Manual de la API del Registro Telemático UPV

1.Introducción

La adaptación de la UPV a los requisitos de la tramitación electrónica se traduce en la utilización del sistema de registro GEISER (Gestión Integrada de Servicios de Registro), que constituye una una solución integral de registro que funciona en modo nube para prestar el servicio para cualquier organismo público, que cubre tanto la gestión de sus oficinas de registro de entrada/salida como la recepción y envío de registros en las unidades tramitadoras destinatarias de la documentación.

La aplicación GEISER cubre todas las necesidades de gestión de la documentación presentada por los ciudadanos en las oficinas de registro presencial de las administraciones, así como las necesidades de Registro de las sedes electrónicas. También cubre la gestión de recepción y envío de registros entre las unidades tramitadoras destinatarias de la documentación a las que prestan servicio las oficinas de registro dotadas de GEISER.

GEISER asegura la supresión del movimiento de papel mediante la digitalización de la documentación presentada por el ciudadano en las oficinas cumpliendo el Esquema Nacional de Interoperabilidad, de intercambio de registros en formato electrónico con todas las administraciones integradas en SIR, y de integración con las aplicaciones de tramitación para el intercambio de asientos y documentación, así como los requerimientos legales contemplados en la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas (https://boe.es/diario_boe/txt.php?id=BOE-A-2015-10565).

Por tanto, la adopción de la solución de GEISER para el registro implica el cambio de la API del registro telemático actual que utiliza el sistema MASTIN, para que en su lugar utilice el módulo de interoperabilidad con el servicio común de registro GEISER. Este módulo se llama REGECO y proporciona servicios de registro y de consulta de asientos.

2,Web Service del Registro Telemático

El servicio web del registro telemático recibe como parámetros un XML y, en su caso, una serie de ficheros anexos.

El WSDL en desarrollo se define en:

https://aplidesa.upv.es/rtupv-app/services/RegistroTelematicoDispatcherWS?wsdl

Se invocará siempre a la operación lanzaDispatcher, que examinará el XML que recibe para saber qué operación tiene que realizar.

En producción la url es:

https://aplicat.upv.es/rtupv-app/services/RegistroTelematicoDispatcherWS?wsdl

a.Autenticación de las aplicaciones cliente.

El registro telemático comprobará que las aplicaciones llamantes tengan permisos para el mismo. Las aplicaciones cliente tendrán un usuario y contraseña para el acceso al registro telemático. La autenticación será básica, es decir, que estas aplicaciones, en las peticiones al registro telemático tendrán que enviar en la cabecera Authorization el valor:

Basic usuario:password

Donde la cadena usuario:password debe ir codificada en base 64.

Se asignará un usuario a cada aplicación cliente.

b.Creación de un apunte en el registro.

El servicio web de creación y envío de un asiento recibe como parámetros un XML con los datos del asiento a crear y sus documentos anexos. El servicio web realizará una petición de registro y envío de asiento a REGECO.

Geiser proporciona 2 tipos de registros de asientos:

- **Registro:** Se realiza un apunte en el registro, pero no se lleva a cabo ningún envío.
- **Registro y envío simple**: Se envía a la unidad de destino, pero el asiento va a la oficina de registro que da servicio a esa unidad para que se confirme en la oficina de registro a través de Geiser.

- **Registro y envío directo a la unidad**: Se envía a la unidad de destino, pero directamente pasa a la unidad, los usuarios de la oficina de registro que da servicio a la unidad no tienen que hacer nada. Directamente lo verían en Geiser los usuarios que tienen asignado el ámbito correspondiente a esa unidad de destino.

El XML de entrada para ambos servicios será el mismo excepto el nombre del nodo raíz que será el que indique qué tipo de registro se realiza:

- **API_CREAREGISTROSINENVIO:** No se envía, simplemente se asigna un número de registro al apunte.
- **API_CREAREGISTROSIMPLE:** Se enviará a la oficina de registro que dé servicio a la unidad de destino.
- **API_CREAREGISTRO:** Se enviará directamente a la unidad de destino.

Operación	Creación de registro
Parámetros de entrada	- XML con datos a registrar. - Documentos anexos en formato DataHandler
Datos de salida	XML con los datos de entrada más los datos resultado del registro (número de registro, CSV, justificante en PDF...)
Observaciones	Usuario y contraseña – No se usan actualmente *

DTD de entrada:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT API_CREAREGISTRO (DATOS_DEL_REGISTRO, FIRMA_DATOS_DEL_REGISTRO?)>
<!ATTLIST API_CREAREGISTRO
        type CDATA #REQUIRED
        formato CDATA #REQUIRED
        identificador CDATA #REQUIRED
        version CDATA #REQUIRED >
        idioma_error CDATA #REQUIRED >
<!ELEMENT DATOS_DEL_REGISTRO (
        DATOS_LIBRO_REGISTRAL,
        DATOS_UNIDAD,
        DATOS_INTERESADO+,
        OTROS_DATOS,
        DATOS_EXPEDIENTES,
        DOCUMENTOS_ANEXOS+) >
<!ELEMENT DATOS_LIBRO_REGISTRAL (
        TIPO_LIBRO,
        AMBITO_LIBRO_REGISTRAL,
        ANYO,
        CODIGO_OFICINA_REGISTRAL) >
<!ELEMENT TIPO_LIBRO ( #PCDATA ) >
<!ELEMENT AMBITO_LIBRO_REGISTRAL ( #PCDATA ) >
<!ELEMENT ANYO ( #PCDATA ) >
<!ELEMENT CODIGO_OFICINA_REGISTRAL ( #PCDATA ) >
<!ELEMENT DATOS_UNIDAD (
        CODIGO_UNIDAD,
        TIPO_UNIDAD,
        DECODIFICACION_UNIDAD?) >
<!ELEMENT CODIGO_UNIDAD ( #PCDATA ) >
```

