2. WAINE

WAINE (Web Application INterface Engine) es un motor de desarrollo de aplicaciones web que, basándose en modelos de Interfaces de Usuario IU, permite crear las aplicaciones cumpliendo unos objetivos básicos tales como *Independencia, Seguridad, Personalización, Minimización de la programación y Eficiencia.* [2]

WAINE aprovecha la relación entre los modelos que componen un MBUIDE y las BBDD relacionales para almacenar todos los componentes de la aplicación en un BDRelacional llamada Meta-Base de datos.

Esta Meta-base de datos almacena los elementos claves del interfaz de usuario tales como *usuarios, grupos, permisos, acciones, parámetros…*

Sin embargo, la construcción a base de sentencias SQL de una aplicación es claramente inviable. Por ello WAINE define un lenguaje descriptivo basado en XML llamado ASL que ayuda a la definición de los elementos del MBUIDE. Con este lenguaje se escribe el código de la aplicación el cual mediante un script o comando se traduce a la MDB para su posterior visualización en el servidor.

El funcionamiento básico se resume en el siguiente esquema:

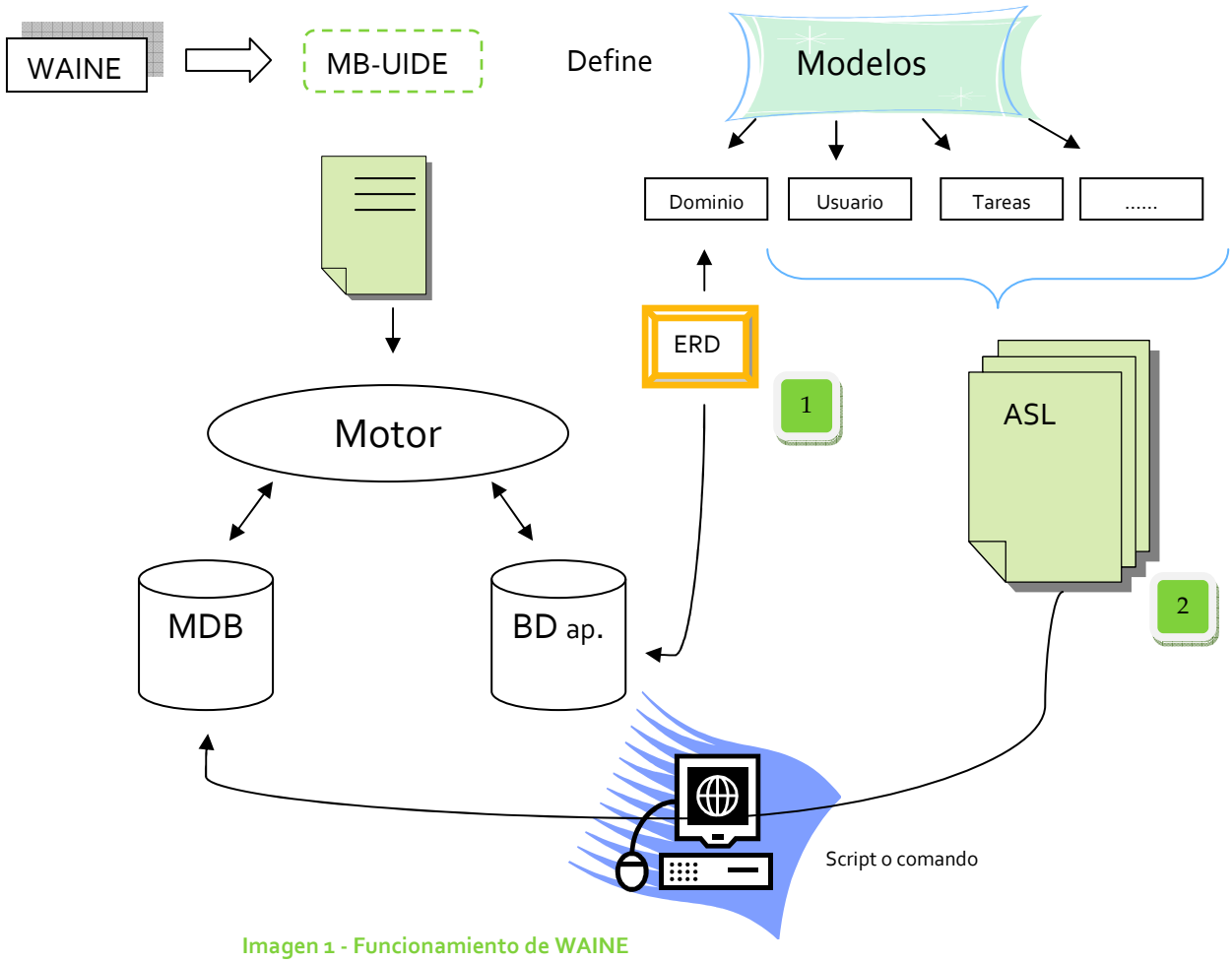

Imagen 1 - Funcionamiento de WAINE

2.1 ARQUITECTURA

Las aplicaciones creadas por WAINE siguen una estructura de niveles como la mostrada en el esquema anterior, donde destacan el nivel de datos (Base de datos y MDB), el nivel de aplicación (Código de la aplicación con la generación de formularios) y el nivel de presentación (Cliente).

En la siguiente imagen se muestra la arquitectura de la que se compone WAINE.

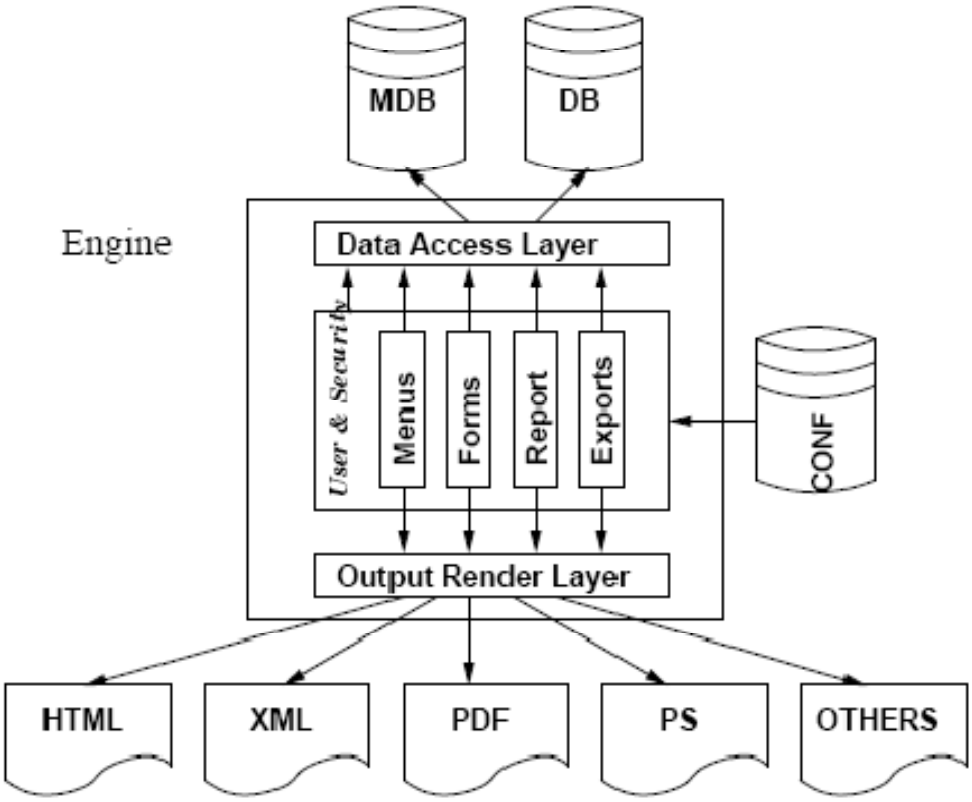

Imagen 2 - Arquitectura de WAINE

Cabe destacar que se implementa siguiendo cuatro elementos básicos:

1. DB (Base de datos) de la Aplicación, donde se guarda la información relativa a la aplicación.
2. La Meta-Base de datos donde se almacena el modelo de la aplicación.
3. El Motor de la aplicación, que accede a la MDB para generar la aplicación.
4. El Repositorio de configuraciones CONF, para poder personalizarlo o extenderlo de forma sencilla a base de parametrizar o crear clases del sistema.

