en el proyecto, sólo hay 4 cables). Uno de los cables es el de referencia (masa o GND) y el otro es el de transmisión (en reposo se encuentra a -12V) y se conectan directamente al diodo, el de referencia al cátodo y el de transmisión al ánodo. De esta forma, en reposo el diodo es un circuito abierto.

4. Como lo que se pretende es obtener un cierto nivel de tensión positiva a la entrada del diodo cuando se quiere accionar el relé, hay que actuar sobre la transmisión de datos por el puerto serie. Para ello se fija la velocidad de transmisión al mínimo posible (1 bps) y se sacan tantos ceros como sean necesarios para conseguir una salida positiva un cierto tiempo. Directamente proporcional al número de ceros que se saquen será el tiempo que esté accionado el relé.

3.3.2.4 Indicación del puerto (“Puerto a usar para el sistema de apertura:”)

En este campo de la ventana de configuración se indica el puerto al que se va a conectar el dispositivo que accionará el sistema de apertura de la puerta. Por defecto es el puerto serie número 1.

3.3.2.5 Entrada de contraseña (“Contraseña:”)

En este campo del interfaz del cliente, al introducir la clave, sólo aparecen asteriscos por razones obvias, pero internamente el programa toma los caracteres tal cual se introducen y debe ser capaz de separar la clave de los caracteres de inicio y fin (hay que recordar que para introducir una clave hay que comenzar por “/” y acabar en “*”). Hecho esto, envía la clave al servicio Web que se encarga de comprobar si, con la clave introducida, se puede acceder a ese laboratorio en ese instante de tiempo.

Pero para que todo esto sea robusto, hay que garantizar además ciertos aspectos:

1. La aplicación no se puede bloquear por una pulsación prolongada. Para lograrlo se ha asegurado que el único carácter válido para comenzar una clave sea “/” y el resto se ignoren completamente; la pulsación de un carácter diferente como comienzo no se recoge ni por pantalla ni internamente. En caso de que la pulsación prolongada sea de “/” tampoco habrá problema, ya que sólo se recoge una tanto por pantalla como internamente y el resto se van borrando según se va pulsando; es decir, el hecho de introducir “/” borra todo lo anterior y sólo queda “/”.

2. Siempre se debe comenzar la introducción de la clave con “/” y acabar con “*”. La primera parte ya se ha comentado en el primer punto; en cuanto a la segunda se obra con la misma filosofía para que el único carácter que valida el envío de la clave (siempre que antes se haya pulsado “/”) es “*”.

3. No se debe poder introducir una clave arbitrariamente larga. Esto casi formaría parte del primero de los aspectos a garantizar, ya que, en el caso de introducir una “/” y luego hacer una pulsación prolongada también se podría bloquear el sistema a no ser que, tal y como se ha hecho, se ponga un máximo, en este caso

de 10 caracteres incluyendo "/" y "*". Si se alcanza ese máximo se borra toda la clave introducida y, como debe haber una "/" al comienzo, si se continúa pulsando ese carácter prolongadamente el programa no lo recoge ni por pantalla ni internamente.

Dado que todo esto está implementado, se asegura que al servicio Web siempre le va a llegar un clave que no produzca error en el mismo.

3.3.2.6 Indicación de lo ocurrido en el último intento de acceso ("Último acceso:")

En este campo del interfaz del cliente se indica el resultado del último intento de acceso, obtenido a partir de lo que devuelve el servicio Web. Sus posibles resultados son "acceso permitido" o "acceso denegado".

3.3.3 DIAGRAMA DE CLASES DEL CLIENTE VÍA TECLADO DEL SERVICIO WEB

En la figura 3.4 se pueden ver esquemáticamente las clases implementadas para obtener la funcionalidad del cliente vía teclado del servicio Web. En los recuadros, que representan cada uno a una clase, se pueden apreciar en cada separación y de arriba a abajo, el nombre de la clase, las propiedades de la clase y los métodos que implementa. La línea de puntos indica un comentario, las líneas continuas indican asociación entre clases (una clase que hace uso de otras) y la flecha indica herencia en el sentido de la flecha (la clase apuntada es la clase padre).

La clase principal es "MeterPwd". A partir de ella se llama a una instancia de la clase "VentanaPreguntaIP", que es la encargada de recoger la dirección IP o nombre del servidor donde se encuentra alojado el servicio Web. Una vez que se ha introducido correctamente este dato, haciendo uso de una instancia de la clase "ObtieneLaboratorios" se consigue una lista de los laboratorios que están en la base de datos, y se puede proceder a configurar el resto de los parámetros. Cuando ya todos los parámetros están configurados, se pasa el control a la clase "IHM", que son las siglas de interfaz hombre-máquina, y que justamente

hace eso: se queda esperando a recibir las pulsaciones de teclado que identifican a una contraseña para proceder o no a la apertura de puertas. El hecho de que las pulsaciones de teclado sean sólo las permitidas tal y como se especificaba en el punto 3.3.2.5 se asegura gracias a la clase "LimitadorTexto". Una vez introducida una contraseña siguiendo esas reglas, se procede a comprobar si se permite el acceso a esa contraseña a través del servicio Web, y, en caso de permitirse, almacenarla en caché con una instancia de "ManagerDeCache". La apertura efectiva de puertas se realiza con una instancia de "AbrirPuerta".

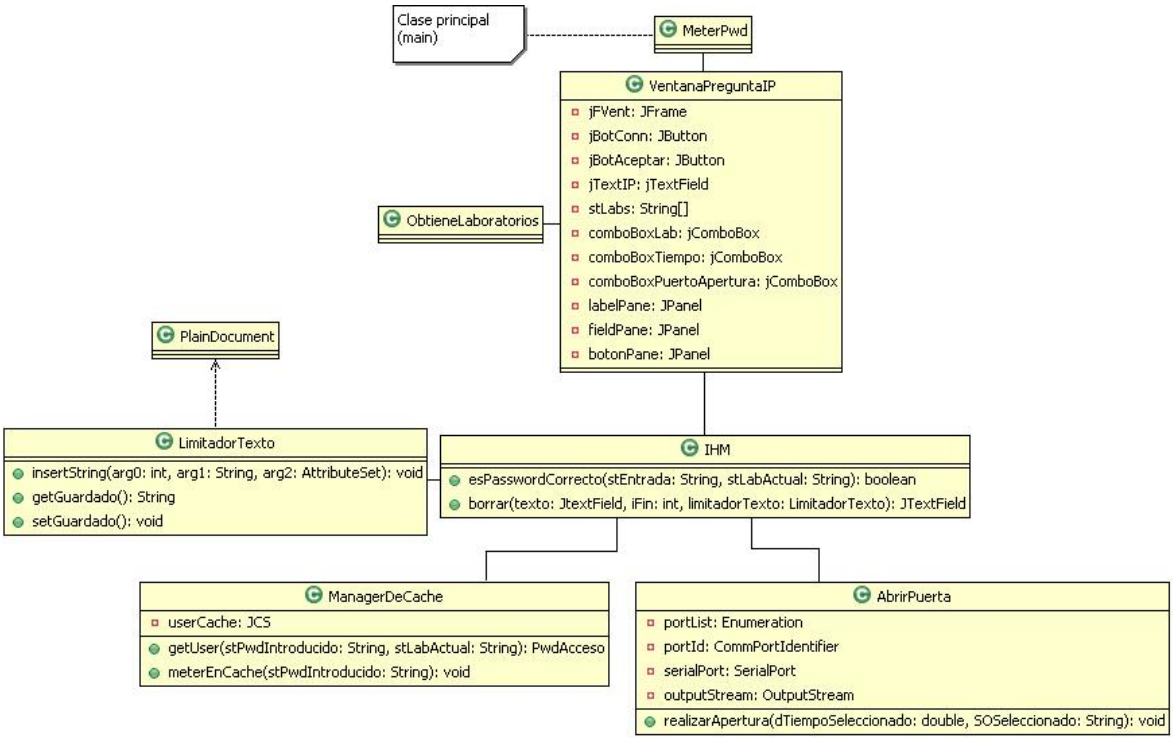

Figura 3.4. Diagrama de clases del cliente vía teclado del servicio Web

3.4 CLIENTE VÍA LECTOR RFID DEL SERVICIO WEB

3.4.1 DESCRIPCIÓN

Al igual que en el caso del cliente vía teclado, el interfaz gráfico resultante de cara al usuario es muy sencillo en todo a la hora de configurar sus parámetros, tal y como se puede apreciar en las figuras 3.5 y 3.6.

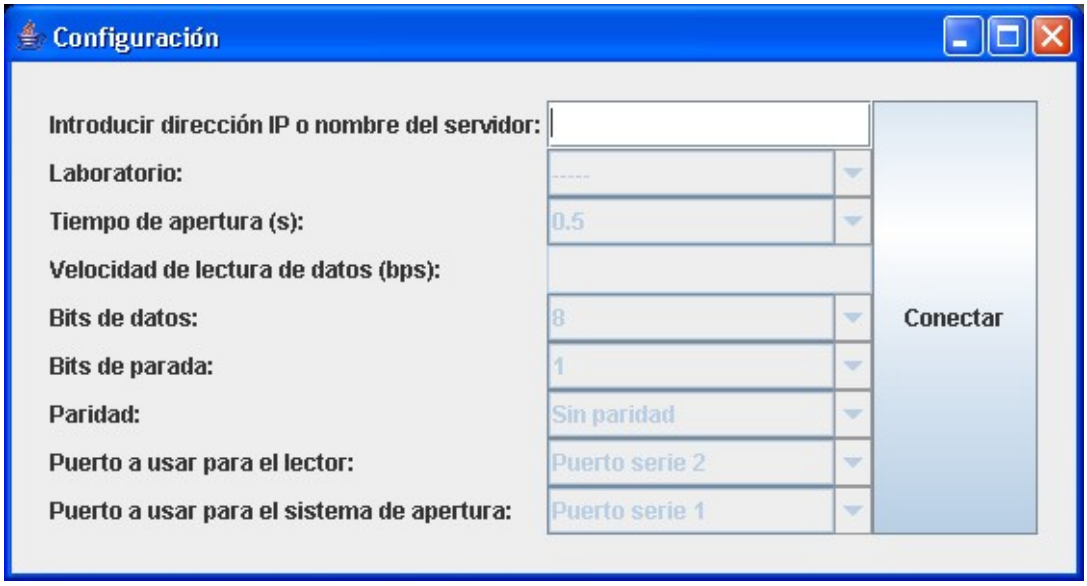

Figura 3.5: Ventana inicial del cliente vía lector RFID

Como sucedía con el cliente vía teclado, el único campo que se puede modificar es el destinado a insertar la dirección o el nombre del servidor y, una vez hecho esto y si ha tenido éxito la conexión con el servidor, se habilitan el resto de campos y se deshabilita el del servidor, quedando la ventana como se ve en la figura 3.6. En esta figura se ha rellenado ya el campo de la velocidad de lectura de datos con la velocidad del lector que se ha usado para las pruebas.

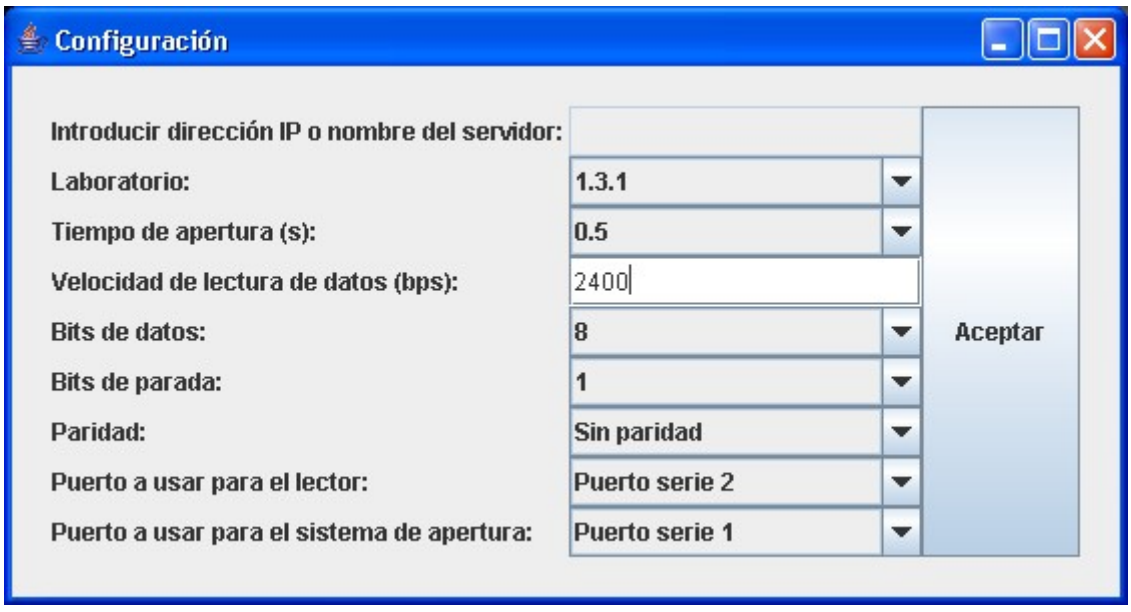

Figura 3.6: Ventana de configuración de parámetros del cliente vía lector RFID

Hay tres campos que se repiten respecto al cliente vía teclado, que son “Laboratorio”, “Tiempo de apertura (s)” y “Puerto a usar para el sistema de apertura” y tienen exactamente la misma función. El resto son parámetros propios de la lectura a través del puerto serie, gracias a los cuales se puede configurar cualquier lector (RFID o no) que se pueda conectar a dicho puerto. De esta forma se puede configurar con cuatro parámetros (“Velocidad de lectura de datos”, “Bits de datos”, “Bits de parada” y “Paridad”) la modalidad de lectura en el puerto seleccionado y en qué puerto se va a realizar la lectura. Cabe resaltar que, obviamente no se puede usar el mismo puerto para la lectura y para el sistema de apertura de puertas, y, de seleccionarse así, aparece un mensaje indicando del error. También se comprueba, que existan y estén disponibles los puertos serie; de no ser así, igualmente se indica con una ventana emergente que detalla el error (figura 3.7).

De la misma forma que ocurre con el cliente vía teclado, una vez seleccionadas todas las opciones se pasa al modo de espera de contraseñas, tal y como se ve en la figura 3.8.

Figura 3.7: Excepción de puerto no existente

Figura 3.8: Interfaz del cliente que recoge los datos del lector

3.4.2 DIAGRAMA DE CLASES DEL CLIENTE VÍA LECTOR RFID DEL SERVICIO WEB

En la figura 3.9 se pueden ver esquemáticamente las clases implementadas para obtener la funcionalidad del cliente vía lector RFID del servicio Web. La simbología es la misma que la mostrada en el apartado 3.3.3.: en los recuadros, que representan cada uno a una

clase, se pueden apreciar en cada separación y de arriba a abajo, el nombre de la clase, las propiedades de la clase y los métodos que implementa. La línea de puntos indica un comentario, las líneas continuas indican asociación entre clases (una clase que hace uso de otras) y la flecha indica herencia en el sentido de la flecha (la clase apuntada es la clase padre).

Al igual que en el cliente vía teclado, la clase principal es “MeterPwd”. A partir de ella se llama a una instancia de la clase “VentanaPreguntaIP”, que es la encargada de recoger la dirección IP o nombre del servidor donde se encuentra alojado el servicio Web. Una vez que se ha introducido correctamente este dato, haciendo uso de una instancia de la clase “ObtieneLaboratorios” se consigue una lista de los laboratorios que están en la base de datos, y se puede proceder a configurar el resto de los parámetros. Es de notar que uno de los parámetros, la velocidad del puerto serie a través del cual se conecta el lector RFID, está limitado por una instancia de la clase “LimitadorNumeros”, de manera que en ese campo sólo se puedan escribir números. Cuando ya todos los parámetros están configurados, se pasa el control a la clase “IHM”, que en este caso sólo se encarga de mostrar por pantalla el resultado del intento de acceso. De recoger las contraseñas se encarga una instancia de la clase “LecturaSerie”, la cual, a través de otro hilo de ejecución, recibe todos los datos que entren por el puerto serie e identifica aquellos susceptibles de ser una contraseña. Una vez introducida una contraseña a través del lector, se procede a comprobar si se permite el acceso a esa contraseña a través del servicio Web, y, en caso de permitirse, almacenarla en caché con una instancia de “ManagerDeCache”. La apertura efectiva de puertas se realiza con una instancia de “AbrirPuerta”.

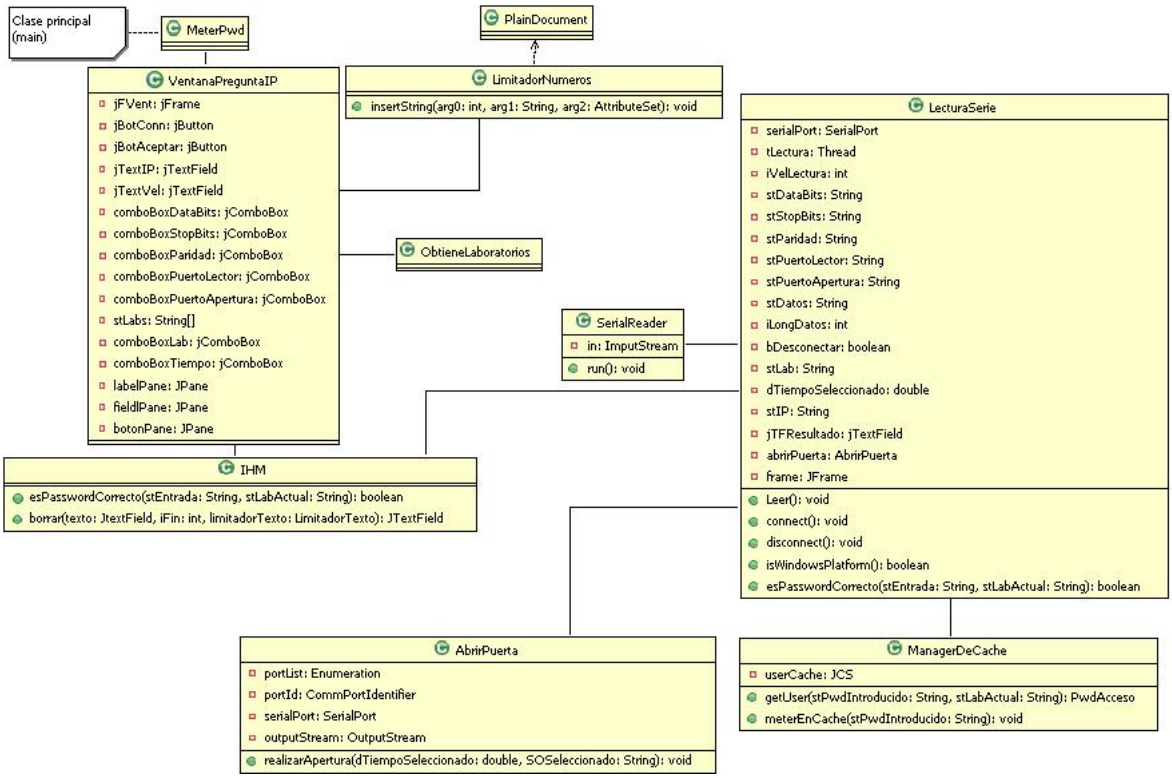

Figura 3.9: Diagrama de clases del cliente vía lector RFID del servicio Web

3.5 SERVICIO WEB

3.5.1 DESCRIPCIÓN

Esta es la parte menos visible del diseño pero sin duda es fundamental para que todo funcione correctamente. Consta de dos clases, una auxiliar (“PwdAcceso”) y la otra (“LeerPwdSW”) que es la que realmente dota de funcionalidad al servicio Web. Esta última tiene todos los métodos necesarios para recoger, validar y registrar los accesos a cada laboratorio. Los servicios siempre son en el mismo sentido: petición del ordenador local del laboratorio, es decir, el encargado de recoger las contraseñas, al servidor del servicio Web. Estos servicios ofrecidos son:

1. Cuando el cliente, es decir, el ordenador de un laboratorio se conecta al servicio Web, lo primero que hace es pedir una lista de los laboratorios que constan en la base de datos para que el técnico pueda identificar el laboratorio en cuestión. Esta lista aparece luego reflejada en la ventana del cliente (figuras 3.2 y 3.6)

2. Cuando ya están todos los parámetros configurados, al recibir el ordenador local del laboratorio una contraseña (vía teclado o vía lector), envía al servicio Web

dicha contraseña y el identificador del laboratorio, para que el servidor decida si se puede o no acceder a ese laboratorio en ese instante concreto con esa contraseña. En función de la respuesta, el ordenador local obra abriendo o no la puerta.

3.5.2 SERVICIO INACCESIBLE

En el caso de no encontrarse el servidor accesible u operativo por cualquier motivo, se ha dotado al sistema de una vía de emergencia consistente en lo siguiente: si alguien ha accedido en las últimas 24 horas con una cierta contraseña y, mientras el servidor está caído, vuelve a intentar acceder se le dará acceso. En caso contrario se le deniega el acceso.

Esto se ha implementado mediante un sistema de caché en Java llamado JCS, en el que, en este caso, se usa la caché de forma inversa a como se acostumbra. Es decir, lo habitual en un sistema con caché es mirar primero en la caché, y si allí no se encuentra el dato buscado, ir a la memoria principal. En este sistema se implementa al revés, para usarla como base de datos local en caso de fallo. Dicha caché se almacena tanto en la memoria principal del ordenador como en el disco duro, para tener así dos niveles.

3.5.3 ESQUEMA DE FUNCIONAMIENTO DEL SERVICIO WEB

3.5.3.1 Introducción

En esta sección se describirá visualmente lo que ya se ha descrito en el punto 3.5.1, viendo en los dos servicios Web implementados, la petición y la respuesta de los mismos, así como el diagrama de clases empleadas para la consecución de los mismos.

3.5.3.2 Lista de laboratorios

3.5.3.2.1 Petición

En la figura 3.10 se puede observar esquemáticamente cómo se realiza la petición al servicio Web, en este caso de la lista de laboratorios disponibles. Se parte del ordenador

local, donde está instalado el cliente del servicio Web, invocando a una clase que realiza las funciones de "*proxy*". A partir de ella comienza el intercambio de mensajes SOAP con el servicio Web (clase "LeerPwdSW") usando como enlace la clase "LeerPwdSWBinding". A la derecha de la figura 3.10 se puede ver la lista de los mensajes que se pueden intercambiar, en la que está resaltada la petición de la lista de los laboratorios ("listaLaboratoriosRequest").

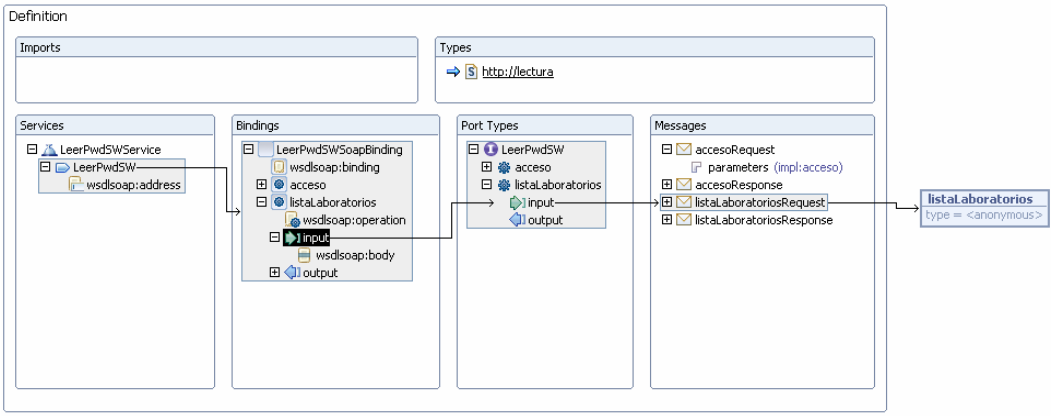

Figura 3.10. Petición de la lista de laboratorios

3.5.3.2.2 Respuesta

En la figura 3.11, donde se puede ver la respuesta del servicio Web, se aprecia que la parte interesante es justamente el formato de la respuesta donde aparece el tipo, que este caso será cadena de texto ("*String*") y en qué cantidad aparecerán ("1..*"). El resto es prácticamente igual que la petición, ya que el interfaz es el mismo.

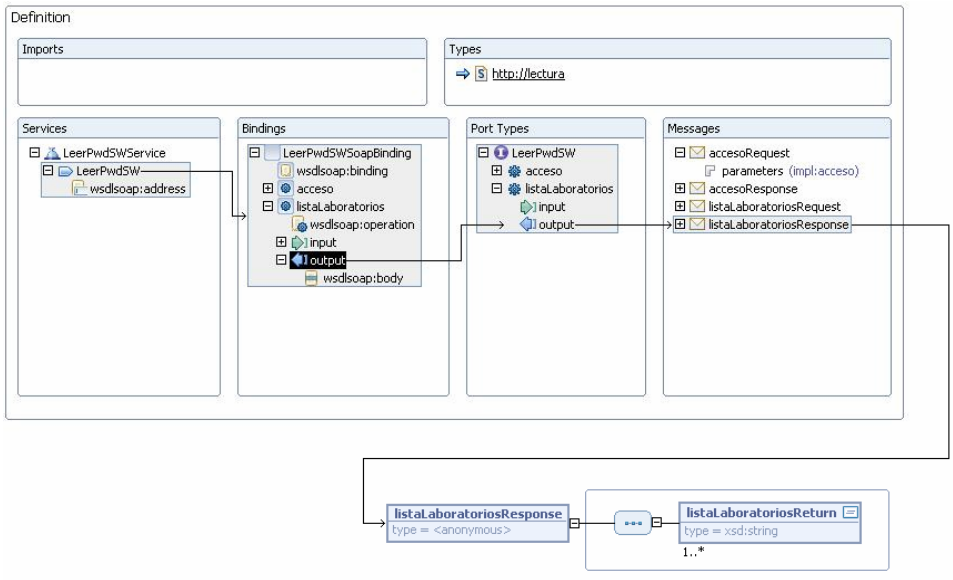

Figura 3.11. Respuesta de la lista de laboratorios

3.5.3.3 Acceso a los laboratorios

3.5.3.3.1 Petición

En la figura 3.12 se puede observar el esquema de la petición de acceso que ejecuta el programa residente en el ordenador local. Los interfaces son los mismos que los explicados en el punto 3.5.3.2.1. Se puede ver que en este caso sí que se pasan parámetros en "accesoRequest" (a diferencia de la petición en el caso del servicio que implementa la obtención de la lista de laboratorios disponibles): la clave introducida ("stPwd") y el identificador del laboratorio ("stLab") desde el que se realiza la petición.

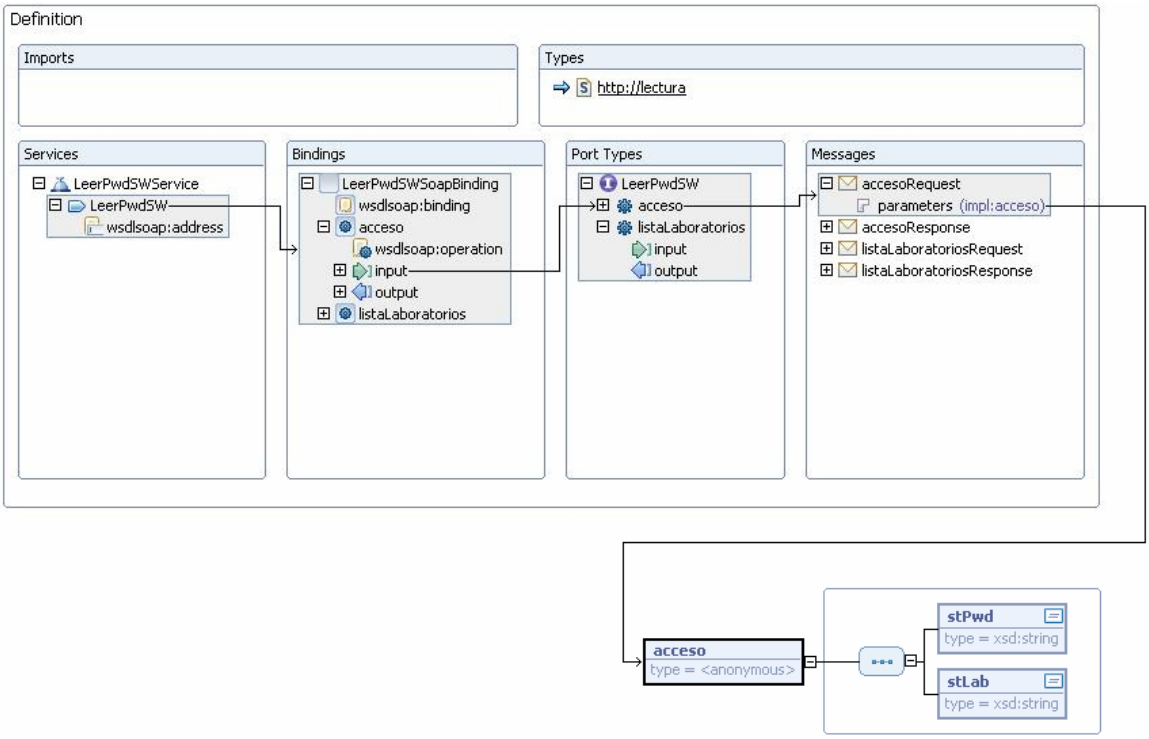

Figura 3.12. Petición de acceso al laboratorio

3.5.3.3.2 Respuesta

En la figura 3.13 se puede observar el esquema de la respuesta de acceso ("accesoResponse") que devuelve dos datos: por una parte dice si la contraseña dada en la petición y asociada al laboratorio que también se indicó en la petición es válida en ese momento (para ello devuelve verdadero o falso), es decir, si se da acceso, y por otra parte devuelve otra vez el identificador del laboratorio en una cadena de texto. Esto se usará en

el ordenador local, como ya se ha visto, para actualizar la base de datos local (caché) y tendrá una gran utilidad como sistema de emergencia por si el servidor cae. En caso de tener condiciones normales, el único dato que tendrá relevancia a la hora de accionar el mecanismo de apertura de puertas será *acceso*, que devuelve verdadero o falso. Es interesante señalar también que la respuesta está encapsulada en un objeto de tipo "PwdAcceso" y que en la figura está representada mediante el recuadro que rodea a los dos parámetros que devuelve ("acceso" y "pwd"), que son dos propiedades de la clase.