```
<!ELEMENT TIPO_UNIDAD ( #PCDATA ) >
<!ELEMENT DECODIFICACION_UNIDAD ( #PCDATA ) >
<!ELEMENT DATOS_INTERESADO (
                    TIPO_PERSONA,
                    NOMBRE,
                    APELLIDO1,
                    APELLIDO2,
                    RAZON_SOCIAL?,
                    TIPO_DOCUMENTO,
                    NUMERO_IDENTIFICACION?,
                    TIPO_TITULAR,
                    TELEFONO?,
                    TELEFONO_MOVIL?,
                    FAX?,
                    DIRECCION_CORREO?,
                    DATOS_DIRECCION* ) >
<!ELEMENT TIPO_PERSONA ( #PCDATA ) >
<!ELEMENT APELLIDO1 ( #PCDATA ) >
<!ELEMENT APELLIDO2 ( #PCDATA ) >
<!ELEMENT RAZON_SOCIAL ( #PCDATA ) >
<!ELEMENT TIPO_DOCUMENTO ( #PCDATA ) >
<!ELEMENT NUMERO_IDENTIFICACION ( #PCDATA ) >
<!ELEMENT TIPO_TITULAR ( #PCDATA ) >
<!ELEMENT TELEFONO ( #PCDATA ) >
<!ELEMENT TELEFONO_MOVIL ( #PCDATA ) >
<!ELEMENT FAX ( #PCDATA ) >
<!ELEMENT DIRECCION_CORREO ( #PCDATA ) >
<!ELEMENT DATOS_DIRECCION (
                    TIPO_DOMICILIO,
                    PAIS?,
                    PROVINCIA?,
                    COMARCA?,
                    MUNICIPIO?,
                    POBLACION?,
                    DIRECCION,
                    CODIGO_POSTAL? ) >
<!ELEMENT TIPO_DOMICILIO ( #PCDATA ) >
<!ELEMENT PAIS ( #PCDATA ) >
<!ELEMENT PROVINCIA ( #PCDATA ) >
<!ELEMENT COMARCA ( #PCDATA ) >
<!ELEMENT MUNICIPIO ( #PCDATA ) >
```

```
<!ELEMENT POBLACION ( #PCDATA ) >
<!ELEMENT DIRECCION ( #PCDATA ) >
<!ELEMENT CODIGO_POSTAL ( #PCDATA ) >
<!ELEMENT OTROS_DATOS (
                CODIGO_ASUNTO?,
                DESCRIPCION_ASUNTO?,
                OBSERVACIONES?,
                TIPO_ENVIO?,
                ESTADO?,
                REFERENCIA_EXTERNA?,
                CODIGO_TIPO_TRANSPORTE?,
                NUMERO_TRANSPORTE? ) >
<!ELEMENT CODIGO_ASUNTO ( #PCDATA ) >
<!ELEMENT DESCRIPCION_ASUNTO ( #PCDATA ) >
<!ELEMENT OBSERVACIONES ( #PCDATA ) >
<!ELEMENT REFERENCIA_EXTERNA ( #PCDATA ) >
<!ELEMENT CODIGO_TIPO_TRANSPORTE ( #PCDATA ) >
<!ELEMENT NUMERO_TRANSPORTE ( #PCDATA ) >
<!ELEMENT DATOS_EXPEDIENTES (
                TIPO,
                TIPOACTOR,
                VALORACTOR,
                A_CODIGO_EXPEDIENTE,
                B_CODIGO_PROCED?,
                N_CODIGO_PROCED?,
                N_ASUNTO?,
                DATOS_PROPIOS? ) >
<!ELEMENT TIPO ( #PCDATA ) >
<!ELEMENT TIPOACTOR ( #PCDATA ) >
<!ELEMENT VALORACTOR ( #PCDATA ) >
<!ELEMENT A_CODIGO_EXPEDIENTE ( #PCDATA ) >
<!ELEMENT B_CODIGO_PROCED ( #PCDATA ) >
<!ELEMENT N_CODIGO_PROCED ( #PCDATA ) >
<!ELEMENT N_ASUNTO ( #PCDATA ) >
<!ELEMENT DATOS_PROPIOS (
                ID_FORMULARIO,
                TABLA ) >
<!ELEMENT ID_FORMULARIO ( #PCDATA ) >
<!ELEMENT TABLA (
                NOMBRE_TABLA,
```

```
                CAMPO+ ) >
<!ELEMENT NOMBRE_TABLA ( #PCDATA ) >
<!ELEMENT CAMPO (
                NOMBRE,
                VALOR ) >
<!ELEMENT VALOR ( #PCDATA ) >
<!ELEMENT DOCUMENTOS_ANEXOS (
                NOMBRE_FICHERO,
                CODIGO_DOCUMENTO?,
                DESCRIPCION_DOCUMENTO?,
                ASOCIO_EXPEDIENTE?,
                TIPO_DOCUMENTO,
                VALIDEZ?,
                FIRMA_DOCUMENTO?) >
<!ELEMENT NOMBRE_FICHERO ( #PCDATA ) >
<!ELEMENT CODIGO_DOCUMENTO ( #PCDATA ) >
<!ELEMENT DESCRIPCION_DOCUMENTO ( #PCDATA ) >
<!ELEMENT ASOCIO_EXPEDIENTE ( #PCDATA ) >
<!ELEMENT FIRMA_DOCUMENTO (
                ENTIDAD_CERTIFICADORA,
                FORMATO_FIRMA,
                VALOR_FIRMA )>
<!ELEMENT FIRMA_DATOS_DEL_REGISTRO (
                ENTIDAD_CERTIFICADORA,
                FORMATO_FIRMA,
                VALOR_FIRMA )>
<!ELEMENT ENTIDAD_CERTIFICADORA ( #PCDATA ) >
<!ELEMENT FORMATO_FIRMA ( #PCDATA ) >
<!ELEMENT VALOR_FIRMA ( #PCDATA ) >
```

DTD de salida:

El XML de salida será el mismo que el de entrada, pero añadiéndole un tag más. La estructura sería la siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT API_CREAREGISTRO (DATOS_DEL_REGISTRO, FIRMA_DATOS_DEL_REGISTRO?, JUSTIFICANTE,FIRMA_JUSTIFICANTE)>
<!ATTLIST API_CREAREGISTRO
                type CDATA #REQUIRED
                formato CDATA #REQUIRED
                identificador CDATA #REQUIRED
                version CDATA #REQUIRED >
                idioma_error CDATA #REQUIRED >
<!ELEMENT DATOS_DEL_REGISTRO (
```

```
                DATOS_LIBRO_REGISTRAL,
                DATOS_UNIDAD,
                DATOS_INTERESADO+,
                OTROS_DATOS,
                DATOS_EXPEDIENTES,
                DOCUMENTOS_ANEXOS+) >
...

<!ELEMENT JUSTIFICANTE (
                NUMERO_REGISTRO,
                FECHA_REGISTRO,
                EXPEDIENTE,
                CLAVE_CONSULTA,
                FICHERO)
<!ELEMENT NUMERO_REGISTRO ( #PCDATA ) >
<!ELEMENT FECHA_REGISTRO ( #PCDATA ) >
<!ELEMENT EXPEDIENTE ( #PCDATA ) >
<!ELEMENT CLAVE_CONSULTA ( #PCDATA ) >
<!ELEMENT FICHERO ( #PCDATA ) >
<!ELEMENT FIRMA_JUSTIFICANTE (
                ENTIDAD_CERTIFICADORA,
                FORMATO_FIRMA,
                VALOR_FIRMA )>
<!ELEMENT ENTIDAD_CERTIFICADORA ( #PCDATA ) >
<!ELEMENT FORMATO_FIRMA ( #PCDATA ) >
<!ELEMENT VALOR_FIRMA ( #PCDATA ) >
```