Además de estos cuatro elementos, la arquitectura consta de dos capas de uso común:

1. Capa de Render de salida: Encargada de generar los archivos de salida tales como HTML, PDF, etc.
2. Capa de acceso de datos: Tal como su nombre indica permite el acceso a las fuentes de datos tales como la BD y la MDB.

En cuanto a la Meta-Base de Datos, su estructura es la siguiente:

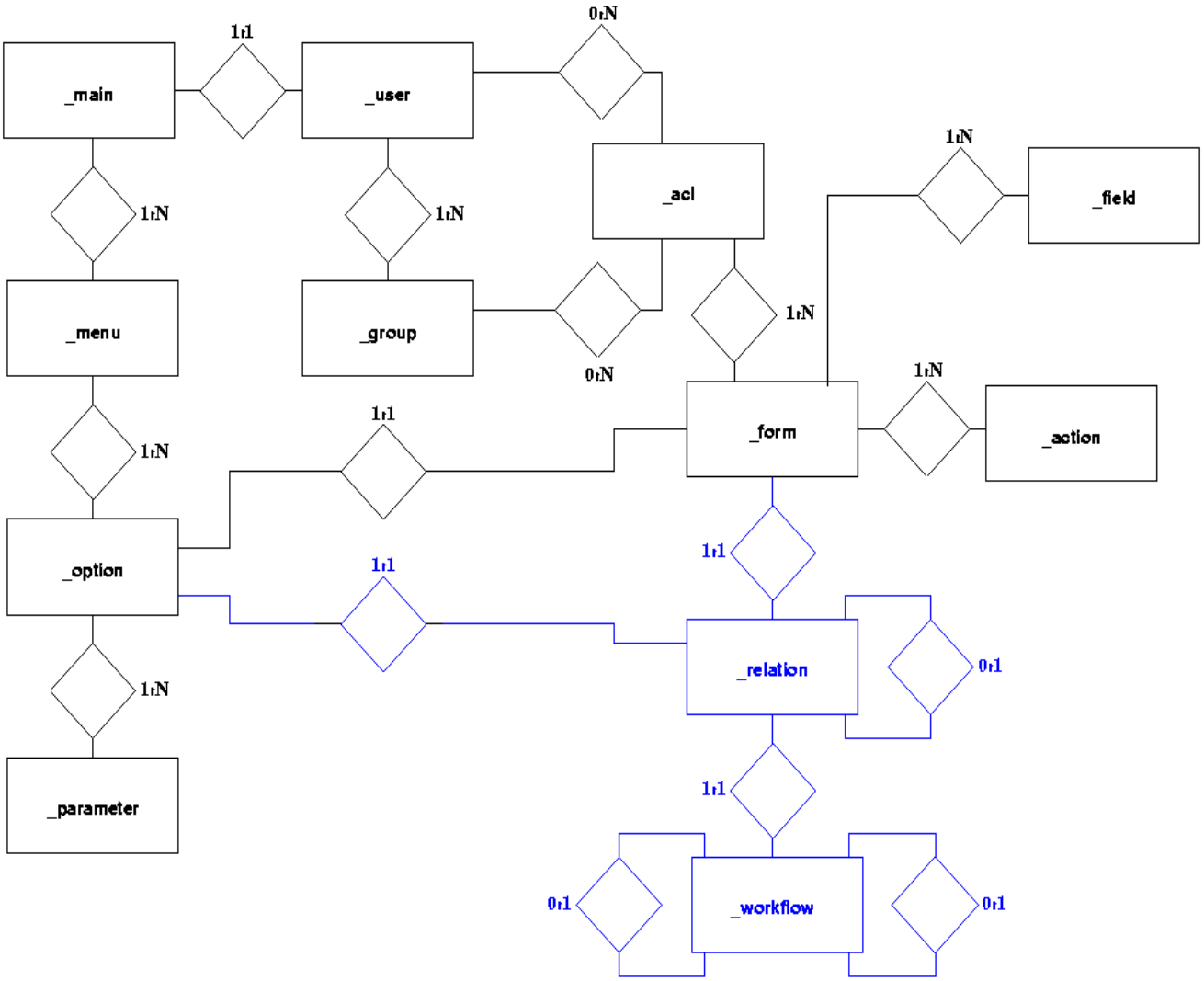

Imagen 3 – MDB

Como se muestra en el esquema anterior la MDB está estructurada en distintas tablas con sus respectivos campos, algunos de los cuales pasamos a detallar:

