Proyecto

Notas Online. Sistema web para notas.

ÍNDICE

1. Estructura de la aplicación

La aplicación consta de la siguiente estructura:

1. Ventana principal:

    1. Consiste en un login partido, un campo para rellenar el dni y la contraseña del profesor y el otro para que se identifiquen los alumnos.

    2. Una vez identificados se comprueban las credenciales, (el usuario y la contraseña son correctos). Si falta cualquier campo o alguno es incorrecto, muestra el error 405.

2. Cuenta de profesor:

    1. Una vez iniciada la sesión de profesor, se abre la ventana con el título "Notas OnLine. Asignatura del profesor (nombre del profesor)", links con las asignaturas que imparte y un botón de cierre de sesión.

    2. Cuando se pulsa en una asignatura que queremos consultar, se nos muestra una lista de los alumnos matriculados en esa asignatura (definido en el servlet listaAlumnos.java), como título "Alumnos matriculados" y abajo se indica el nombre de la asignatura. Además hay tres botones, uno de volver atrás, uno para sacar la nota media y otro para guardar la nota y se actualice.

En la lista de alumnos, indicados por dni, para consultar más detalles del algún alumno, pulsamos en su dni. En esta página se muestra la foto del alumno, más sus datos personales (esta funcionalidad está definida en el servlet detallesAlumnos.java). Hay un botón para volver atrás que devuelve a la lista de alumnos matriculados en la asignatura del profesor, esto se realiza con una función que vuelve atrás : *window.history.back()*.
Además donde está la lista de alumnos, tenemos un campo de texto donde se indica la nota de cada alumno.

3. Cuenta de alumno:

    1. Una vez iniciada la sesión de alumno, se abre la ventana con el título "Notas OnLine. Mis asignaturas. (Nombre del alumno)", links con las asignaturas en las que está matriculado, un botón de cierre de sesión, la foto asignada al alumnos y un botón para sacar la nota media de sus asignaturas.

    2. Cuando se pulsa en la asignatura de la nota que queremos consultar, se abre otra página donde se muestra la nota final con la foto del alumno y un botón para volver atrás que se ejecuta con el método *window.history.back()*.

2. Tablas de contenido

2.1 Clases definidas:

- **servletProfesores.java**: Después de cargar el login, con el inicio de sesión, se ejecuta este servlet que corresponde a los profesores.
    - Consta de un doGet formado por los siguientes parámetros:
    - Un **HttpSession** que define la sesión del profesor.
    - Dos Strings (**user** y **pass**) que corresponden al usuario y a la contraseña que se les pasa por el formulario del html.
    - En las variables verificar, key y cookies se asignan el user y el pass. Como la **key** es el user, asignamos a la **JSESSIONID** la key y añadimos ese valor a una **Cookie** que luego se añade al response.

- Para recuperar las asignaturas que corresponden al profesor usamos el método **getAsignaturasDeProfesor(key, cookie, user)**, que explicaré en más detalle cuando llegue a la clase funciones.

- Para recuperar las asignaturas que corresponden al alumno usamos el método **getAsignaturasDeAlumnos(key, user)**, que explicaré en más detalle cuando llegue a la clase funciones.

- Respecto a los ifs. Se trata de dos ifs diferenciados aunque muy parecidos que hacen una función parecida pero uno para alumnos y otro para profesores. Se comprueba si se trata de un profesor o un alumno con el radio button que hemos puesto en el formulario:

    - if("Profesor".equals(usuarios)){}

    - if("Alumno".equals(usuarios)){}

- En caso de que un alumno intente iniciar sesión como profesor o viceversa, se manda un error 500.

- En la última parte cabe destacar que inicializamos un text/html para que en caso de que el inicio de sesión sea correcto, (profesores().size()>0) y ademas (usuario y contraseña correctos con los código de error -1 para la contraseña y 500 para el usuario), imprima las asignaturas impartidas por el profesor. Para ello hemos hecho un for que recorre el ArrayList de "asignaturasProfe" e imprime todas las asignaturas. (Lo mismo para los alumnos)

- En caso de que falte algún campo al iniciar sesión o haya algún error tanto en el dni como en la contraseña, se muestra un error 405, mediante "response.sendError(405)".

- **listaAlumnos.java**: sirve para proporcionar la lista de alumnos con su dni y la nota, cuando el profesor selecciona una de las asignaturas que está impartiendo.

    - Se define una key y una clave, esta última recoge el valor de "asignatura" que se le pasa mediante el "servletProfesores", y la key sirve para los alumnos. Entonces se define un ArrayList para todos los alumnos y se comprueba si están matriculados en la asignatura seleccionada. En caso de ser así, devuelve una lista con su dni y la nota. Esto se implementa con el "for" definido en la parte de abajo, un for que recorre el ArrayList de los alumnos, mediante alumnos.size() y comprobamos que la variable i del for sea par, porque ahí es donde se contienen los dni, y es lo que queremos que aparezca.

  Además el println, a parte de imprimir los dnis de los alumnos, también imprime un "input" para modificar la nota que aparece.

- **detallesAlumnos.java** :

    - Recibe de "listaAlumnos.java" el dni seleccionado, llamado "alum" y lo almacenamos en user.

    - Creamos un ArrayList que recoge los detalles de los alumnos, que son el nombre y apellidos y el dni. Los pasa de JSON a texto con el bucle for que recorre. Y más abajo aparece la foto que le hemos asignado con img, que se le asigna por dni.

    -

- **funciones.java:** En esta clase se definen los métodos que implementan las funciones principales para los servlets, son los siguientes:

    - getKey(String user, String pass):

- **getListaAsignaturasdeAlumno(String key, String user)**: este método sirve para devolver las asignaturas de un alumno, para ello usamos la url indicada como urlAsigAlum para sacar las asignaturas de la API, luego asociamos esta función al método "GET" para cuando se llame en el servlet, y al final se recoge el resultado JSON y se pasa a String, por lo que se devuelve en formato texto.

- **getAsignaturasDeProfesor(String key, String cookie, String user)**: en este método se devuelven las asignaturas impartidas por un profesor. Como en el método anterior, accedemos a la url indicada por urlAsigProfes, se le asocia el método "GET" para el servlet y en el for, se devuelve en formato String la lista de asignaturas por el acrónimo(elemento JSON de la API).

