3.3.- SERVIDOR.

3.3.1. Introducción.

El servicio implementado en este proyecto se encarga de la gestión de Guías de Práctica Clínica en XML. Las prestaciones y funcionalidades principales de este servicio son:

1. Flexibilidad y portabilidad de Guías de Practica Clínica, es decir, que los elementos del archivo XML de la guía puedan cambiar sin que el servicio se vea afectado. Por ejemplo: se quiere acceder al elemento "*autor*" de la guía y se define que está en una ruta "*path1*". Si esta ruta cambia o se utiliza otro archivo de guía XML que tenga otra ruta, el servicio dejaría de funcionar.

   La solución adoptada es la creación del ***"archivo de configuración de rutas"*** donde se pueden cambiar estas rutas y el servicio seguiría funcionando. Para ello nada más iniciar el servicio éste realiza una ***configuración*** inicial en el bean *Bean_rutas,* únicamente si dicho archivo es válido con respecto al esquema editado. *Bean_rutas* tendrá por tanto la asociación del nombre de un parámetro a utilizar en el servicio con la ruta correspondiente dentro del *fichero de GPC* en XML.

2. Una vez cargados los ficheros de configuración vistos en el apartado 3.2, se muestra la posible gestión de las GPC. Entre ellas está la del ***almacenamiento de una GPC*** en la base de datos. Este servicio permite almacenar la guía en una colección específica que representará la especialidad de la GPC dentro del entorno médico.

3. Pero no cualquier archivo de guía en XML bien formado puede almacenarse en la base de datos, sino que también debe de ser válido. Para ello el servidor realiza una ***validación*** de las GPC que se introducen gracias a un esquema creado con XML Schema. Esta validación es la misma que se realiza con el *archivo de configuración de rutas* pero con diferente esquema. Si el documento XML de la guía no es válido, éste no es almacenado.

   En el caso que cambiara la estructura de los documentos XML de las guías, habría que crear otro esquema para ese tipo de documentos.

4. Otra gestión que se permite es la ***eliminación de una GPC.*** Para el caso de que se quiera eliminar de la base de datos una guía, se tiene la opción de poder hacerlo tan sólo pasando el identificador y la colección de la GPC.

5. El servicio también permite el ***listado de las GPCs*** que están almacenadas, sea cual sea su colección, o también permite listar sólo

aquellas que pertenezcan a una colección. Así mismo también muestra una guía que acaba de ser almacenada en la colección de una base de datos.

6. Una gestión importante es la de poder ***consultar una GPC.*** Al seleccionar una GPC se permite poder consultar el valor de un campo de la guía.

7. Y, por último, proporciona un servicio de ***búsqueda de GPCs*** según uno o varios criterios, pudiendo combinar dichas búsquedas con operadores lógicos.

3.3.2. Descripción y función de las clases.

3.3.2.1. Validar.java (paquete *config*).

La clase *Validar* se encarga de la validación de un documento XML con respecto a su XML Schema. Se encuentra dentro del paquete *"config"*. Se utiliza tanto para la validación de los archivos de configuración como para la del *fichero de GPC* XML.

Para la validación se utiliza el analizador SAX, tal como se vió en el apartado 2.5. Se configura el analizador, se le asigna un manejador y se realiza el proceso de validación.

Tiene un único método *"Validación"* que requiere el paso de dos parámetros:

- String archivoXML: documento XML a validar.
- String esquema: esquema realizado en XML Schema del documento a validar.

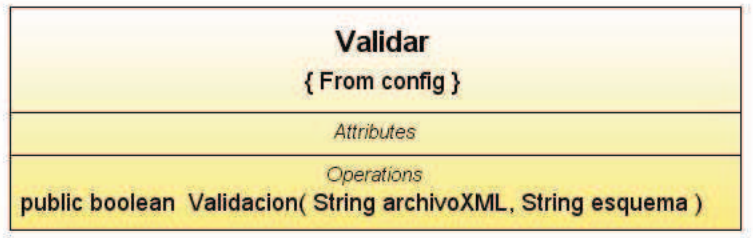

Ilustración 35.- *Validar.java.*

3.3.2.2. Configurar.java (paquete *config*).

La clase *Configurar* se encarga de cargar el archivo de configuración de rutas XML, es decir, las rutas de cada campo de las GPC son almacenadas en los campos correspondientes del *Bean_rutas*. Se encuentra dentro del paquete *config*. Se utiliza al comienzo del servicio y sólo carga el fichero una vez. Se considera dentro de la etapa de validación.