- **_main:** contiene el nombre del menú que se le mostrará a los usuarios de la aplicación. Cada aplicación puede tener distintos menús según el usuario que acceda y por tanto estará aquí reflejado.
    - mainid: identificador del main.
    - caption: nombre asociado al main.

- **_group:** contiene la información relativa a los grupos que accederán a la aplicación.
    - gid: identificador del grupo.
    - name: nombre asociado al grupo.
    - descr: descripción del grupo existente.

- **_user:** contiene la información de los usuarios de la aplicación tales como nombre, usuario, password, etc.
    - uid: identificador del usuario.
    - gid: identificador del grupo al que está asociado.
    - name: nombre o "user" del usuario.
    - passwd: clave o "password" asociado al usuario.
    - mainid: identificador del main asociado al usuario, es decir, el menú que verá el usuario al acceder.
    - descr: descripción del usuario.

- **_acl:** contiene la información sobre los permisos de acceso o ejecución sobre los formularios de modo que para dos usuarios la vista de la aplicación puede ser distinta.
    - uid: identificador del usuario al cual se le aplica la regla.
    - gid: identificador del grupo al cual se le aplica la regla.
    - formid: identificador del formulario sobre el que se aplica la regla.
    - read: indica si tiene permiso de lectura.
    - ins: indica si tiene permiso de inserción de datos.
    - del: indica si tiene permiso de borrado de datos.
    - upd: indica si tiene permiso de actualización.
    - action: indica si tiene permiso de ejecución de acciones.

- **_menu:** contiene información relativa a los menús que componen el main de cada usuario.
    - menuid: identificador del menú.
    - mainid: identificador del main asociado al menú. Todos los menús que sean del mismo main compartirán este valor.
    - ord: indica el orden de aparición del menú con respecto a otros dentro del main.
    - caption: mensaje o texto asociado al menú.
    - img: imagen asociada al menú.

- **_option:** contiene información relativa a las opciones asociadas al menú. Un main se compone de varios menús, y este a su vez de varios option u opciones.
    - optionid: identificador del option.
    - menuid: identificador del menú al que va asociado. En este caso se asocia al menú y no al main porque en varios main puede existir el mismo menú.
    - ord: indica el orden de aparición dentro del menú.
    - caption: mensaje o texto asociado.
    - structid: identificador de la estructura asociada.
    - url: dirección que se ejecuta al seleccionar el option.
    - img: imagen asociada al option.

- **_parameter:** contiene la información asociada a los parámetros que pueden contener los option tales como el formulario asociado, la inclusión de botones o el navegador.
    - parameterid: identificador del parámetro.
    - structid: identificador de la estructura a la que está asociada.
    - name: nombre del parámetro.
    - value: valor del parámetro asociado.

- **_form:** contiene la información relativa a los formularios de la aplicación tales como orden de los campos, fuente de datos, filtros, etc.
    - formid: identificador del formulario.

- source: fuente de datos del formulario. Esta fuente de datos puede ser una tabla de la base de datos o de la MDB tales como _user o _group.
- caption: mensaje o texto asociado al formulario.
- filter: regla de filtrado de los datos.
- orderby: indica el campo por el cual se ordena el formulario en una consulta.
- theme: contiene el nombre del tema utilizado para la visualización del formulario en la aplicación.