- **getListaAlumnos(String key, String auxAsig):** este método sirve para devolver la lista de Alumnos matriculados en una asignatura impartida por un profesor. Como anteriormente, recuperamos de la API mediante la url los alumnos por Asignatura y le asignamos el método "GET". En este caso se sacan en formato de String los dnis del alumno y la nota (para la lista de los alumnos de un profesor se devuelven los dnis junto a las notas). Todos los parámetros entre comillas, en este caso "alumno" y "nota" son los elementos JSON de la API.

- **getDetallesAlumnos(String key, String user):** este método sirve para mostrar el dni, los apellidos, el nombre y la foto(definida en el servlet) de cada uno de los alumnos que seleccione el profesor que inicie sesión. Aquí se hace lo mismo que en el resto de los métodos pero se devuelven el dni, el nombre y los apellidos del objeto JSON correspondiente a la url asignada al principio.

- **getProfesor(String key, String dni) y getAlumnos(String key, String dni):** estos dos métodos sirven para el login del

correspondiente rol, devolver el usuario que inicia sesión con sus credenciales. Estos métodos realizan todas las tareas previas similares al resto de métodos, y devuelve el dni y el nombre del objeto JSON correspondiente al alumno o al profesor.

    - **getAsignaturas(String key, String dni):** se usa en el servlet de alumnos para devolver las asignaturas con su acrónimo y su nombre completo. La dinámica es igual que en los métodos anteriores.

- **modificarNotaAlumno.java / ajax.js**: el ajax esta relacionado con la parte html de " listaAlumnos.java". El método ajax lanza una petición mediante "PUT" al servlet modificarNotaAlumno.java que luego recibe el valor que le hemos puesto, "nota", y lo guarda y lo añade a "listaAlumnos.java". En modificarNotaAlumnos.java cuando se recibe la petición se accede a la API mediante la url que nosotros hemos llamado "urlNota".

- **notasAlumnos.java:** este método lo usamos para sacar la nota y la imagen de dicho alumno. Y el alumno lo conseguimos pasándoselo por parámetro desde servletAlumnos.java mediante: href=\"notasAlumno", en este se recogen el alumno, la asignatura y la nota, se le manda a notasAlumnos.java y este último lo imprime.

2.2 Contenido web

- Login.html: Página inicial de nuestra aplicación, donde deberemos identificarnos para poder entrar en ella.

Esta página HTML se compone, además de la cabecera típica de la aplicación, de un formulario principal, tanto como para el acceso del profesorado como para acceso del alumnado.

Este formulario está bien diferenciados y se observa claramente cómo identificarte dependiendo de si eres profesor o alumno, teniendo que pulsar sobre tu opción dentro del formulario, ya que si por ejemplo eres alumno y pulsas la opción de profesor no podrás iniciar sesión correctamente. También, el formulario se compone de dos inputs, uno para que el usuario ponga su dni y otro para que ingrese su contraseña. Además, tenemos un botón de tipo submit en el que habrá que pulsar para que te redirija a tu página principal de usuario en caso de que tus credenciales sean correctas.

Esto se consigue mediante la propiedad action en la etiqueta inicial del formulario, donde se debe de poner el servlet al que se dirigirá, en nuestro caso "ServletProfesores". En este servlet se realiza la comprobación de si los credenciales introducidos por el usuario son correctos. Esto se explica más a fondo en los servlet ServletProfesores.

En la parte derecha de la pantalla hemos añadido un contenedor donde indicamos el número de nuestro grupo junto con los nombres completos de los componentes.

Además, para el estilo de esta página hemos usado CSS y Bootstrap 4, teniendo que añadir en el header lo necesario para que esto funcionase.
Se puede observar que en el atributo class de muchos elementos se han añadido las cosas necesarias para darle el estilo como dictan las reglas de Bootstrap.

Como se puede observar en el código hemos creado varios contenedores (div) para estilizar los formularios mediante Bootstrap ya que queríamos que fueran de las mismas dimensiones y estuvieran alineados. Todo el Bootstrap se realiza en esta misma página, como he dicho anteriormente, en los atributos class de los elementos. Sin embargo, también usamos una página .css aparte, que es invocada mediante un link y que a continuación procedemos a detallar.

- styles.css: En esta página dotamos a la página inicial de nuestra aplicación de los estilos que son necesarios para que el Bootstrap pueda funcionar correctamente, centrándonos sobretodo en los estilos de los formularios, los cuales tienen los mismos atributos(class e id) y que por lo tanto quedan estructurados de la misma forma.

En esta página de hipertexto también dotamos de estilos al body, donde cabe destacar el background-image, donde le ponemos en la url una foto llamada hola.jpeg para que sea el fondo de la página login.

- general.html: Esta es una página hecha para servir de base de todas las páginas de nuestra aplicación, solamente está compuesta por una cabecera y por unos estilos que, como he dicho, servirán de base y de los que estará compuesta nuestra aplicación.

Esto lo hacemos para ahorrarnos trabajo y no tener que crear estilos para cada página que creemos, así solo nos limitamos a programar la parte específica de cada página. Además, con esto conseguimos un diseño responsable y consecuente de nuestra aplicación ya que tenemos los mismos estilos y cabeceras en todas las páginas.

Todo esto lo conseguimos mediante la línea de código que se verá a continuación, la cual se verá en todos los servlets y que tiene la función de invocar a esta página html para servir de base:

- getServletContext().getRequestDispatcher("general.html").include(request, response);

Por último, en esta página también hemos añadido un archivo css, el cual hemos añadido su link en el header , y que describimos a continuación:

- **general.css:** En esta página de hipertexto solamente hemos añadido pequeños detalles para que todas las páginas de nuestra aplicación tuvieran un buen diseño.

Por ejemplo, la mayoría de modificaciones las hemos hecho a divs que se encuentran en diferentes páginas para que la estructura de la misma estuviera correcta y funcional para el usuario. También hemos añadido diferentes tipos de letras, las cuales las hemos encontrados en Google Fonts.