Como en el caso de la validación, esta clase utiliza el analizador SAX para recorrer el documento e ir almacenando en el bean las rutas. Esto lo hace gracias a sus métodos que capturan los eventos lanzados por el analizador.

El método que se utiliza para realizar esta función es el método *"lee"* y únicamente precisa de un parámetro:

- *String archivoXML:* el archivo de configuración de rutas XML.

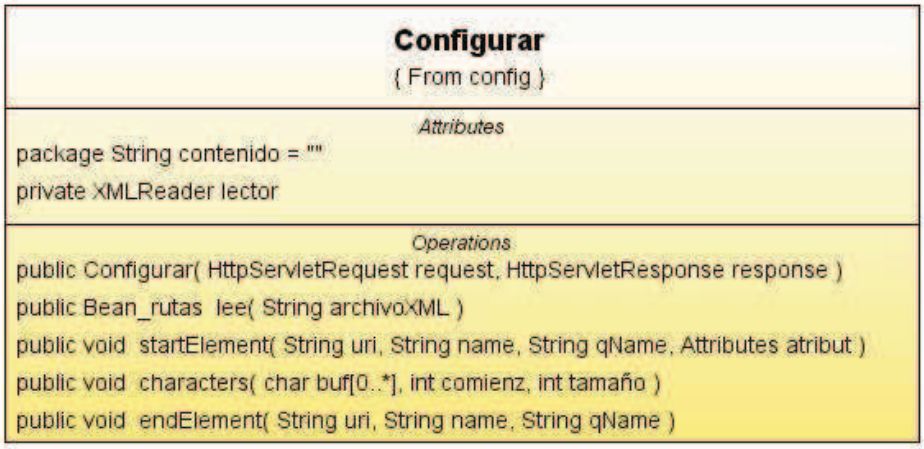

Ilustración 36.- ***Configurar.java.***

3.3.2.3. Bean_servicio (paquete *config*).

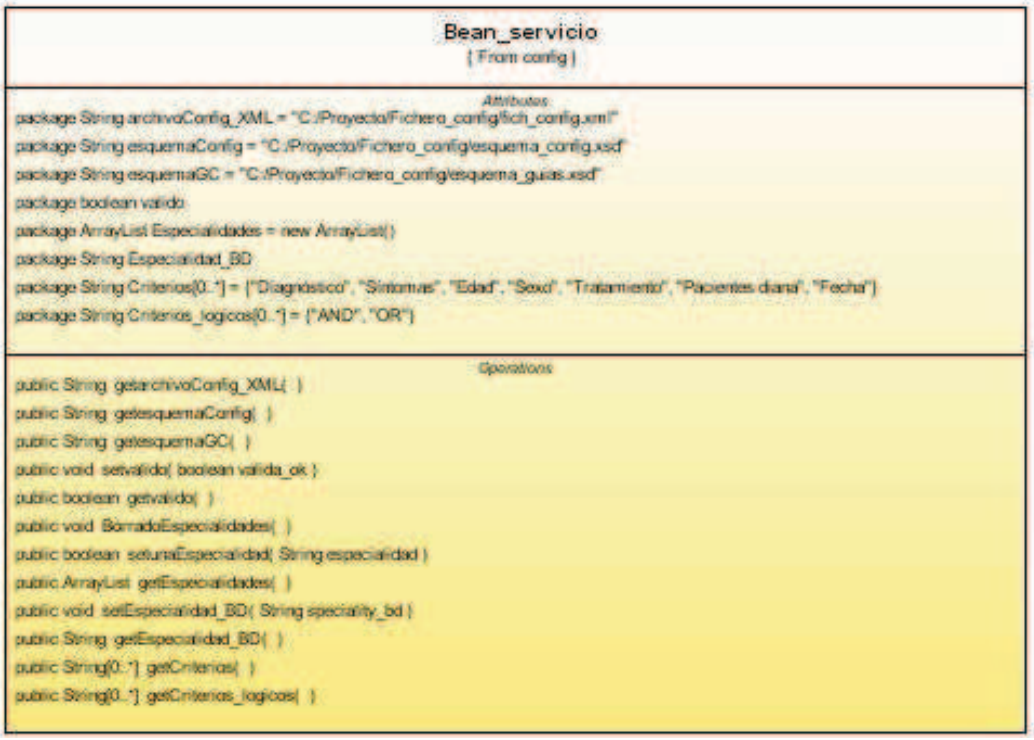

Ilustración 37.- ***Bean_servicio.***

Bean_servicio es una clase Bean que contiene información relacionada con el servicio dotándolo de una mayor flexibilidad y portabilidad. Se encuentra dentro del paquete *config.*