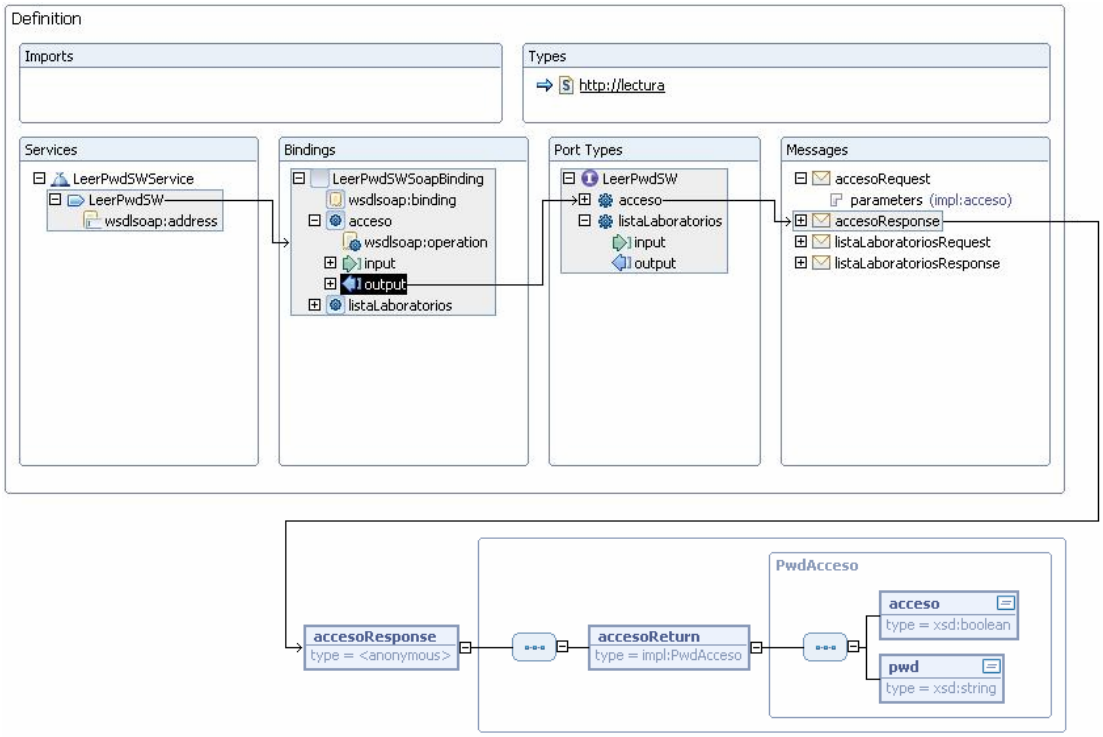

Figura 3.13. Respuesta de acceso al laboratorio

3.5.3.4 Diagrama de clases del servicio Web

En la figura 3.14 se muestran las clases empleadas para conseguir la funcionalidad del servicio Web. Al igual que en las figuras 3.4 y 3.9, aparece cada clase representada por un recuadro, y dentro del mismo, se puede apreciar en cada separación y de arriba a abajo, el nombre de la clase, las propiedades de la clase y los métodos que implementa. Las líneas continuas indican asociación entre clases (una clase que hace uso de otras).

En este caso es la clase “LeerPwdSW” la principal, la cual se sirve de la clase “ParámetrosBBDD” para obtener los parámetros necesarios para poder conectarse a la base de datos y que, en su procedimiento *acceso(String stPwd,String stLab)* (que es uno de los servicios), hace uso de un objeto del tipo “PwdAcceso” para empaquetar los datos que se le piden al servicio Web. El otro servicio (también básico) se da a través del procedimiento *listaLaboratorios()* para devolver al cliente una lista actualizada de los laboratorios disponibles.

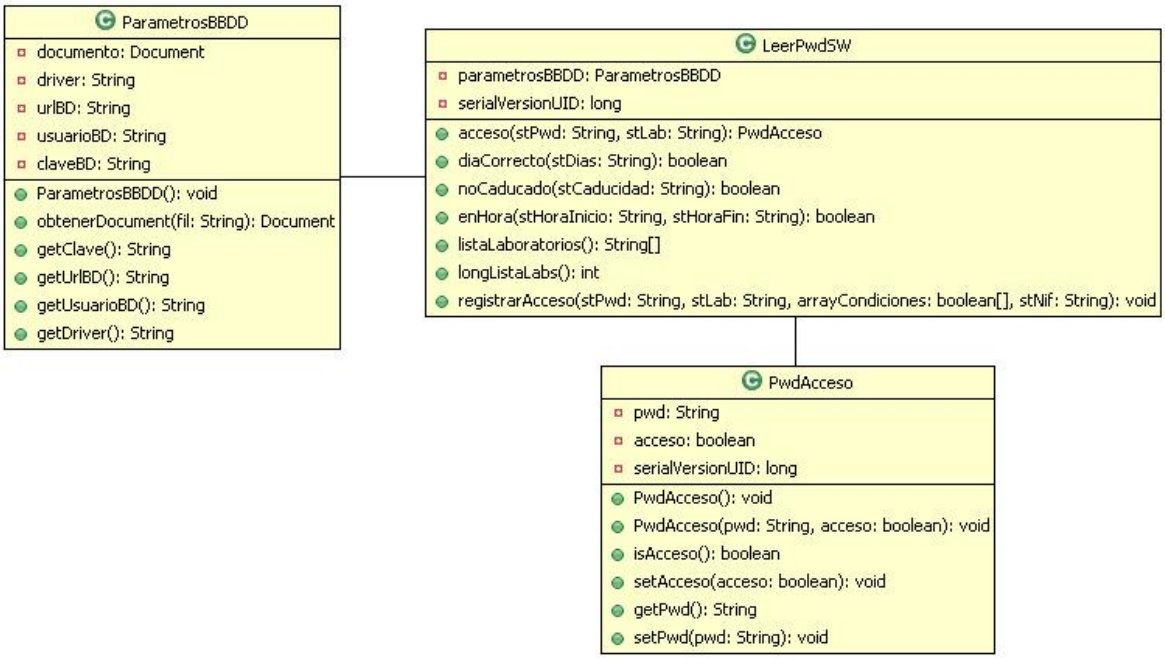

Figura 3.14. Diagrama de clases del servicio Web

3.5.4 SEGURIDAD

Un aspecto que aún no se ha comentado respecto al servicio Web y que probablemente sea de los más importantes es que está dotado de encriptación. Esto es muy necesario, ya que los datos que se van a intercambiar por la red son contraseñas de acceso a los laboratorios y en ningún momento deben ser susceptibles de ser escuchados en claro, es decir, sin encriptación. Para conseguir esto se ha usado un conocido algoritmo, como es el *Blowfish*, en su versión para cadenas de caracteres, de forma que el servicio Web puede transmitir de forma estándar los datos en archivos XML tal y como se ha explicado en el punto 2.3.

Para dotarlo de una seguridad lo más alta posible sin sacrificar para ello la velocidad se ha usado una clave de 65536 bits para encriptar los datos al enviarlos al servicio Web. Una vez que llegan al servidor se desencriptan y se envía la respuesta al cliente también encriptada. El cliente obtiene así la respuesta que buscaba de una forma segura.

3.6 ACCESO FÍSICO

3.6.1 ANTECEDENTES

Actualmente, la parte física del sistema es como sigue:

- Teclado microprocesado a la entrada del laboratorio.
- Tarjeta ISA (*Industry Standard Architecture*) de adquisición de datos para pasar la contraseña introducida al ordenador local (es el ordenador local el que decide si un usuario puede o no entrar).
- Se acciona el mecanismo de apertura de puertas a través de una placa conectada al ordenador mediante el puerto serie.

3.6.2 NUEVO SISTEMA

El sistema renovado sería el siguiente:

- En el caso de usar el teclado, sería un teclado no microprocesado, que se puede conectar al ordenador local del laboratorio a través del puerto USB o del PS/2 (*IBM Personal System/2*). En el caso del lector, sería un lector (RFID) conectado a uno de los puertos serie del ordenador local del laboratorio.
- Se acciona el mecanismo de apertura de puertas a través de un módulo optoacoplado conectado al puerto serie, con salida en un relé que es el que finalmente acciona el abrepuertas.

3.6.3 REALIZACIÓN

En el caso del teclado puede ser uno numérico cualquiera como los que se conectan a los portátiles, por lo que no necesita mucha explicación. En el caso del lector, la instalación también es muy simple, ya que sólo se trata de conectar el lector a un puerto serie del ordenador.

Sí es más interesante ver como queda la parte que acciona físicamente el abrepuertas y que en la figura 3.15 se puede ver ya montado al completo. Consta del módulo optoacoplador, al que se conectan el cable serie del ordenador y la alimentación, por medio de un transformador de 12 V de tensión continua y al menos 100 mA de corriente para poder accionar el mecanismo de apertura de puertas, y tiene dos hilos de salida al mecanismo que realiza la apertura de la puerta.

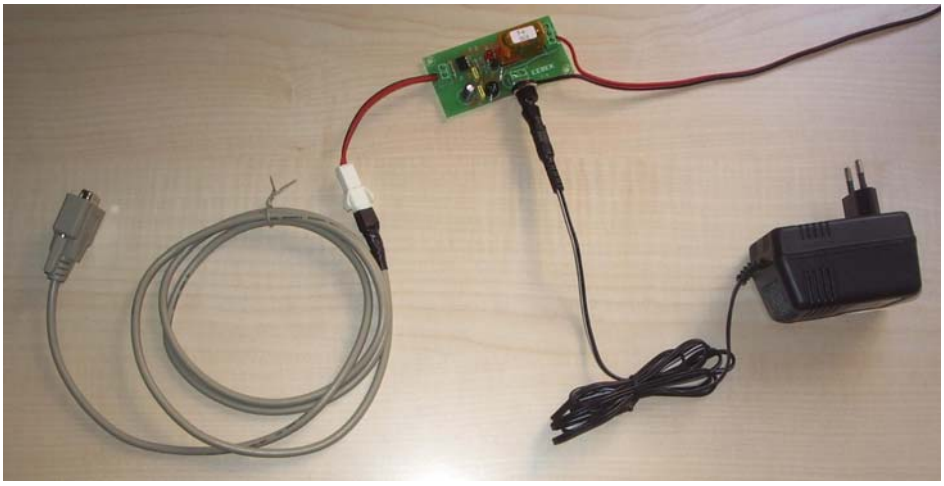

Figura 3.15. Sistema que acciona el abrepuertas

En la figura 3.16 se puede ver claramente el puerto serie utilizado con el conector que se ha adaptado para realizar el conexionado a la placa, que es el módulo optoacoplado.

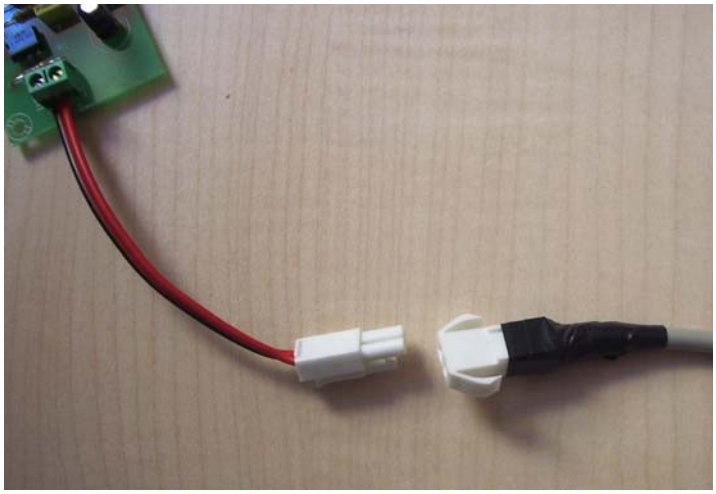

Figura 3.16. Detalle del conector del puerto serie

En la figura 3.17 se puede ver con más detalle el módulo optoacoplador y la conexión a la fuente de alimentación. Caben resaltar dos elementos fundamentales: el diodo acoplado con el receptor, integrado en el chip IC1 según se ve en la imagen y el relé (con la pegatina que pone “T-4 C616”) que conecta la fuente de alimentación a la salida cuando se acciona el receptor óptico.

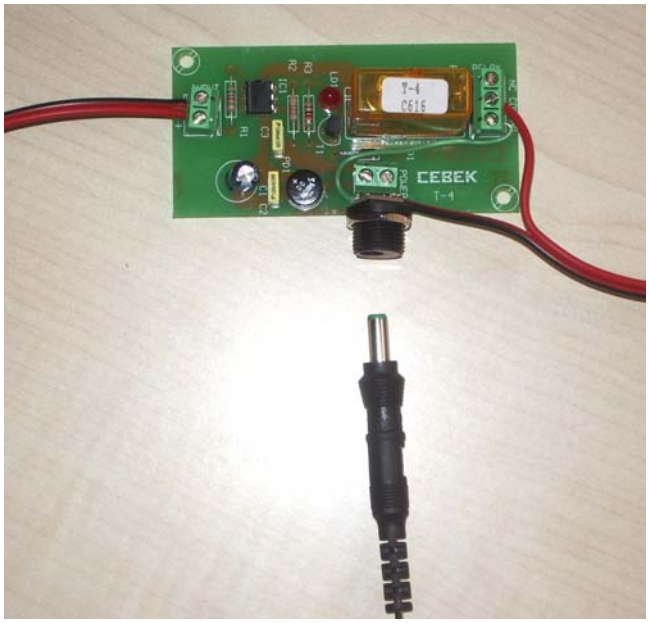

Figura 3.17. Detalle del optoacoplador y la conexión a la fuente de alimentación

En la figura 3.18 se puede apreciar otra vista del módulo optoacoplador y en particular de la base a la que se conecta la fuente de alimentación, además del interior del relé.

Figura 3.18. Otra vista del módulo optoacoplador

3.6.4 JUSTIFICACIÓN

Las alternativas a este sistema son bastante similares, a saber:

- Usar uno de los cables de alguno de los puertos del ordenador para dar la corriente necesaria para abrir la puerta.
- Realizar un módulo usando como pieza clave un transistor que conmute entre corte y saturación.
- Construcción de un sistema equivalente al módulo que se ha comprado hecho también con optoacoplador.

La primera de las alternativas fue rápidamente desechada al ver que no se puede conseguir suficiente corriente de salida procedente de los puertos de comunicación del ordenador para accionar el abrepuertas.

En cuanto a usar un transistor para realizar la conmutación habría sido una solución válida, y con un precio final parecido al del sistema que se ha usado finalmente. La ventaja que se obtiene con el optoacoplador es que nos aseguramos de que en ningún caso, por malfuncionamiento del dispositivo va a fluir una corriente de entrada al ordenador que pudiera averiarlo. También se pueden realizar circuitos de protección, pero habrían

complicado el circuito final, mientras que con la solución implementada es más sencillo identificar cualquier problema que pudiera producirse.

Respecto a construir un sistema equivalente al módulo que se ha comprado no tiene demasiado sentido dado el bajo precio del mismo. Además se cuenta con una garantía de funcionamiento y de cualquier forma sigue siendo sencillo identificar cualquier problema que pueda producirse y proceder a su reparación, aún si el proveedor de estos módulos dejase de fabricarlos, ya que todos los elementos del circuito son bastante comunes.

CAPÍTULO 4: DISEÑO DE LA APLICACIÓN WEB DE GESTIÓN DE ACCESOS

4.1 INTRODUCCIÓN

En este capítulo se describirá la parte del proyecto relacionada con el sitio Web de gestión de los accesos a los laboratorios. Se dará una visión general del sitio Web para luego adentrarse en partes más detalladas del mismo. Con este objetivo se usarán diagramas de flujo que permitan comprender de un vistazo la organización del sistema implementado. Asimismo se explicarán en detalle aspectos relacionados con todo el proyecto (con el sitio Web y también con la parte menos visible: el servicio Web y el cliente del servicio Web) como son la creación y estructuración de la base de datos y la seguridad.

4.2 FUNCIONAMIENTO

4.2.1 FUNCIONAMIENTO GENERAL

A la hora de diseñar el sitio Web siempre se ha buscado obtener los mejores resultados con la mayor sencillez posible, sobre todo de cara al usuario. Para conseguir estos resultados se ha combinado el uso de JavaBeans en las JSP (para conseguir páginas lo más libres de código posibles y, por tanto, lo más sencillas posibles) con JavaScript (para comprobar los formularios y ver que están libres de errores antes de enviarlos al servidor) y hojas de estilo para aplicar un formato agradable a la vista. En los siguientes párrafos se explicará un poco más en detalle cada una de estas tecnologías y su papel en este proyecto.

De un vistazo, la funcionalidad de la Web a grandes rasgos se puede resumir en el diagrama de flujo de la figura 4.1.

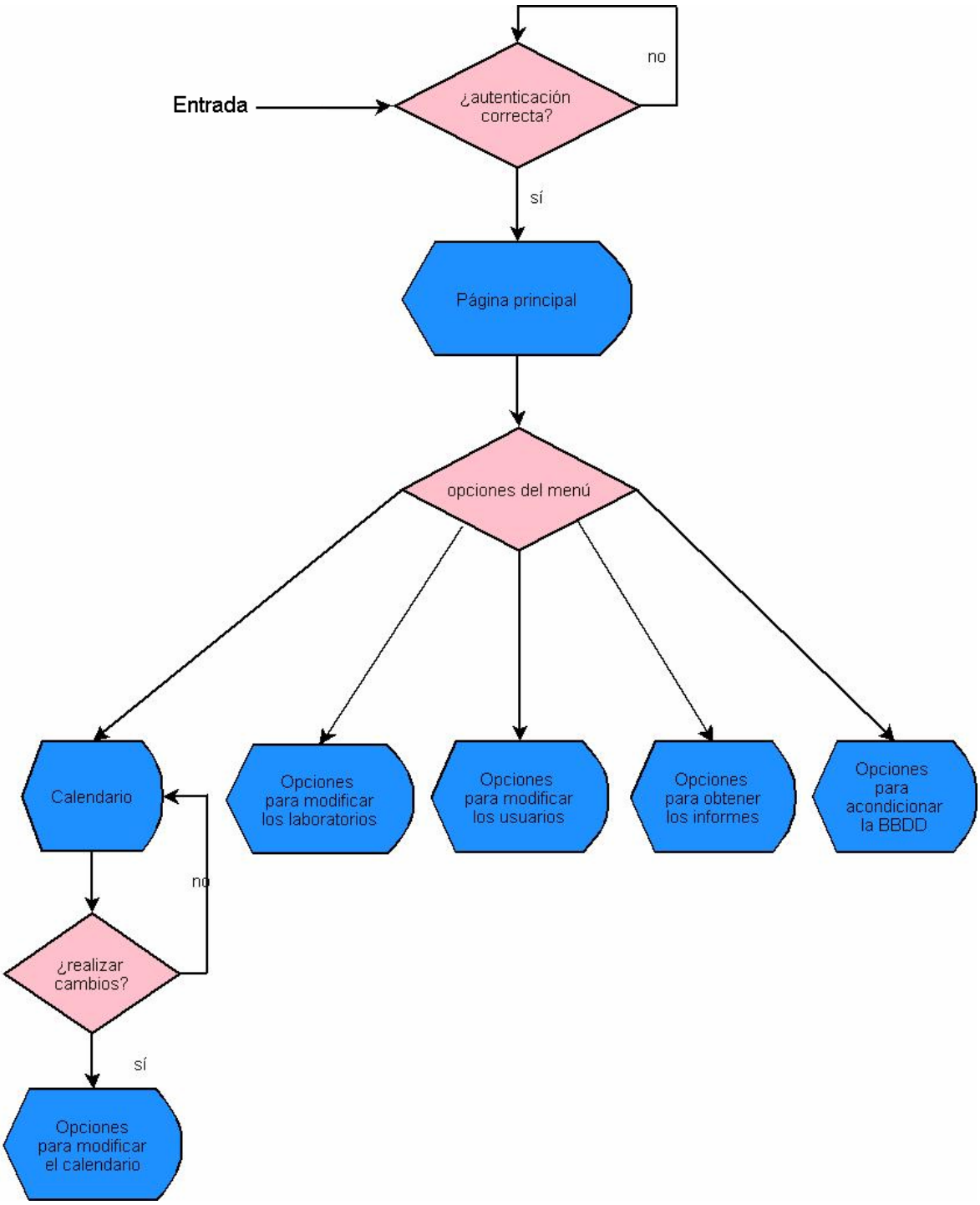

Figura 4.1. Diagrama de flujo general de la página

El funcionamiento bloque por bloque es el siguiente:

- Autenticación: en este punto se pide nombre de usuario y contraseña para dar acceso o no al sitio Web. Se explicará más en detalle en el punto 4.4 de este mismo capítulo.

- Página principal: si la autenticación ha sido correcta se pasa a esta parte, en que se muestra, en una composición de dos marcos HTML la página principal. Uno de ellos (*frameLeyenda.html*) es una barra a la izquierda de la pantalla con todos los enlaces a las acciones que se pueden realizar. La otra (*Principal.html*) es el contenedor donde se mostrarán todos los contenidos de los enlaces de la primera. La primera página que se puede ver muestra por tanto, aparte del título y la barra lateral con los enlaces que se acaba de comentar, que está optimizado para una resolución de 1024x768 y que se mantenga activado JavaScript en el navegador para mantener el correcto funcionamiento.

- Opciones del menú: como se ha visto ya, esta parte está implementada en la propia página principal como un conjunto de enlaces cuyo contenido pasa a volcarse en un contenedor en la página principal al pinchar en ellos. El contenido de cada una de las opciones se explicará en los siguientes apartados de este capítulo. Sólo cabe resaltar una que es ligeramente diferente, el calendario, en el que se muestra una pantalla de bienvenida (*CalendarioBienvenida.jsp*) con el calendario del año actual, y sólo si se marca el botón de “Realizar modificaciones en el calendario” se pasa a ver las opciones que hay para modificar.

Como denominador común en todas las opciones del menú está el uso de los JavaBeans, de JavaScript y de hojas de estilo (estas últimas están presentes en la totalidad de las páginas de las que se compone la aplicación), así como de una página de error que se muestra en caso de ocurrir algún fallo mostrando la causa raíz y la página en que se ha originado. Se trata de *paginaError.jsp*.

En cuanto a los JavaBeans, su papel ha sido fundamental para dar claridad a las páginas JSP, ya que la mayor parte del código ha recaído en ellos dejando para las JSP sólo lo estrictamente necesario para mostrar los contenidos requeridos en cada situación a través de los resultados que ofrecen los JavaBeans. JavaScript también ha sido primordial, ya que la aplicación es, prácticamente un compendio de formularios para llegar a un determinado objetivo. En esas circunstancias lo idóneo es usar JavaScript en todos ellos para asegurar

que los datos que se envían están libres de errores y para informar al usuario en todo momento del punto en qué se ha equivocado. También se ha usado JavaScript para dar claridad a los formularios; en las opciones "Calendario", "Laboratorios" y "Usuarios" hay partes diferenciadas claramente (añadir, modificar, borrar, mover…) y cada una de esas partes es independiente de las demás. Por ello se muestra sólo el título de cada una de esas partes y haciendo clic en su título se ve u oculta la información necesaria para realizar la acción que se desee. Y para dar forma a todo lo que se presenta por pantalla se han usado las hojas de estilo, en este caso una sola (*Estilos.css*) para dar homogeneidad a toda la aplicación.

En los siguientes apartados se mostrará exactamente el papel de cada JavaBean y del código JavaScript usados en cada una de las opciones del menú.

4.2.2 OPCIÓN CALENDARIO

En esta opción intervienen las páginas *Calendarios.jsp, IntroducirFiesta.jsp, MoverFiesta.jsp, BorrarFiesta.jsp, IntroducirFechasEspeciales.jsp, ModificarEspecial.jsp* y *BorrarEspecial.jsp*, y los JavaBeans del paquete calan. La página de inicio es *Calendarios.jsp*, en la que se ve un calendario del año actual en que se marcan en rojo los festivos (los festivos fijos como Navidad, etc. ya aparecen marcados por defecto), en azul los días con horario especial y en negro los restantes. Desde esta página se pueden realizar todas las operaciones de modificación del calendario del año que se trate. Además se incluye un menú desplegable donde se puede cambiar el año que se está modificando hasta dos años posteriores al año actual.

Para conseguir visualizar el calendario antes mencionado son necesarias las siguientes clases de Java (o JavaBeans en este caso): *ArrayVisual.java*, *BeanInicio.java*, *Dia.java*, *DomingoResurreccion.java* y *LeerDias.java*. Y para ver la lista de días festivos o de tramos de horario especial que aparecen en los menús desplegables para modificarlos o borrarlos se tienen las clases *ListaEspeciales.java* y *ListaFiestas.java*. Entrando un poco más en profundidad, *ArrayVisual.java* sirve para tener una lista desde la que sea relativamente fácil sacar por pantalla directamente todos los días del año, aunque internamente, el manejo de los días sea más fácil tal y como se hace en *LeerDias.java*, donde *Dia.java* (como es obvio por el nombre) es la clase que representa a cada día con sus propiedades y *DomingoResurreccion.java* incluye un algoritmo capaz de calcular en

qué fecha cae la semana santa en cualquier año. Esto se usa para marcar directamente los días festivos que son fijos cada año.

Las páginas que aún no se han explicado son *IntroducirFiesta.jsp*, *MoverFiesta.jsp*, *BorrarFiesta.jsp*, *IntroducirFechasEspeciales.jsp*, *ModificarEspecial.jsp* y *BorrarEspecial.jsp*. Todas ellas son simplemente indicaciones de si la operación que se pretendía realizar se ha llevado a cabo con éxito o no (introducir, mover o borrar fiesta o introducir, modificar o borrar tramo de horario especial respectivamente) y son gobernadas por los JavaBeans siguientes: *IntroducirFiesta.java*, *MoverFiesta.java*, *BorrarFiesta.java*, *IntroducirFechasEspeciales.java*, *ModificarEspecial.java* y *BorrarEspecial.java* respectivamente siguiendo el orden de las páginas JSP anteriores.

Y ya sólo queda una clase Java por explicar, *DiaBBDD.java*, cuya función es, junto a *LeerDias.java*, actualizar la base de datos cuando pasa un año. Para entenderlo bien antes hay que explicar un poco la estructura de la base de datos: se guardan tres tablas, *calendario0*, *calendario1* y *calendario2*, donde el primero corresponde al año actual y los dos siguientes a los dos años posteriores, y en cada tabla se almacenan los días festivos y los días con horario especial que se establezcan cada año. De esta forma cada 1 de Enero, que se detecta que ha pasado un año, el que era el año actual (y ocupaba la tabla *calendario0*) ya no es útil almacenarlo y se debe borrar desplazándose los dos años posteriores hacia las primeras posiciones y dejando el tercero vacío.

En un diagrama de flujo muy simplificado quedaría como en la figura 4.2.