3. Seguridad ante fallos

En la aplicación se pueden dar algunas ocasiones en las que se produzcan fallos, pero para ello hemos implementado una serie de comprobación que evitan que el usuario cometa errores:

- El principal error es el de escribir mal el usuario o la contraseña, o simplemente dejarlo en blanco. Para ello en cada servlet de inicio de sesión (servletProfesores.java y servletAlumnos.java) hemos implementado lo siguiente:

```
if (alumnos.size() > 0) {
        if(!verificar.equals("-1") || !verificar.equals("500")) {
```

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

```
if (alumnos.size() > 0) {
        if(!verificar.equals("-1") || !verificar.equals("500")) {
```

Con esta comprobación se mira si el dni introducido está contenido en el ArrayList y luego con el "-1" se comprueba que la contraseña sea correcta y con

el código “500” se mira si el usuario es correcto. En caso de error mandamos un código “405” de la siguiente forma:

```
else {
                response.sendError(405, "ERROR");
        }
```

- Otro error posible es a la hora de iniciar sesión, que un alumno pudiese iniciar sesión como profesor o viceversa. Esto lo hemos solucionado mandando un error 500 en el caso de que sea un alumno e inicie sesión como profesor y viceversa.

- En el caso de la media hemos puesto un input type text que tiene un mínimo(min=0) y un máximo(max=10), para limitar el 0 y el 10 en caso de algún fallos. Y un método ajax que al poner una nota mayor que 10 automáticamente lo cambia a un 0 antes de darle a guardar.

4. Anotaciones de acceso(logs) y cómo consultar su contenido

Las anotaciones log generan ficheros en los cuales puede haber una secuencia de información generadas cada vez que se ejecuta la aplicación, en la cual podemos adquirir información mediante una fecha en la cual puede aparecer la IP usada en cada momento, o el nombre de usuario el cual esta interactuando con la aplicación, la URI en la que estas en cada momento, y más información que podemos adquirir del servlet. Las anotaciones se pueden poner mediante ‘’@” en los servlets y también se pueden configurar en el XML.

En este caso en nuestro trabajo se nos solicita un log el cual con la fecha y hora actual de LocalDateTime.Now(); se genere un fichero persistente el cual anota esa información junto con el servlet que se ha activado y su respectivo usuario.
En principio iremos al xml y colocaremos estas líneas en el web.xml de nuestro proyecto como en la siguiente captura de imagen:

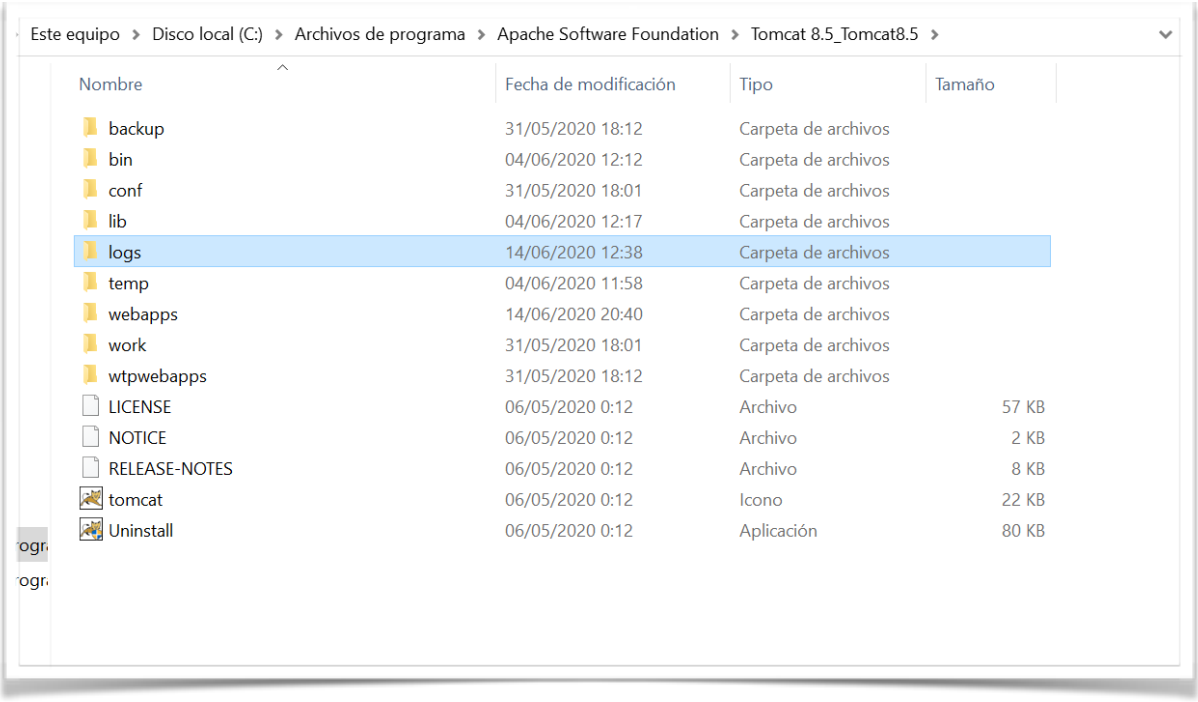
```
<filter>
  <display-name>Logs</display-name>
  <filter-name>Logs</filter-name>
  <filter-class>servlet.Logs</filter-class>
</filter>
<filter-mapping>
  <filter-name>Logs</filter-name>
  <url-pattern>/Logs</url-pattern>
  <url-pattern>/detallesAlumnos</url-pattern>
  <url-pattern>/listaAlumnos</url-pattern>
  <url-pattern>/logout</url-pattern>
  <url-pattern>/modificarNotaAlumno</url-pattern>
  <url-pattern>/notasAlumno</url-pattern>
  <url-pattern>/servletProfesores</url-pattern>
</filter-mapping>
```