Contiene información de la ubicación de los *fichero de configuración y GPC* así como de los esquemas empleados, los criterios que pueden ser elegidos a la hora de realizar una búsqueda, o las colecciones que están creadas, entre otros. De esta forma estos parámetros pueden modificarse sin que el servicio se vea afectado.

3.3.2.4. Bean_rutas (paquete *config*).

Se trata de una clase Bean cuya función es la de asociar el nombre de cada campo de la GPC que se usa en el servicio con la ruta correspondiente dentro del documento XML de la GPC. Se encuentra dentro del paquete *config*.

Permite que si se producen cambios en el *fichero de GPC* XML el servicio no se vea afectado, sólo habría que modificar el *fichero de configuración de rutas* indicando las nuevas rutas.

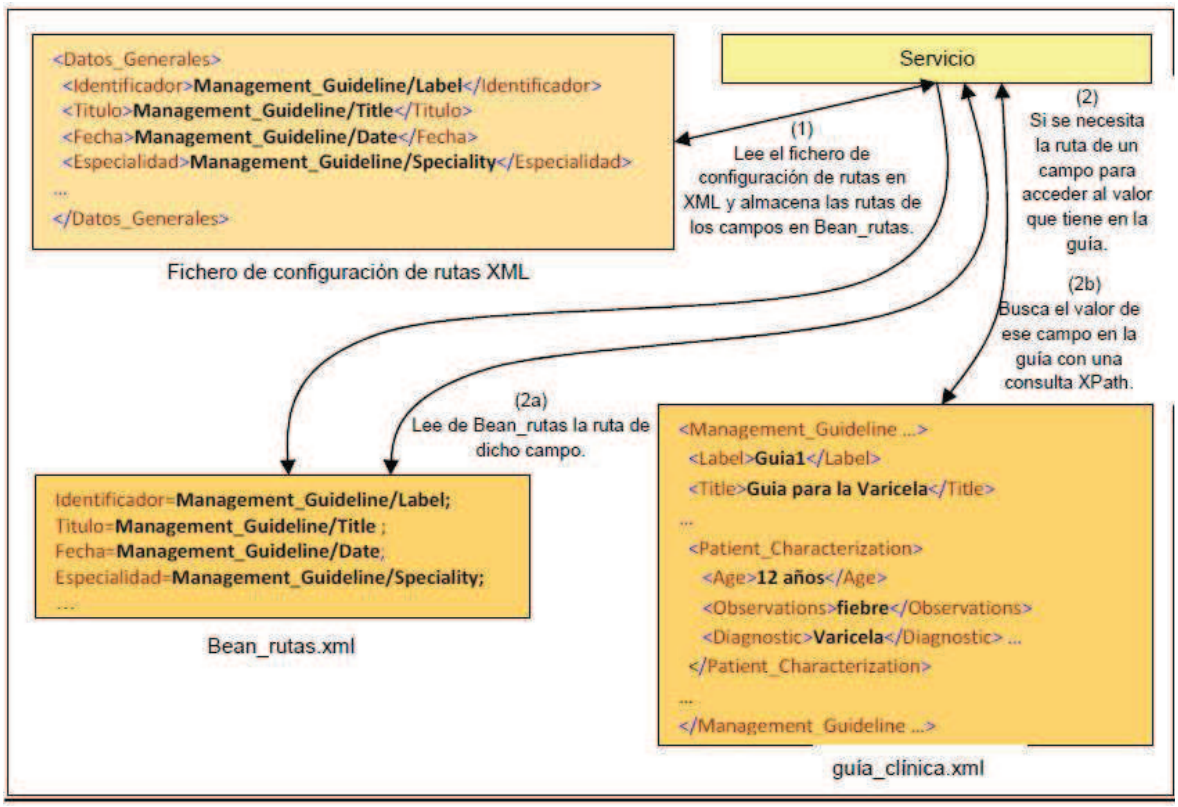

Ilustración 38.- Pasos para el uso de la clase *Bean_rutas.*

El funcionamiento dentro del servicio de este bean consiste: al inicio del programa actualiza sus variables con los valores contenidos en el *fichero de configuración de rutas*. Cuando el servicio necesita de consultar algún campo en un