El formato de los valores de los elementos de los DTDS anteriores será el siguiente:

DATOS_LIBRO_REGISTRAL	
TIPO_LIBRO	=1: ENTRADA <>1: SALIDA
AMBITO_LIBRO_REGISTRAL	No se usa
ANYO	No se usa
CODIGO_OFICINA_REGISTRAL	No se usa
DATOS_UNIDAD	
CODIGO_UNIDAD	Código DIR3 de la unidad de destino (registro de entrada) o de la unidad de origen (registro de salida) - DEPRECATED (usar en su luegar CODIGO_UNIDAD_ORIGEN y CODIGO_UNIDAD_DESTINO).
CODIGO_UNIDAD_ORIGEN	Código DIR3 de la unidad de origen. Es obligatorio si:

	▪ Es un asiento de entrada y no se definen interesados. ▪ Es un asiento de salida.
CODIGO_UNIDAD_DESTINO	Código DIR3 de la unidad de destino. Es obligatorio si: • Es un asiento de entrada. • Es un asiento de salida y no se definen interesados.
TIPO_UNIDAD	No se usa
DECODIFICACION_UNIDAD	No se usa
DATOS_INTERESADO	
TIPO_PERSONA	No se usa
NOMBRE	Nombre del interesado
APELLIDO1	Primer apellido del interesado
APELLIDO2	Segundo apellido del interesado
RAZON_SOCIAL	Razón social
TIPO_DOCUMENTO	Será obligatorio indicarlo si se rellena el campo NUMERO_IDENTIFICACION y podrá tener los siguientes valores: 1- NIF 2- CIF 3- PASAPORTE 4- DOCUMENTO IDENTIFICACIÓN EXTRANJEROS 5- OTROS PERSONA FÍSICA 6- CÓDIGO ORIGEN
NUMERO_IDENTIFICACION	Número del documento indicado. El formato dependerá del valor indicado en TIPO_DOCUMENTO.
TIPO_TITULAR	No se usa
TELEFONO	Teléfono del interesado
TELEFONO_MOVIL	Teléfono móvil del interesado
FAX	No se usa
DIRECCION_CORREO	Email del interesado
DATOS_DIRECCION	
TIPO_DOMICILIO	P = Principal
PAIS	Código del país del interesado según valores en catálogo ISO 3166.
PROVINCIA	Código de la provincia del interesado, este valor es el definido en el OM del Padrón (11/7/1997).
COMARCA	No se usa
MUNICIPIO	Código del municipio del interesado, este valor es el definido en el OM del Padrón (11/7/1997)
POBLACION	No se usa
DIRECCION	Dirección del interesado
CODIGO_POSTAL	Código postal del interesado
OTROS DATOS	
CODIGO_ASUNTO	Código de Asunto. El asunto debe haber sido dado de alta previamente en REGECO
DESCRIPCION_ASUNTO	Resumen del asiento
OBSERVACIONES	Observaciones
TIPO_ENVIO	Sólo aplicable al registro y envío simple: API_CREAREGISTROSIMPLE.

	Tipo de envío que se quiere hacer, los valores válidos son: ENVIO_DESTINO -> Permite hacer un envío a un órgano de destino que esté conectado ENVIO_INTERESADO -> Permite hace un envío a un interesado. Para futuras versiones. No implementado por el momento ENVIO_NOTIFICA -> Permite hace un envío a Notific@. Para futuras versiones. No implementado por el momento Si no se indica este dato, se usará por defecto ENVIO_DESTINO.
ESTADO	Estado en el que se crea el registro. Sólo es aplicable para la operación API_CREAREGISTROSINENVIO. Los valores posibles son: ▪ FINALIZADO (es necesario poner este valor si el destino es un interesado en lugar de una unidad). ▪ PENDIENTE_ENVIO
REFERENCIA_EXTERNA	Referencia externa
CODIGO_TIPO_TRANSPORTE	Puede tener los siguientes valores: SERVICIO_MENSAJEROS / CORREO_POSTAL / CORREO_POSTAL_CERTIFICADO / BUROFAX / EN_MANO / FAX / OTROS
NUMERO_TRANSPORTE	Número/referencia del transporte
DATOS_EXPEDIENTES	
TIPO	No se usa
TIPOACTOR	No se usa
VALORACTOR	No se usa
A_CODIGO_EXPEDIENTE	Código del expediente
B_CODIGO_PROCED	No se usa
N_CODIGO_PROCED	No se usa
N_ASUNTO	No se usa
DATOS_PROPIOS no se usa	
DOCUMENTOS_ANEXOS	
NOMBRE_FICHERO	Nombre del fichero anexo
CODIGO_DOCUMENTO	Identificador del documento
DESCRIPCION_DOCUMENTO	Descripción del documento
ASOCIO_EXPEDIENTE	No se usa
TIPO_DOCUMENTO	Tipo mime del documento
VALIDEZ	Validez del documento anexo. Los valores posibles son: COPIA/COPIA_COMPULSADA/COPIA_ORIGINAL/ORIGINAL. Si no se indica este campo, se pondrá por defecto el valor COPIA.
FIRMA_DOCUMENTO	
ENTIDAD_CERTIFICADORA	Entidad certificadora de la firma del documento anexo
FORMATO_FIRMA	Formato de la firma
VALOR_FIRMA	Valor de la firma
FIRMA_DATOS_DEL_REGISTRO	No se usa

Ejemplo de uso del WS para la creación de un apunte registral:

Existen 2 métodos en el WS que permiten hacer los apuntes registrales:

- lanzaDispatcher: Sus parámetros de entrada son el xml de la petición y un array de DataHandlers.

- lanzaDispatcherF: Sus parámetros de entrada son el xml de la petición y un array de objetos para definir los anexos indicando el anexo codificado en base 64 y el tipo mime del fichero.