- **_field:** contiene la información relativa a los campos del formulario.
  - fieldid: identificador del campo.
  - formid: identificador del formulario asociado.
  - ord: orden de aparición con respecto al resto de campos.
  - caption: mensaje o texto asociado al campo.
  - type: indica el tipo de datos que contiene el campo, tales como string, integer, date, text, image, etc.
  - source: indica el nombre del campo de la tabla de la base de datos a la cual se está refiriendo.
  - len: indica la longitud del campo que se muestra en el formulario.
  - maxlen: máxima longitud del campo que se muestra en el formulario.
  - attr: atributo del campo ya sea de sólo lectura (R) u oculto (H).
  - search: necesita de searchfld para indicar la búsqueda del valor del campo del formulario en cuestión en el campo de otro formulario. Aquí se indica de que formulario se extrae este valor.
  - searchfld: indica el campo del formulario donde buscar el valor requerido.
  - canbenull: indica si el campo se puede dejar en blanco o no.
  - picture: indica el formato de los datos introducidos en el formulario.
  - defvalue: en caso de no rellenar el campo se indica el valor por defecto.
  - msg: mensaje que se muestra si no se cumplen las reglas de canbenull y picture.

- **_action:** contiene la información relativa a las acciones definidas para el formulario en forma de botones.
    - actionid: identificador de la acción.
    - formid: identificador del formulario asociado.
    - ord: orden de aparición con respecto a otras acciones.
    - type: indica el tipo de acción que realizará el botón en el formulario tales como "web", "system" o "execute" donde podemos escribir ordenes SQL.
    - caption: mensaje o texto asociado al botón de la acción.
    - msg: mensaje que se muestra en caso de error.

2.2 ASL

En este apartado se incluyen las diferentes instrucciones ASL que permite WAINE para la definición de los menús, formularios, estructuras, etc.

Lo primero que hay que definir en el código ASL son los grupos y usuarios que habrá en la aplicación inicialmente, definiendo después los menús que componen la aplicación, seguidos de sus estructuras, formularios, acciones, eventos, etc.

2.2.1 GRUPOS Y USUARIOS

Para crear los grupos y los usuarios pertenecientes, hemos de tener presente la definición de las tablas anteriores de la MDB. En la siguiente imagen se muestra como ha de definirse:

```
<group gid="0" name="administrador">
  <user uid="0" name="admin" passwd="admin" mainid="main_admin"
  descr="Administradores de la Aplicacion"/>
  </group>
```

Con estas instrucciones se definen los valores de gid y nombre para el grupo creado, y los valores de uid, gid, nombre, password, mainid y descripción para el usuario perteneciente a ese grupo.

Teniendo esto definido en el ASL, el usuario "admin" puede acceder a la aplicación una vez creada.

El siguiente paso que hemos de dar, es la creación de los menús que compondrán la aplicación así como los submenús y option, y los parámetros correspondientes.

2.2.2 MENUS Y SUBMENUS

Para poder definir los menús que posteriormente los usuarios visualizaran al arrancar la aplicación, es necesario crear inicialmente el main asociado a ese usuario y dentro de éste aquellos menús y submenús que el usuario podrá ver.

Para entender esto, a continuación se muestra un ejemplo del código para definirlo.

```
<main id="main_admin" caption="es=Menu de administracion|en=Admin menu">
        <menu caption="Aplicaciones">
            <option caption="Registro" call="st_aplicaciones"/>
        </menu>
    <menu caption="Usuarios">
        <option caption="Gestion" type="form"
                <param name="formid" value="form_reg_usuario"/>
                <param name="form_type" value="table"/>
                <param name="button_insert" value="1"/>
                <param name="button_delete" value="1"/>
                <param name="button_update" value="1"/>
        </option>
        <option caption="Asignacion de aplicaciones" call="st_usuarios"/>
    </menu>
        …
</main>
```

Como podemos apreciar en el ejemplo, las etiquetas main definen lo que será el menú general del usuario, en este caso el administrador.

Entre las mismas se definen los menús y dentro de éstos los submenús por medio de las etiquetas option.

Como se aprecia en el ejemplo podemos hacer una llamada a una estructura donde se definen los formularios, o simplemente hacerlo directamente en el option indicando el tipo (en este caso un formulario).