fichero de GPC XML, primero lee de este bean la ruta donde se encuentra dicho campo, y con la ruta ya puede formar la estructura XPath para poder consultarlo.

3.3.2.5. AñadirGC.java

Esta clase se encarga del almacenamiento del *fichero de GPC* en XML en una colección específica dentro de la base de datos. Si entre las colecciones creadas no existe la deseada permite decir qué nueva colección se quiere crear para almacenar la GPC.

Posee tres métodos:

- *añadir:* se encarga de almacenar la guía en la colección correspondiente. Exige el paso de cinco parámetros, entre ellos:
  - *String nombre:* nombre de la GPC a introducir.
  - *String ruta_guia:* ruta donde se encuentra la guía a introducir.
  - *String especialidad:* colección donde se desea almacenar la guía.
- *comprueba_nombre:* comprueba que el nombre de la guía no está repetido en la base de datos. Si lo está devolverá *false*. Requiere del paso de tres parámetros, entre ellos:
  - *String nombre_GC:* el nombre de la GPC a introducir para hacer la comprobación.
- *PasaFile_String:* se encarga de coger el *fichero de GPC* de tipo File y pasarlo a String para poderlo almacenar. Exige el paso de:
  - *File file:* archivo de GPC de tipo FILE.

Ilustración 39.- ***AñadirGC.java.***

3.3.2.6. Conexión_BD.java

Esta clase se encarga de establecer la conexión con la colección que se desee y si ésta no existiera se encarga también de crearla. Otra función de esta clase es la de, al inicio del programa, ver las colecciones que existen y almacenarlas en un ArrayList y, si no existiera ninguna, crea una por defecto (*"General"*).

Posee dos métodos:

- *getColeccion:* su función es la de establecer y devolver la colección cuyo nombre se pasa como parámetro. Si esta colección no existe primero se crea y depués se actualiza el ArrayList de *Bean_servicios* que contiene las colecciones que están creadas.

  Requiere el paso, entre otros, de:

  - *String colección_BD:* nombre de la colección que se quiere recuperar.

- *Recopila_Especialidades:* este método se llama al inicio del servicio al realizar la validación y cargar los fichero de configuración. Se encarga de comprobar si en el interior de la colección principal del servicio, *"Gestion_GC"*, existe al menos una colección creada. Si no se cumple, crea la colección por defecto *"General"*. Y, por último, carga en un ArrayList todas las colecciones creadas para poder mostrarlas posteriormente. Este ArrayList está en *Bean_servicios*. No requiere de ningún parámetro y sólo se ejecuta una vez.

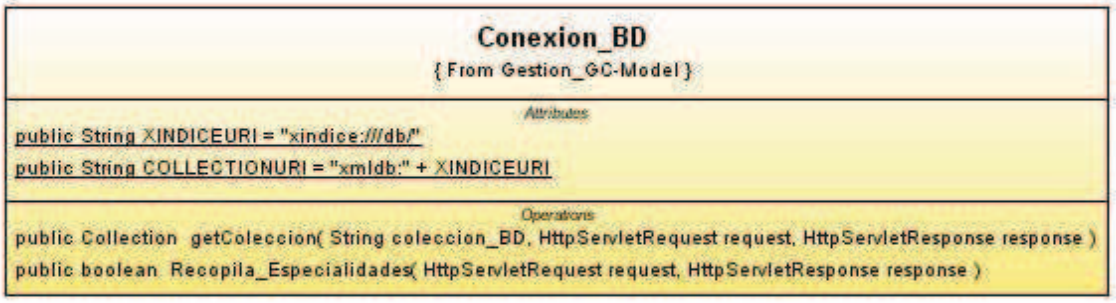

Ilustración 40.- ***Conexion_BD.java.***

3.3.2.7. MostrarGGCC.java

La clase MostrarGGCC almacena en un ResourceSet una única GPC, tras haberla almacenado con la clase AñadirGC, o un conjunto de ellas pertenecientes a una colección, o todas las GPCs almacenadas en cada una de las colecciones.