En Java, podemos hacer una llamada al servicio web para crear un asiento de la siguiente forma haciendo uso del método lanzaDispatcher:

```
import javax.xml.namespace.QName;
import org.apache.axis.client.Call;
import org.apache.axis.cliente.Service;


...

//Obtenemos los datos a enviar como parámetros (xml de entrada y lista de ficheros anexos)

String path1="path/to/file1", path2 = "path/to/file2";

String xmlEntrada="<?xml version=\"1.0\" encoding=\"iso-8859-1\"?> <API_CREAREGISTRO… ";

File file1 = new File(path1);

File file2 = new File(path2);

DataHandler fichxml = new DataHandler(new ByteArrayDataSource(xmlEntrada,”text/xml"));

DataHandler fichero1 = new DataHandler(new FileDataSource(file1));

DataHandler fichero2 = new DataHandler(new FileDataSource(file2));


List<DataHandler> ficheros = Arrays.asList(fichxml, fichero1, fichero2);


//Una vez tenemos los datos hacemos la petición al servicio web.

String endpoint = "http://rtupvdes.upv.es/rtupv-app/services
/RegistroTelematicoDispatcherWS";

Service  service = new Service();

Call call= (Call) service.createCall();

call.setTargetEndpointAddress( new java.net.URL(endpoint) );

call.setOperationName(new QName("http://rtupvdes.upv.es/rtupv-app/services
/RegistroTelematicoDispatcherWS", "lanzaDispatcher"));

call.setProperty(Call.USERNAME_PROPERTY, "usuario");

call.setProperty(Call.PASSWORD_PROPERTY, "password");

String ret = (String) call.invoke( new Object[] {xmlEntrada,ficheros} );
```

El ejemplo se ha realizado usando Apache Axis. Las dependencias para su uso son:

```
    <dependency>
        <groupId>org.apache.axis</groupId>
        <artifactId>axis</artifactId>
        <version>1.4</version>
    </dependency>
  <dependency>
        <groupId>javax.xml</groupId>
        <artifactId>jaxrpc-api</artifactId>
        <version>1.1</version>
  </dependency>
```

En el XML se debe definir como primer documento anexo el xml de entrada al servicio seguido del resto de documentos anexos. Y en la lista de ficheros pasados como argumento se pasan los ficheros en el mismo orden en que se encuentran definidos en el xml de Entrada.

En caso de que el registro se realice correctamente el String devuelto contiene el xml de Salida con el número de registro, fecha de registro, etc.

En caso de errores en la creación de registro, el servicio web devolverá en el String de resultado el mensaje de descripción del error, por ejemplo:

“5 - identificadorInteresado: el formato del NIF no es válido”.

*Para los ejemplos en java hemos utilizado Apache Axis 1.4.

En PL/SQL, podemos hacer una llamada al servicio web para crear un asiento haciendo uso del método lanzaDispatcherF. Para ello deberemos montar el xml correspondiente a la petición SOAP (ver ejemplo de búsqueda de apuntes). En el cuerpo de la petición tendremos lo siguiente:

```
<ws:lanzaDispatcherF soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <xmlEntrada xsi:type="xsd:string">
        <![CDATA[
            <?xml version="1.0" encoding="iso-8859-1"?>
                <API_CREAREGISTRO idioma_error="" version="1.0" identificador="" formato="Mastin" type="E">
                    <DATOS_DEL_REGISTRO><DATOS_LIBRO_REGISTRAL>...</DATOS_DEL_REGISTRO>
                </API_CREAREGISTRO>]]>
    </xmlEntrada>

    <files xsi:type="reg:ArrayOf_tns1_Fichero" soapenc:arrayType="mod:Fichero[]" xmlns:reg="https://aplidesa.upv.es/rtupv-app/services/RegistroTelematicoDispatcherWS"    xmlns:mod="http://modelo.rt.upv.es">
        <fichero xsi:type="reg:Fichero">
            <contenido xsi:type="xsd:string"><![CDATA[cadena_contenido_base64]]></contenido>
            <tipoMime xsi:type="xsd:string">application/xml</tipoMime>
        </fichero>
        <fichero xsi:type="reg:Fichero">
            <contenido xsi:type="xsd:string"><![CDATA[cadena_contenido_base64]]></contenido>
            <tipoMime xsi:type="xsd:string">application/pdf</tipoMime>
        </fichero>
    </files>
</ws:lanzaDispatcherF>
```

c.Consulta de apuntes.

La consulta de registro permite consultar los datos de un asiento por número de registro.

Operación	Consulta de registro
Parámetros de entrada	- XML con datos a consultar
Datos de salida	- XML con el resultado de la consulta
DTD de entrada:	

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT API_CONSULTAREGISTRO (DATOS_CONSULTA)>
<!ATTLIST API_CONSULTAREGISTRO
                type CDATA #REQUIRED
                formato CDATA #REQUIRED
                identificador CDATA #REQUIRED
                version CDATA #REQUIRED>
                idioma_error CDATA #REQUIRED >
```

```
<!ELEMENT DATOS_CONSULTA (
                NUMERO_REGISTRO,
                FECHA_REGISTRO,
                CLAVE_CONSULTA,
                DEVOLVER_DOCUMENTOS,
                DOCUMENTOS_FIRMADOS)>
<!ELEMENT NUMERO_REGISTRO (#PCDATA)>
<!ELEMENT FECHA_REGISTRO (#PCDATA)>
<!ELEMENT CLAVE_CONSULTA (#PCDATA)>
<!ELEMENT DEVOLVER_DOCUMENTOS (#PCDATA)>
<!ELEMENT DOCUMENTOS_FIRMADOS (#PCDATA)>
```

DTD de salida:

El XML de salida será el mismo que el de entrada, pero con los resultados de la consulta.