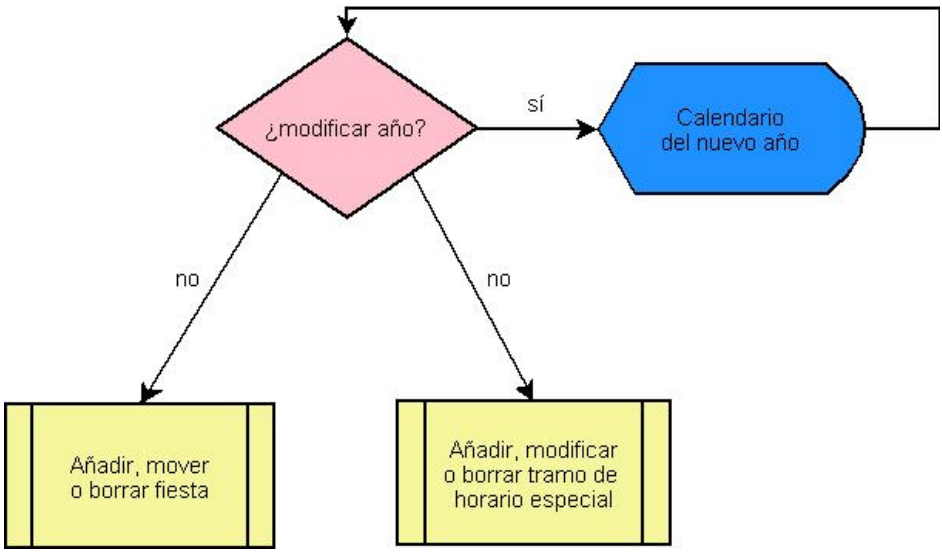

Figura 4.2. Diagrama de flujo de la opción de modificar calendarios

El código JavaScript ha estado encaminado en esta parte a asegurar que no quedan campos vacíos en los formularios a la hora de introducir fiestas o tramos de horario especial, que no se puedan meter horas de inicio posteriores a las de fin ni fechas de inicio posteriores a las de fin, que no se puedan poner días inválidos en un mes (por ejemplo 30 de Febrero) y en todos los casos a confirmar las operaciones antes de que se lleven a cabo definitivamente. Todas estas acciones están programadas en un archivo externo, dentro de la capeta */WebContent/javascript* llamado *Calendarios.js*. También se podría haber incluido el código JavaScript directamente en las páginas JSP, pero la intención siempre es dejarlas lo más limpias posible para facilitar su comprensión, por lo que la mejor opción es dejar todo el código en el archivo externo.

4.2.3 OPCIÓN LABORATORIOS

En esta opción intervienen las páginas *Laboratorios.jsp*, *IntroducirLab.jsp*, *ModificarLab.jsp* y *BorrarLab.jsp*, y los JavaBeans del paquete *listalabs*.

La estructura es bastante similar a la opción presentada anteriormente de los calendarios y también lo será para la opción de los usuarios. Aquí, la página desde la que se pueden realizar todas las operaciones es *Laboratorios.jsp* y las demás son indicaciones del resultado de la operación que se trate, es decir, añadir, modificar o borrar laboratorio respectivamente.

En la figura 4.3 se puede apreciar, al igual que se hizo en el caso de la opción de los calendarios, a grandes rasgos el funcionamiento de esta opción, sin entrar en el detalle de las clases Java o las páginas JSP que intervienen.

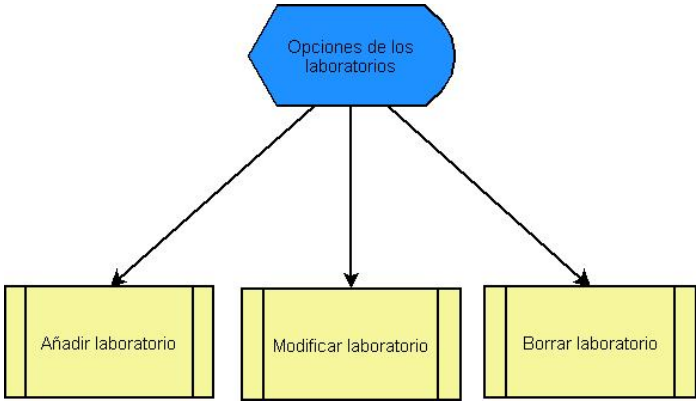

Figura 4.3. Diagrama de flujo de la opción de los laboratorios

También aquí se usan como apoyo fundamental las clases del paquete que contiene los JavaBeans y que en este caso se llama *listalabs*. Dichas clases son *Laboratorio.java*, *ListaLabs.java*, *Introducir.java*, *Modificar.java* y *Borrar.java*. La clase *Laboratorio.java* se usa para instanciar objetos con las propiedades de los laboratorio, que serán dos: identificador y descripción. *ListaLabs.java* es una clase que permite obtener la lista de los laboratorios incluidos en la base de datos con las propiedades antes descritas, es decir, es una lista de esos objetos. Su utilidad se ve a la hora de mostrar por pantalla todos los laboratorios, para modificarlos, borrarlos, o, como se verá en el punto 4.2.5, para obtener un informe con todos los laboratorios. En cuanto a *Introducir.java*, *Modificar.java* y *Borrar.java*, hay que decir que son JavaBeans bastante bien autodefinidos por su nombre. Sirven en este caso para introducir, modificar y borrar laboratorios de la base de datos respectivamente, y se emplean en las páginas *IntroducirLab.jsp*, *ModificarLab.jsp* y *BorrarLab.jsp* respectivamente, valga la redundancia.

Nuevamente, el código JavaScript empleado en esta parte ha estado encaminado en primer lugar a que en los formularios usados no puedan dejarse campos en blanco ni se introduzcan valores inválidos y en segundo lugar a pedir la confirmación antes de realizar la operación solicitada de forma definitiva. La novedad respecto a los calendarios es el uso de JavaScript para visualizar la descripción de los laboratorios que se seleccionan a medida que se van seleccionando. Todo ello está programado en el archivo */WebContent/javaScript/Laboratorios.js.*

4.2.4 OPCIÓN USUARIOS

En esta opción intervienen las páginas AsignarLaboratorioUsuario.jsp, BorrarLaboratorioUsuario.jsp, BorrarUsuario.jsp, IntroducirUsuario.jsp, ModificarDatosUsuario.jsp, ModificarLaboratorioUsuario.jsp, ModificarUsuarios.jsp y Usuarios.jsp, y los JavaBeans del paquete listasuarios.

Aquí la forma de operar es básicamente la misma que para la opción de los laboratorios con la salvedad de que, al tener más detalles, se complica un poco. La página de partida es *Usuarios.jsp*, desde la que se pueden realizar o iniciar todas las acciones. En todas esas acciones siempre está presente la clase *Usuario.java*, que sirve para instanciar objetos con todas las propiedades necesarias para manejar los datos de los usuarios.

La primera acción que se puede realizar es introducir a un usuario nuevo, acción que se realiza gracias a la clase *IntroducirUsr.java* y que lleva a la página *IntroducirUsuario.jsp*, donde se indica si se ha llevado a cabo con éxito la operación. La otra acción que se puede llevar a cabo desde la página principal es borrar a un usuario existente, acción que lleva a la página *BorrarUsuario.jsp*, donde se indica si se ha borrado con éxito. Para ello se hace uso de las clases *BorrarUsuario.java* y *ListaUsuarios.java*, donde la primera se encarga de realizar el borrado efectivo del usuario seleccionado y la segunda, de mostrar por pantalla una lista detallada de los usuarios existentes en el sistema.

Por último, se pueden realizar modificaciones sobre un usuario determinado, para lo cual se muestra en la página *Usuarios.jsp* una lista de todos los usuarios ordenados por el apellido gracias a la clase *ListaUsuarios.java*. Una vez seleccionado un usuario para modificar, hay que tener en cuenta varias posibles modificaciones: modificar datos personales, añadir laboratorios a los que podrá acceder, modificar los días u horas en las que tiene acceso a alguno en concreto o borrar alguno de los laboratorios a los que puede acceder. Todo esto se muestra en la página *ModificarUsuarios.jsp*. Las opciones son:

- Modificar datos personales: se hace uso de la página *ModificarDatosUsuario.jsp* que emplea la clase *ModificarUsr.java* para realizar la modificación y la clase *UsuarioSeleccionado.java* para ver por pantalla antes de realizar las modificaciones los datos antiguos.

- Añadir un laboratorio a su lista personal de laboratorios a los que tiene acceso: se usa la página *AsignarLaboratorioUsuario.jsp*, que a su vez emplea el JavaBean *AsignarLaboratorioUsuario.java* para completar la asignación.

- Modificar las horas o días de acceso de uno de los laboratorios a los que ya tiene acceso: se hace uso de la página *ModificarLaboratorioUsuario.jsp*, que a su vez emplea las clases *ModificarLabUsr.java*, *LabAmpliado.java* y *ListaLabsAmpliada.java*. La primera realiza la modificación, la segunda permite tener un objeto de tipo laboratorio con más atributos para poder cambiar dinámicamente la Web al seleccionar uno de ellos y la tercera mantiene una lista de los mismos, que es lo que se muestra por pantalla para seleccionar uno y modificarlo.

- Borrar un laboratorio de su lista personal de laboratorios a los que tiene acceso: se usa la página *BorrarLaboratorioUsuario.jsp*, que a su vez emplea la clase *BorrarLaboratorioUsuario.java* para realizar el borrado.

También la parte de JavaScript es aquí un poco más complicada, ya que hay más cosas que controlar. Por una parte se sigue controlando lo mismo que para los laboratorios, que no haya campos vacíos en los formularios y pedir siempre confirmación para realizar definitivamente cualquier operación. También se obtienen datos de los usuarios dinámicamente para mostrarlos por pantalla a medida que se seleccionan y se muestran también datos de los laboratorios que tienen asignados al modificar alguno de los que tienen. Además de esto se hacen otro tipo de comprobaciones como que el NIF que se introduzca sea correcto, y las fechas válidas. Y otro tipo de acciones como deshabilitar los campos innecesarios en ciertos tipos de usuarios para imposibilitar que se introduzcan datos inválidos o que a posteriori no serían tenidos en cuenta. Todo ello está programado en el archivo */WebContent/javaScript/Usuarios.js.*

En la figura 4.4 se puede ver de una forma muy simplificada el funcionamiento de la opción de los usuarios. Aquí, la caja de "Modificar usuario", lógicamente se expandiría en las opciones ya vistas.

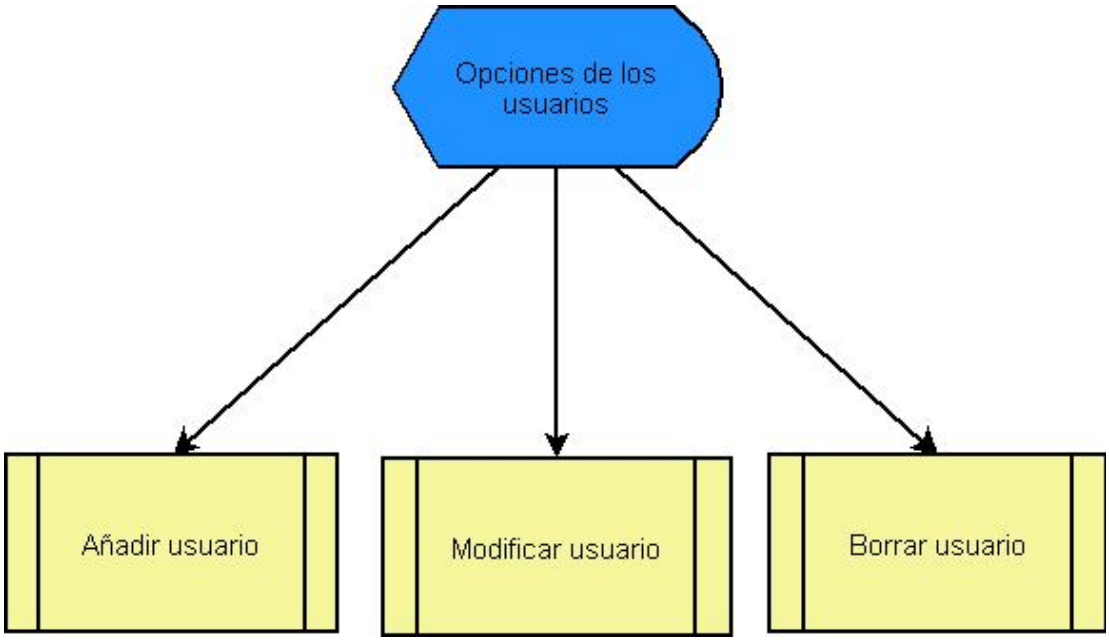

Figura 4.4. Diagrama de flujo de la opción de los usuarios

4.2.5 OPCIÓN INFORMES

4.2.5.1 Funcionamiento básico

En esta opción intervienen las páginas *Informes.jsp*, *OpcionesFestivos.jsp*, *OpcionesRegistro.jsp*, *OpcionesUsuarios.jsp*, *PrePDFFestivos.jsp*, *PrePDFRegistro.jsp*, *PrePDFLaboratorios.jsp* y *PrePDFUsuarios.jsp* y las clases del paquete informes. Este paquete está formado por clases auxiliares necesarias para generar los informes, JavaBeans de apoyo a las páginas JSP y un servlet que es el que se encarga de generar todos los informes en formato PDF. Es necesario usarlo, como ya se dijo en el capítulo 2, porque no se puede hacer cualquier cosa sólo con JSP. Estas son muy útiles para trabajar con datos de tipo texto, pero no así con datos de tipo binario como el formato PDF.

La página principal de esta opción es *Informes.jsp*, desde la cual se accede a un listado de enlaces para obtener informes sobre laboratorios, calendarios, usuarios o el registro de accesos a los laboratorios. Estos enlaces llevan a *PrePDFLaboratorios.jsp*, *OpcionesFestivos.jsp*, *OpcionesUsuarios.jsp* y *OpcionesRegistro.jsp* respectivamente.

La primera página, *PrePDFLaboratorios.jsp*, usa las clases *GeneraXMLListaLaboratorios.java* e *InformePDF.java*, que como ya se ha dicho es el *servlet* que interviene en la generación de todos los documentos en formato PDF. La clase *GeneraXMLListaLaboratorios.java* es la encargada de generar el documento en formato XML que toma el *servlet* como entrada. Este documento, en este caso estará formado por una lista de todos los laboratorios del Departamento de Tecnología Electrónica con su descripción. En cuanto al *servlet*, es importante recordar que debe estar definido en el archivo */WebContent/WEB-INF/web.xml* para que se haga la llamada correctamente.

La página *OpcionesFestivos.jsp* muestra un menú desplegable para seleccionar de qué año se desea obtener el informe de los días festivos y de horario especial, y de allí se envía al *servlet* que genera el documento en formato PDF a través de las clases *BeanInformeFestivos.java*, en el que se guarda el año seleccionado anteriormente, y *GeneraXMLListaFestivos.java*, que genera el documento en formato XML que toma el *servlet* como entrada.

OpcionesUsuarios.jsp muestra una serie de opciones para seleccionar un conjunto de usuarios con unas ciertas características (por defecto se muestran todos los usuarios). A partir de aquí el esquema de funcionamiento es casi igual al de los calendarios: se envían

las opciones seleccionadas al *servlet* que genera el documento PDF a través de las clases *BeanInformeUsuarios.java*, que almacena todas esas opciones marcadas, y *GeneraXMLListaUsuarios.java*, que genera el documento en formato XML que toma el *servlet* como entrada. Aquí la única diferencia es que se usa una clase auxiliar llamada *DatosUsr.java*, para almacenar todos los datos de un usuario (que se encuentran en varias tablas en la base de datos) como apoyo de *GeneraXMLListaUsuarios.java*.

En cuanto a *OpcionesRegistro.jsp*, es la página encargada de mostrar las opciones para seleccionar aquellos registros que cumplan una serie de características. Al igual que para los usuarios, por defecto están marcados todos los registros para sacar un informe completo. Por lo demás el funcionamiento de todo se repite: se envían las opciones seleccionadas al *servlet* que genera el documento PDF a través de las clases *BeanInformeRegistro.java*, que almacena todas esas opciones marcadas, y *GeneraXMLListaRegistro.java*, que genera el documento XML que toma el *servlet* como entrada. Al igual que en los usuarios, para los registros también es necesaria una clase auxiliar, llamada *Acceso.java*, que sirve para almacenar todos los datos de un acceso y que sirve de apoyo a *GeneraXMLListaRegistro.java*.

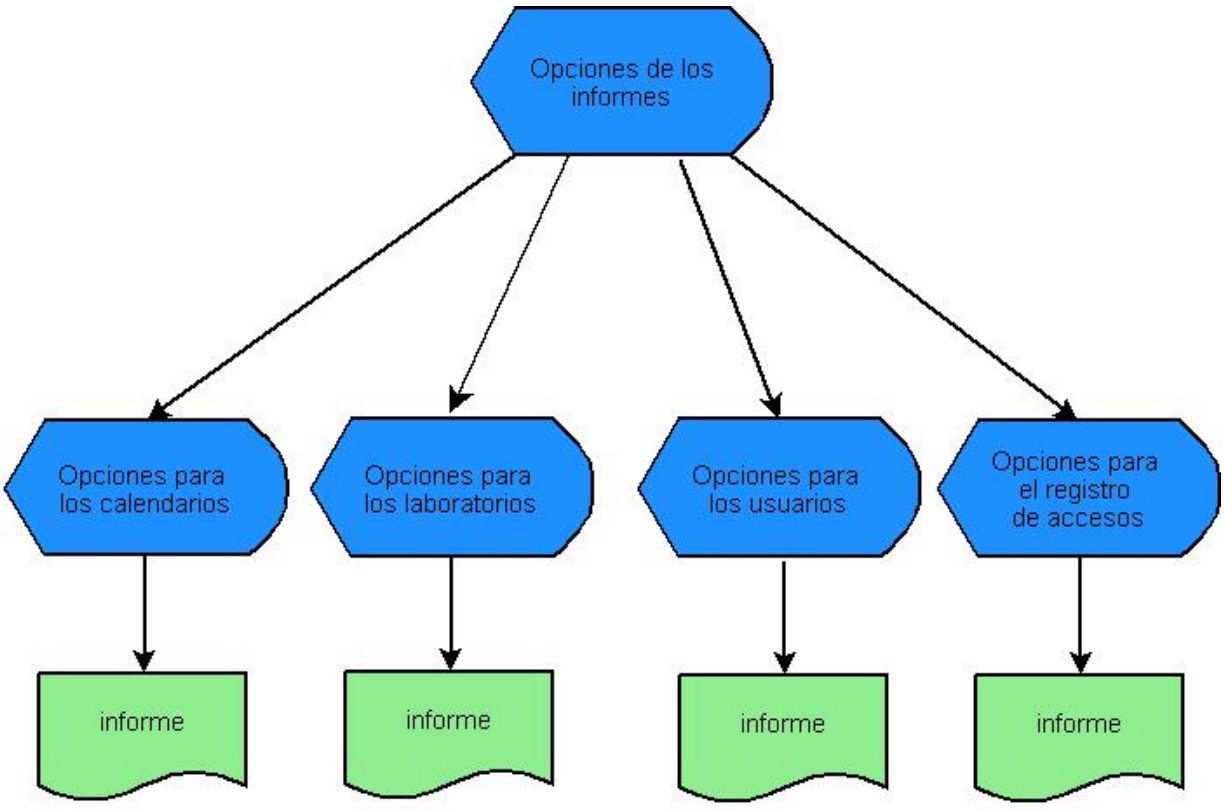

Figura 4.5. Diagrama de flujo de la opción del registro de accesos

El uso de JavaScript en este caso no está encaminado a no dejar campos vacíos, ya que todas las opciones que se muestran son menús desplegables o botones de tipo radio, y si no se marca nada, por defecto se toma la opción que lo seleccione todo. Tampoco se pide confirmación porque estas operaciones no son críticas como en el caso de modificar laboratorios, calendario o usuarios; aquí sólo se sirven informes. En lugar de todo eso se comprueba que no se meten horas de inicio posteriores a las de fin, fechas de inicio posteriores a las de fin, horas o fechas inválidas. El único sitio donde se comprueba que se ha seleccionado algo es en las opciones de los calendarios, donde se ve si se ha seleccionado o no algún año. Otra acción importante que se realiza con JavaScript es habilitar o deshabilitar una zona de las opciones del registro de accesos, según la opción que se marque en un botón de tipo radio, donde se selecciona si se desea ver los accesos de un cierto espacio de tiempo (por ejemplo desde el lunes al miércoles a cualquier hora) o un rango de horas dentro de un cierto intervalo de tiempo (por ejemplo de las 10 a las 12 de una cierta semana). Todo ello está programado en el archivo */WebContent/javaScript/Informes.js.*

Un resumen de todo lo explicado de forma esquemática está en la figura 4.5.

4.2.5.2 Consideraciones adicionales

Para asegurar la compatibilidad en Windows y en Linux en cuanto a la codificación usada se han tenido que poner varias medidas:

- Usar un filtro, *UTF8Filter.java*, que asegure que todas las comunicaciones a través de los formularios de las páginas se realicen usando la codificación UTF-8 (*8-bit Unicode Transformation Format*). Esta codificación asegura que se va a poder visualizar correctamente cualquier símbolo de cualquier lenguaje conocido en el mundo. A través de este filtro se puede asegurar que los datos que van a ser convertidos en un documento en formato PDF llegarán con la codificación adecuada. Para que realice su función debe ser definido al inicio del archivo */WebContent/WEB-INF/web.xml* de la aplicación Web, indicando como patrón a filtrar "/*" para que realice el filtrado de todos los datos.

- Crear la base de datos con la codificación UTF-8, para que los datos sean almacenados siempre en esta codificación.

- Asegurarse de usar el entorno de desarrollo en codificación UTF-8, para que los valores de las cadenas que se meten directamente en las bases de datos estén codificadas en UTF-8.

Es muy importante la codificación en la aplicación Web, ya que si no se asegurase seguir este estándar, según se instalase en un sistema operativo u otro, se asumiría la codificación del sistema en el que se instale. Es decir, si se instala en Windows, la aplicación tomaría como codificación a seguir la CP1252 (que es una variante de Microsoft del *Latin 1* ó ISO 8859-1) y en Linux la UTF-8. De esta forma, si el archivo de texto usado para crear la base de datos está en una codificación, la aplicación Web entiende que está en la codificación del sistema operativo en que esté, y estas dos codificaciones no coinciden nunca se tendría el control de cómo se representan los caracteres. Habría que crear las bases de datos específicamente para cada sistema operativo. Siguiendo estas medidas se asegura que tanto para Windows como para Linux se va a usar siempre la codificación UTF-8 en todo momento y por lo tanto siempre van a aparecer caracteres válidos en los documentos PDF.

4.2.6 OPCIÓN ACONDICIONAR BBDD

En esta opción intervienen las páginas *AcondicionarBBDD.jsp* y *AcondicionarRegistros.jsp* y el único JavaBean necesario para realizar las operaciones requeridas, *EliminarRegistros.java*, del paquete *bbdd*.

Es una opción muy sencilla, pero no por ello menos importante, ya que es la encargada de limpiar de la base de datos los registros más antiguos a petición del técnico. Para ello se muestran en la página *AcondicionarBBDD.jsp* dos menús desplegables con el mes y el día y una caja de texto para introducir el año a partir del cual se quiere realizar la eliminación de registros. Desde aquí, previa comprobación de que los datos introducidos no son inválidos y confirmación se procede a la eliminación. Esto se consigue con el JavaBean *EliminarRegistros.java* y se envía el resultado de la operación a la página *AcondicionarRegistros.jsp*, donde se indica si se ha llevado a cabo con éxito o no.

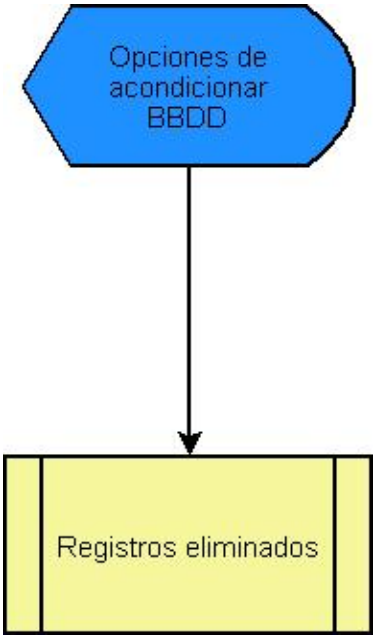

Figura 4.6. Diagrama de flujo de la opción de eliminación de registros antiguos

El uso de JavaScript en este caso está restringido a, como ya se ha dicho, comprobar que se meten datos válidos y a pedir confirmación para realizar efectivamente la eliminación de registros antiguos. Esto está implementado en el archivo */WebContent/javaScript/AcondicionarBBDD.js.*

La representación del diagrama de flujo, de forma simplificada, que representa todo esto está en la figura 4.6.

4.3 BASE DE DATOS

4.3.1 ESTRUCTURA BÁSICA

El sistema implementado necesita de una base de datos llamada *accesolabdte*, con doce tablas, estructuradas de la siguiente forma:

- 3 tablas para guardar cada una un año del calendario, llamadas *calendario0*, *calendario1* y *calendario2*. La primera almacena los días festivos y de horario especial del año actual, y las dos siguientes las de los dos años posteriores.
- 1 tabla para guardar los usuarios con sus datos llamada *usuarios*.

- 1 tabla para guardar los laboratorios con sus datos llamada *laboratoriosdte*.
- 1 tabla para guardar los tipos posibles de usuarios llamada *tiposusuarios.*
- 2 tablas para relacionar las 3 anteriores. Una llamada *usuariosalaboratoriosdte* encargada de relacionar la tabla *usuarios* con la tabla *laboratoriosdte* y otra llamada *usuariosatiposusuarios* encargada de relacionar la tabla *usuarios* con la tabla *tiposusuarios*. A través de ellas, un cambio en una de las tablas que relacionan se refleja en la otra. Por ejemplo, si se borra un laboratorio se debe borrar la entrada de la tabla que relaciona un usuario con ese laboratorio.
- 1 tabla para registrar todos los intentos de acceso que se produzcan llamada *registroaccesos.*
- 3 tablas para habilitar y gestionar la autenticación del técnico encargado de manejar la gestión de los laboratorios del Departamento de Tecnología Electrónica: *usuariosTomcat*, que contiene los posibles usuarios, *roles*, que contiene los posibles roles de los usuarios y *usuariosTomcat_roles*, que contiene la relación entre las dos tablas anteriores.

Todas estas tablas, la relación entre ellas y los campos de los que está formada cada una de las tablas estás representadas de forma esquemática en la figura 4.7, que se explica a continuación.

En la parte superior de la figura se pueden ver las tablas de los calendarios y la correspondiente al registro de accesos, que no tienen relación con las demás tablas. En la parte central están representadas la tabla de los usuarios, la de los tipos de usuarios y la de los laboratorios, y la relación que las une entre sí a través de otras dos tablas. Además, en esas dos tablas de relaciones se definen algunas características de los usuarios intrínsecas a la relación de las tablas que se vinculan. Por último, en la parte inferior se representan las tres tablas que permiten gestionar la autenticación.

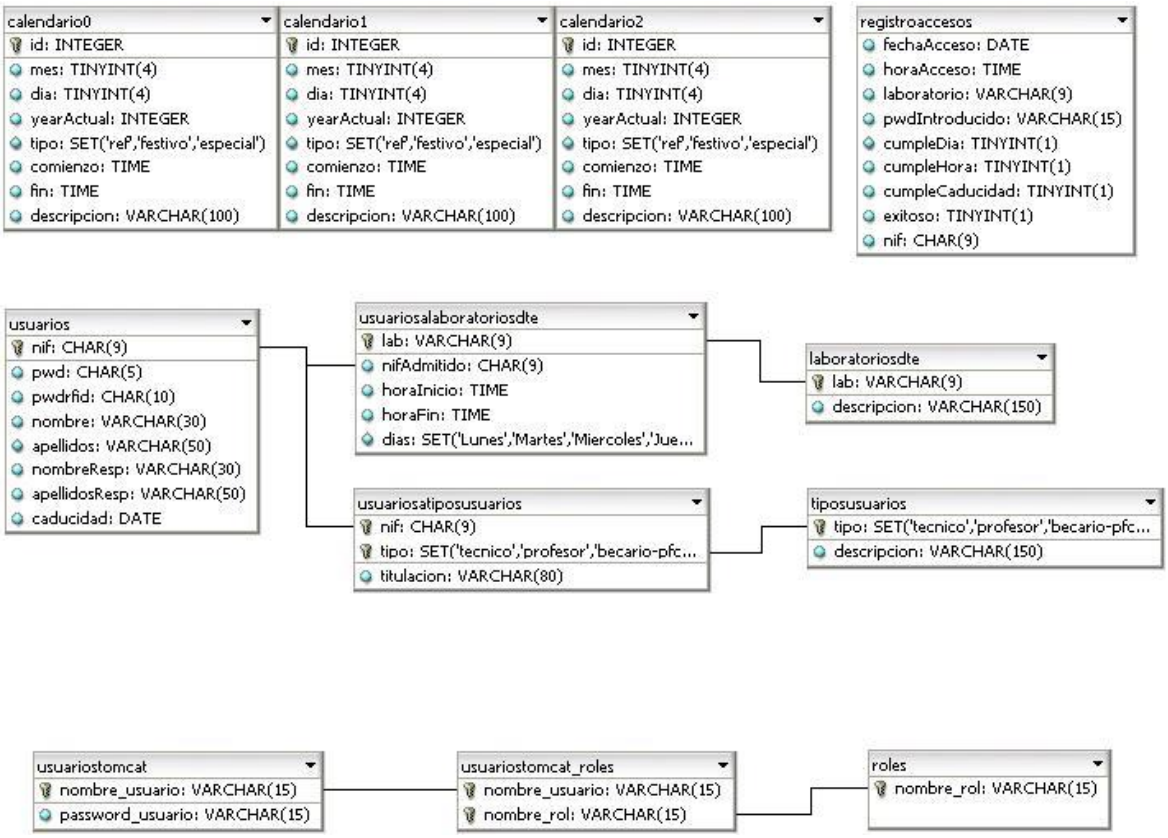

Figura 4.7. Tablas de la base de datos

Se puede apreciar (figura 4.7) que la estructura de las tablas correspondientes a los calendarios es exactamente la misma y que la tabla que contiene el registro no tiene un índice (marcado en la figura con una pequeña llave al lado del campo índice). Esto es así porque en realidad, las búsquedas que se produzcan en esa tabla a la hora de recuperar datos no se van a centrar en uno o dos campos, sino que los criterios de búsqueda pueden comprender a muchos campos, e incluso a casi todos, por lo que se pierde la utilidad de tener un índice. En cambio, en las tablas por medio de las cuales se relacionan otras dos tablas, se observa que hay dos índices, uno para cada relación.