Se basa en que todas las GPC en XML deben tener como etiqueta principal *"/Management_Guideline",* luego mediante una búsqueda con XPath se recoge el o los recursos para almacenarlos en un ResourceSet para posteriormente poder ser mostrados.

Posee del único método *"mostrar_guias"* que solicita:

- *String decide:* indica si lo que se debe mostrar es una GPC o un conjunto de ellas.

- *String colección:* indica la colección donde se encuentran la/s GPC/s a mostrar.

- *String nombre:* si es una única GPC la que se debe mostrar, se debe indicar su nombre, si no da igual el valor que tome este parámetro.

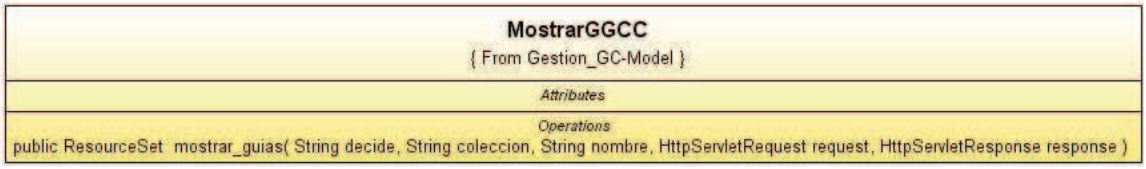

Ilustración 41.- ***MostrarGGCC.java.***

3.3.2.8. EliminarGC.java

EliminarGC es la clase encargada de eliminar una GPC de una colección. Su método *"eliminar"* sólo requiere de:

- *String nombre_GC:* nombre de la GPC a eliminar.

- *String coleccion_GC:* colección donde se encuentra a GPC a eliminar.

Ilustración 42.- ***EliminarGC.java.***

3.3.2.9. Buscar_GGCC.java

Se trata de la clase más elaborada y extensa. Buscar_GGCC tiene como función la búsqueda lógica de GPCs. Esta búsqueda se ha realizado de tal forma que permita introducir los comandos lógicos "AND" y "OR" pudiendo concatenar búsquedas y hacerlas mucho más concretas y puntuales.

Se denomina búsqueda lógica a la concatenación de búsquedas simples mediante criterios lógicos. Por ejemplo: "*Diagnóstico=Varicela AND Síntomas=fiebre"* es una búsqueda lógica que contiene dos búsquedas simples unidas por un criterio lógico.

Posee seis métodos:

- *buscarGGCC_criterio:* método principal donde recae toda la funcionalidad de esta clase. Se encarga de realizar las operaciones que correspondan llamando a otros métodos de esta clase para así poder proporcionar el resultado deseado. Requiere de dos parámetros principalmente:

- *String consulta_xpath:* es una cadena que contiene toda la información para realizar la búsqueda lógica. Se precisa de una cadena de la forma:

```
#Campo_1=Valor_campo_1#... (AND/OR)….#Campo_2=Valor_campo_2#....(AND/OR)….
(….)….#Campo_n=Valor_campo_n#
```

  Donde "*Campo_k"* corresponde con el campo a buscar (diagnóstico, sexo, edad…); *"Valor_campo_k"* es el valor que se quiere de ese campo (Varicela, Varón, 25…); y por último para concatenar las búsquedas se utilizan los operadores lógicos *AND* u *OR.*

- *String especialidad:* la colección donde se quiere realizar la búsqueda lógica. Permite el caso de buscar en todas las colecciones si se le pasa por colección "*TODAS".*

La idea de la realización de la búsqueda lógica es hacer la primera búsqueda simple, hacer la segunda y con el resultado de ambas aplicar el operador lógico correspondiente; posteriormente se realizaría la tercera búsqueda y con el resultado anterior aplicar el operador lógico y así hasta la última búsqueda.

- *obtiene_ruta_XPath:* este método se encarga de obtener la ruta del campo que se le pasa en una GPC. Para ello es necesario:

  - *String criterio:* campo elegido para la búsqueda.