En este último caso tendremos los siguientes parámetros para definir el formulario:

- *formid:* identificador del formulario.
- *form_type:* donde indicamos el tipo de formulario a usar, a elegir entre form, table, list, grid, etc.
- *button_insert*: presencia o no del botón de inserción.
- *button_delete*: presencia o no del botón de borrado.
- *button_update*: presencia o no del botón de actualización.
- *source_filter_field*: campo por el cual se filtra el formulario.
- *source_filter_value*: valor del campo anterior.
- *source_filter_where*: regla de filtrado con la instrucción where propia de SQL.
- *Navigator_field*: campo filtro del navegador en el caso de usar navegador.
- *Navigator_position*: posición del navegador dentro del formulario.

Estos campos mostrados son los mismos que tendrían que definirse en la estructura en caso de haber hecho una llamada a alguna de ellas.

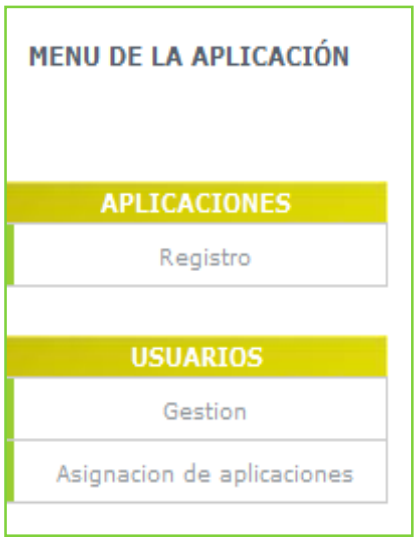

Imagen 4 - Menú del administrador

2.2.3 STRUCT

Como se ha comentado en el apartado anterior, dentro de los submenús se pueden hacer llamadas a "struct" que contendrán la estructura de los formularios.

Con los elementos struct podemos tener estructuras complejas del formulario dividiendo la pantalla en consultas a varios de ellos, declarando el orden de ellas por medio del parámetro "ord" presente en la MDB.

Se ilustrará esto con un ejemplo.

```
<struct id="st_gasign" type="relation">
        <param name="form_split" value="rows=20%,*"/>
        <param name="formid" value="form_reg_grupo"/>
        <param name="form_type" value="combo"/>
        <param name="fields_hidden" value="1"/>

        <param ord="2" name="formid" value="form_gr_ap"/>
        <param ord="2" name="form_type" value="table"/>
        <param ord="2" name="button_insert" value="1"/>
        <param ord="2" name="button_delete" value="1"/>
        <param ord="2" name="button_update" value="1"/>
        <param ord="2" name="source_filter_field" value="gid"/>

</struct>
```

A la hora de definir el struct hemos de especificar de qué tipo será, dentro de los tres tipos posibles:

- Relation: es el más común o usado de los tres y tal como su nombre indica se utiliza para establecer relaciones entre los formularios usados.
- Tab: En este caso los formularios se muestran en distintos tabs o pestañas que debemos seleccionar para verlos.
- Split: Este último tipo divide la pantalla en varias zonas donde podemos encontrar formularios o estructuras. En este caso no hay relación entre ellos.

A continuación presentamos un ejemplo de cada uno de ellos.