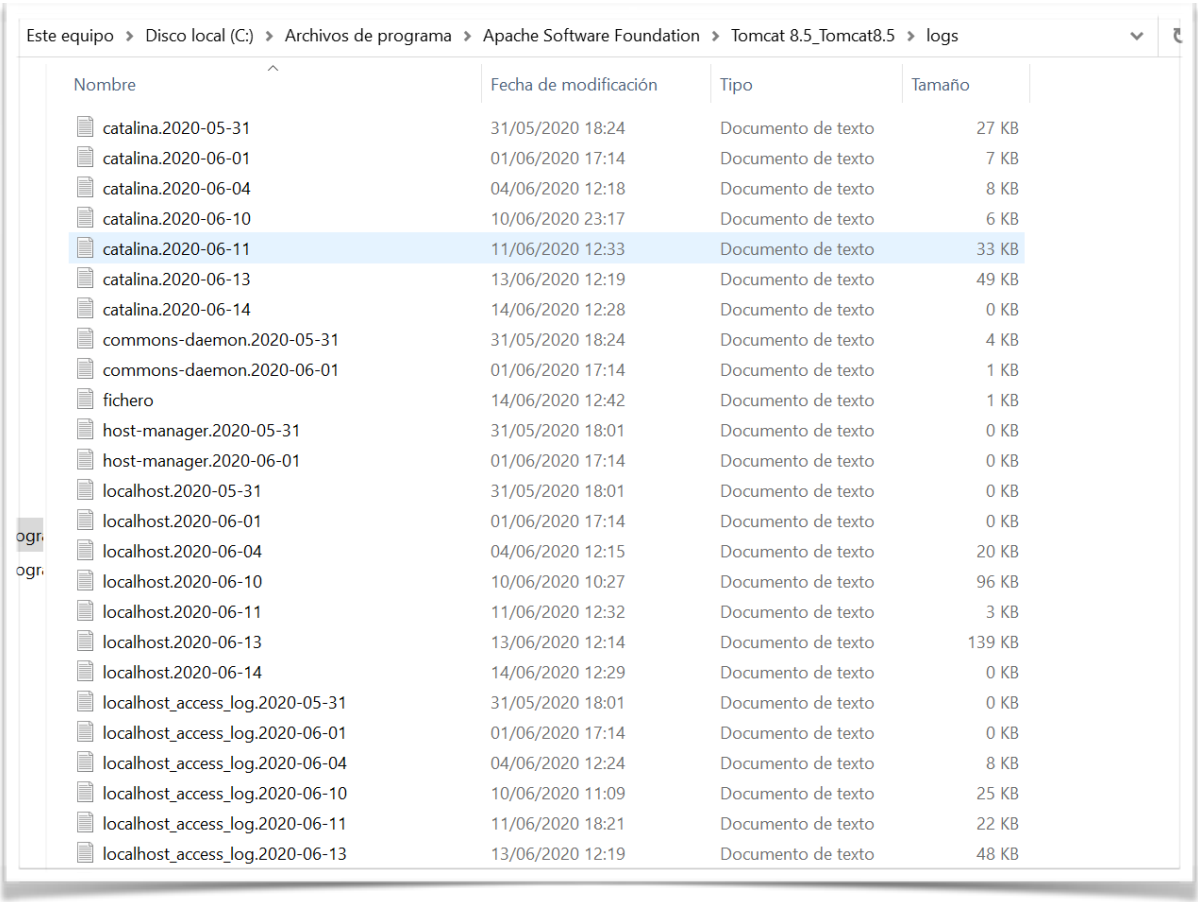
Tendremos que implementar un Filtro el cual se encargue de generar el fichero, y escriba en el según las condiciones anteriores. Por tanto en el web.xml declaramos primero el filtro mediante <filter>, con su respectivo nombre <filter-name> y la ubicación que tiene la clase filtro en nuestro proyecto mediante <filter-class>, todo esto se genera de manera automática al crearlo en *New->Filter* en eclipse. Para poder realizar el mapeo del filtro , ya que el filtro debe interactuar sobre algún archivo, introduciremos en <url-pattern> todos los archivos con los que queremos que el filtro interactúe, en este caso meteremos todos nuestros servlets para que pueda detectar cual se ha activado en cada caso.
Posteriormente, en el código del filtro tendremos su destroy(), el doFilter() y el init().

```java
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
    // TODO Auto-generated method stub
    // place your code here


    HttpServletRequest httpServletRequest = (HttpServletRequest) request;
    HttpServletResponse httpServletResponse = (HttpServletResponse) response;
    BufferedWriter bw = null;
    FileWriter fw = null;
    String prof = httpServletRequest.getParameter("dni");


    try {

        String acceso = LocalDateTime.now().toString() + " " + prof  +  " " + " acceso a "+  httpServletRequest.getRequestURI().substring(10) +  "\r\n";

        File file = new File("./logs.txt");
        // Si el archivo no existe, se crea por primera vez
        if (!file.exists()) {
            file.createNewFile();
        }
        // flag true entonces adjuntamos información al archivo. y el getAbosluteFile es para habilitar la funcion de append
        fw = new FileWriter(file.getAbsoluteFile(), true);
        bw = new BufferedWriter(fw);
        bw.write(acceso);
        //File[] files = file.listFiles();
        //Arrays.sort(files, Comparator.comparingLong(File::lastModified));
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
    //Cierra instancias de FileWriter y BufferedWriter
            if (bw != null)
                bw.close();
            if (fw != null)
                fw.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
```