```
<!ELEMENT RESULTADOS_CONSULTA (DATOS_ASIENTO)*>
<!ELEMENT DATOS_ASIENTO (
                        NUMERO_REGISTRO,
                        FECHA_PRESENTADO,
                        FECHA_REGISTRADO,
                        CSV,
                        AMBITO_CREACION,
                        AMBITO_ACTUAL,
                        TIPO_ASIENTO,
                        ESTADO
                         ORGANO_DESTINO,
                        CODIGO_ASUNTO,
                       DESCRIPCION_ASUNTO,
                        RESUMEN,
                        NOMBRE_USUARIO,
                        CODIGO_TIPO_TRANSPORTE,
                         NUMERO_REGISTRO_ORIGEN,
                         DATOS_INTERESADO*,
                       DOCUMENTOS_ANEXOS*
                         )>
<!ELEMENT NUMERO_REGISTRO (#PCDATA)>
<!ELEMENT FECHA_PRESENTADO (#PCDATA)>
<!ELEMENT FECHA_REGISTRADO (#PCDATA)>
<!ELEMENT CSV (#PCDATA)>
<!ELEMENT AMBITO_CREACION (#PCDATA)>
<!ELEMENT AMBITO_ACTUAL (#PCDATA)>
```

<!ELEMENT TIPO_ASIENTO (#PCDATA)>

<!ELEMENT ESTADO (#PCDATA)>

<!ELEMENT ORGANO_DESTINO (#PCDATA)>

<!ELEMENT CODIGO_ASUNTO (#PCDATA)>

<!ELEMENT DESCRIPCION_ASUNTO (#PCDATA)>

<!ELEMENT RESUMEN (#PCDATA)>

<!ELEMENT NOMBRE_USUARIO (#PCDATA)>

<!ELEMENT CODIGO_TIPO_TRANSPORTE (#PCDATA)>

<!ELEMENT NUMERO_REGISTRO_ORIGEN (#PCDATA)>

<!ELEMENT DATOS_INTERESADO(

TIPO_DOCUMENTO,

NUMERO_IDENTIFICACION,

NOMBRE,

APELLIDO1,

APELLIDO2)>

<!ELEMENT DOCUMENTOS_ANEXOS(

NOMBRE_FICHERO,

CODIGO_DOCUMENTO?,

DESCRIPCION_DOCUMENTO?,

TIPO_DOCUMENTO?,

CONTENIDO?)>

En la parte de los anexos se añadirá el contenido si así se indica en la consulta. El contenido del anexo estará codificado en BASE 64 se llamará <CONTENIDO>. Por ejemplo, la sección <DOCUMENTOS_ANEXOS> quedaría:

<DOCUMENTOS_ANEXOS>

<NOMBRE_FICHERO>prueba.pdf</NOMBRE_FICHERO>

<CODIGO_DOCUMENTO>000000320856_157c0dbf-105b-4698-b6fd-42452ccaf1b0</CODIGO_DOCUMENTO>

<DESCRIPCION_DOCUMENTO/>

<TIPO_DOCUMENTO>application/pdf</TIPO_DOCUMENTO>

<CONTENIDO>...</CONTENIDO>

</DOCUMENTOS_ANEXOS>

La consulta devuelve una tupla para cada estado distinto por el que pase un asiento, por lo que puede haber varios apuntes para el mismo número de registro.

DATOS_CONSULTA	
NUMERO_REGISTRO	Número de registro. Obligatorio ya que es el dato por el que se consulta en REGECO
FECHA_REGISTRO	Fecha de registro. En formato: YYYYMMDDHHMMSS
CLAVE_CONSULTA	Clave de consulta
DEVOLVER_DOCUMENTOS	S/N (indica si hay que devolver los documentos anexos).
DOCUMENTOS_FIRMADOS	No se usa

En REGECO sólo se puede filtrar por número de registro. Sin embargo mantenemos el resto de parámetros por si nos sirven para filtrar en caso de varios asientos con el número de registro por copia de asiento en GEISER, etc.

Ejemplo de uso del WS para la consulta de apuntes:

En el siguiente código la variable de tipo String ret contendrá el XML de salida con el formato indicado anteriormente.

```
String endpoint = "http://rtupvdes.upv.es/rtupv-app/services/RegistroTelematicoDispatcherWS
";


Service  service = new Service();

Call call;

String ret = "";

String consulta = "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?> <API_CONSULTAREGISTRO… “;


try {

call = (Call) service.createCall();

        call.setTargetEndpointAddress( new java.net.URL(endpoint) );

        call.setOperationName(new QName("http://rtupvdes.upv.es/rtupv-app/services
/RegistroTelematicoDispatcherWS", "lanzaDispatcher"));

        call.setProperty(Call.USERNAME_PROPERTY, "usuario");

call.setProperty(Call.PASSWORD_PROPERTY, "password");

        ret = (String) call.invoke( new Object[] {consulta} );

} catch (ServiceException e) {

// TODO Auto-generated catch block

        e.printStackTrace();

}
```

d.Búsqueda de apuntes

La búsqueda de registro permite buscar asientos en base a varios parámetros de entrada. El resultado será la lista de asientos que cumplen el criterio de búsqueda.

No devolvemos la documentación. Para obtener la documentación de un asiento concreto se usa la consulta de asiento por número de registro.

Operación	Búsqueda de registro
Parámetros de entrada	- XML con datos de los asientos a buscar
Datos de salida	- XML con el resultado de la búsqueda
DTD de entrada:	
<?xml version="1.0" encoding="iso-8859-1"?> <!ELEMENT API_BUSCAREGISTRO (DATOS_BUSQUEDA)> <!ATTLIST API_ BUSCAREGISTRO type CDATA #REQUIRED formato CDATA #REQUIRED identificador CDATA #REQUIRED version CDATA #REQUIRED>	

idioma_error CDATA #REQUIRED >

<!ELEMENT DATOS_BUSQUEDA(

TIPO_ASIENTO,

FECHA_REGISTRADO_DESDE,

FECHA_REGISTRADO_HASTA,

ORGANO_ORIGEN,

ORGANO_DESTINO,

FECHA_PRESENTADO,

FECHA_PRESENTADO_DESDE,

FECHA_PRESENTADO_HASTA,

FECHA_EVENTO_DESDE,

FECHA_EVENTO_HASTA,

CODIGO_ASUNTO,

ESTADO,

IDENTIFICADOR_INTERESADO

)>

<!ELEMENT TIPO_ASIENTO (#PCDATA)>

<!ELEMENT FECHA_REGISTRADO_DESDE (#PCDATA)>

<!ELEMENT FECHA_REGISTRADO_HASTA (#PCDATA)>

<!ELEMENT ORGANO_ORIGEN (#PCDATA)>

<!ELEMENT ORGANO_DESTINO (#PCDATA)>

<!ELEMENT FECHA_PRESENTADO (#PCDATA)>

<!ELEMENT FECHA_PRESENTADO_DESDE (#PCDATA)>

<!ELEMENT FECHA_PRESENTADO_HASTA (#PCDATA)>

<!ELEMENT FECHA_EVENTO_DESDE (#PCDATA)>

<!ELEMENT FECHA_EVENTO_HASTA (#PCDATA)>

<!ELEMENT CODIGO_ASUNTO (#PCDATA)>

<!ELEMENT ESTADO (#PCDATA)>

<!ELEMENT IDENTIFICADOR_INTERESADO (#PCDATA)>

DTD de salida:

El XML será el mismo de entrada pero se añadirá los elementos correspondientes a los asientos encontrados:

<!ATTLIST API_ BUSCAREGISTRO

type CDATA #REQUIRED

formato CDATA #REQUIRED

identificador CDATA #REQUIRED

version CDATA #REQUIRED>

idioma_error CDATA #REQUIRED >

<!ELEMENT DATOS_BUSQUEDA …>

<!ELEMENT RESULTADOS_BUSQUEDA (DATOS_ASIENTO)*>

<!ELEMENT DATOS_ASIENTO (

NUMERO_REGISTRO,