  - *Bean_rutas ruta_GPC:* bean donde se indica la ruta que sigue el campo elegido dentro de la GPC. Es necesario para la construcción de la cadena XPath a devolver.

- *crea_consultaXPath:* de la cadena anteriormente comentada, esta función se encarga de que con la ruta del campo, "*xpath",* y el *"Valor_campo"* formar una cadena conforme la estructura de búsqueda XPath. Requiere de:

  - *String xpath:* ruta del campo del que se quiere formar la cadena XPath.

  - *String valor_criterio:* valor del campo deseado.

- *separa_criterios:* con la estrutura de la cadena anteriormente mencionada en *"buscarGGCC_criterio",* este método se encarga de crear una tabla bien estructurada con dicha información que representa la búsqueda lógica completa.

  Para ello contabiliza las búsquedas simples que deben realizarse, que corresponderán con las filas de la tabla, y almacena en la primera columna el criterio de la búsqueda, en la segunda el valor de ese criterio y en la

tercera si existe una nueva búsqueda simple, contendrá el valor lógico con el que se relaciona, y si no existe, almacena null. Únicamente necesita de la cadena "*consultax_path*" de *"buscarGGCC_criterio".* Un ejemplo de cómo quedaría dicha tabla sería el siguiente:

Campo_1	*Valor_criterio_1*	*(AND/OR)*
Campo _2	*Valor_campo_2*	*(AND/OR)*
Campo _3	*Valor_campo_3*	*(AND/OR)*
…	…	…
Campo _4	*Valor_campo_4*	*(AND/OR)*

Tabla 54.- Tabla de *separa_criterios.*

Síntomas	*Fiebre*	*OR*
Sintomas	*Picor*	*AND*
Diagnóstico	*Varicela*	*AND*
…	…	…
Autor	*Daniel*	*null*

Tabla 55.- Tabla ejemplo de *separa_criterios.*

- *funcion_OR:* es llamada para realizar la primera búsqueda simple dentro de la búsqueda lógica; y también al aplicar el operador lógico OR entre las búsquedas simples anteriormente realizadas y la actual. Requiere, entre otros, de los parámetros:

  - *ResourceSet resultadoSetGPC:* conjunto de los recursos resultado de las anteriores búsquedas.

  - *String xpath:* consulta XPath a realizar.

  - *String especialidad:* colección donde se desea realizar la búsqueda.

  - *Bean_servicio serv_config:* Bean que se requiere para obtener las colecciones que existen para el caso de una búsqueda en todas las colecciones.

  La forma de realizar esta búsqueda es la siguiente: se tiene un ResourceSet con los resultados de las búsquedas simples anteriores y la nueva consulta XPath de la siguiente búsqueda simple. Se realiza dicha consulta y se añaden todos los resultados al anterior ResourceSet. Esto puede conllevar a que existan GPCs repetidas. Luego, por último se comprueba si existe alguna repetida. Si existiera se guarda el índice que ocupa dentro del ResourceSet para posteriormente borrarlo de éste.

- *funcion_AND:* aplica el operador lógico AND entre las búsquedas simples anteriormente realizadas y la actual. Requiere, entre otros, de los parámetros:

  - *ResourceSet resultadoSetGPC:* conjunto de los recursos resultado de las anteriores búsquedas.

  - *String xpath:* consulta XPath a realizar.

  Para conseguir la funcionalidad AND se ha seguido el siguiente procedimiento: se extrae uno a uno los recursos almacenados en el

ResourceSet de las búsquedas simples anteriormente realizadas y se hace la nueva búsqueda en cada uno de ellos. Si la búsqueda no tiene éxito, se almacena el índice de dicho recurso dentro del ResourceSet para posteriormente eliminarlo de él.

Ilustración 43.- *Buscar_GGCC.java.*

3.3.2.10. Consultar.java

Esta clase se encarga de devolver un ResourceSet con el resultado de la consulta de una GPC. Para realizar la consulta XPath debe hacer uso de la función obtiene_rutaXPath de Buscar_GGCC.java.

Su único método *realizar_Consulta* exige el paso de, entre otros, los siguientes parámetros:

- *String nombre_GPC:* identificador de la GPC que se desea consultar.
- *String coleccion:* colección a la que pertenece dicha GPC.
- *String criterio:* criterio que se desea consultar de la GPC.

Ilustración 44.- *Consultar.java.*