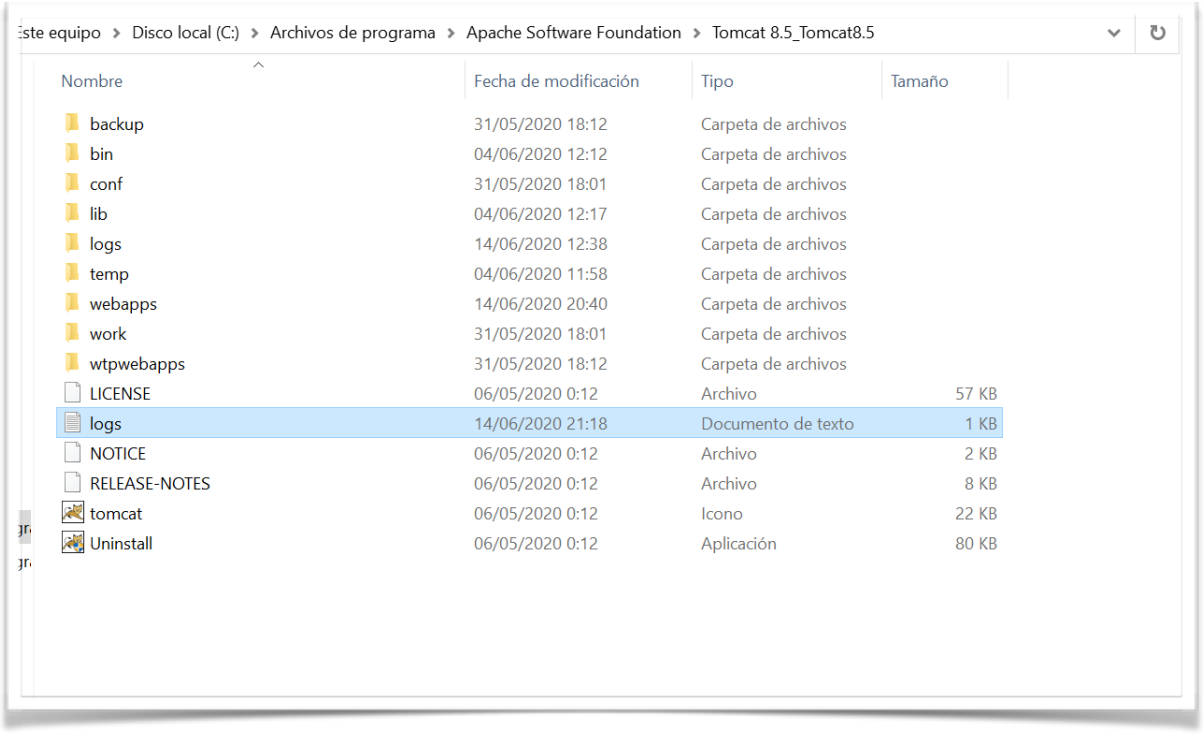
En el cual tendremos en el doFilter() el código implementado el cual declararemos los HttpServletRequest para poder hacer las peticiones de nuestros servlets como es el getRequestURI para saber en que ubicación estamos en cada momento del proyecto, y esto es debido a las <url-pattern> declaradas también que hacen que el filtro trabaje sobre ellos.
Crearemos el archivo File con ubicación de manera absoluta y llamando por último a chain.doFilter, ya que todos los filtros deben llamar al método chain.doFilter para poder ejecutar el servlet original después del filtrado.

Tomcat con ayuda de la herramienta ‘’Catalina’’ por defecto genera logs los cuales pueden ubicarse en la propia carpeta del tomcat:


Este equipo › Disco local (C:) › Archivos de programa › Apache Software Foundation › Tomcat 8.5_Tomcat8.5 ›
Nombre
Fecha de modificación
Tipo
Tamaño
backup 31/05/2020 18:12 Carpeta de archivos
bin 04/06/2020 12:12 Carpeta de archivos
conf 31/05/2020 18:01 Carpeta de archivos
lib 04/06/2020 12:17 Carpeta de archivos
logs 14/06/2020 12:38 Carpeta de archivos
temp 04/06/2020 11:58 Carpeta de archivos
webapps 14/06/2020 20:40 Carpeta de archivos
work 31/05/2020 18:01 Carpeta de archivos
wtpwebapps 31/05/2020 18:12 Carpeta de archivos
LICENSE 06/05/2020 0:12 Archivo 57 KB
NOTICE 06/05/2020 0:12 Archivo 2 KB
RELEASE-NOTES 06/05/2020 0:12 Archivo 8 KB
tomcat 06/05/2020 0:12 Icono 22 KB
Uninstall 06/05/2020 0:12 Aplicación 80 KB