```
                                FECHA_PRESENTADO,
                                FECHA_REGISTRADO,
                                CSV,
                                AMBITO_CREACION,
                                 AMBITO_ACTUAL,
                                TIPO_ASIENTO,
                                ESTADO
                                ORGANO_DESTINO,
                                CODIGO_ASUNTO,
                               DESCRIPCION_ASUNTO,
                                NUMERO_REGISTRO_ORIGEN,
                                DATOS_INTERESADO*
                                )>
<!ELEMENT NUMERO_REGISTRO (#PCDATA)>
<!ELEMENT FECHA_PRESENTADO (#PCDATA)>
<!ELEMENT FECHA_REGISTRADO (#PCDATA)>
<!ELEMENT CSV (#PCDATA)>
<!ELEMENT AMBITO_CREACION (#PCDATA)>
<!ELEMENT AMBITO_ACTUAL (#PCDATA)>
<!ELEMENT TIPO_ASIENTO (#PCDATA)>
<!ELEMENT ESTADO (#PCDATA)>
<!ELEMENT ORGANO_DESTINO (#PCDATA)>
<!ELEMENT CODIGO_ASUNTO (#PCDATA)>
<!ELEMENT DESCRIPCION_ASUNTO (#PCDATA)>
<!ELEMENT NUMERO_REGISTRO_ORIGEN (#PCDATA)>
<!ELEMENT DATOS_INTERESADO(
                                TIPO_DOCUMENTO,
                                NUMERO_IDENTIFICACION,
                                NOMBRE,
                                APELLIDO1,
                                APELLIDO2)>
```

DATOS_BUSQUEDA	
TIPO_ASIENTO	ENTRADA/SALIDA
FECHA_REGISTRADO_DESDE	Fecha inicial de registro. Formato: YYYYMMDDHHMMSS
FECHA_REGISTRADO_HASTA	Fecha final de registro. Formato: YYYYMMDDHHMMSS
ORGANO_ORIGEN	Código DIR3 del órgano de origen
ORGANO_DESTINO	Código DIR3 del órgano de destino
FECHA_PRESENTADO	Fecha de presentación. Formato: YYYYMMDDHHMMSS
FECHA_PRESENTADO_DESDE	Fecha inicial de presentación. Formato: YYYYMMDDHHMMSS
FECHA_PRESENTADO_HASTA	Fecha final de presentación. Formato: YYYYMMDDHHMMSS

FECHA_EVENTO_DESDE	Fecha inicial del último evento en el asiento. Formato: YYYYMMDDHHMMSS
FECHA_EVENTO_HASTA	Fecha final del último evento en el asiento. Formato: YYYYMMDDHHMMSS
CODIGO_ASUNTO	Código del asunto
ESTADO	Estado del asiento. Los valores posibles se muestran en la tabla de estados.
IDENTIFICADOR_INTERESADO	Identificador del interesado.
DATOS_ASIENTO	
NUMERO_REGISTRO	ENTRADA/SALIDA
FECHA_PRESENTADO	Fecha de presentación del registro. Formato: YYYYMMDDHHMMSS
FECHA_REGISTRADO	Fecha de registro. Formato: YYYYMMDDHHMMSS
CSV	CSV
AMBITO_CREACION	Ámbito (unidad u oficina) de creación del registro.
AMBITO_ACTUAL	Ámbito (unidad u oficina) donde se encuentra el registro
TIPO_ASIENTO	ENTRADA/SALIDA
ESTADO	Estado del asiento. Los valores posibles se muestran en la tabla de estados.
ORGANO_DESTINO	Código DIR3 del órgano destino del asiento
CODIGO_ASUNTO	Código del asunto
DESCRIPCION_ASUNTO	Descripción del asunto
NUMERO_REGISTRO_ORIGEN	Número de registro de origen (si el asiento deriva de otro)

Los estados posibles en que se puede encontrar un apunte son los siguientes:

ESTADOS	
SIN_DATOS	No puede ser enviado porque no tiene todos los datos obligatorios.
PENDIENTE_ENVIO	Pendiente de envío al destino.
ENVIADO_PENDIENTE_CONFIRMACION	Enviado pendiente de confirmación (estado sólo visible desde origen).
ENVIADO_PENDIENTE_CONFIRMACION_MANUAL	Enviado pendiente de confirmación cuando el destino está desconectado (sólo visible desde origen).
RECIBIDO_PENDIENTE_CONFIRMACION	Recibido pendiente de confirmación (sólo visible desde destino).
RECIBIDO_PENDIENTE_CONFIRMACION_MANUAL	Recibido pendiente de confirmación cuando el destino está desconectado (sólo visible desde destino).
ENVIADO_CONFIRMADO	Confirmado en destino (sólo visible desde origen)
RECIBIDO_CONFIRMADO	Confirmado en destino (sólo visible desde destino)
ENVIADO_RECHAZADO	Rechazado en destino (sólo visible desde origen)
RECIBIDO_RECHAZADO	Rechazado en destino (sólo visible desde destino)
ANULADO	Anulado
REENVIADO	Reenviado.
EN_TRAMITE	Confirmado por unidad tramitadora
ASIGNADO	Asignado a un subórgano
FINALIZADO	Último estado de un asiento.
REENVIADO_RECHAZADO	Reenviado y rechazado por destino a la oficina del último reenvio.
RECTIFICADO	Modificado para su envío por SIR
ENVIO_PROCESO	Estado temporal pendiente de ser enviado.

RECIBIDO_RECHAZADO_CIUDADANO	Rechazado de una unidad a su oficina de registro con destino a un interesado.

Ejemplo de uso del WS para la búsqueda de apuntes:

En este caso el cliente del web service será idéntico al de la consulta, pero el xml de entrada deberá ser el correspondiente al servicio de búsqueda de asientos:

