

3.5.- LA APLICACIÓN WEB.

3.5.1. La Aplicación Web.

La aplicación Web permite al usuario poder hacer uso del servicio mediante la interfaz Web que se muestra gracias al cliente de la aplicación.

La aplicación consta de un servlet principal que es el motor de la aplicación. En este caso es “*organizacion.java*”, y se encarga de gestionar toda la aplicación: redireccionar a las páginas JSP que correspondan para mostrarlas al usuario y llamar a los diferentes métodos Java del servicio según la función a realizar.

El esquema general de funcionamiento de cada una de las gestiones que realiza el servicio sería la siguiente: se inicia la aplicación y ésta se encarga de mostrarle una página de bienvenida e invitarle a poder hacer uso del servicio. Cuando se acepta, se comprueba que los ficheros de configuración son válidos y se cargan en los beans correspondientes. Si éstos no fueran válidos se volvería a mostrar de nuevo esta página de bienvenida.

Tras esto se puede elegir la gestión a realizar. Al elegir una de ellas “*organizacion.java*” se encarga de mostrarle la página correspondiente para obtener los datos necesarios para hacer la gestión. Una vez se han introducido, si el servlet tiene que hacer alguna comprobación la hace y pasa los parámetros necesarios a los métodos Java que correspondan. Estos le devuelven el resultado y el servlet con ellos llama a la página JSP para mostrarlos.

Este es el funcionamiento básico. En resumen, se trata de comunicar el cliente con el servicio, lo que pide el cliente se lo dice al servlet, éste llama al servicio que realiza la gestión y le devuelve el resultado y, por último, el servlet le da el resultado al cliente para que lo muestre.

3.5.2. Servlet.

3.5.2.1. Organización.java

Esta clase es el servlet de la aplicación, procesa la petición del cliente y envía la respuesta de vuelta al contenedor. A la hora de enviar información al servlet se tienen los métodos *doGet* y *doPost* que, aunque tengan diferencias, en este servicio realizan la misma función ya que *doGet* se encarga de redirigir la información a *doPost*.

La acción a realizar depende del valor que se obtenga de la variable “*accion*”:

- “*accion= configuracion*”: realiza la validación del *fichero de configuración de rutas* con respecto a un esquema proporcionado gracias a la clase **Validar**. Para ello coge la ruta de dichos archivos de *Bean_servicio*. Posteriormente carga dicho

archivo en el *Bean_rutas* con la clase **Configurar** únicamente si la validación ha tenido éxito.

- “*accion=anadir*”: muestra la página Web que recopila la información necesaria para almacenar una GPC en una colección de la base de datos: nombre, especialidad y ruta de la GPC.
- “*accion=anadirGC*”: almacena la guía en la base de datos con la clase **AñadirGC** comprobando si el nombre, la colección y la ruta de la GPC han sido especificados así como que el nombre no esté repetido. Una vez hecho esto le pasa la nueva guía en un *ResourceSet* a *mostrarGGCC.jsp* para que sea mostrada.
- “*accion=mostrar_guias*”: muestra la página Web donde se debe indicar la colección cuyos recursos se desean mostrar o, en su defecto, los recursos de todas las colecciones.
- “*accion=mostrar_col*”: recoge la colección que se desea mostrar y le pasa a *mostrarGGCC.jsp* un *ResourceSet* con los recursos (GPCs) a mostrar. Este *ResourceSet* es proporcionado por la clase **MostrarGGCC.java**.
- “*accion=mostrar_guia*”: recoge el identificador de la nueva guía que se ha almacenado en la base de datos y que se va a mostrar así como la colección donde se ha almacenado dicho recurso. Le pasa a *mostrarGGCC.jsp* un *ResourceSet* con el recurso (GPCs) a mostrar. Este *ResourceSet* es proporcionado por la clase **MostrarGGCC.java**.
- “*accion=buscar*”: muestra la página Web que recoge la información para realizar la búsqueda con o sin criterios lógicos: especialidad, criterio, valor del criterio y criterio lógico que concatena con la siguiente búsqueda (este último sólo es necesario si se quiere realizar una búsqueda con criterios lógicos y es indiferente cuando se quiere parar de añadir criterios lógicos y realizar la búsqueda)
- “*accion=Realizar búsqueda*”: recoge los valores de la colección dónde realizar la búsqueda, el último criterio y su valor y la cadena de consulta que se ha ido formando para la búsqueda lógica. Con esto la clase **Buscar_GGCC** devuelve en un *ResourceSet* los recursos resultado de la búsqueda completa.
- “*accion=Seguir con la búsqueda*”: muestra una página Web que permite ir formando búsquedas concatenadas mediante criterios lógicos. Es decir, se va indicando en cada búsqueda el criterio con el valor que se desea que tenga, por ejemplo, *Diagnóstico=Varicela* y se indica el criterio lógico (AND/OR) que se desea para concatenar con la siguiente búsqueda quedando, por ejemplo: “*Diagnóstico=Varicela AND Síntomas=fiebre*”. De esta forma se tienen dos búsquedas unidas por un criterio lógico. Al conjunto se le asigna el nombre de búsqueda lógica.

La cadena que se va formando en esta búsqueda lógica debe pasarse tanto si se desea seguir con la búsqueda lógica añadiendo más búsquedas individuales como si se desea realizar definitivamente la búsqueda lógica.

- “*accion=eliminar*”: se encarga de eliminar una GPC mediante la clase **EliminarGC** para ello recoge el identificador y la colección de la GPC a suprimir.
- “*accion=consultar*”: muestra una página Web donde se puede elegir el apartado de la GPC a consultar.
- “*accion=ver_consulta*”: recoge los parámetros necesarios para realizar la consulta: nombre de la GPC a consultar, colección a la que pertenece y criterio que se desea consultar. Con esto se encarga de realizar la consulta de la GPC mediante la clase **Consultar**.

El servlet utiliza otro método más llamado “*gotoPage*” que redirige al usuario desde la página Web actual a otra que se indica en el parámetro “*String direccion*”.

organizacion { From Gestion_GC-Model }	
Attributes	
Operations	
public void	doPost(HttpServletRequest request, HttpServletResponse response)
public void	doGet(HttpServletRequest request, HttpServletResponse response)
public void	gotoPage(String direccion, HttpServletRequest request, HttpServletResponse response)

Ilustración 45.- *organizacion.java*.