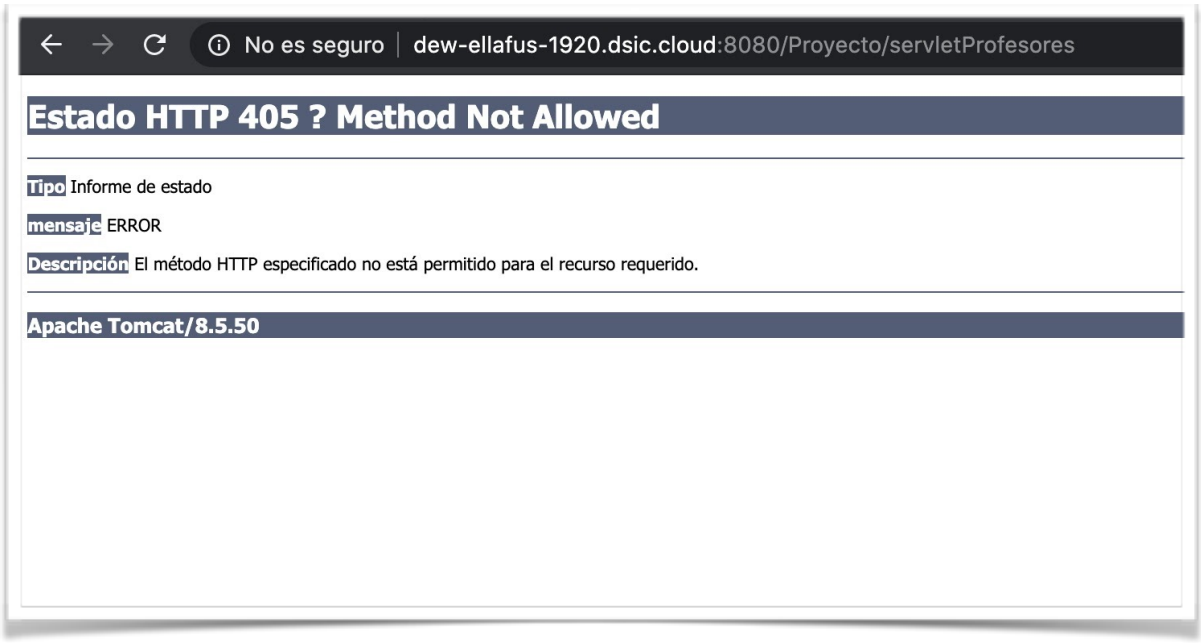

Este equipo › Disco local (C:) › Archivos de programa › Apache Software Foundation › Tomcat 8.5_Tomcat8.5 › logs
Nombre
Fecha de modificación
Tipo
Tamaño
catalina.2020-05-31 31/05/2020 18:24 Documento de texto 27 KB
catalina.2020-06-01 01/06/2020 17:14 Documento de texto 7 KB
catalina.2020-06-04 04/06/2020 12:18 Documento de texto 8 KB
catalina.2020-06-10 10/06/2020 23:17 Documento de texto 6 KB
catalina.2020-06-11 11/06/2020 12:33 Documento de texto 33 KB
catalina.2020-06-13 13/06/2020 12:19 Documento de texto 49 KB
catalina.2020-06-14 14/06/2020 12:28 Documento de texto 0 KB
commons-daemon.2020-05-31 31/05/2020 18:24 Documento de texto 4 KB
commons-daemon.2020-06-01 01/06/2020 17:14 Documento de texto 1 KB
fichero 14/06/2020 12:42 Documento de texto 1 KB
host-manager.2020-05-31 31/05/2020 18:01 Documento de texto 0 KB
host-manager.2020-06-01 01/06/2020 17:14 Documento de texto 0 KB
localhost.2020-05-31 31/05/2020 18:01 Documento de texto 0 KB
localhost.2020-06-01 01/06/2020 17:14 Documento de texto 0 KB
localhost.2020-06-04 04/06/2020 12:15 Documento de texto 20 KB
localhost.2020-06-10 10/06/2020 10:27 Documento de texto 96 KB
localhost.2020-06-11 11/06/2020 12:32 Documento de texto 3 KB
localhost.2020-06-13 13/06/2020 12:14 Documento de texto 139 KB
localhost.2020-06-14 14/06/2020 12:29 Documento de texto 0 KB
localhost_access_log.2020-05-31 31/05/2020 18:01 Documento de texto 0 KB
localhost_access_log.2020-06-01 01/06/2020 17:14 Documento de texto 0 KB
localhost_access_log.2020-06-04 04/06/2020 12:24 Documento de texto 8 KB
localhost_access_log.2020-06-10 10/06/2020 11:09 Documento de texto 25 KB
localhost_access_log.2020-06-11 11/06/2020 18:21 Documento de texto 22 KB
localhost_access_log.2020-06-13 13/06/2020 12:19 Documento de texto 48 KB

Una vez dentro podemos detectar todos los logs que ha generado Tomcat según los movimientos que hemos realizado con él.
En nuestro caso no tenemos ubicación relativa para la creación de nuestro archivo que contenga estos logs, por tanto en nuestra ubicación aparecerán archivos txt similares a estos.



5. Funcionamiento con imágenes

En esta primera captura, vemos la pantalla de Login, donde tenemos que seleccionar si somos profesor o alumno(viene por defecto marcado en profesor).
Una vez puesto el DNI y la contraseña, si estás como profesor y pones datos de profesor, o si estás como alumno y pones datos de alumno, entras correctamente.
En caso de poner un profesor teniendo marcado alumno o viceversa, salta un error, y en caso de poner mal el login, también. En la siguiente imagen se puede observar.



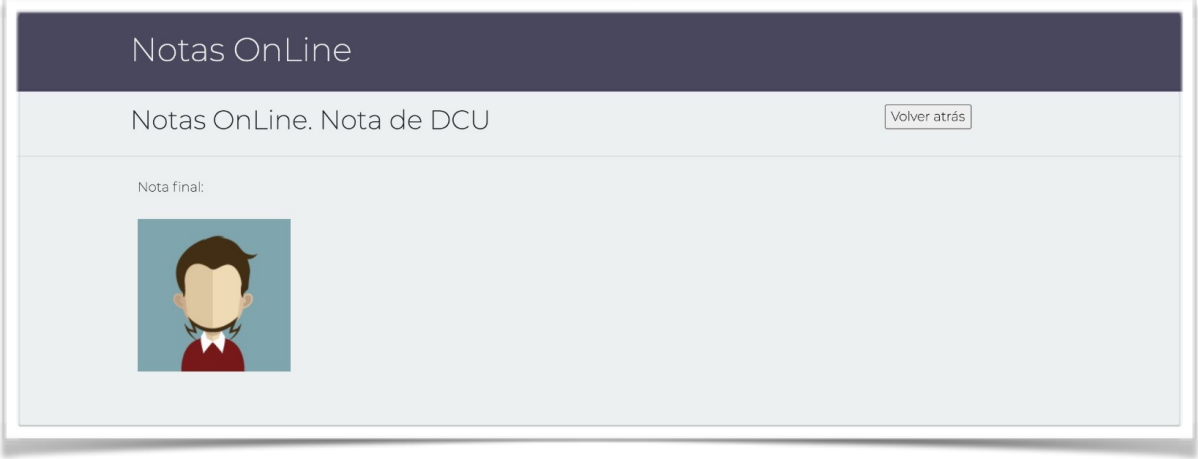
En esta captura, vemos el nombre de profesor, una lista con sus asignaturas y un botón de cerrar sesión, el cual cerrará sesión si lo pulsamos. En caso de clicar a una asignatura, nos llevará a la siguiente captura.