En general, en todas las tablas que conforman las bases de datos se usan unos ciertos índices en función de la búsqueda que luego se vaya a realizar en las bases de datos. Todos los índices y su descripción están resumidos en la tabla 4.1.

Tabla	Índice	Descripción
calendario0	id	Es simplemente un identificador numérico (de tipo INT) con autoincremento
calendario1	id	Es simplemente un identificador numérico (de tipo INT) con autoincremento
calendario2	id	Es simplemente un identificador numérico (de tipo INT) con autoincremento
usuarios	nif	El índice es el NIF del usuario, que siempre es único (VARCHAR)
laboratoriosdte	lab	El índice es el propio nombre del laboratorio, que siempre es único (VARCHAR)
tiposusuarios	tipo	El índice es el propio tipo (en principio sólo serán 4), de tipo VARCHAR
usuariosalaboratoriosdte	lab, nifAdmitido	Hay 2 índices que producirán el efecto cascada al modificar un registro de las tablas que relaciona (ambos de tipo VARCHAR)
usuariosatiposusuarios	nif, tipo	Hay 2 índices que producirán el efecto cascada al modificar un registro de las tablas que relaciona (ambos de tipo VARCHAR)
registroAccesos	-	No tiene índice, ya que las búsquedas se realizarán por fecha y esta se puede repetir, por lo que no puede ser índice. Poner otro índice no tiene sentido
usuariosTomcat	nombre_usuario	El índice es el nombre, que debe ser único (VARCHAR)
roles	nombre_rol	El índice es el rol, que debe ser único (VARCHAR)
usuariosTomcat_roles	nombre_usuario, nombre_rol	Hay 2 índices que indican la relación de las tablas anteriores (de tipo VARCHAR)

Tabla 4.1. Resumen de los índices de las tablas y su descripción

4.3.2 CONSIDERACIONES ADICIONALES

Para asegurar un funcionamiento óptimo de la base de datos se ha empleado el formato de tablas InnoDB, que permite tener claves foráneas en la tabla, de manera que si se borra un campo referenciado en una de las tablas a través de una clave foránea se borrarán todas las referencias. Por ejemplo, si se borra un laboratorio y éste está relacionado con un usuario que tenga acceso al mismo, se debe borrar también de la lista de laboratorios a los que tiene acceso ese usuario. Por lo tanto, debe borrarse ese registro de la tabla correspondiente. Con el tipo de tablas InnoDB es posible definir las tablas de manera que se realicen esas actualizaciones.

Por otra parte, y como ya se comentó en el punto 4.2.5.2, es necesario definir la base de datos de manera que tome como codificación estándar para todas sus tablas la codificación UTF-8, que asegura que se puede almacenar cualquier símbolo de cualquier lenguaje conocido en el mundo. En el caso de este proyecto interesa en principio sólo que las palabras con tildes se almacenen correctamente, pero lógicamente el uso de un estándar siempre es recomendable.

4.4 AUTENTICACIÓN Y SEGURIDAD

4.4.1 AUTENTICACIÓN

Para implementar la autenticación en el sitio Web, que sólo debe poder consultar y manipular el técnico, se ha empleado la base de datos del proyecto, y en concreto tres tablas que son *usuariosTomcat, roles* y *usuariosTomcat_roles.* En ellas se han definido los posibles usuarios, los posible roles existentes y la relación entre ellos, de forma que sólo un usuario con una determinada contraseña podrá acceder a los contenidos del *Security Realm* (o dominio de seguridad, como se podría traducir en español). Este dominio de seguridad es una carpeta del servidor Tomcat con acceso restringido.

Así pues, con las tablas creadas no queda más que modificar un archivo de la configuración del Tomcat y otro del dominio que queremos proteger:

- */Tomcat 5.5/conf/server.xml*: En el que hay que incluir la información relativa al tipo de base de datos que se usa (MySQL), el usuario de la base de datos, su contraseña, y las tablas que se usan para gestionar los usuarios del dominio que

se pretende proteger. Este fragmento de código es el que se muestra en la figura 4.8. Nótese como en los comentarios del mismo se indica la configuración inicial, con la base de datos instalada en el ordenador local, y cómo se debería cambiar este archivo en caso de tener la base de datos alojada en otro ordenador. Todos los cambios que se realizaran en este fichero se deberían realizar al mismo tiempo en el archivo de configuración (*configuracion.xml*) situado en la carpeta *configuracion* de la carpeta que contiene la aplicación Web. No obstante, estos cambios se comentan en más detalle en el capítulo 5, ya que son más propios de la instalación.

```
<!--
  CONTENIDO ORIGINAL DEL REALM. En caso de malfuncionamiento, esta configuración con la base
 de datos en el ordenador local siempre funciona.
**********************************************************************************************************************************
    <Realm className="org.apache.catalina.realm.JDBCRealm" connectionName="root"
connectionPassword="pelotero" connectionURL="jdbc:mysql://localhost/accesolabdte" debug="99"
driverName="org.gjt.mm.mysql.Driver" roleNameCol="nombre_rol" userCredCol="password_usuario"
userNameCol="nombre_usuario" userRoleTable="usuariosTomcat_roles" userTable="usuariosTomcat"/>
**********************************************************************************************************************************
    Para cambiar la contraseña de la base de datos modificar "connectionPassword"
    Para cambiar el usuario de la base de datos modificar "connectionName"
    Para cambiar el protocolo de la base de datos modificar "jdbc:mysql" por el conector y el sistema gestor
    de bases de datos correspondiente
    Para cambiar el ordenador donde reside la base de datos modificar "localhost" por el nombre o dirección IP
    que corresponda
    Para cambiar el driver de la base de datos modificar "driverName"
-->
        <Realm className="org.apache.catalina.realm.JDBCRealm" connectionName="root"
connectionPassword="pelotero" connectionURL="jdbc:mysql://localhost/accesolabdte" debug="99"
driverName="org.gjt.mm.mysql.Driver" roleNameCol="nombre_rol" userCredCol="password_usuario"
userNameCol="nombre_usuario" userRoleTable="usuariosTomcat_roles" userTable="usuariosTomcat"/>
```

Figura 4.8. Fragmento del archivo server.xml relativo a los dominios de seguridad

- */WebContent/WEB-INF/web.xml*: En el que hay que añadir la información necesaria para definir la zona con seguridad, los métodos por los que no se puede acceder a ella, el tipo de seguridad que se implementa y determinar qué roles son los que tienen acceso a ella. Para ello se indica que se va a proteger toda la carpeta, de manera que ningún recurso de la aplicación Web sea accesible, que no se puede acceder ni por el método POST ni por el método GET, y se deja sólo a *admin* como tipo de usuario autorizado a entrar. Además se indica el valor de seguridad en el transporte y el método de autenticación que se va a emplear. Los posibles valores de seguridad en el transporte son NONE (ningún tipo de cifrado), CONFIDENTIAL (usando comunicación SSL) o INTEGRAL (significa que, aunque alguien pueda ver lo que se envía, se

garantiza que no se cambia por el camino). El que se ha usado es CONFIDENTIAL, ya que lo más interesante para esta aplicación es tener una comunicación segura. En cuanto a los métodos de autenticación disponibles son los siguientes: BASIC (salta una ventana en la que pregunta usuario y contraseña), FORM (igual que BASIC, con la diferencia que se integra en la aplicación con el formato que se quiera), CLIENT-CERT (sólo ve el certificado del cliente) y DIGEST (algo más avanzado que BASIC, ya que la clave viaja encriptada). En este caso se ha usado el método BASIC, ya que la única diferencia con respecto a DIGEST es que la clave no viaja encriptada, cosa que se soluciona al usar comunicación SSL, por lo que la combinación de transporte CONFIDENTIAL con método de autenticación BASIC es la más acertada.

4.4.2 SEGURIDAD

El sistema completa su seguridad con el uso de tres certificados digitales para así implementar una comunicación SSL con todas las garantías que se requerían en el punto 2.18.1 de autenticidad, confidencialidad, integridad y no repudio.

Todos esos certificados se han creado usando la herramienta OpenSSL explicada en el punto 2.19 y se les ha dado un periodo de validez de 10 años, ya que no se va a pagar por ello a ninguna empresa reconocida (en ese caso estos certificados serían bastante caros). El orden es el siguiente:

- Primero se debe crear un certificado de autoridad certificadora; esta será la que firme los certificados de cliente y servidor y los valide. Este certificado se debe importar en el almacén de claves del servidor para que posteriormente se reconozcan como válidos los certificados firmados por la autoridad certificadora.

- Con la autoridad certificadora creada ya se puede crear un certificado de servidor, firmarlo con ella, e importarlo en el almacén de claves del servidor.

- Por último se crea un certificado de cliente, se firma con la autoridad certificadora y se importa al almacén de claves del servidor para que posteriormente reconozca al cliente. Ese certificado se pasa a un formato válido para que cualquier navegador lo importe, como es pkcs12, y será este archivo el

que se lleve al cliente para importarlo en el navegador que se use. De esta forma el cliente se autentica de dos formas, por una parte con el mecanismo antes visto en el punto 4.4.1 y por otra, con el certificado digital, mucho más difícil de suplantar.

Con todos estos elementos de seguridad el sistema queda blindado de varias formas y, por lo tanto, a salvo de intrusiones y escuchas ilícitas.

CAPÍTULO 5: INSTALACIÓN Y MANUAL DE USUARIO

5.1 INSTALACIÓN

5.1.1 CONSIDERACIONES PREVIAS

Se ha probado la instalación en varios sistemas operativos, con varias combinaciones posibles de cliente y servidor, a saber:

- Windows 98 Segunda Edición
- Windows 2000 Profesional
- Windows XP Tablet PC Edition 2005 Versión 2002 Service Pack 2
- Windows XP Profesional Versión 2002 Service Pack 2
- Windows XP Profesional x64 Edition Versión 2003 Service Pack 1
- Vesión beta de Windows Vista
- Ubuntu Linux 6.06 Dapper Drake para procesador Intel de 32 bits
- Ubuntu Linux 6.06 Dapper Drake para procesador AMD de 64 bits

Dado que las instalaciones en Ubuntu se realizan (salvo el Tomcat) automáticamente a través de la conexión a Internet no hay motivo para tener las aplicaciones que se necesitan almacenadas en ninguna parte. Es por esto que no se proveen en el CD **(***Compact Disk***)** estas aplicaciones en su versión para Linux. Sí se proveen en cambio para Windows, aunque es de esperar que cualquier versión que se descargue posterior a las provistas también funcione correctamente.

La instalación para otras distribuciones de Linux es similar, pero dado que se ha trabajado siempre con Ubuntu, siempre se va a referir en este capítulo a la instalación en Ubuntu.

La instalación del servidor se ha separado en varias partes, ya que, según se ha hecho la aplicación, se puede instalar la base de datos en un ordenador, el sitio Web en otro y el servicio Web en otro si se desea.

5.1.2 ORDENADOR LOCAL DEL LABORATORIO

La instalación, paso a paso, del sistema en el ordenador que realiza la apertura de la puerta es la que sigue:

- Instalar el JRE (*Java Runtime Environment*) al menos de la versión jre1.5.0_06. En el caso de Windows todo se reduce a ejecutar el archivo *Instalación sistema/Instalación Windows/ jre-1_5_0_08-windows-i586-p.exe* del CD. En el caso de Ubuntu hay que ejecutar en una terminal las siguientes instrucciones:
    - sudo aptitude install sun-java5-jre
    - sudo update-alternatives --config java y elegir la opción 3, es decir la nueva instalación.
- Crear las siguientes variables de entorno:
- *JAVA_HOME*: indica el directorio de instalación de la máquina virtual de Java. En Windows habitualmente se encuentra en *C:\Program Files\Java\jxxy.y.y_yy*, donde "*xx*" varía según sea *jre* o *jdk* y "*y.y.y_yy*" varía según la versión. En Ubuntu se encuentra en */usr/lib/jvm/java-a.a.a-sun-b.b.b.bb*, donde *a.a.a-sun-b.b.b.bb* indican la versión.
- *JRE_HOME*: indica la ubicación del entorno de ejecución de Java. Se encuentra en *$JAVA_HOME/jre*, donde con *$JAVA_HOME* en Linux se especifica el directorio al que apunta la variable de entorno *JAVA_HOME*. En el caso de Windows sería *%JAVA_HOME%*.

  Para ello en Windows se accede a propiedades de mi PC, opciones avanzadas, variables de entorno y se indican en cada caso los directorios donde se encuentran ubicadas. En Linux hay que editar el archivo */home/ USUARIO/.bashrc*, donde *USUARIO* identifica el directorio de trabajo de la sesión que se quiere habilitar para trabajar como cliente y escribir al final del archivo:
    - *export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun-1.5.0.06*
    - *export JRE_HOME=${JAVA_HOME}/jre*
    - *export CATALINA_HOME=/usr/local/apache-tomcat-5.5.17*

  Donde pueden variar los números de las versiones de Java y del Tomcat, según la que se instale.

- Copiar la librería RXTXcomm.jar en el directorio $JAVA_HOME/jre/lib/ext tanto para Linux como para Windows (donde $JAVA_HOME es la ubicación del directorio donde está instalada la máquina virtual de Java). Además hay que copiar el archivo rxtxSerial.dll en $JAVA_HOME/jre/bin en el caso de Windows, y en el caso de Linux el archivo rxtxSerial.so en el directorio $JAVA_HOME/jre/lib/i386. Todos estos archivos se encuentran en el CD en la carpeta *Instalación sistema/Archivos a copiar en cliente SW\rxtx-2.1-7-bins-r2*.
- Copiar la carpeta *InsercionPwdTeclado* (que contiene a la aplicación de apertura de puertas, *InsercionPwdTeclado.jar*), la carpeta *InsercionPwdLector* (que contiene a la aplicación de apertura de puertas, *InsercionPwdLector.jar*) o ambas (ya que podrían funcionar al mismo tiempo), según las necesidades, en una carpeta del ordenador local. En esa carpeta, que se encuentra en *Instalación sistema/Aplicaciones*, está todo lo necesario para que funcione tanto en Linux como en Windows, así como las instrucciones de uso en un archivo de texto, *Leeme.txt*, según se quiera tener o no información del proceso de envío de datos. Tanto en Windows como en Linux la aplicación se ejecuta con un doble clic en el correspondiente archivo de extensión "*.jar*".
- Una vez completados todos los pasos anteriores sólo queda conectar el dispositivo que realiza la apertura de puertas al puerto serie del ordenador y el teclado o el lector que se use al puerto correspondiente del ordenador.

5.1.3 CLIENTE DE LA APLICACIÓN WEB

Para poder utilizar la aplicación Web desde el ordenador cliente únicamente es necesario instalar el certificado digital del cliente (*client1.p12*) en el navegador que se vaya a usar para gestionar la Web de accesos a los laboratorios. Este certificado se encuentra en la carpeta del CD *Instalación sistema/ssl/client*. Para ello normalmente los navegadores tienen una opción de importar certificados. En los navegadores en los cuales se han hecho las pruebas se hace de la siguiente forma:

- En Firefox se hace a través del menú de herramientas, seleccionando opciones y luego seleccionando avanzado. Allí se busca dentro de la pestaña correspondiente a seguridad, la sección de certificados, desde la que se puede importar.
- En Internet Explorer a través del menú de herramientas, seleccionando opciones de Internet, después contenido y finalmente certificados, desde donde se puede importar el certificado.

5.1.4 SERVIDOR WEB

5.1.4.1 Sitio Web de la gestión de accesos a los laboratorios del DTE

Para habilitar el servidor de la aplicación Web, la instalación paso a paso es la siguiente:

- Instalar el JRE al menos de la versión jre1.5.0_06. de la misma forma que se explica en el punto 5.1.2 para el ordenador local del laboratorio.
- Instalar Tomcat, en su versión 5.5 o superior. En el caso de Windows está disponible en el CD del proyecto en la carpeta *Instalación sistema/Instalación Windows*. Para Linux se puede descargar de la página de *Apache Tomcat* y descomprimir en /usr/local. En la ruta donde se haya instalado se debe ahora cambiar el archivo *server.xml* del directorio *conf* por el que se suministra en el CD del proyecto en la carpeta *Instalación sistema/Archivos externos a copiar en servidor SW /*. Además es necesario copiar los archivos de la carpeta *Instalación sistema/Archivos externos a copiar en servidor SW /lib* del CD en la carpeta *common/lib*. En todo caso siempre se ha supuesto que el servidor de aplicaciones Tomcat se usa en solitario, por lo que el archivo *server.xml* se ha configurado para usarse en el puerto 80. En caso de tener funcionando en el mismo ordenador un servidor HTTP, como Apache por ejemplo, este puerto estaría ocupado por él, de manera que habría que poner al servidor de aplicaciones Tomcat en otro puerto, como el 8080, que es el que viene por defecto. Este cambio habría que hacerlo sobre el archivo *server.xml* que se suministra. De esta forma, para acceder a él, en la dirección IP habría que indicar el puerto. Por ejemplo, si fuera el ordenador local, la dirección IP sería "*localhost:8080*".
- Crear las mismas variables de entorno que se explicaron en el cliente en el punto 5.1.2 y además añadir la siguiente:

- *CATALINA_HOME*: indica la ubicación del directorio donde se encuentra instalado Tomcat. En Windows se suele instalar en *C:\Program Files\Apache Software Foundation\Tomcat 5.5* en el caso de Tomcat 5.5. En Linux es aconsejable descomprimirlo en un directorio accesible para todos los usuarios como */usr/local/*.

De la misma forma explicada antes en Windows se accede a propiedades de mi PC, opciones avanzadas, variables de entorno y se indica el directorio donde se encuentra ubicada. En Linux hay que editar el archivo */home/ USUARIO/.bashrc*, donde *USUARIO* identifica el directorio de trabajo de la sesión que se quiere habilitar para trabajar como cliente y escribir al final del archivo:

- *export CATALINA_HOME=/usr/local/apache-tomcat-5.5.17*

➢ Copiar el archivo desplegable *GestionLaboratorios.war*, disponible en la carpeta del CD *Instalación sistema/Aplicaciones*, en la carpeta *$CATALINA_HOME/webapps*, donde *$CATALINA_HOME* indica el directorio donde se encuentra instalado Tomcat. Una vez desplegado, en el archivo *configuracion.xml* de la carpeta configuracion habría que modificar los campos necesarios en caso de que la base de datos estuviera instalada en otro ordenador, o se quisiera cambiar la contraseña, el usuario, o incluso el sistema gestor de bases de datos. Estos cambios habría que hacerlos al mismo tiempo en el archivo server.xml del directorio conf de la instalación de Tomcat.

5.1.4.2 Servicio Web

Para habilitar el servicio Web en un servidor los pasos a seguir serían los mismos que en el apartado anterior, en caso de ser en un ordenador diferente, cambiando sólo que en el último paso se copiaría el archivo desplegable *InsercionPwdSW.war*, disponible en la carpeta del CD *Instalación sistema/Aplicaciones*, en la carpeta *$CATALINA_HOME/webapps*, donde *$CATALINA_HOME* indica el directorio donde se encuentra instalado Tomcat. Una vez desplegado, en el archivo *configuracion.xml* de la carpeta *WEB-INF/classes/configuracion* habría que modificar los campos necesarios en caso de que la base de datos estuviera instalada en otro ordenador, o se quisiera cambiar la contraseña, el usuario, o incluso el sistema gestor de bases de datos. Estos cambios habría

que hacerlos al mismo tiempo en el archivo *server.xml* del directorio *conf* de la instalación de Tomcat.

5.1.4.3 Base de datos

Para instalar la base de datos se deben seguir los siguientes pasos:

- Instalar el SGBD MySQL usando como usuario *root* y como contraseña *pelotero*. Estos datos son los que por defecto usará la aplicación Web y el servicio Web para acceder a la base de datos, por lo que de elegir otros habría que modificar, como ya se ha comentado, el archivo *server.xml* de Tomcat y el archivo *configuracion.xml* perteneciente a las aplicaciones que se han desplegado en Tomcat. Para la instalación en Windows basta con ejecutar el archivo correspondiente a la instalación de MySQL provisto en la carpeta *Instalación sistema/Instalación Windows /*. En Ubuntu, la instalación se reduce a ejecutar la siguiente instrucción:
    - *sudo aptitude install mysql-server*

  Pero como por defecto no tiene contraseña, es necesario introducirla. Para ello se ejecuta la siguiente instrucción:
    - *sudo /usr/bin/mysqladmin -u root password pelotero*

  Donde *pelotero* es la contraseña que se ha usado en el proyecto,.
- Instalar la base de datos. Para ello primero se debe copiar la carpeta *configuracion BBDD* al equipo en que se quiere instalar la base de datos. En esa carpeta, que se encuentra en *Instalación sistema/Aplicaciones*, está todo lo necesario para que se despliegue tanto en Linux como en Windows, así como las instrucciones de uso en un archivo de texto, *Leeme.txt*, según se quiera tener o no información del proceso de creación de la base de datos. Tanto en Windows como en Linux la aplicación se ejecuta con un doble clic en *Configuracion.jar*.
- En el caso en que se quisiera cambiar la contraseña de acceso a la aplicación Web, fijada por defecto como *pelotero*, sólo habría que cambiar en la base de datos alojada en el servidor el valor de un registro de una tabla. En concreto, en la tabla *usuariosTomcat*, habría que modificar el valor del campo *password_usuario*. Esto se puede hacer sencillamente con la sentencia "*UPDATE usuariosTomcat SET password_usuario = nuevoPassword;*". Para

hacer este cambio antes de desplegar la base de datos simplemente habría que cambiar en el archivo *configuracion.xml* de la carpeta *configuracion BBDD* la sentencia "*INSERT INTO usuariosTomcat VALUES ('root', 'pelotero'),('tomcat', 'tomcat')*" modificando los valores de "root" y "pelotero" por el usuario y la contraseña respectivamente que se deseen. Al ejecutar el archivo de despliegue de la base de datos, *Configuracion.jar*, tomará los valores que se pasan en el archivo *configuracion.xml*.

5.2 MANUAL DE USUARIO

5.2.1 ORDENADOR LOCAL DEL LABORATORIO

Para poner en marcha el sistema, tanto en Windows como en Linux, lo único que hace falta es ejecutar el archivo *InsercionPwdTeclado.jar* o *InsercionPwdLector.jar* según se use uno u otro. A partir de ahora se llamará en este apartado *InsercionPwdXXX* tanto a *InsercionPwdTeclado* como a *InsercionPwdLector* para generalizar en los puntos en los que la explicación es la misma. Hay varias formas de poner en marcha el sistema:

- Para Windows:
  - Con doble clic en el archivo *InsercionPwdXXX.jar*
  - Con doble clic en *InsercionPwdXXX.bat*; así aparece una consola que detalla las acciones que se realizan. En caso de no desear que aparezca esa consola se puede usar la primera opción o editar el archivo "*InsercionPwdXXX.bat*" y cambiar "java" por "javaw".

- Para Linux:
  - Con doble clic en el archivo *InsercionPwdXXX.jar*
  - Estando posicionados en el directorio donde se hallen *InsercionPwdXXX.jar* e *InsercionPwdXXX.sh*, teclear *sh InsercionPwdXXX.sh*.
  - Darle permisos de ejecución con "*chmod a+x InsercionPwdXXX.sh*" a *InsercionPwdXXX.sh* y teclear en consola *./InsercionPwdXXX.sh*.

Una vez en ejecución el programa es diferente según sea la versión del teclado o la del lector.

5.2.1.1 Versión de teclado

En la versión del teclado aparece por pantalla la ventana de la figura 3.1, en la que se pide la dirección IP o nombre del servidor que aloja el servicio Web. En cuanto se introduce la dirección IP o nombre del servidor y se valida con el botón asociado a esa acción aparece la ventana de la figura 3.2. En esa ventana, tal y como se explica en el capítulo 3, hay tres campos configurables, que son:

- Laboratorio: aquí se elige, de un menú desplegable el laboratorio en el que se está instalando el cliente.
- Tiempo de apertura (s): aquí se elige, también de un menú desplegable el tiempo que se va a accionar el mecanismo de apertura de puertas en un rango de 0,5 a 5 segundos con paso de 0,5 segundos.
- Puerto a usar para el sistema de apertura: por último, se elige, también de un menú desplegable el puerto al que se ha conectado el sistema de apertura de puertas.

Hecho esto, ya sólo queda validar la configuración elegida con el botón “Aceptar”, de manera que la aplicación queda como se ve en la figura 3.3, con un campo de texto que recoge las pulsaciones válidas del teclado y un campo no editable que indica el resultado del último intento de acceso. Sólo queda recordar que para que la aplicación reciba correctamente los caracteres introducidos, la ventana debe estar activa, es decir, en primer plano y con el campo de la contraseña seleccionado (esta es la configuración con la que se inicia por defecto).

5.2.1.2 Versión de lector

En la versión del lector aparece por pantalla la ventana de la figura 3.5, en la que se pide la dirección IP o nombre del servidor que aloja el servicio Web. En cuanto se introduce la dirección IP o nombre del servidor y se valida con el botón asociado a esa acción aparece la ventana de la figura 3.6. En esa ventana, tal y como se explica en el capítulo 3, hay ocho campos configurables, que son:

- Laboratorio: aquí se elige, de un menú desplegable el laboratorio en el que se está instalando el cliente.

- Tiempo de apertura (s): aquí se elige, también de un menú desplegable el tiempo que se va a accionar el mecanismo de apertura de puertas en un rango de 0,5 a 5 segundos con paso de 0,5 segundos.
- Velocidad de lectura de datos (bps): donde, en un campo de texto (donde sólo es posible escribir números) se puede indicar exactamente la velocidad a la que el lector envía los datos por el puerto serie.
- Bits de datos: es un parámetro de comunicación, al igual que la lectura de datos, también configurable. Se elige de un menú desplegable.
- Bits de parada: es también un parámetro de comunicación configurable. Se elige igualmente de un menú desplegable.
- Paridad: es también un parámetro de comunicación configurable. Se elige igualmente de un menú desplegable.
- Puerto a usar para el lector: se elige, de un menú desplegable el puerto a través del cual se va a realizar la lectura.
- Puerto a usar para el sistema de apertura: por último, se elige, también de un menú desplegable el puerto al que se ha conectado el sistema de apertura de puertas. No es posible, evidentemente, que los dos últimos campos sean iguales. Por defecto están marcados dos puertos diferentes, pero si por equivocación se eligiera el mismo, el programa da un aviso del error.

Hecho esto, ya sólo queda validar la configuración elegida con el botón “Aceptar”, de manera que la aplicación queda como se ve en la figura 3.8, con un campo de texto no editable que indica el resultado del último intento de acceso.

5.2.2 CLIENTE DE LA APLICACIÓN WEB

5.2.2.1 Entrada a la aplicación

Si se supone que el servidor está en la dirección IP 192.168.1.7, la dirección que habría que poner en el navegador para acceder al sitio Web de gestión de accesos sería http://192.168.1.7/GestionLaboratorios/GestionLaboratoriosDTE.html. Al enviar esa petición desde el navegador aparecería una pantalla como la de la figura 5.1, en la que se da información acerca del certificado del servidor. Se debe aceptar entonces para proseguir con la autenticación, esta vez del cliente.