```
String endpoint = "http://rtupvdes.upv.es/rtupv-app/services
/RegistroTelematicoDispatcherWS";


Service  service = new Service();

Call call;

String ret = "";

String consulta = "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?> <API_BUSCAREGISTRO… ";


try {

call = (Call) service.createCall();

        call.setTargetEndpointAddress( new java.net.URL(endpoint) );

        call.setOperationName(new QName("http://rtupvdes.upv.es/rtupv-app/services
/RegistroTelematicoDispatcherWS", "lanzaDispatcher"));

        call.setProperty(Call.USERNAME_PROPERTY, "usuario");

call.setProperty(Call.PASSWORD_PROPERTY, "password");

        ret = (String) call.invoke( new Object[] {consulta} );

} catch (ServiceException e) {

// TODO Auto-generated catch block

        e.printStackTrace();

}
```

Si se quiere consumir el servicio desde PL/SQL, un ejemplo para usar el servicio de búsqueda sería el siguiente:

```
declare

  soap_request varchar2(30000);

  soap_respond clob;

  http_req utl_http.req;

  http_resp utl_http.resp;

  resp XMLType;

  i integer;

  l_header varchar2(1000);

  eob boolean:=false;

  buffer VARCHAR2(32767):='';

  v_offset number := 1;

  v_chunk_size number := 10000;

  l_xml clob:=EMPTY_CLOB();

begin

  dbms_lob.createtemporary(soap_respond, true);
```

```
  l_header:='Basic '||utl_raw.cast_to_varchar2(utl_encode.base64_encode(utl_raw.cast_to_raw('testUser:
testPassword')));

  soap_request:= '<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ws="http://ws.rt.upv.es" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">

   <soapenv:Header/>

   <soapenv:Body>

      <ws:lanzaDispatcher soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

         <xmlEntrada xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="iso-8859-1"?>

            <API_BUSCAREGISTRO type="E" formato="" identificador="" version="" idioma_error="" validarFirma="">

            <DATOS_BUSQUEDA>

                <TIPO_ASIENTO>ENTRADA</TIPO_ASIENTO>

                <FECHA_REGISTRADO_DESDE></FECHA_REGISTRADO_DESDE>

                <FECHA_REGISTRADO_HASTA></FECHA_REGISTRADO_HASTA>

                <ORGANO_ORIGEN/>

                <ORGANO_DESTINO/>

                <FECHA_PRESENTADO/>

                <FECHA_PRESENTADO_DESDE/>

                <FECHA_PRESENTADO_HASTA/>

                <FECHA_EVENTO_DESDE/>

                <FECHA_EVENTO_HASTA/>

                <CODIGO_ASUNTO>UPV_0001</CODIGO_ASUNTO>

                <ESTADO/>

                <IDENTIFICADOR_INTERESADO/>

            </DATOS_BUSQUEDA>

            </API_BUSCAREGISTRO>]]>

         </xmlEntrada>

         <dataHandlers xsi:type="reg:ArrayOf_xsd_DataHandler" soapenc:arrayType="xsd:DataHandler[]" xmlns:reg="
http://rtupvdes.upv.es/rtupv-app/services/RegistroTelematicoDispatcherWS"/>

      </ws:lanzaDispatcher>

   </soapenv:Body>

</soapenv:Envelope>

';


 http_req:= utl_http.begin_request

            ( 'http://rtupvdes.upv.es/rtupv-app/services/RegistroTelematicoDispatcherWS?lanzaDispatcher'

            , 'POST'

            , 'HTTP/1.1'

            );

  utl_http.set_header( http_req, 'Content-Type', 'text/xml; charset=UTF-8' );

  utl_http.set_header(http_req, 'Content-Length', length(soap_request));

  utl_http.set_header(http_req, 'SOAPAction', ''); -- required to specify this is a SOAP communication

  utl_http.set_header(http_req, 'Authorization', l_header);

  utl_http.write_text(http_req, soap_request);

  http_resp:= utl_http.get_response(http_req);
```

```
  -- LEEMOS LA RESPUESTA:
  while not(eob)
    loop
      begin
        utl_http.read_text(http_resp, buffer, 32767); -- buffer = VARCHAR2(32767)
        if buffer is not null and length(buffer)>0 then
          dbms_lob.writeappend(soap_respond, length(buffer), buffer);

        end if;
      exception when UTL_HTTP.END_OF_BODY then
        eob := true;
      end;
    end loop;

  utl_http.end_response(http_resp);

  -- PARSEAMOS LA RESPUESTA A XML Y OBTENEMOS EL CONTENIDO DEL SOAP ENVELOPE
  resp:= XMLType(soap_respond);

  l_xml := resp.getClobVal();