Aquí vemos una lista con los alumnos, su nota, y un campo de texto para modificar su nota, la cual se actualiza automáticamente cuando la insertamos, pero tenemos un botón para recargar la página con los datos modificados.
Por otra parte, tenemos un botón para la obtención de la nota media de los alumnos, la cual funciona aunque no todos los alumnos tengan nota, contando a estos alumnos como un 0 de nota.
Por defecto, si no tiene notas, devolverá un 0, pero cuando insertamos notas y las guardamos, saldrá lo siguiente al darle click a sacar nota media:

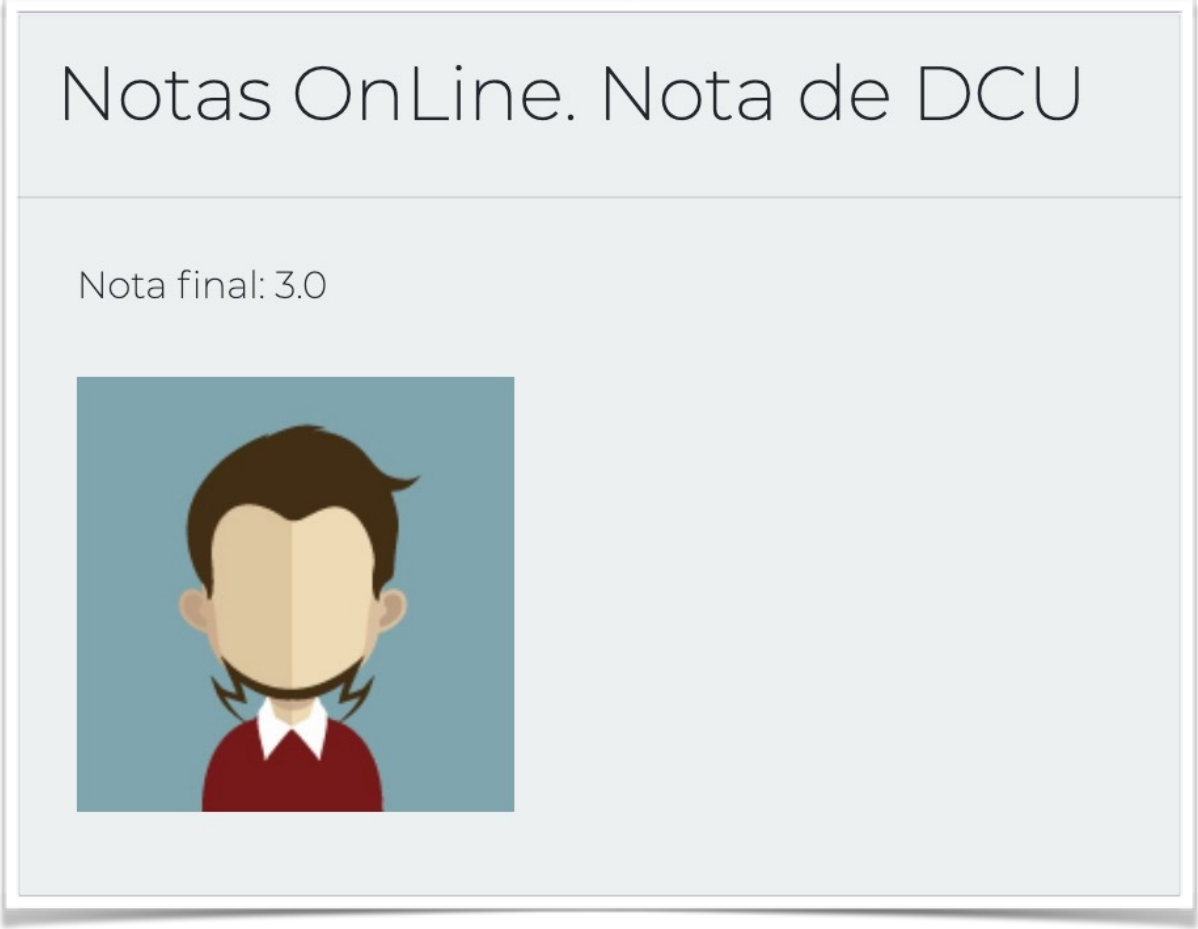
Si pulsamos a algún alumno, nos lleva a la siguiente captura, donde vemos los detalles del alumno(Su DNI, nombre y apellidos).

En el caso de la captura anterior, no sale la nota, puesto que es lo que vemos al cargar la página, pero si pulsamos al botón de sacar nota media, veremos lo siguiente:

Para cerrar sesión, disponemos como en el caso de profesor, de un botón dedicado a esa funcionalidad.
Si se pulsa a alguna de las asignaturas, nos llevará a la siguiente captura.



En esta captura, vemos la nota del alumno en esa asignatura y su foto. Inicialmente no tendrá nota, hasta que el profesor no se la asigne. Una vez se la asigne, se verá lo siguiente:



Como en el profesor, también tenemos un botón de volver atrás para volver a la pantalla de las asignaturas del alumno.

6. Actas del proyecto

6.1 Séptima sesión

En esta séptima sesión, hemos estado pensando en cómo meter las fotografías, y hemos decidido que las fotografías tengan de nombre los DNI de los alumnos, para así invocarlas con los DNI.
Estamos investigando cómo hacer el PUT correctamente para asignar las notas a los alumnos, que nos está dando bastantes problemas, nos salían errores y excepciones y no teníamos ni idea de por qué salían, así que fuimos muy a prueba y error, con cosas que fuimos buscando por internet.

6.2 Octava sesión

En esta octava sesión, hemos estado informándonos sobre el bootstrap y hemos hecho una primera toma de contacto de nivel práctico, empezando a aplicarlo parcialmente sobre un par de pantallas, para ver cómo se comportaba y de paso, poder decidir el estilo que le queríamos dar.
Por otra parte, también hemos estado intentando averiguar cómo sacar la nota media, pero no hemos tenido éxito.

Teníamos bastantes problemas a la hora de hacer la petición ajax, ya que nos saltaba una excepción CORS, para la cual instalamos una extensión para el navegador, que evitaba estas excepciones.

Más tarde te llamamos para preguntar varias dudas, y en una de aquellas nos ayudaste con lo que estábamos atascados, que era que no conseguíamos hacer el PUT de insertar nota de alumno, y el error era porque estábamos haciendo directamente la petición ajax al nivel de datos, saltándonos también la capa lógica. Así que nos pusimos a ello, y al final del día, conseguimos implementar la nota sin saltarnos la capa lógica, con lo que también nos eliminamos la necesidad de usar la extensión CORS, ya que con un correcto direccionamiento de las peticiones, no es necesaria.

Lo último que miramos en esta sesión es la autenticación en el login, ya que, aunque hacía login, no nos sacaba algunas veces los errores que debía, y nos pusimos a intentarlo separando rolpro y rolalu, pero no conseguimos nada, puesto que nos salía una página en blanco, en lugar del error correspondiente.