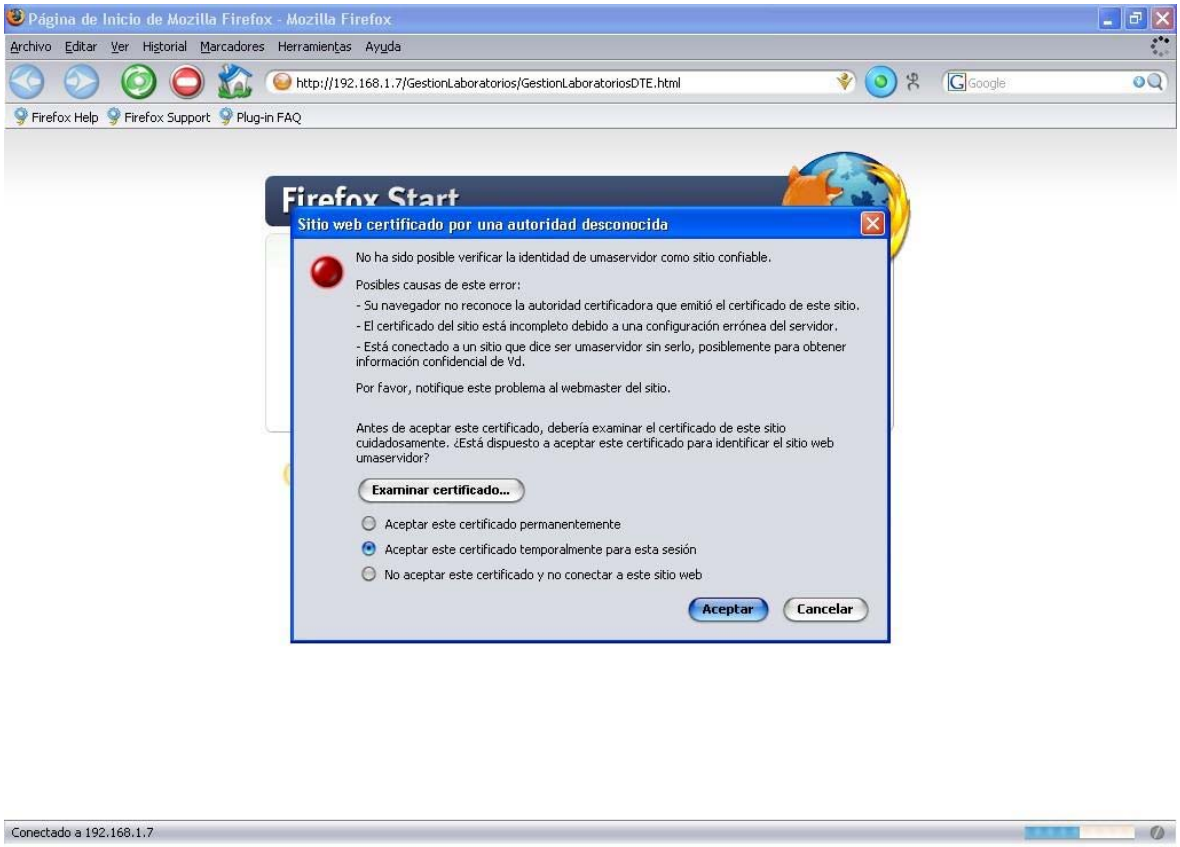

Figura 5.1. Información del certificado del servidor

Una vez que se ha aceptado el certificado del servidor se pasa a una situación como la de la figura 5.2, en la que se pide el nombre del usuario y la contraseña que habilitan el acceso al sitio Web. Este es uno de los dos mecanismos por los que se autentica al cliente de cara al servidor, y el primero de los dos en activarse. Cuando se remita el nombre del usuario con la contraseña, el servidor pedirá además el certificado del cliente. El envío del mismo ocurre de forma transparente al usuario, de manera que de cara al mismo la única forma en que se autentica parece ser con la contraseña.

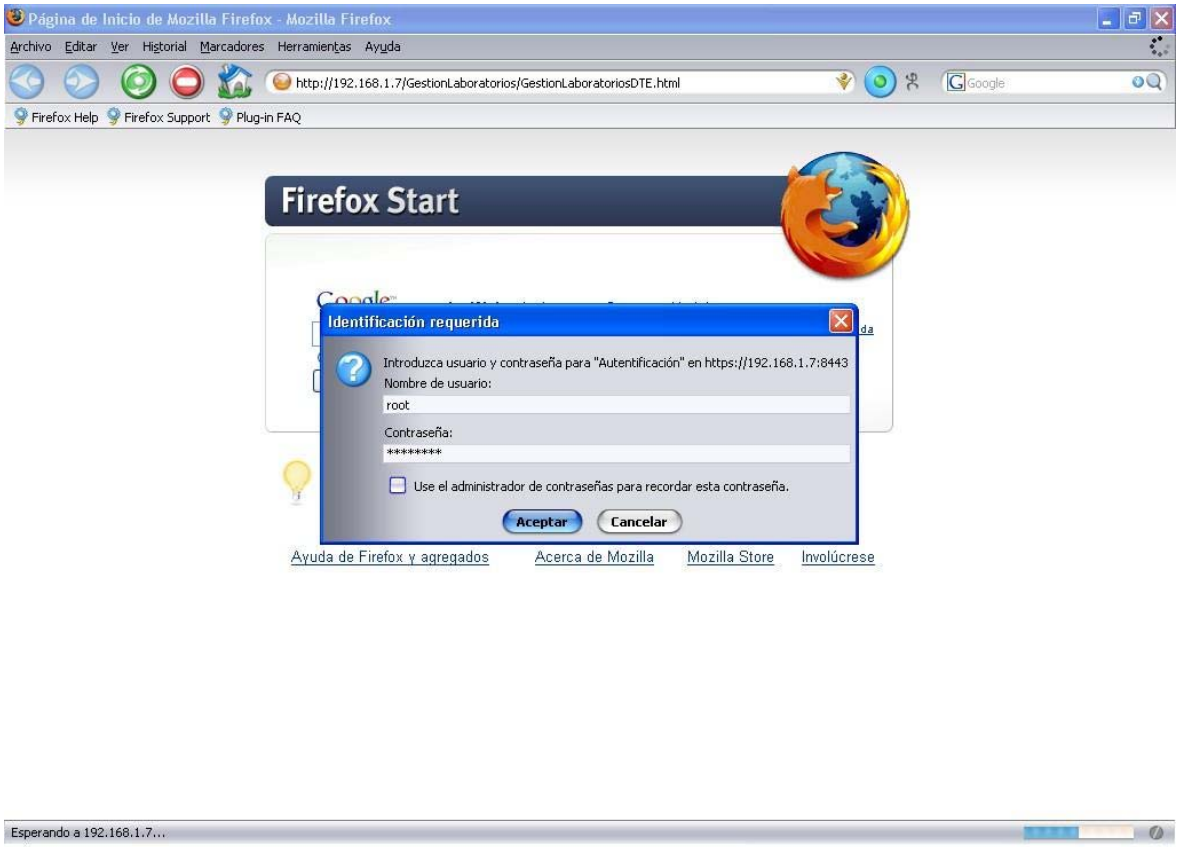

Figura 5.2. Autenticación del cliente

Cuando cliente y servidor se han reconocido mutuamente, el servidor da acceso a los contenidos del sitio Web que ha pedido el cliente. La página inicial que se muestra, ya con conexión SSL, es la de la figura 5.3. Se puede apreciar como, efectivamente se ha establecido una conexión segura, al ver que las letras que indican la dirección ahora están en color verde y aparece un candado, tanto al lado de la dirección como en la parte inferior derecha de la página. Los signos que indican que la conexión es segura varían de un navegador a otro y dependiendo de la personalización que se le dé al mismo, pero normalmente hay un cambio de color en la barra de direcciones y aparece un candado en la misma.

A la página inicial que se muestra en la figura 5.3 siempre se puede volver marcando el enlace "Principal" de la barra lateral. En esta página se hacen dos indicaciones para el óptimo funcionamiento de la aplicación. En cuanto a la indicación de la resolución de la pantalla, el hecho de tener otra resolución simplemente provocará que la representación gráfica varíe respecto a aquella que se diseñó. Por lo que respecta al uso de JavaScript sí que es absolutamente necesario su uso, ya que no sólo se emplea para comprobar que los datos que se envían se hacen de forma correcta, sino que en muchos casos es a través de JavaScript como se realiza el envío definitivo.

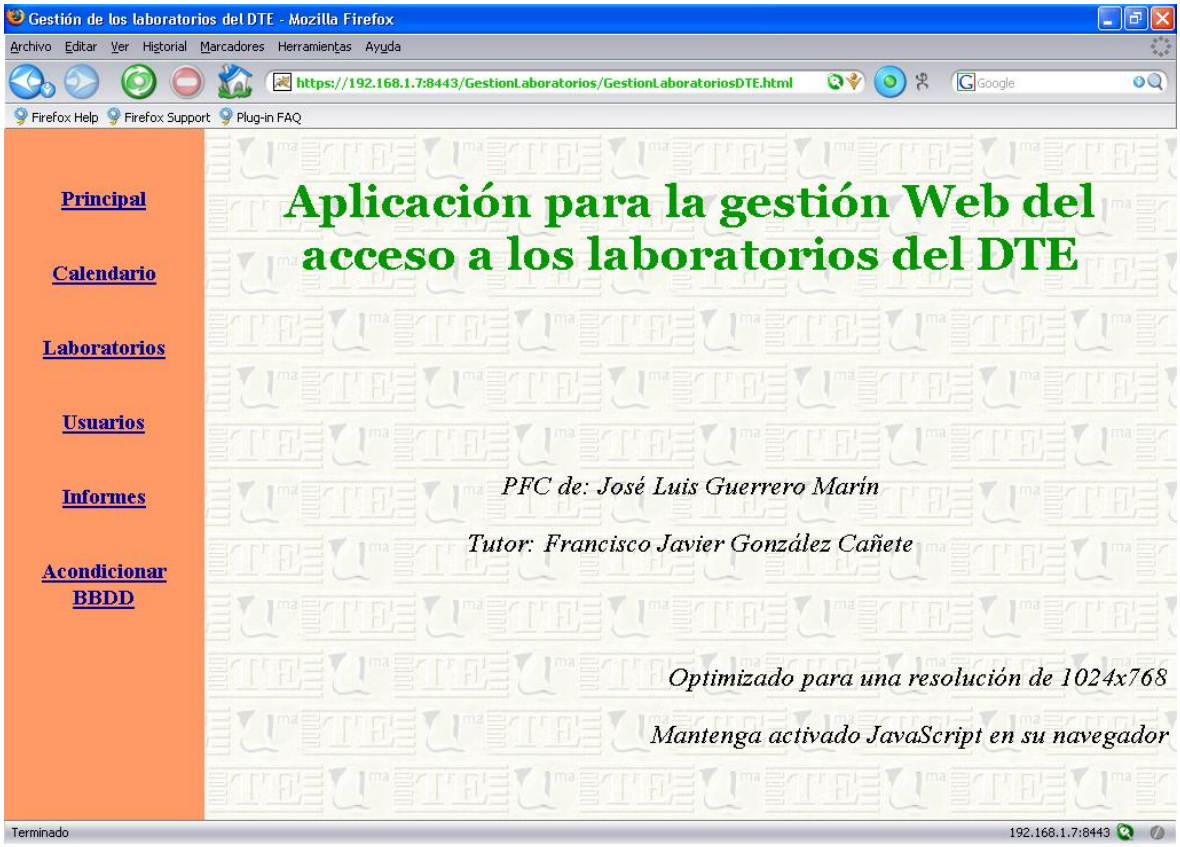

Figura 5.3. Página principal del sitio Web de gestión de accesos

5.2.2.2 Sección Calendario

Al entrar en esta sección, la primera pantalla que aparece es como la de la figura 5.4, en la que se puede ver al completo el calendario del año actual con los días festivos marcados en rojo y los días con horario especial, si hubiera alguno, en azul. Si se quisiera realizar alguna modificación en el calendario habría que descender por la página hasta encontrar el botón “Realizar modificaciones en el calendario”, que conduciría hasta otra página, en la que sí se podrían modificar los días festivos y los de horario especial del año actual y los dos siguientes.

Figura 5.4. Página de bienvenida del calendario

La página donde se pueden realizar las modificaciones está dividida en cuatro partes, de arriba a abajo como sigue:

- Menú desplegable donde se puede seleccionar el año que se va a modificar.
- Calendario con el mismo aspecto que el de bienvenida, siempre referido al año seleccionado en el menú desplegable comentado en el punto anterior.
- Fiestas: Al pulsar sobre el título se ven todas los subapartados de los que está compuesto (añadir, mover o borrar fiestas del calendario) y al pulsar nuevamente sobre el título se ocultan. En la figura 5.5 se puede ver en detalle este apartado, ya desplegado. Dentro del subapartado "Añadir" se encuentra un campo de texto en el que describir en menos de 100 caracteres la fiesta a añadir y dos menús desplegables donde seleccionar el mes y día donde se marcará la fiesta. El subapartado "Mover" es básicamente igual, la única diferencia es que ahora no aparece un campo para la descripción. En su lugar aparece un menú desplegable para indicar cuál, de las fiestas ya introducidas en la base de datos se desea mover. El subapartado "Borrar" es el más sencillo, constando sólo de

un menú desplegable con las fiestas almacenadas. En todos los subapartados, al validar la acción mediante el botón correspondiente (“Añadir”, “Mover” o “Borrar”) se comprueba que los datos sean correctos y siempre se procede a confirmar la operación.

Figura 5.5. Subapartado de fiestas

- Tramos de horario especial: es muy parecido al apartado anterior, con la salvedad de que en este caso se introducen las fechas por tramos de días. Igualmente, al pulsar sobre el título se ven todas los subapartados de los que está compuesto (añadir, mover o borrar fiestas del calendario) y al pulsar nuevamente sobre el título se ocultan. En los tramos de días que se seleccionen hay que definir unas horas de acceso a los laboratorios, fuera de las cuales sólo podrían acceder técnicos y profesores. Como se ve en la figura 5.6, también esta subdividido en tres apartados. Por una parte “Añadir”, en la que se definen esos tramos de horario especial, por otra “Modificar”, en la que se pueden cambiar las horas de acceso del tramo que se seleccione dentro de los que consten en la base de datos y por último “Borrar”, que elimina un tramo al completo. Hay que destacar que los días festivos tienen más prioridad que los tramos de horario especial. Es decir, si se pone un tramo de horario especial que incluya algún día festivo, este continuará siendo festivo y a todos los efectos sólo podrán acceder a los laboratorios técnicos y profesores. En cambio, si un día marcado como especial se hace festivo, pierde la condición de día con horario especial.

Como nota, cabe destacar que aparte de los domingos, por defecto están marcados algunos días de fiesta en el calendario. Estos días son los correspondientes a fiestas nacionales que en ningún caso se van a modificar, por lo que no se pueden mover ni borrar, ni aparecen siquiera en los menús desplegables.

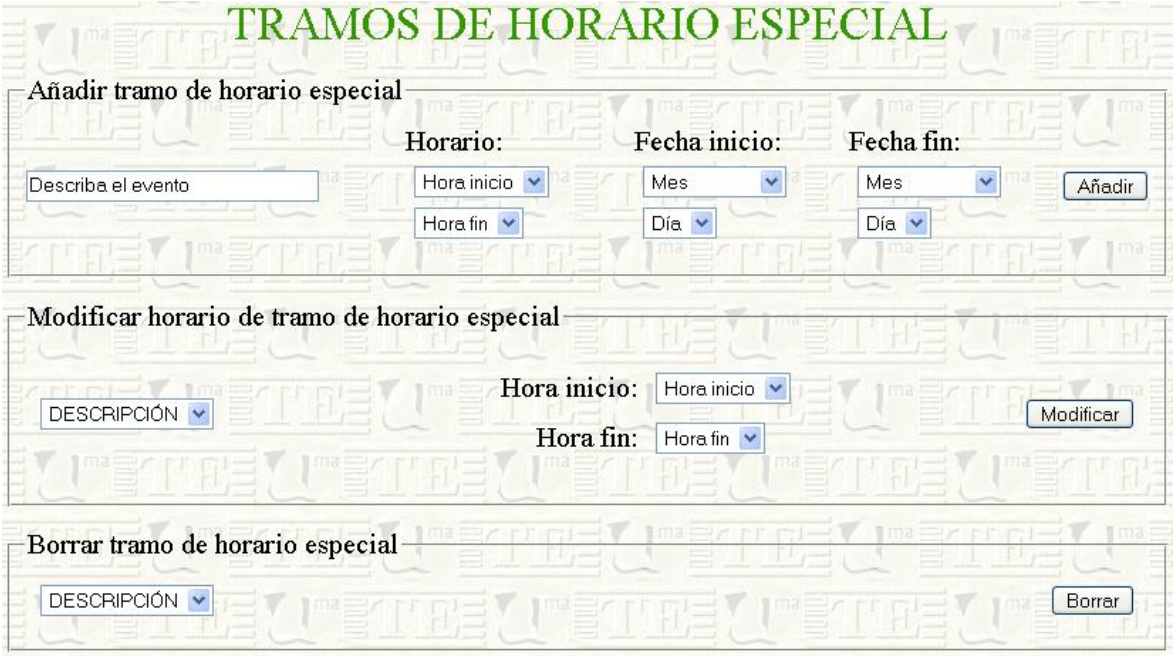

Figura 5.6. Subapartado de tramos de horario especial

5.2.2.3 Sección Laboratorios

Esta sección está dividida en tres apartados: “Añadir”, “Modificar” y “Borrar”, y todos ellos están sobre la misma página, como se muestra en la figura 5.7. Tienen, al igual que en la sección “Calendario”, la característica de desplegarse o replegarse cada opción pulsando sobre el título. Estos apartados están organizados de la siguiente forma:

- Añadir (figura 5.8): permite introducir nuevos laboratorios a la base de datos. Contiene dos campos de texto; en el primero se dispone de 9 caracteres para escribir el nombre del laboratorio, mientras que en el segundo se dispone de hasta 100 caracteres para introducir la descripción.
- Modificar (figura 5.9): permite modificar la descripción de los laboratorios ya introducidos en la base de datos. Para ello dispone de un menú desplegable con la lista de los laboratorios ya incluidos en la base de datos y un campo de texto donde meter la nueva descripción del laboratorio.

- Borrar (figura 5.10): permite borrar un laboratorio de la base de datos. Para ello simplemente se debe seleccionar de la lista el laboratorio que se desea borrar y se valida con el botón “Borrar”.

Al igual que en la sección “Calendario” todas la operaciones piden confirmación antes de realizarse efectivamente.

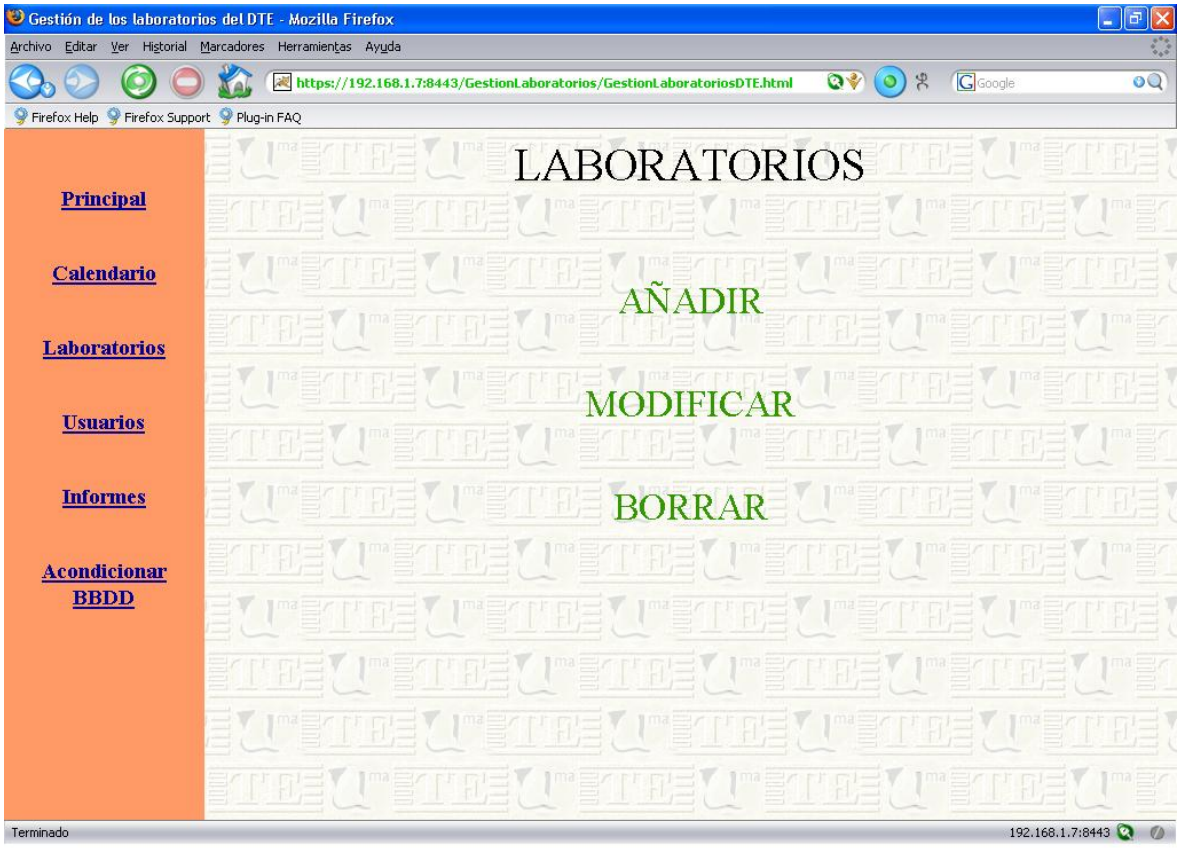

Figura 5.7. Sección laboratorios

Figura 5.8. Detalle de la opción “Añadir”

Figura 5.9. Detalle de la opción "Modificar"

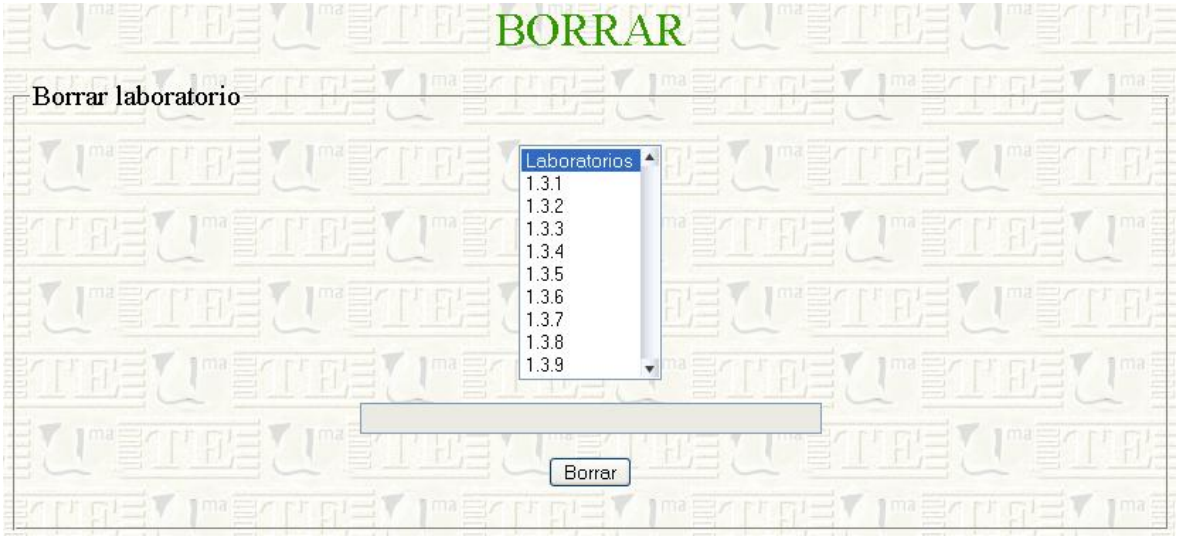

Figura 5.10. Detalle de la opción "Borrar"

5.2.2.4 Sección Usuarios

Al entrar en esta sección aparece una página como la que se muestra en la figura 5.11. En ella, de arriba abajo, las opciones son: "añadir usuario", "modificar usuario" y "ver datos/borrar usuario". Se puede observar que se repite más o menos la estructura que tiene todo el sitio Web, como se ha visto en la sección del calendario y la de los laboratorios, de manera que, al igual que en esas otras secciones, pulsando sobre el título se despliegan o repliegan. De esta forma el interfaz es muy intuitivo y una vez que se ha manejado mínimamente una parte del sitio Web es muy sencillo adivinar cómo funciona el resto.

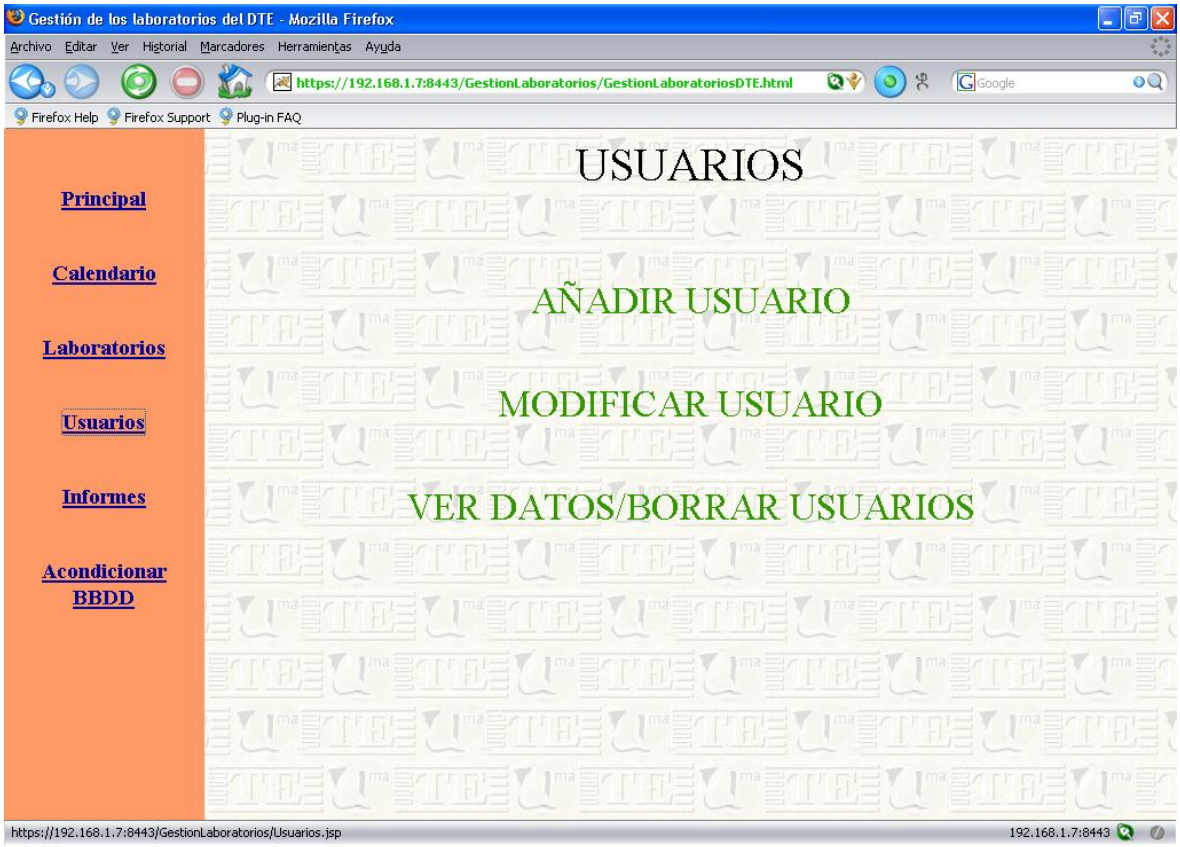

Figura 5.11. Sección usuarios

En el apartado de "añadir usuario" se pueden observar dos subapartados: uno referente a los datos propios del usuario que se va a añadir (figura 5.12) y otro en el que se le asigna al menos un laboratorio (figura 5.13), siempre que el usuario no sea de tipo privilegiado (técnico o profesor), en cuyo caso no tiene sentido que se le asigne un laboratorio, ya que tienen acceso a todos por defecto.

Figura 5.12. Campos deshabilitados al seleccionar el tipo de usuario "profesor"

En el subapartado de los datos personales del usuario hay que rellenar todos los campos correctamente. En caso contrario aparece una ventana de alerta indicando el tipo de error (campo en blanco, NIF – Número de Identificación Fiscal – mal introducido, fecha incorrecta…). Además, en el caso de que el usuario a introducir sea un técnico o un profesor, se deshabilitan los campos que no tienen sentido en ese caso, como son la titulación a la que pertenece y el nombre y los apellidos del responsable. Se puede ver esta situación en la figura 5.12.

En cuanto al subapartado en el que se le asigna un laboratorio, basta con indicar, de un menú desplegable, el laboratorio, el margen de horas y los días que tendrá acceso. Una vez que se han rellenado correctamente todos los datos que correspondan al tipo de usuario que se quiere añadir a la base de datos se puede proceder al envío con el botón "Añadir usuario" (figura 5.13).

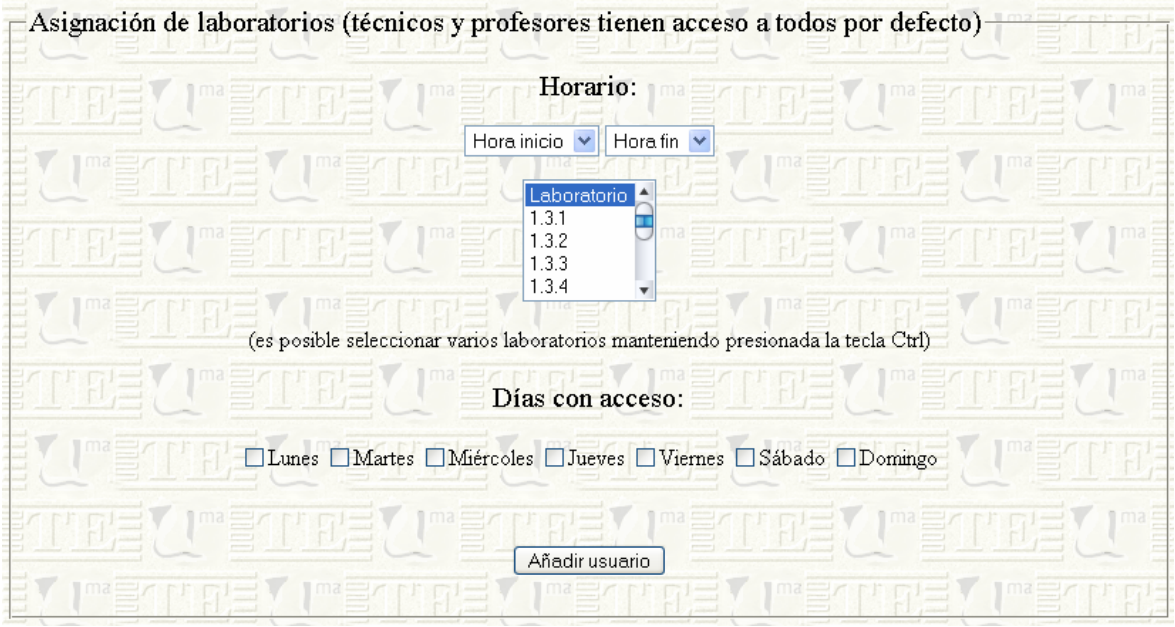

Figura 5.13. Añadir usuario

El siguiente apartado es el referente a "Modificar usuario", que consta de un menú desplegable en el que elegir, de una lista ordenada alfabéticamente según el apellido, uno de los usuarios para realizar modificaciones sobre él (figura 5.14).