  if (l_xml is null) then
    dbms_output.put_line('Es nulo');
  else
  --IMPRIMIMOS LA RESPUESTA
    loop
      exit when v_offset > dbms_lob.getlength(l_xml);
      dbms_output.put_line( dbms_lob.substr( l_xml, v_chunk_size, v_offset ) );
      v_offset := v_offset + v_chunk_size;
    end loop;
  end if;
end;
```

3. Estructura de la base de datos.

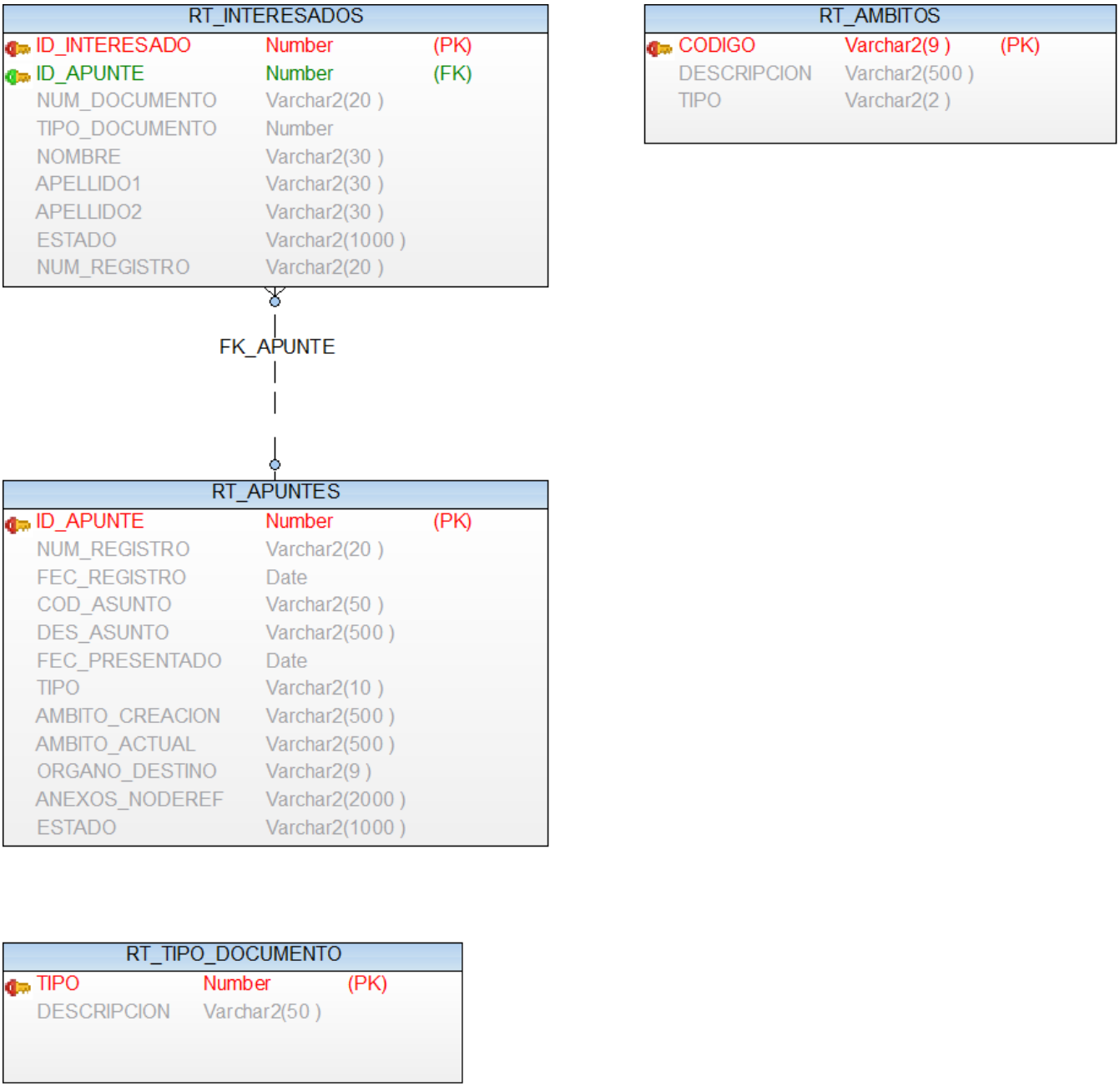
Los datos de los apuntes realizados en el registro se guardarán en la base de datos en un proceso nocturno que se ejecutará diariamente.

Las bases de datos serán las siguientes:

- Desarrollo: Base de datos REGDES, usuario DESAREGTEL.
- Producción: Base de datos REGISTRO, usuario REGTEL.

Si el apunte tiene anexos, éstos se guardarán en Alfresco.

La estructura de la base de datos es la siguiente:



El campo anexos_noderef de la tabla RT_APUNTES contendrá la referencia al nodo de alfresco correspondiente a la carpeta que contiene los anexos del apunte. Si el apunte no tiene anexos, este campo estará vacío.

Para cada asiento registral con su número de registro correspondiente puede haber múltiples tuplas en la tabla RT_APUNTES, debido a que en el sistema Geiser, un apunte tiene un estado distinto según el ámbito en el que se encuentre o el estado por el que pase.

Por ejemplo, el siguiente apunte está ya en tramitación y por tanto ha pasado por el estado ENVIADO_CONFIRMADO y actualmente está en el estado EN_TRAMITE:

ID_APUNTE	NUM_REGISTRO	FEC_REGISTRO	COD_ASUNTO	DES_ASUNTO	FEC_PRESENTADO	TIPO	AMBITO_CREACION	AMBITO_ACTUAL	ORGANO_DESTINO	ANEXOS_NODEREF	ESTADO
146	O00022801e1700000004	26/06/2017 14:22:36	001	Solicitud becas comedor 2017/2018	26/06/2017 14:22:36	ENTRADA	Registro General (CAMPUS DE VERA)	Secretaría General	U02700007		EN_TRAMITE
179	O00022801e1700000004	26/06/2017 14:22:36	001	Solicitud becas comedor 2017/2018	26/06/2017 14:22:36	ENTRADA	Registro General (CAMPUS DE VERA)	Secretaría General	U02700007		ENVIADO_CONFIRMADO

4.Adaptación de las aplicaciones existentes.

AUTENTICACIÓN

Para el uso del web service las aplicaciones clientes tendrán que utilizar la autenticación básica.

Se quitan de las llamadas a los métodos del servicio web los parámetros usuario y clave que existían anteriormente en todas las operaciones.

CREACIÓN DE ASIENTOS

Las diferencias en el modo de uso del servicio web del registro telemático anterior y esta nueva versión son:

- Hay 2 modos de realizar el registro, el envío simple (a la oficina de registro que da servicio a la unidad de destino y el envío directo hasta unidad (el asiento va directamente a la unidad de destino).
  Si se usa el mismo formato del XML (nodo raíz API_CREAREGISTRO) que se usaba para el registro de mastín, se realizará el envío directo hasta unidad.
  Si se quiere que el asiento pase por la oficina de registro que da servicio a la unidad de destino habrá que cambiar el XML para poner el nodo raíz API_CREAREGISTROSIMPLE.

- El sistema Geiser está integrado con DIR3 por lo que la codificación de las unidades y oficinas de registro debe ser esa misma. Las aplicaciones clientes del registro telemático deberán actualizar los códigos de las oficinas y unidades que utilizan al registrar para usar los códigos DIR3 correspondientes.

- El sistema Geiser requiere más especialización en la definición del tipo de documento de identificación de los interesados, por lo que habrá que usar los nuevos tipos indicados en las tablas anteriores y poner los documentos completos (NIF con números y letra, etc).

- Se han eliminado los parámetros “usuario” y “clave” porque no estaban usando.

CONSULTA Y BÚSQUEDA DE ASIENTOS

Los datos de los asientos creados con GEISER/REGECO se guardarán diariamente en la base de datos. Sin embargo la estructura de las tablas es distinta a la estructura que se usaba en Mastín, así que en este caso habrá que realizar una de las siguientes acciones:

- Utilizar el servicio web de consulta y búsqueda tal y como se ha definido anteriormente.
- Modificar las consultas a la base de datos que se hagan en la aplicación para adaptarlas al nuevo esquema (en este caso habrá que tener en cuenta que la información no estará del todo actualizada en el momento de la consulta).