6.3 Novena sesión

En esta sesión, hemos estado investigando cómo hacer la nota media de las notas de una asignatura de un profesor, que lo teníamos bastante abandonado debido a que, como unas sesiones antes no podíamos poner la nota, era un poco absurdo intentar sacar la media, porque no podíamos saber con certeza si estaba cogiendo el dato o no. Ahora ya con el dato sobre la web, empezamos a hacer pruebas. Tuvimos problemas porque primero nos hacía la media sólo de 1, pero la sacaba muchas veces, en lugar de sacarla una vez pero recorriendo todos los valores.
Al final de esta sesión, a base de prueba y error, arreglando cosas sueltas por algunos sitios, fuimos filtrando errores hasta descubrir cómo hacerlo correctamente, y lo hicimos.

6.4 Décima sesión

En esta sesión, hemos implementado correctamente la autenticación en el XML, para que compruebe el rol y en base a eso, permita o deniegue el acceso.
También hemos implementado el bootstrap en el resto de páginas, de modo que todas tuvieran un diseño coherente y consistente.
Empezamos a mirar cómo sacar los logs de los accesos, pero no lo teníamos muy claro aún, por lo que no conseguimos sacarlo aún.
También pensamos en hacer un solo servlet de login, para simplificarlo más a nivel interno.

6.5 Undécima sesión

En esta sesión, estuvimos viendo cómo hacer la nota media de un alumno, y lo conseguimos haciendo la petición correctamente.
Hemos conseguido generar el fichero de logs y que los escriba correctamente.

6.6 Duodécima sesión

En esta sesión hemos cambiado la ventana de login, que primero pasó a tener un solo login, y entraba en caso de entrar como alumno o profesor, pero no había un login específico de cada uno, por lo que pusimos un radio para profesor y otro para alumno, para que seleccione como quién se quiera identificar.
Dedicamos el resto de la sesión a intentar capturar los errores en caso de que te conectases como profesor marcando alumno y viceversa, o poniendo mal un login.

6.7 Decimotercera sesión

En esta sesión nos hemos dedicado a adaptar todo el documento con el CSS, para hacer cosas más funcionales y bonitas. Hemos estructurado mejor los divs y hemos creado las clases y los ids necesarios para que funcionen, modificando esos divs en el archivo general.css para que se mostrasen como queríamos, menos algún estilo que lo metimos interno en div, con el atributo style.

6.8 Decimocuarta sesión

En esta sesión, hemos decidido poner nuestros nombres en un círculo acorde al fondo de la pantalla de login, para que no quedase raro.
Nos dimos cuenta de que si acababas de ejecutar el jar y no habías puesto ninguna nota, al darle a calcular nota media, te devolvía un NaN, y cambiamos para que en ese caso, devolviera un 0.

7. Bibliografía

- https://www.codeproject.com/Articles/1157120/How-to-Call-RESTful-API-Web-Service-in-Servlet
- https://migazoft.wordpress.com/2015/10/08/login-basico-con-servlets-ajax/
- https://migazoft.wordpress.com/2015/10/08/login-basico-con-servlets-ajax/
- https://www.javatpoint.com/example-of-login-form-in-servlet
- http://www.edu4java.com/es/servlet/servlet3.html
- https://stackoverflow.com/questions/8997598/importing-json-into-an-eclipse-project
- https://coderanch.com/t/479697/java/HttpURLConnection-URLConnection-login-sites
- https://underc0de.org/foro/java/(tutorial)-peticiones-ajax-a-servlets/
- https://api.jquery.com/jquery.ajax/
- https://stackoverflow.com/questions/7667416/javascript-error-unexpected-identifier-when-trying-to-do-jquery-ajax-call
- https://www.w3schools.com/js/js_cookies.asp
- https://living-sun.com/es/html/287897-make-difference-between-2-buttons-html-jsp-servlets.html

- https://stackoverrun.com/es/q/9923769
- https://stackoverrun.com/es/q/1688390
- https://www.galisteocantero.com/sesiones-en-servlets-y-jsp-ejemplo-login-logout/
- https://stackoverflow.com/questions/855835/servlet-parameters-and-doput
- https://es.stackoverflow.com/questions/36022/insertar-datos-en-mysql-y-java-en-eclipse/36024
- https://books.google.es/books?id=is2J44U4DpsC&pg=PA473&lpg=PA473&dq=metodo+insert+java&source=bl&ots=108e5krcuA&sig=ACfU3U0ui1-FNiYjIbhULn9UtCsa6UMuUw&hl=es&sa=X&ved=2ahUKEwjU18nqgtLpAhU7D2MBHWBBBRgQ6AEwDnoECAkQAQ#v=onepage&q=metodo%20insert%20java&f=false
- https://www.studentstutorial.com/ajax/insert-data
- https://stackoverflow.com/questions/13196485/servlet-doput-request-getparameter
- https://es.stackoverflow.com/questions/236026/redireccionar-de-index-jsp-a-otro-jsp-seg%C3%BAn-bot%C3%B3n-presionado
- https://jarroba.com/ajax-con-jsp-y-servelts/
- https://stackoverflow.com/questions/13274279/authentication-filter-and-servlet-for-login
- https://stackoverflow.com/questions/2349633/doget-and-dopost-in-servlets

- https://www.codejava.net/java-ee/servlet/how-to-implement-authentication-filter-for-java-web-application
- https://www.ibm.com/support/knowledgecenter/es/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/tsec_servlet.html
- https://www.javatpoint.com/requestdispatcher-in-servlet
- https://devqa.io/how-to-parse-json-in-java/
- https://stackoverflow.com/questions/42540520/attempting-to-login-to-a-rest-api-with-ajax
- http://www.java2s.com/Code/Jar/j/Downloadjsonsimple11jar.htm
- https://www.it-swarm.dev/es/java/necesito-una-opcion-alternativa-httpclient-en-android-para-enviar-datos-php-ya-que-ya-no-es-compatible/1052779463/
- https://www.codeproject.com/Articles/1157120/How-to-Call-RESTful-API-Web-Service-in-Servlet
- https://www.ibm.com/support/knowledgecenter/es/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/tsec_web.html