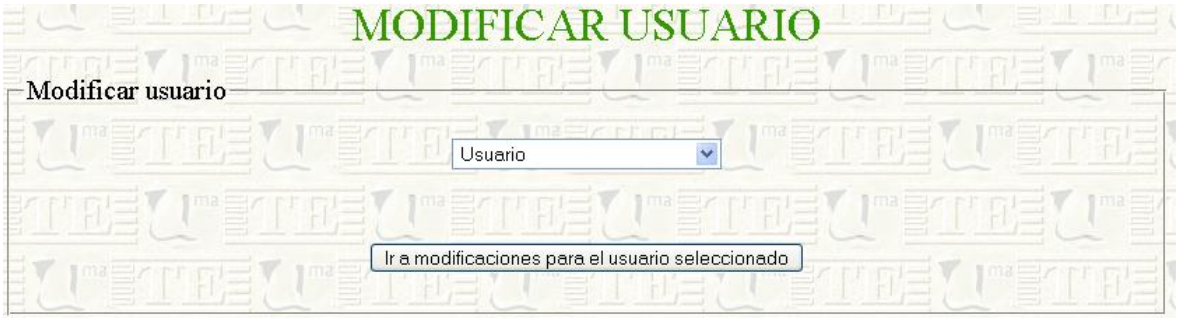

Figura 5.14. Selección de usuario a modificar

Cuando se ha seleccionado un usuario sobre el que realizar las modificaciones aparece una página dividida en cuatro partes como la de la figura 5.15 (con la misma característica del resto del sitio Web de desplegarse y replegarse pulsando sobre el título).

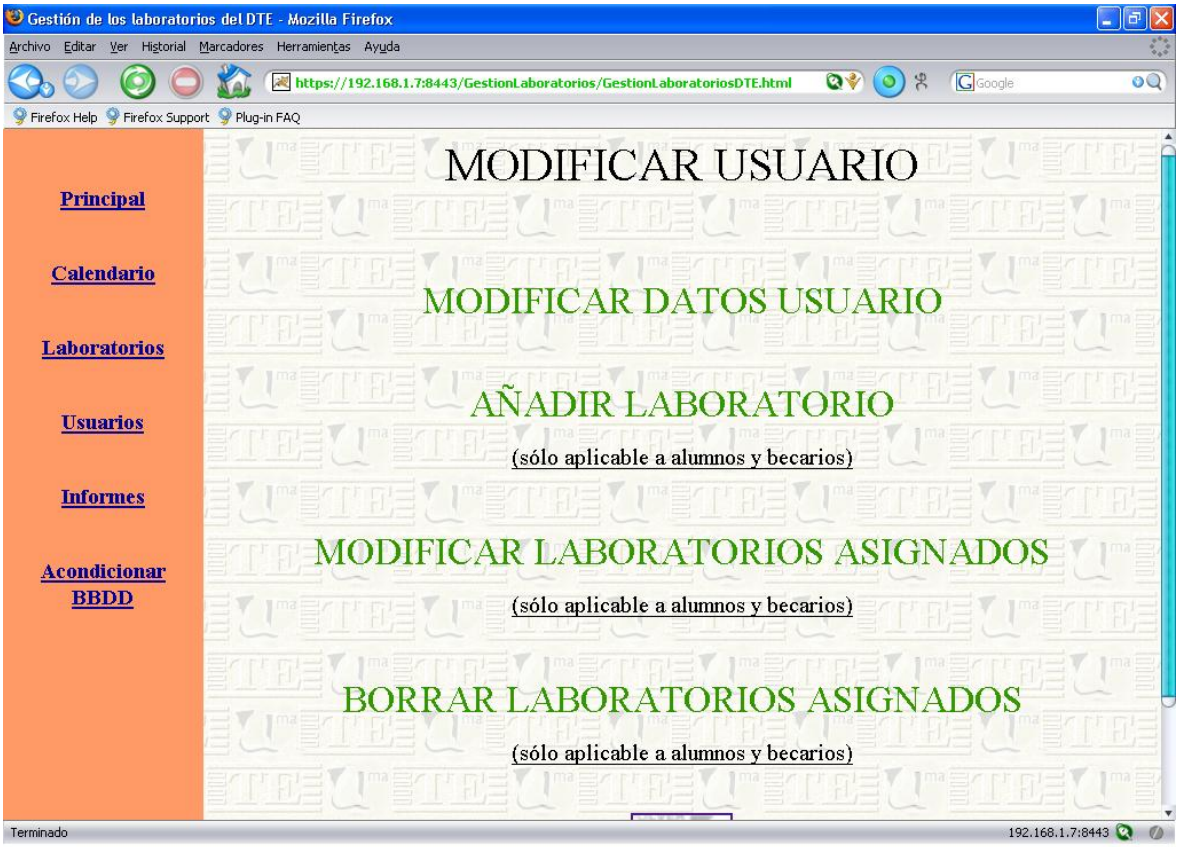

Figura 5.15. Modificar usuario

Las cuatro partes de las que se compone esta página son:

- Modificar datos usuario (figura 5.16).
- Añadir laboratorio (figura 5.17).
- Modificar laboratorios asignados (figura 5.18).
- Borrar laboratorios asignados (figura 5.19).

En la figura 5.16 se puede ver el resultado de seleccionar un cierto usuario para modificar (en el subapartado “Modificar datos usuario”). Aparecen todos sus datos almacenados, para realizar las modificaciones necesarias sobre ellos. Nuevamente, si se dejan campos en blanco, fechas erróneas o valores imposibles (por ejemplo nombre y apellidos del responsable iguales a los del usuario siendo el usuario un alumno o becario) aparece un mensaje de alerta indicando el error.

Figura 5.16. Modificar datos usuario

Los otros tres subapartados (añadir, modificar o borrar laboratorios) sólo se pueden editar si el usuario está almacenado en la base de datos como alumno o becario en el momento en el que se pretende editar, ya que los profesores y técnicos tienen acceso a todos los laboratorios por defecto.

En la figura 5.17 se muestra el subapartado correspondiente a “Añadir laboratorio”, en el se puede ver que siempre se indica, al igual que en el subapartado anterior, el NIF del usuario que se está modificando. Aparecen dos menús desplegables donde es posible marcar la hora de comienzo y de fin del acceso al laboratorio, otro que permite seleccionar de la base de datos el laboratorio al cual se va a dar acceso y siete casillas de marcado donde se pueden señalar los días de la semana en que se va a poder producir dicho acceso al laboratorio en cuestión.

Figura 5.17. Añadir laboratorio a un usuario

En la figura 5.18, correspondiente al subapartado de "Modificar laboratorios asignados", se puede ver que al seleccionar un laboratorio de los que tiene asignado el usuario que se está modificando se marcan todos los valores que tenía almacenados para ese laboratorio. Se observa además que la estructura es la misma que al añadir un laboratorio a un usuario. De esta forma el interfaz queda lo más sencillo posible de cara al usuario.

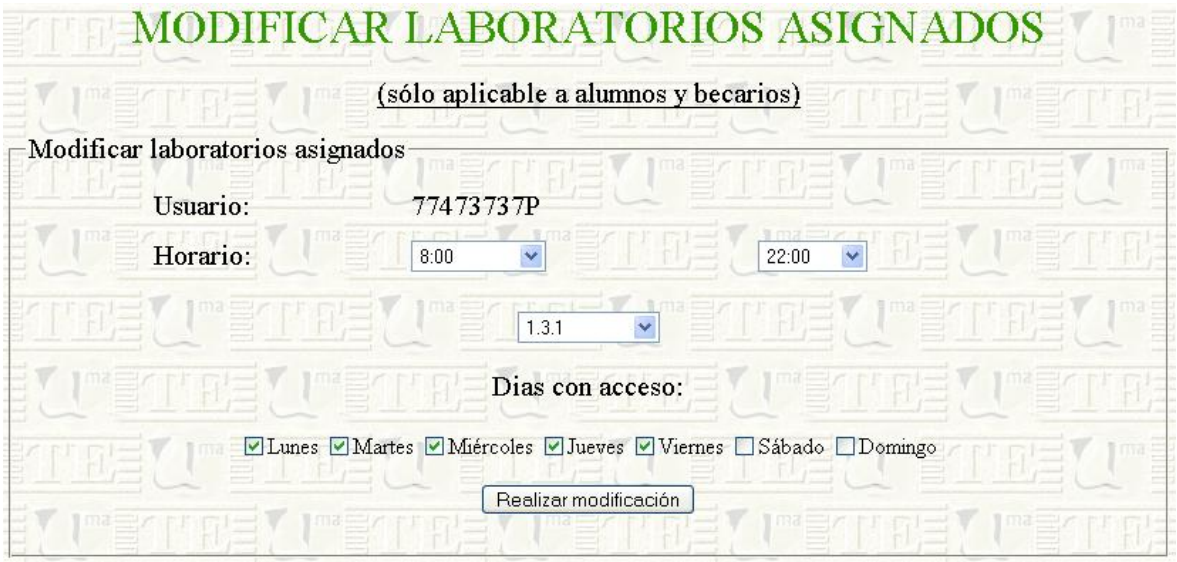

Figura 5.18. Modificar laboratorios asignados

El último de los apartados dentro de "Modificar usuario" es "Borrar laboratorios asignados", que se muestra en la figura 5.19. Consta de un menú desplegable en el que seleccionar el laboratorio que se quiere borrar, siempre dentro de los que tiene asignados el usuario que se está modificando y un botón para confirmar la operación. Hay que señalar

además que todas estas operaciones piden además una confirmación por medio de una ventana de alerta antes de realizarse.

Figura 5.19. Borrar laboratorios asignados

Por último, dentro de la sección "Usuarios", se encuentra la opción "Ver datos/Borrar usuarios" en la que, al seleccionar un usuario para borrar de la lista, aparecen sus datos de forma resumida como se puede ver en la figura 5.20. Si se prosigue borrando el usuario se pide confirmación y tras ésta se procede al borrado.

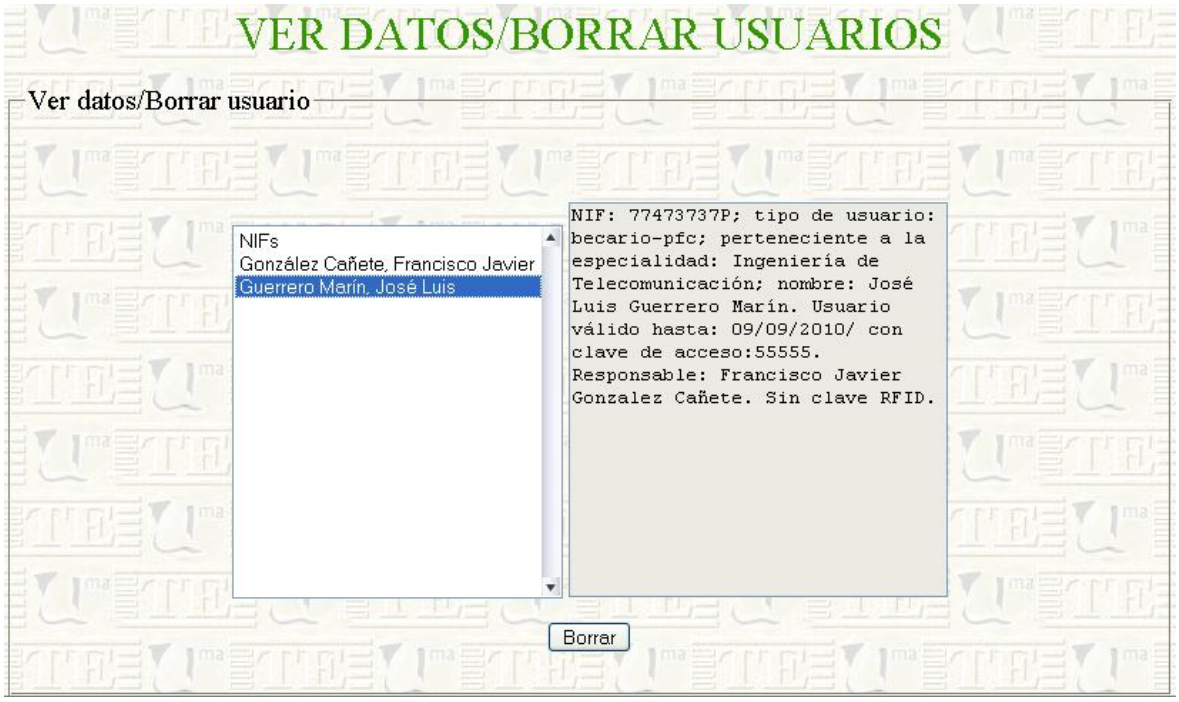

Figura 5.20. Ver datos/Borrar usuarios

5.2.2.5 Sección Informes

La página principal de esta sección se puede ver en la figura 5.21. En ella se muestran todos los informes que se pueden obtener dinámicamente en función de lo que se necesite. Dichos informes se muestran a través de las siguientes opciones:

- "Listado de fiestas y tramos de horario especial": con esta opción se pueden obtener informes de todas las fiestas y días con horario especial almacenados en la base de datos.
- "Listado de laboratorios": permite obtener listados de los laboratorios y su descripción.
- "Listado de usuarios": permite obtener listados de los usuarios filtrándolos por diversos criterios.
- "Registro de accesos": permite obtener informes de los intentos de acceso a los laboratorios.

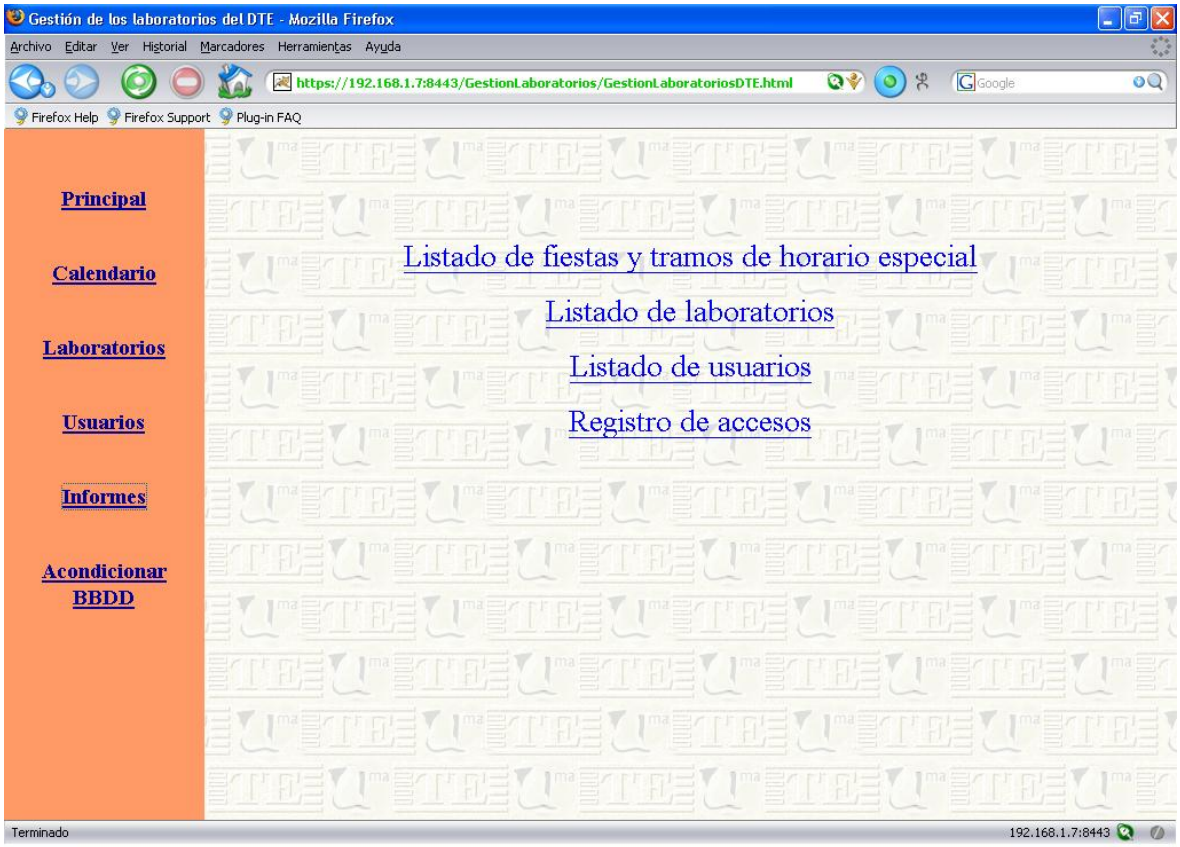

Figura 5.21. Sección Informes

La primera de las opciones dentro de esta sección es “Listado de fiestas y tramos de horario especial”, desde la que se puede obtener un informe de todos los días marcados en el calendario como festivos o como días con horario especial. Al acceder a ese apartado aparece un menú desplegable como el de la figura 5.18 donde se puede seleccionar el año del que se pretende obtener el informe, de los almacenados en la base de datos. Una vez seleccionado el año aparece el informe correspondiente, en su versión HTML, para mostrar rápidamente lo que se desea (figura 5.19).

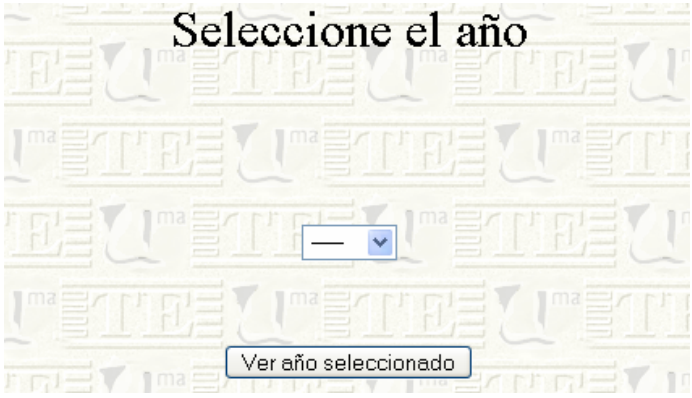

Figura 5.22. Selección del año para obtener informe

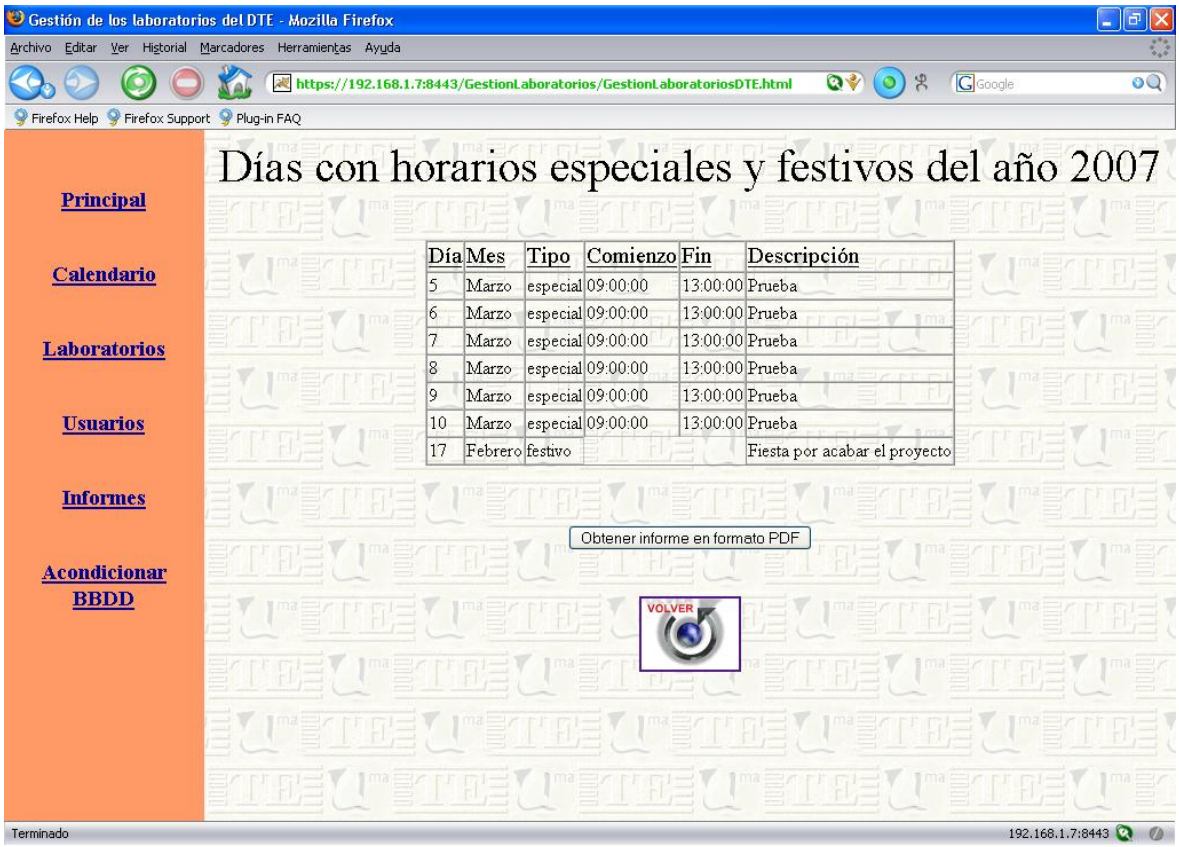

Figura 5.23. Informe del calendario en formato HTML

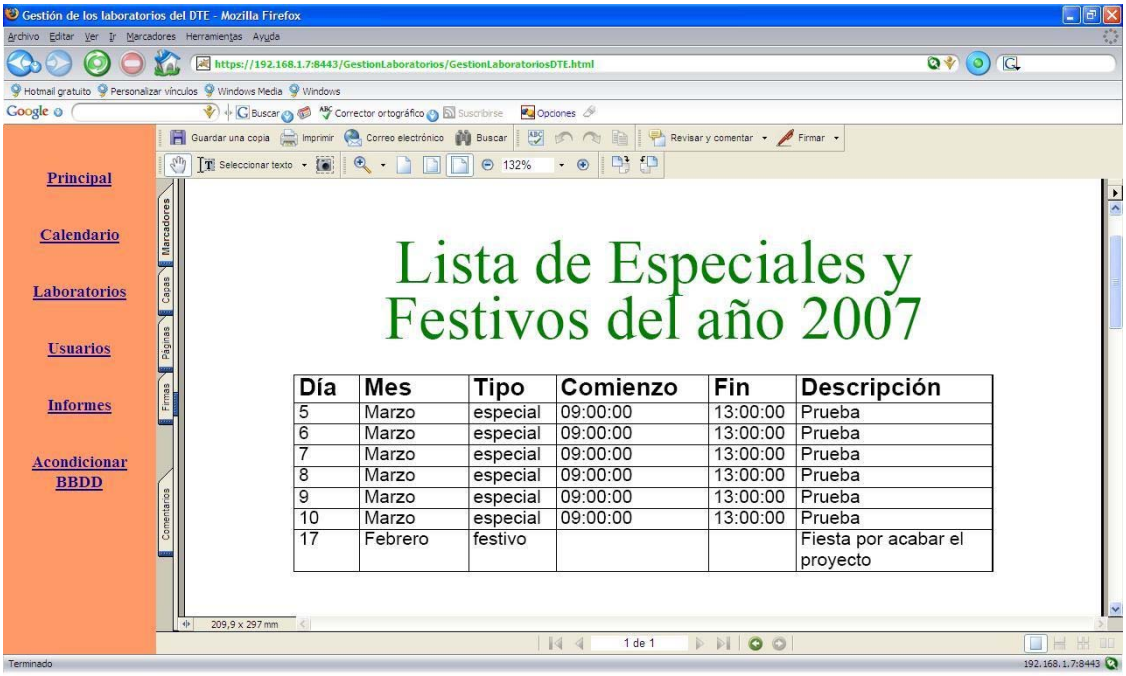

Figura 5.24. Informe del calendario en formato PDF

Si finalmente se desea obtener el informe para impresión o almacenarlo se puede proceder a obtener el informe en formato PDF con el botón asociado a esa acción. El resultado del informe de la figura 5.23 es el de la figura 5.24.

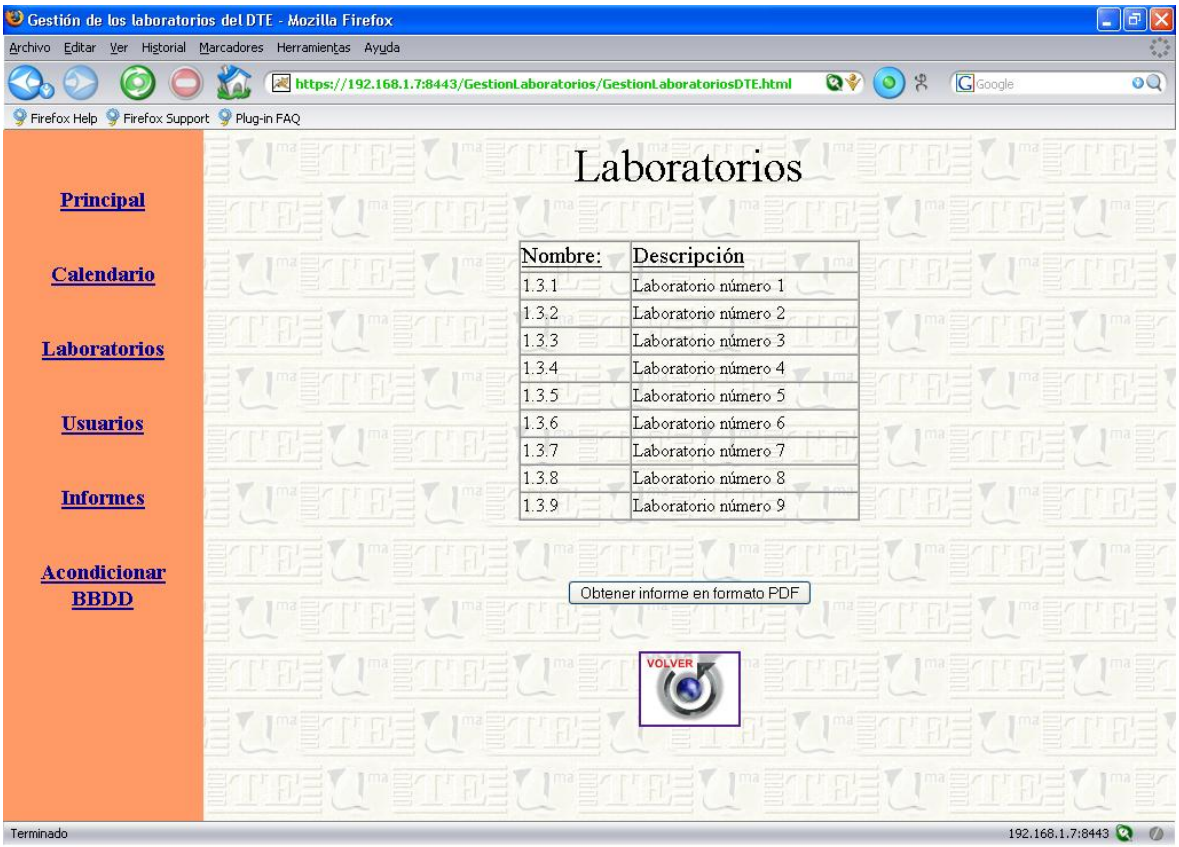

Figura 5.25. Listado de laboratorios en formato HTML

La segunda opción dentro de la sección de informes es “Listado de laboratorios”. Desde esta sección se puede obtener un listado de los laboratorios que en ese momento se encuentren en la base de datos, tal y como se muestra en la figura 5.25.

Del listado de la figura 5.25 también se puede obtener su correspondiente versión en PDF usando el botón asociado a esa acción, “Obtener informe en PDF”, al igual que el informe del calendario. El resultante informe correspondiente al listado de la figura 5.25 se muestra en la figura 5.26.

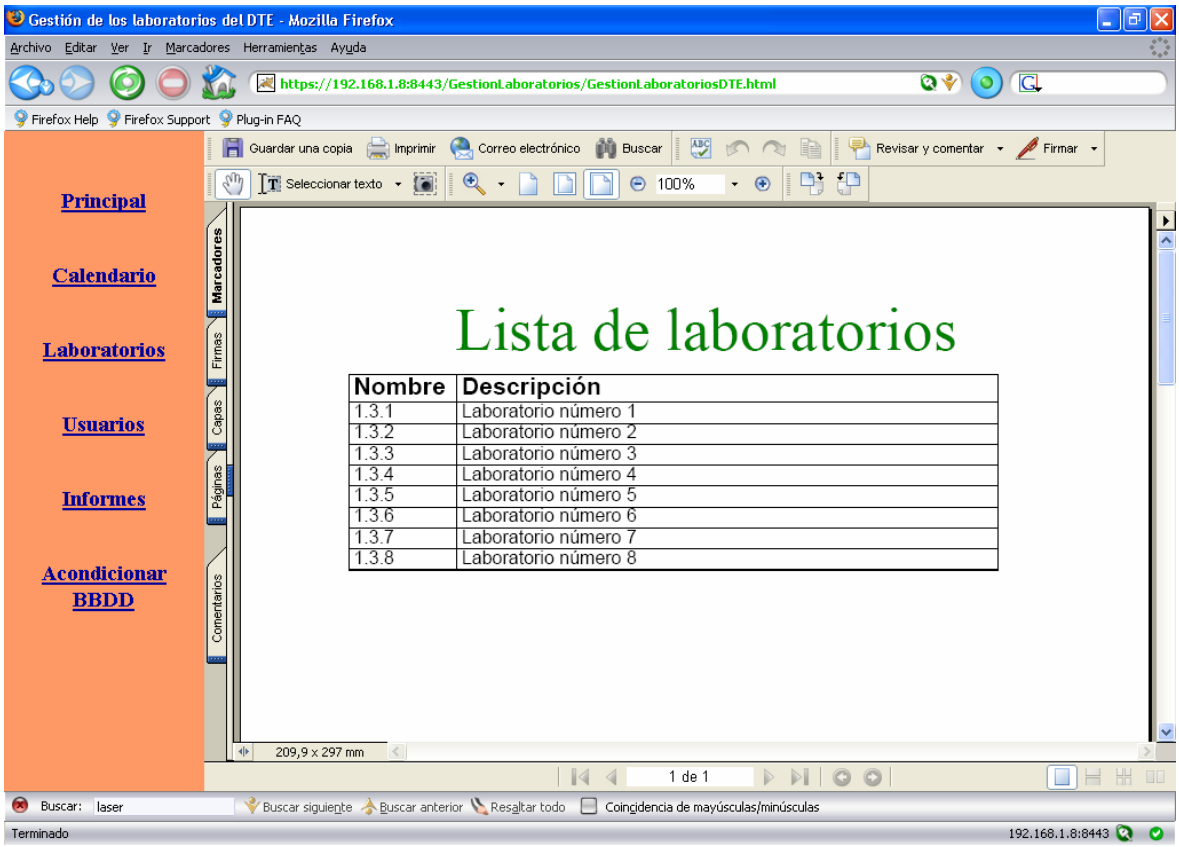

Figura 5.26. Listado de laboratorios en formato PDF

La tercera opción dentro de los informes es “Listado de usuarios”, desde la que se puede generar un informe de los usuarios filtrándolos según varias condiciones, tal y como se ve en la figura 5.27:

- “Tipo de usuario”: esta opción permite mostrar sólo un perfil de usuario (alumno, becario-pfc, profesor o técnico). Por defecto se muestran todos.
- “Titulación (alumnos, becarios-pfc)”: esta opción permite mostrar sólo los usuarios pertenecientes a una especialidad, que obviamente no pueden ser profesores o técnicos, aunque hay una opción para mostrarlos a ellos (“Técnicos

y profesores"). Por defecto se muestran los de todas las especialidades y los técnicos y profesores.

- "Con acceso al laboratorio": esta opción filtra a los usuarios que tienen acceso a un determinado laboratorio.

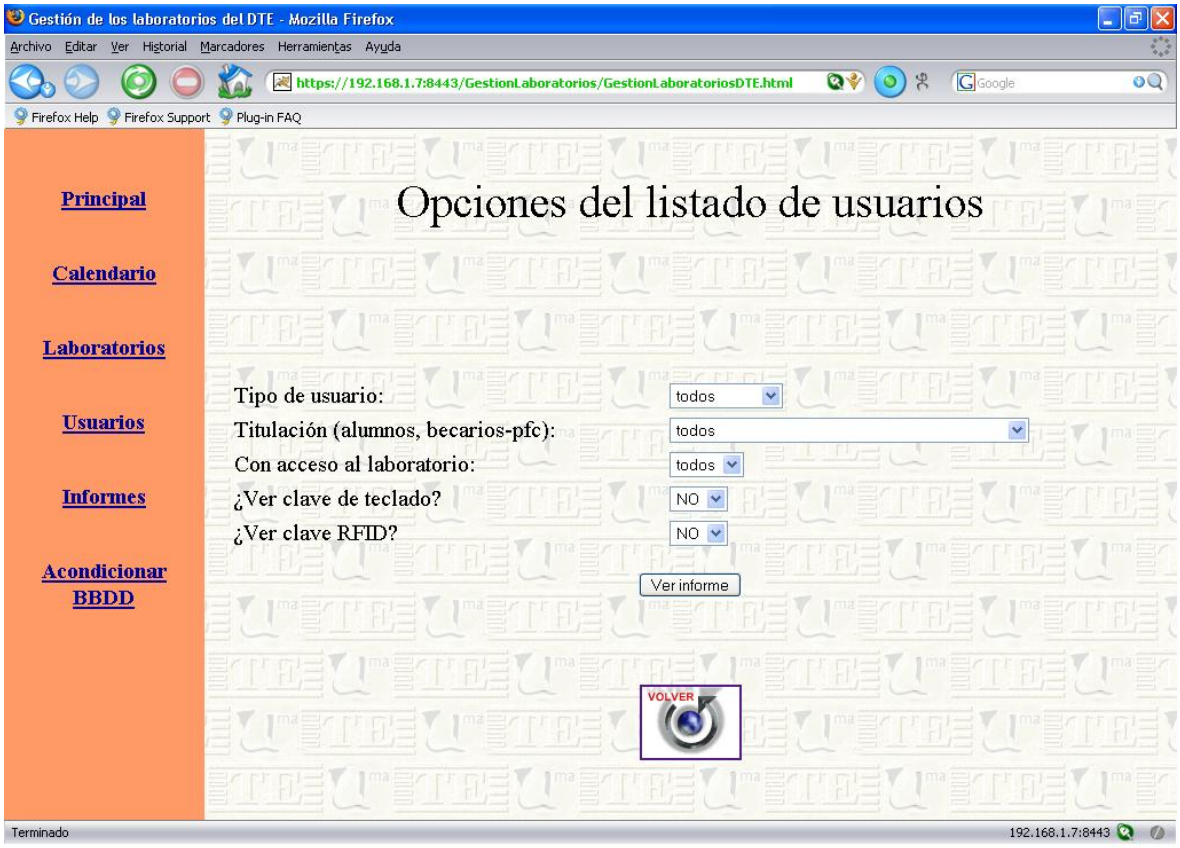

Figura 5.27. Opciones del listado de usuarios

Hay dos últimas opciones, que son las de "¿Ver clave de teclado?" y "¿Ver clave RFID?", que realmente no son filtros. Simplemente muestra o no la clave de teclado y/o RFID de cada usuario en el informe. Esto se hace de cara a que si se imprime esta información, no haya riesgo de que se divulguen las claves de acceso. Todas esas opciones se pueden combinar entre sí para obtener un informe de lo que se está buscando exactamente. Una vez marcadas las opciones que se deseen (de no marcar ninguna aparecerían todos los usuarios sin verse su clave) se obtiene un informe en formato HTML como el de la figura 5.28. En el mismo se puede apreciar que las claves no son visibles, que los usuarios aparecen ordenados por su apellido en la primera columna y que el tipo de usuario aparece en la última columna para ver los campos más interesantes de un vistazo.

Por último, en la figura 5.29 se muestra el informe en formato PDF de lo mismo que aparece en la figura 5.29, el cual se obtiene a partir del botón habilitado para ello, "Obtener informe en formato PDF".

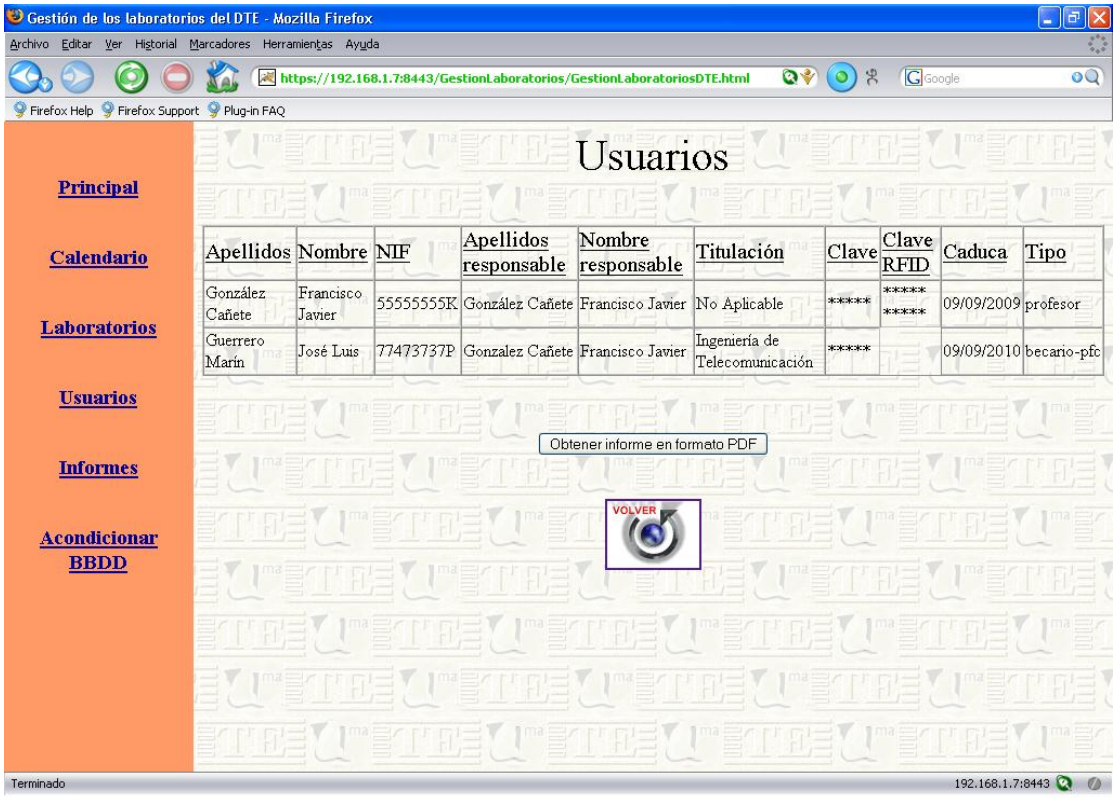

Figura 5.28. Informe de los usuarios en formato HTML

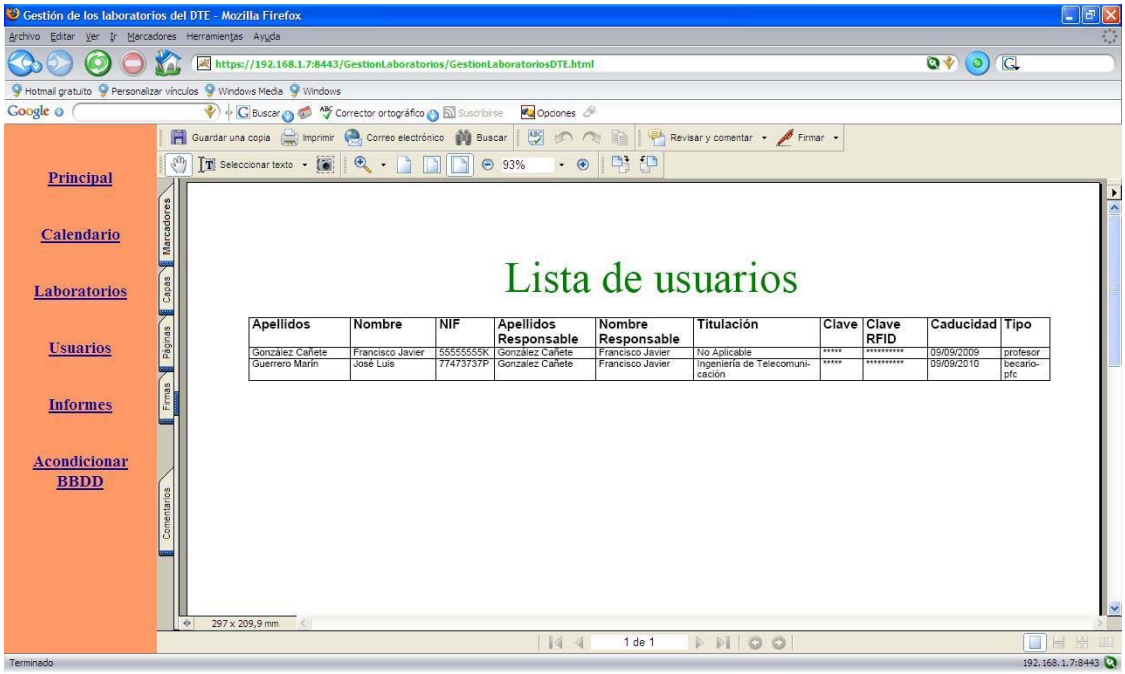

Figura 5.29. Informe de los usuarios en formato PDF

Finalmente, la última opción dentro de los informes es “Registro de accesos”, que permite obtener un informe completo de todos los intentos de acceso a los laboratorios del DTE. Al igual que en el caso del listado de los usuarios proporciona una serie de opciones

por las que filtrar los registros de la base de datos, de manera que se muestren sólo los datos que realmente interesan. Estas opciones son las que aparecen en la figura 5.30.

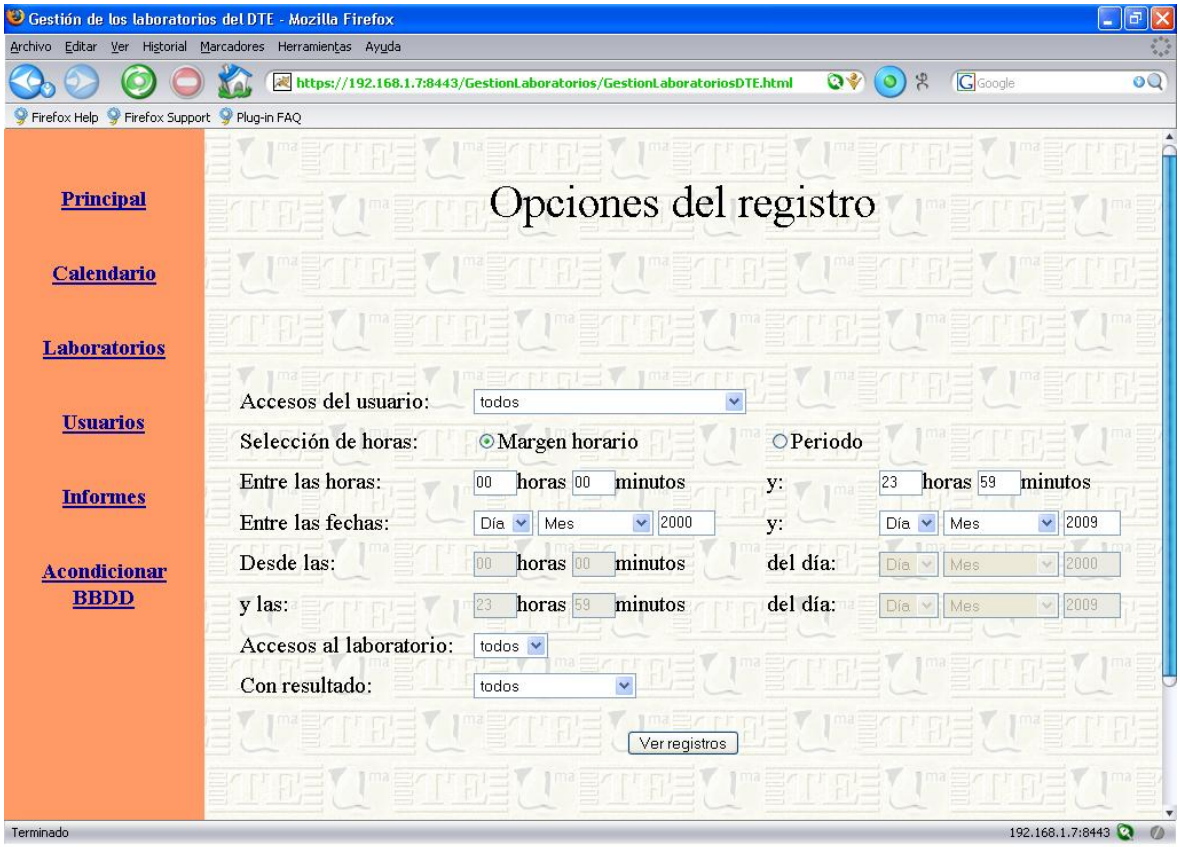

Figura 5.30. Opciones del registro de accesos

Estas opciones se detallan a continuación:

- "Accesos del usuario": permite ver los accesos de un usuario en concreto o de todos los usuarios.
- "Selección de horas": dos alternativas son posibles en este caso, "Margen horario", que permite seleccionar un rango de horas entre una fecha de inicio y una de fin, y "Período", que permite seleccionar un espacio completo de tiempo entre una fecha de inicio y una de fin. Cuando se marca una de las alternativas, se habilitan los campos correspondientes a esa alternativa y se deshabilitan los correspondientes a la otra.
- "Accesos al laboratorio": permite ver los accesos a un determinado laboratorio o a todos.
- "Con resultado": permite ver los accesos exitosos, no exitosos o ambos.

Por defecto todas estas opciones están marcadas para ver todos los registros existentes en la base de datos. Cuando se han marcado todos los filtros que se desean y se continúa viendo el registro de accesos aparece una pantalla como la de la figura 5.31.

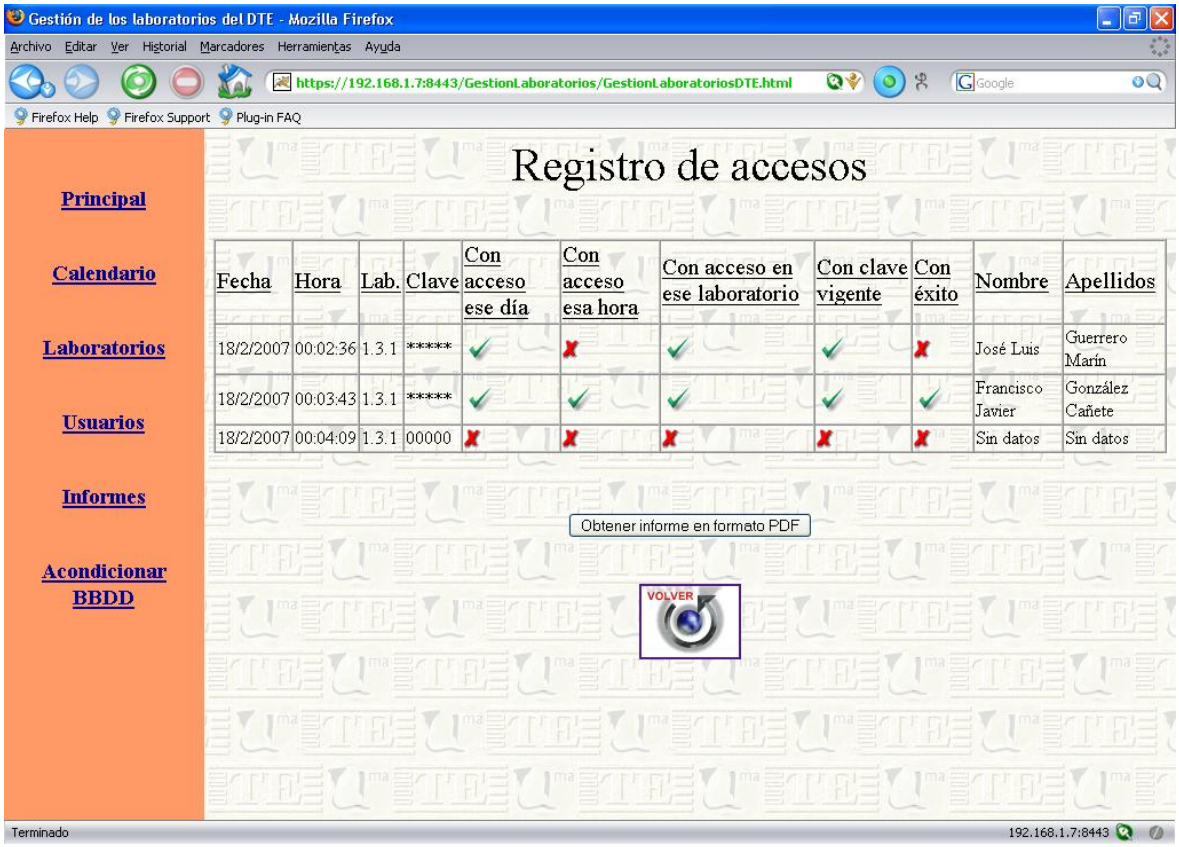

Figura 5.31. Informe del registro de accesos en formato HTML

En el informe en formato HTML de la figura 5.31 se puede ver como aparecen todos los datos relevantes de los intentos de acceso a los laboratorios. En primer lugar aparecen ordenados según la fecha y hora, seguidos de la clave introducida para el ingreso al laboratorio. En caso de ser una clave válida no se muestra su valor para evitar que, una vez impreso, el informe pudiera acabar cayendo en malas manos y ser utilizado con fines maliciosos. Los siguientes cuatro campos indican el motivo (día incorrecto, hora incorrecta, laboratorio al que no se tiene acceso o clave caducada) por el que no se da acceso al laboratorio en caso de que así sea. El siguiente indica si finalmente se tiene acceso o no (será afirmativo si se cumplen las cuatro condiciones anteriores) y los dos últimos identifican al usuario que ha intentado entrar en caso de que se reconozca.

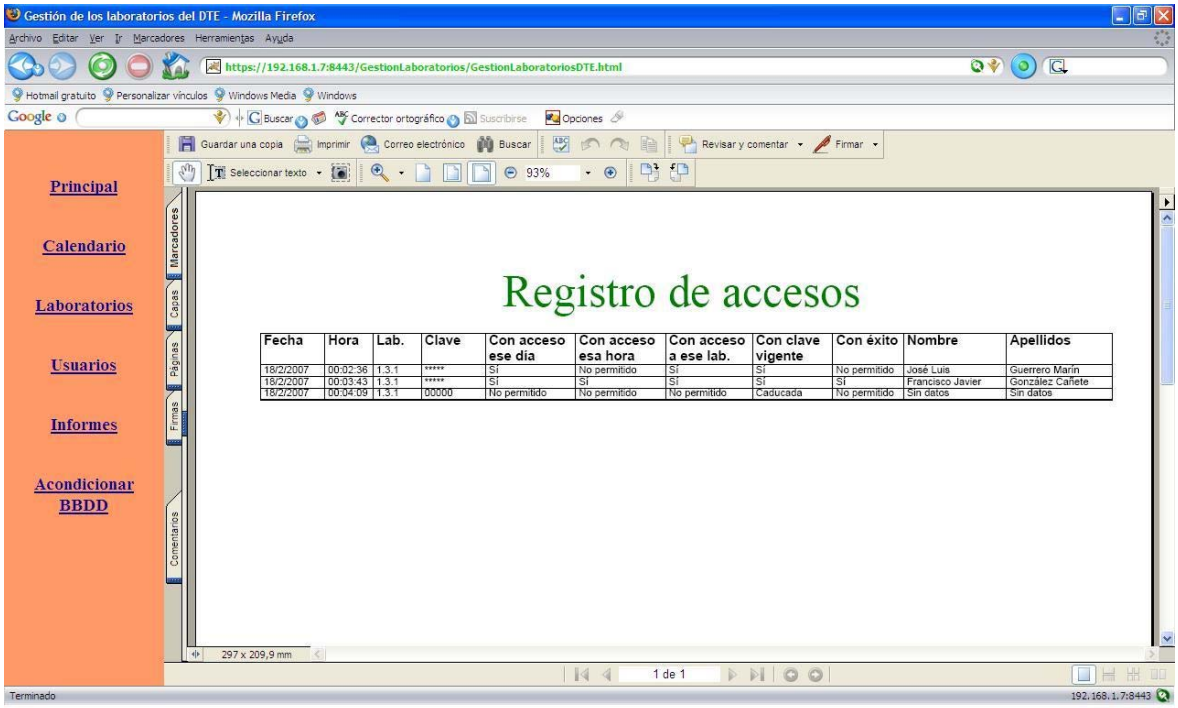

Figura 5.32. Informe del registro de accesos en formato PDF

Si finalmente se procede a obtener el informe en formato PDF del registro de accesos se obtiene un documento como el de la figura 5.32, que resulta de la conversión dinámica a formato PDF de los datos de la figura 5.31.

5.2.2.6 Sección Acondicionar BBDD

Esta última opción se incluyó al final del proyecto para evitar que los registros de accesos que vienen almacenados en la base de datos crecieran indefinidamente. Ya que cada acceso o intento de acceso a cualquier laboratorio viene registrado en la base de datos, al cabo de un cierto tiempo, la cantidad de estos datos es bastante considerable y, normalmente, los registros más antiguos van perdiendo importancia. Cuando se considere que ya no tienen relevancia se pueden eliminar gracias a esta opción. Para ello simplemente se debe indicar la fecha a partir de la cual se deben borrar los registros (desde esa fecha hacia atrás), tal y como se puede ver en la figura 5.33.

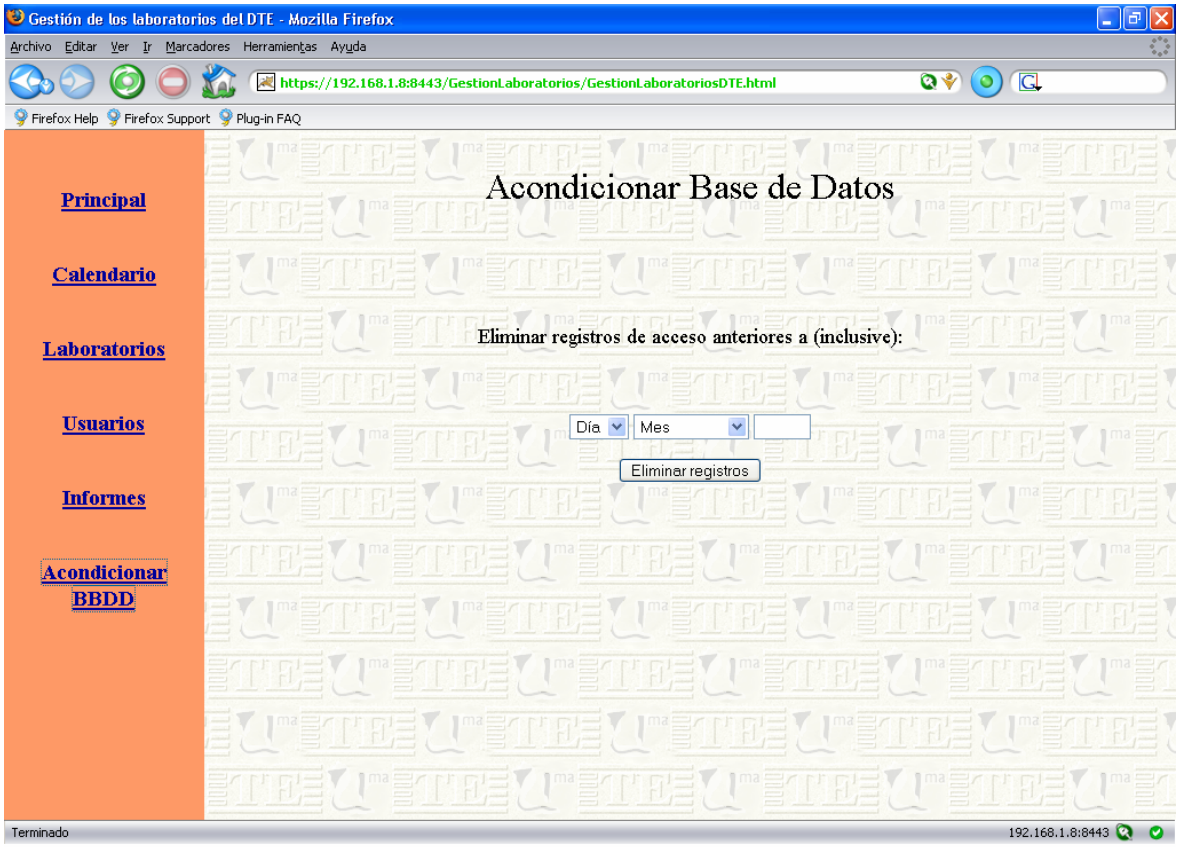

Figura 5.33. Acondicionar Base de Datos

5.2.3 SERVIDOR DE LA APLICACIÓN WEB

En el servidor, lo único que hace falta para ponerlo en marcha es arrancar el servidor de aplicaciones Tomcat. Para ello se obra de la siguiente manera:

- En Windows: se arranca el monitor Tomcat y se hace clic con el botón derecho sobre el icono de la barra de tareas, seleccionando “Start service”. Para apagarlo se hace lo mismo seleccionando “Stop service”.
- En Ubuntu Linux: en una consola se teclea “sudo $CATALINA_HOME/bin/./startup.sh” para arrancar el servidor o “sudo $CATALINA_HOME/bin/./shutdown.sh” para pararlo.

En el caso en que se quisiera cambiar la contraseña de acceso a la aplicación Web, fijada por defecto como *pelotero*, sólo habría que cambiar en la base de datos alojada en el servidor el valor de un registro de una tabla. En concreto, en la tabla *usuariosTomcat*, habría que modificar el valor del campo *password_usuario*. Esto se puede hacer sencillamente con la sentencia “*UPDATE usuariosTomcat SET password_usuario = nuevoPassword;*”. Para hacer este cambio antes de desplegar la base de datos simplemente

habría que cambiar en el archivo *configuracion.xml* de la carpeta *Instalación sistema/Aplicaciones/configuracion BBDD* del CD del proyecto la sentencia "*INSERT INTO usuariosTomcat VALUES ('root', 'pelotero'),('tomcat', 'tomcat')*" modificando los valores de "root" y "pelotero" por el usuario y la contraseña respectivamente que se deseen. Al ejecutar el archivo de despliegue de la base de datos, *Configuracion.jar*, tomará los valores que se pasan en el archivo *configuracion.xml*.

5.2.4 CONSIDERACIONES SOBRE SEGURIDAD

En el desarrollo de este proyecto, la seguridad ha sido un factor muy importante, ya que se trata del acceso físico a unos laboratorios con material valioso. Es por ello que una de las herramientas usadas para identificar al técnico que deba gestionar el los accesos ha sido la identificación por medio de un certificado digital, además del nombre de usuario y su contraseña. Por otra parte también se ha creado un certificado digital para el servidor, que se ha almacenado en el archivo ".keystore" en la ruta "GestionLaboratorios/ssl" que crea el archivo desplegable "GestionLaboratorios.war" al copiarlo a la carpeta "webapps" del Tomcat. Dicho certificado tiene una validez de 10 años desde el momento de su creación (08/09/2006), así como el certificado digital del cliente, por lo que, pasado ese tiempo deberán renovarse o crearse otros nuevos. O bien crear otros con un periodo de validez más corto, si así se considera más seguro. Para crearlos o renovarlos se deberá tener instalado OpenSSL y el JDK (Java Development Kit). La metodología para crear unos certificados nuevos con la herramienta OpenSSL es la siguiente:

AUTORIDAD CERTIFICADORA (CA)

1. Crear directorios para almacenar las claves de autoridad certificadora (CA), servidor y cliente. Se asume "ssl/ca", "ssl/server" y "ssl/client" respectivamente.
2. Crear una clave privada y una petición de certificado para la CA mediante: **openssl req -new -newkey rsa:1024 -nodes -out ssl/ca/ca.csr -keyout ssl/ca/ca.key**
3. Crear un certificado autofirmado para la CA propia (donde en lugar de 3650 se pueden poner el número de días válidos que se quieran): **openssl x509 -trustout -signkey ssl/ca/ca.key -days 3650 -req –in ssl/ca/ca.csr -out ssl/ca/ca.pem**
4. Importar el certificado de la CA en el almacén de claves de autoridades certificadoras del JDK usando "bin/keytool" dentro de la ruta donde esté instalado el JDK de Java ($JAVA_HOME): **keytool -import -keystore**

$JAVA_HOME/jre/lib/security/cacerts –file ssl/ca/ca.pem -alias my_ca. (La clave por defecto es “changeit”)

5. Crear un archivo para guardar los números de serie de las CA. El archivo empieza con el 2. Es decir, basta con crear un archivo de texto que sólo tenga escrito “02”. Se considera que este archivo será “ssl/ca/ca.srl”.

SERVIDOR WEB

6. Crear un almacén de claves para el servidor mediante: **bin\keytool -genkey -alias tomcat -keyalg RSA -keysize 1024 –storepass pelotero** (“pelotero” es la clave elegida para el almacén de claves)
7. Crear una petición de certificado para el servidor mediante: **bin\keytool -certreq -keyalg RSA -alias tomcat –file ssl/server/server.csr –keypass pelotero –storepass pelotero** (aquí también se ha elegido “pelotero” como clave del certificado). Abrir la petición de certificado “server.csr” creada con un editor de textos y cambiar “NEW CERTIFICATE REQUEST” por “CERTIFICATE REQUEST”.
8. Firmar con la entidad certificadora la petición de certificado del servidor mediante: **openssl x509 -CA ssl/ca/ca.crt -CAkey ssl/ca/ca.key –CAserial ssl/ca/ca.srl -req -in ssl/server/server.csr –out ssl/server/server.crt -days 3650**
9. Importar el certificado de servidor en el almacén de claves del servidor mediante: **bin\keytool -import -alias tomcat –keypass pelotero –storepass pelotero -trustcacerts -file c:/openssl/bin/ssl/server/server.crt**. Debe aparecer un mensaje de confirmación que indica que el certificado ha sido instalado.
10. Importar el certificado de la CA en el almacén de claves del servidor (este paso es necesario sólo si, como en este caso, se habilita la autenticación del cliente por certificado digital). Esto se hace mediante: **bin\keytool -import -alias my_ca –keypass pelotero –storepass pelotero -trustcacerts -file c:/openssl/bin/ssl/ca/ca.crt**
11. Habilitar la conexión SSL en Tomcat. Esto ya se ha hecho, modificando el archivo “server.xml” que se provee.

CLIENTE WEB

12. Crear una petición de certificado de cliente mediante: **openssl req -new -newkey rsa:1024 -nodes -out ssl/client/client1.req –keyout ssl/client/client1.key**. El CN (*Common Name*) debe coincidir con el nombre que se haya dado al usuario

autenticado, definido en el archivo “server.xml”, en el dominio de seguridad del Tomcat o “*Tomcat Realm*”. En este caso se ha usado “*root*”.

13. Firmar el certificado de cliente con la autoridad certificadora mediante: **openssl x509 -CA ssl/ca/ca.pem -CAkey ssl/ca/ca.key –CAserial ssl/ca/ca.srl -req -in ssl/client/client1.req –out ssl/client/client1.pem -days 3650**
14. Generar un archivo de tipo PKCS12 que contenga la clave y el certificado del servidor mediante: **openssl pkcs12 -export -clcerts -in ssl/client/client1.pem –inkey ssl/client/client1.key -out ssl/client/client1.p12 –name "certificadocliente"**
15. Importar el certificado de cliente al navegador que se use para gestionar la Web.
16. Habilitar la autenticación por certificado de cliente en el Tomcat. Esto se realiza modificando el archivo “server.xml” del Tomcat, cambiando el valor de “clientAuth=false” a “clientAuth=true”. Esta modificación ya está realizada en el archivo “server.xml” que se provee.

Siguiendo todos estos pasos, el almacén de claves del servidor (“.keystore”), que luego se mete en la carpeta “GestionLaboratorios/ssl” se habrá creado en el directorio raíz de trabajo del usuario (en Windows sería en “C:\Documents and Settings\usuario”) y el certificado de cliente que se debe importar en el navegador se crea en la carpeta “client”, en la ruta donde se haya creado la estructura “ssl/client”.

CAPÍTULO 6: PLAN DE PRUEBAS

6.1 CONSIDERACIONES PREVIAS

Al margen de la infinidad de pruebas realizadas en otros equipos en la fase de desarrollo, se ha diseñado un plan de pruebas para aislar y resolver cualquier tipo de error que pudiera producirse tanto en el funcionamiento como en la presentación de cualquiera de los elementos que componen el sistema. Para completar con éxito este plan de pruebas se han usado dos equipos de la Universidad de Málaga conectados a través de la red de datos de la misma. Estos equipos tienen las siguientes características:

- Equipo nº 1: AMD Athlon (TM) XP1500+ 1,34GHz, 1 GB de memoria RAM con sistema operativo Windows XP Profesional Versión 2002 Service Pack 2.
- Equipo nº 2: AMD Athlon (TM) 64 Processor 3200+ 2,21GHz, 1 GB de memoria RAM con sistema operativo Windows XP Profesional x64 Edition Versión 2003 Service Pack 1 y Ubuntu 6.06.1 LTS (Dapper Drake) en su versión para AMD de 64 bits.

En las pruebas se han usado tres configuraciones diferentes:

- Equipo nº 1 como servidor y equipo nº 2 como cliente usando en este último Windows como sistema operativo.
- Equipo nº 1 como servidor y equipo nº 2 como cliente usando en este último Linux como sistema operativo.
- Equipo nº 2 como servidor usando como sistema operativo Linux y equipo nº 1 como cliente.

Además hay que señalar que en los casos en los que el equipo cliente ha usado como sistema operativo Windows se han hecho las pruebas utilizando como navegadores los más populares en diferentes versiones: Internet Explorer 6 de 32 y 64 bits, Internet Explorer 7 de 32 y 64 bits y Mozilla Firefox 1.5 y 2.

En todos los casos se han instalado todas las características del servidor (base de datos, aplicación Web y servicio Web) en el mismo equipo usado como servidor y todas las del cliente (navegador con su certificado de cliente y aplicación de apertura de puertas) en el mismo equipo usado como cliente. Igualmente se podría haber hecho de una forma

distribuida, tal y como se explica en el capítulo de instalación, pero realmente no aporta nada que no se contemple en las pruebas realizadas.

El plan de pruebas se divide en dos partes:

1. Parte *hardware*: todo lo relacionado con el acceso físico al laboratorio, incluyendo la seguridad del servicio Web.
2. Parte *software*: todo lo relacionado con la funcionalidad del sitio Web y la seguridad del acceso al mismo.

6.2 PRUEBAS DE LA PARTE HARDWARE

Las pruebas realizadas y superadas con éxito en la parte *hardware* han sido las siguientes:

- Respecto a la apertura con el teclado:
    1. Se abre con clave correcta
    2. No se abre con clave incorrecta
    3. No admite ningún símbolo inicial a la entrada que no sea "/" ni de finalización que no sea "*"
    4. No admite más de 10 símbolos de entrada
    5. Si pasan más de 5 segundos sin teclear nada se borran todos los símbolos
    6. El sistema no se bloquea con pulsación prolongada
    7. Se registran todos los intentos de acceso
    8. Si se pierde la comunicación con el servidor la caché responde abriendo a una clave válida introducida en las 24 horas anteriores
    9. Si se pierde la comunicación con el servidor la caché responde denegando el acceso a una clave válida que no ha sido introducida en las 24 horas anteriores
    10. Se abre si se dan todas las condiciones: fecha permitida a ese usuario, hora permitida a ese usuario, laboratorio permitido a ese usuario, caducidad de la clave de ese usuario y si no es un usuario privilegiado (técnico o profesor) siempre que esté dentro de los horarios del laboratorio y no es festivo
    11. No se abre si no se dan todas las condiciones del punto anterior y queda registrado el motivo

- Respecto a la apertura con el lector:
  1. Se abre con una tarjeta con clave correcta
  2. No se abre con una tarjeta con clave incorrecta
  3. Se registran todos los intentos de acceso
  4. Si se pierde la comunicación con el servidor la caché responde abriendo a una clave válida introducida en las 24 horas anteriores
  5. Si se pierde la comunicación con el servidor la caché responde denegando el acceso a una clave válida que no ha sido introducida en las 24 horas anteriores
  6. Se abre si se dan todas las condiciones: fecha permitida a ese usuario, hora permitida a ese usuario, laboratorio permitido a ese usuario, caducidad de la clave de ese usuario y si no es un usuario privilegiado (técnico o profesor) siempre que esté dentro de los horarios del laboratorio y no es festivo
  7. No se abre si no se dan todas las condiciones del punto anterior y queda registrado el motivo

- Respecto a la seguridad:

  Para comprobar que la clave viaja encriptada se ha usado el IDE Eclipse sacando por pantalla los mensajes transmitidos y viendo como efectivamente aparecen cifrados.

6.3 PRUEBAS DE LA PARTE SOFTWARE

Las pruebas realizadas en este sentido engloban todo lo que no se ha abarcado en el punto anterior, es decir el funcionamiento del sitio Web y la seguridad del mismo.

- Respecto al funcionamiento del sitio Web:
  1. Comprobar que todos los enlaces funcionan correctamente.
  2. En el calendario, que se guardan los cambios de los 3 años que se almacenan.
  3. En el calendario, que no se puede poner un día incorrecto para un mes (30 de Febrero…).
  4. En el calendario, que se añaden las fiestas correctamente.

5. En el calendario, que se mueven las fiestas correctamente.
6. En el calendario, que se borran las fiestas correctamente.
7. En el calendario, que no se puede poner una hora de inicio posterior a una hora de fin.
8. En el calendario, que no se puede poner una fecha de inicio posterior a una de fin.
9. En el calendario, que se añade un tramo de horario especial correctamente.
10. En el calendario, que se modifica un tramo de horario especial correctamente.
11. En el calendario, que al seleccionar un tramo de horario para modificar se vean los horarios vigentes antes de modificar.
12. En el calendario, que se borra un tramo de horario especial correctamente.
13. En el calendario, que se confirman todas las operaciones.
14. En los laboratorios, que no se pueden dejar campos vacíos.
15. En los laboratorios, que se añade un laboratorio correctamente.
16. En los laboratorios, que se modifica un laboratorio correctamente.
17. En los laboratorios, que se ven los datos de los laboratorios correctamente.
18. En los laboratorios, que se borra un laboratorio correctamente.
19. En los laboratorios, que se confirman todas las operaciones.
20. En los usuarios, que no se puede meter el NIF incorrectamente (español o extranjero).
21. En los usuarios, que cuando se mete un usuario de tipo privilegiado (técnico o profesor) se deshabilitan las opciones que no son aplicables.
22. En los usuarios, que no se pueden dejar campos vacíos.
23. En los usuarios, que no se pueden introducir fechas de caducidad incorrectas (año anterior al 2006, 30 de febrero…).
24. En los usuarios, que cuando se mete un usuario de tipo no privilegiado hay que asignarle un laboratorio.
25. En los usuarios, que se ven los datos de los usuarios correctamente.
26. En los usuarios, al modificar uno, que se ven todos sus datos anteriores.
27. En los usuarios, al modificar un laboratorio de su lista de laboratorios con acceso, que se ven todos los datos vigentes antes de modificar.
28. En los usuarios, al modificar uno, que no se pueden dejar campos vacíos.

29. En los usuarios, al modificar uno, que si se modifica el tipo a privilegiado (técnico o profesor), se deshabiliten los campos oportunos y se hagan los cambios necesarios en la base de datos si se confirma la operación, y al hacer el cambio inverso se habiliten esos campos.
30. En los usuarios, al modificar uno, que si se ha seleccionado un usuario privilegiado sólo tenga habilitados los campos de sus datos personales, ya que el resto no tienen sentido en ese tipo de usuario.
31. En los usuarios, al modificar uno, y añadirle laboratorio, que no se pueden dejar campos vacíos.
32. En los usuarios, al modificar uno, y modificar un laboratorio de los ya asignados, que no se pueden dejar campos vacíos.
33. En los usuarios, y al modificarlos, que se confirman todas las operaciones.
34. En los informes, en el listado de fiestas, que hay que seleccionar un año.
35. En los informes, en el listado de fiestas, que se ven correctamente todas las fiestas y tramos de horario especial del año seleccionado y se crea el PDF correctamente.
36. En los informes, en el listado de laboratorios, que se ven correctamente todos los laboratorios y se crea el PDF correctamente.
37. En los informes, en el listado de usuarios, que si no se selecciona ninguna opción se ven todos los usuarios.
38. En los informes, en el listado de usuarios, que se puede hacer todas las combinaciones posibles para visualizar exactamente lo que se desea.
39. En los informes, en el listado de usuarios, que se crea el PDF con los datos que se deseen correctamente.
40. En los informes, en el registro de accesos, que si no se selecciona ninguna opción se ven todos los intentos de acceso.
41. En los informes, en el registro de accesos, que se puede hacer todas las combinaciones posibles para visualizar exactamente lo que queremos.
42. En los informes, en el registro de accesos, que se puede cambiar entre margen de horas e intervalo de tiempo y se habilitan y deshabilitan los campos oportunos.
43. En los informes, en el registro de accesos, que no se pueden poner horas de inicio posteriores a las de fin.

44. En los informes, en el registro de accesos, que no se pueden poner fechas de inicio posteriores a las de fin.
45. En los informes, en el registro de accesos, que se crea el PDF correctamente con los datos que hemos seleccionado.
46. En el acondicionamiento de la base de datos, que no se puedan dejar campos en blanco.
47. En el acondicionamiento de la base de datos, que no se pueden poner fechas incorrectas (30 de febrero…).
48. En el acondicionamiento de la base de datos, que se pide confirmación de la operación.
49. En el acondicionamiento de la base de datos, que se eliminan los registros seleccionados de la base de datos.

- Respecto a la seguridad:

Comprobar que si no se tiene el certificado de cliente instalado y se mete el usuario y la contraseña correctamente no se puede acceder al sitio Web.

6.4 CONSIDERACIONES ADICIONALES

Aparte del plan exhaustivo de pruebas, como ya se dijo en el capítulo 5, el sistema se ha probado (con diferentes combinaciones de cliente y servidor) en los siguientes sistemas operativos:

- Windows 98 Segunda Edición
- Windows 2000 Profesional
- Windows XP Tablet PC Edition 2005 Versión 2002 Service Pack 2
- Windows XP Profesional Versión 2002 Service Pack 2
- Windows XP Profesional x64 Edition Versión 2003 Service Pack 1
- Vesión beta de Windows Vista
- Ubuntu Linux 6.06 Dapper Drake para procesador Intel de 32 bits
- Ubuntu Linux 6.06 Dapper Drake para procesador AMD de 64 bits

De todos esos sistemas operativos, en el único en que se encontraron problemas fue en Windows Vista, a la hora de usarlo como servidor, debido a que la instalación del JRE aún (a la fecha de las pruebas) no está muy afinada y por lo tanto el Tomcat no puede arrancar.

CAPÍTULO 7: CONCLUSIONES Y LÍNEAS FUTURAS

El objetivo de este Proyecto Fin de Carrera ha sido conseguir un sistema alternativo, totalmente renovado y mejorado para gestionar el acceso a los laboratorios del Departamento de Tecnología Electrónica. Dicho sistema alternativo comprende los siguientes campos:

- Acceso físico a los laboratorios, para lo cual ha habido que diseñar un interfaz adecuado entre el ordenador local del laboratorio y el sistema de apertura de puertas.
- *Software* que gestiona la comunicación entre el ordenador local del laboratorio y el sistema de apertura de puertas.
- *Software* que gestiona la comunicación entre el ordenador local del laboratorio y un servicio Web, el cual puede acceder a la información necesaria para saber si un usuario tiene acceso al laboratorio en un determinado momento o no.
- Sitio Web de gestión de los accesos a los laboratorios del DTE, desde el cual se pueden gestionar:
    - Los días y las horas a las que se puede tener acceso a los laboratorios.
    - Los laboratorios que se adhieran a este sistema.
    - Los usuarios de los laboratorios con sus permisos de acceso.
    - Obtener informes de los días festivos y con horario especial, de los laboratorios, de los usuarios y del registro de accesos a cualquier laboratorio que emplee este sistema.
    - Limpieza de los registros antiguos de la base de datos.

Todo ello ha sido resuelto asegurando total compatibilidad probada con Windows y Linux consiguiendo así un sistema potente y multiplataforma. Se han generado además documentos de ayuda en el popular formato "*Javadoc*", incluidos en el CD del proyecto, donde se describen en detalle todas las funciones y clases desarrolladas.

Todos los requisitos que se planteaban en el anteproyecto, el capítulo 1 de esta memoria, y aquí a modo de resumen en las líneas anteriores, se han resuelto, incluyendo mejoras que han ido surgiendo tales como:

- Obtener informes no sólo del registro de accesos, sino de todos los elementos almacenados en la base de datos: calendario, laboratorios y usuarios.
- Incluir una opción en el sitio Web de gestión de los accesos para la limpieza de los registros más antiguos de la base de datos, eliminando así datos inútiles y optimizando el rendimiento de la misma.
- Conseguir un *software* para el interfaz entre el ordenador local del laboratorio y el sistema de apertura de puertas capaz de reconocer automáticamente el sistema operativo en que se encuentra, eliminando así una fuente de errores y haciéndola más sencilla de cara al usuario.
- Incluir una opción en el software para el interfaz entre el ordenador local del laboratorio y el sistema de apertura de puertas para definir el tiempo que se acciona el abrepuertas y qué puerto del ordenador será el que accione el abrepuertas.
- Ampliar el sistema de entrada, no sólo al acceso por teclado, sino al reconocimiento por tarjetas RFID integrándolo en un sistema mucho más completo y versátil. Para reconocer dichas tarjetas RFID se usa un lector adecuado para ello conectado al puerto serie, y cuyos parámetros son completamente configurables. Además, el sistema no está limitado a usar cualquier otro tipo de lector, siempre y cuando se conecte al puerto serie y la clave siga el mismo protocolo. De cualquier modo, si el protocolo fuese diferente, se ha preparado el sistema de manera que los cambios serían mínimos.

A modo de resumen, tras implementar todas estas características, se puede asegurar que el sistema cumple con los siguientes criterios:

- **Multiplataforma**: el sistema puede ser instalado tanto en Windows como en Linux con total garantía de funcionamiento siempre que se sigan las instrucciones de instalación del capítulo 5. Por otra parte el sitio Web es accesible desde los navegadores más populares (Internet Explorer y Mozilla Firefox) con toda su funcionalidad.

- **Distribuido**: los diferentes componentes del sistema (sistema gestor de bases de datos, servicio Web y aplicación Web) pueden ser instalados en máquinas distintas.
- **Seguridad**: se ha garantizado la seguridad en la comunicación con el sitio Web por medio de SSL usando certificados de servidor y de cliente, y autenticación por usuario y contraseña. Además se ha encriptado la comunicación de la aplicación que acciona el abrepuertas con el servicio Web.
- **Compatibilidad de idiomas**: en el desarrollo del sistema la codificación empleada ha sido UTF-8, que es el estándar que incluye todos los símbolos de todos los lenguajes conocidos en el mundo, por lo que se asegura que cualquier símbolo va a estar representado correctamente tanto en el sitio Web, como en los informes en PDF que se pueden obtener.
- **Facilidad de uso**: al usuario se le presenta un interfaz lo más sencillo posible en todo momento, tanto en el sitio Web como en la aplicación que acciona el abrepuertas.
- **Versatilidad**: es posible cambiar la configuración del sistema (puertos de los servidores, usuarios, contraseñas, sistema gestor de bases de datos, localización del servicio Web, puerto para el sistema de apertura de puertas, puerto para el lector, parámetros del lector, y tiempos de apertura de puertas) simplemente modificando valores de un archivo XML o a través de menús desplegables de la aplicación.
- **Escalable y extensible**: dado que la aplicación que acciona el abrepuertas toma la lista de laboratorios directamente de un servicio Web, el hecho de agregar más laboratorios al sistema no implica modificar absolutamente nada del sistema, ya que la aplicación recibirá sus datos automáticamente. Además, todos los informes se generan dinámicamente, por lo que en este sentido tampoco necesita cambios por el hecho de introducir o borrar usuarios o laboratorios. La única ampliación que necesitaría de modificaciones sería extender este sistema a otros departamentos, en cuyo caso los cambios serían muy simples. Como además se ha separado la lógica de la representación, por medio de hojas de estilo, cualquier modificación que afecte a la apariencia física del interfaz gráfico se puede hacer de forma separada e integrarlo posteriormente con la plataforma.

Aunque se han cubierto todos los objetivos planteados en el anteproyecto y los que han surgido a lo largo de la creación del mismo, se comentan a continuación las siguientes ampliaciones de interés:

- Extender el sistema a otros departamentos de la Universidad de Málaga o de otras universidades.
- Hacer una versión reducida del sitio Web para que sea accesible desde dispositivos móviles.
- Dotar al sitio Web de capacidad para generar estadísticas y gráficas de uso de los laboratorios de un usuario, o de grupos de usuarios.
- Dotar a la aplicación de apertura de puertas de la capacidad de generar una alarma, en forma de mensaje al móvil del técnico, si se detectan un cierto número de intentos de acceso fallido en un breve intervalo de tiempo sobre un mismo laboratorio.

Bibliografía

[1] Subrahmanyam Allamaraju et al., “Programación Java Server con J2EE Edición 1.3” Editorial Anaya Multimedia, año 2002.

[2] http://www.eclipse.org. Página oficial del entorno de desarrollo de Eclipse.

[3] http://www.adictosaltrabajo.com/tutoriales. Portal Web con multitud de tutoriales sobre nuevas tecnologías.

[4] http://www.mysql.com. Página oficial del SGBD MySQL.

[5] http://www.w3schools.com/. Página con tutoriales sobre servicios Web.

[6] http://www.danielclemente.com/html/consejos.html. Recopilación de todo lo básico que hay que saber del lenguaje HTML.

[7] http://www.forosdelweb.com/showthread.php?t=117856#post326203. FAQ (*Frequently Asked Questions*) sobre JavaScript.

[8] Álvaro Zabala Ordóñez, “JDBC: accediendo a BBDD desde Java”, Revistas Profesionales SL., número 105 (páginas 24-29), número 106 (páginas 24-28), número 107 (páginas 32-39), año 2003.

[9] Adolfo Aladro García, “MySQL, el SGBD de Internet”, Revistas Profesionales SL., número 110 (páginas 38-42) y número 111 (páginas 30-33), año 2004.

[10] http://sistemasorp.blogspot.com/. Página con información sobre domótica.

[11] http://www.rxtx.org/. Página con las librerías necesarias y documentación en línea para acceder al puerto serie mediante Java.

[12] http://www.javahispano.org. Portal Web de la comunidad de habla hispana sobre Java

[13] http://www.programacion.net. Portal Web en español sobre programación.

[14] http://www.openssl.org. Página donde se encuentra disponible la herramienta de creación de certificados digitales OpenSSL con su documentación.

[15] http://tomcat.apache.org. Página Web del servidor de aplicaciones Tomcat.

[16] http://es.wikipedia.org/wiki/Portada. Enciclopedia libre en Internet.

[17] http://www.cebek.com. Página Web del fabricante del módulo optoacoplador.

[18] http://www.parallax.com/detail.asp?product_id=28140. Página Web del módulo lector RFID.

[Framiñán'97] José Manuel Framiñán Torres. Manual imprescindible de Java Madrid, Anaya multimedia, 1997

[García de Jalón'99] Javier García de Jalón, José Ignacio Rodríguez, Aitor Imaz. Aprenda servlets de Java como si estuviera en primero. Tecnum, San Sebastián, 1999. Documento accesible por Internet en formato PDF en la dirección http://www.tecnun.es/asignaturas/Informat1/ayudainf/aprendainf/JavaServlets/servlets.pdf#search=%22caracter%C3%ADsticas%20servlets%22

[ISO 8879'86] International Standards Organization. ISO8879, 1986. Documento accesible por Internet en formato PDF en la dirección http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387

APÉNDICE A: PRESUPUESTO

En la realización de este presupuesto se han tomado como referencias los precios individuales de los componentes que se han usado en el prototipo de prueba, que es una versión totalmente funcional del sistema. En caso de implantarse ampliamente este sistema es los precios vendrían adecuados al volumen de compra de los componentes.

Toda la parte *software* es gratuita por la filosofía misma de este proyecto en que todas las herramientas usadas son *software* libre. Por lo tanto la única parte que implica un gasto económico es la que respecta al *hardware*, es decir, la referente al acceso físico. Los ordenadores usados como servidores y el ordenador local del laboratorio se supone que ya están disponibles.

Componente	**Unidades**	**Precio unitario (€)**
Módulo optoacoplador CEBEK	1	12,10
Fuente de alimentación 12 V 300 mA CEBEK	1	14
Cable serie RS-232	1	6
Cable paralelo bicolor (por metro)	2	0,22
Conectores de alimentación (macho y hembra)	1	1,79
Conectores de paso (macho y hembra)	1	1
Teclado	1	25
Lector RFID Parallax 28140	1	44
Tarjeta RFID	1	3

Tabla A.1. Presupuesto

En la tabla A.1 se detallan los precios de cada componente del sistema de apertura de puertas funcionando para un laboratorio. La suma total asciende a 50,23€ para el sistema sólo con teclado que se ha montado y con los precios (IVA incluido) de los componentes usados para ese prototipo. En el caso de usar el sistema sólo con el lector la suma sería 69,23€ más las tarjetas que se usaran (a 3€ cada una). En el caso de usar ambos sistemas simultáneamente en un laboratorio, el coste unitario sería 94,23€ (más tarjetas), siempre teniendo en cuenta que los precios se reducirían al aumentar el volumen de compra.

Hay que resaltar que este sería el único coste de la puesta en funcionamiento del sistema, ya que el resto de herramientas usadas son gratuitas y la instalación de los componentes software se puede realizar en Linux con todas las garantías.