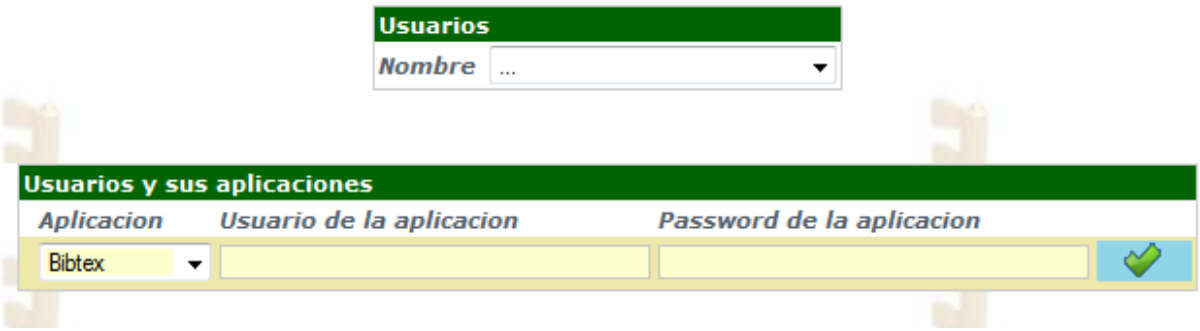

Imagen 5 - Struct con relation

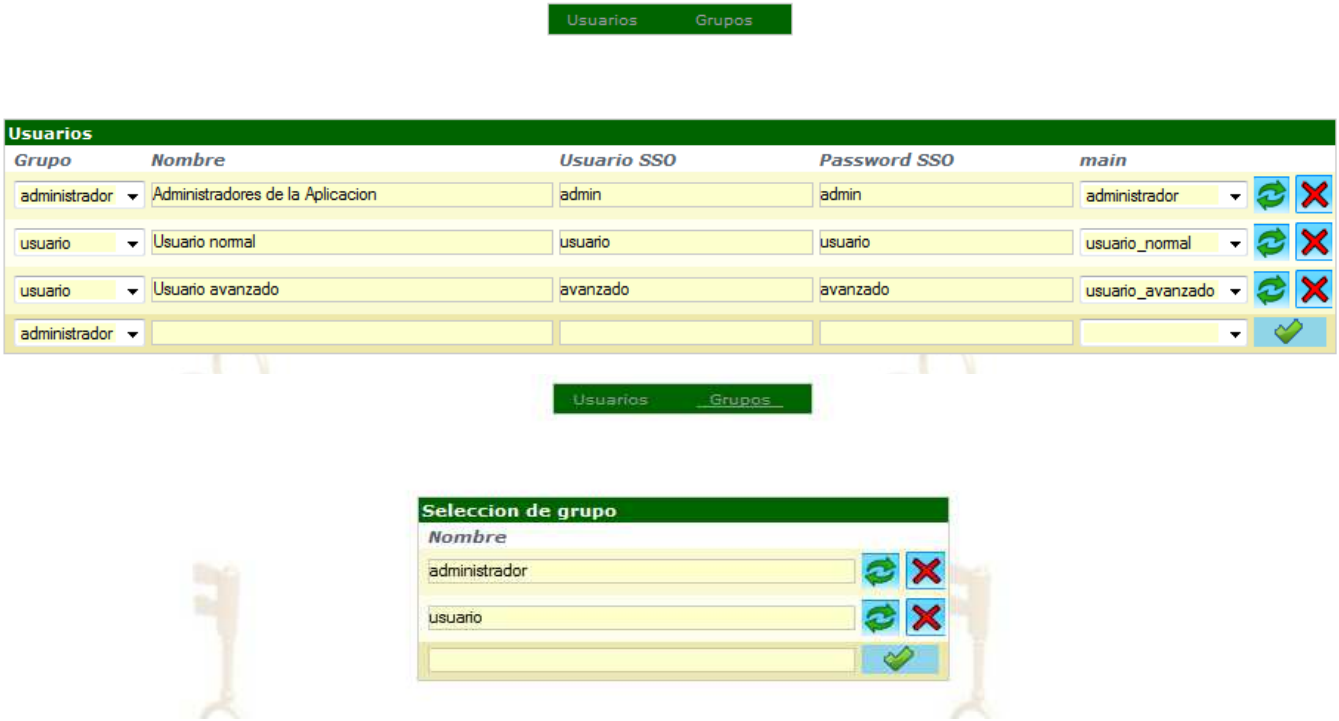

Imagen 6 - Struct con tabs usuario arriba y grupo abajo

Imagen 7 - Struct con Split

Para definir los Struct se utilizan parámetros similares a los del apartado anterior con los option, sólo que se añaden algunos mas.

- *structid:* identificador de la estructura.
- *form_type:* a elegir entre relation, split y tab.
- *form_split:* con este parámetros indicamos el valor en filas (rows) o columnas (cols) que ocupará cada tipo de la estructura.
- *tab_name:* nombre de la pestaña en caso de haber elegido esta opción.
- *formid:* identificador del formulario.
- *form_type:* donde indicamos el tipo de formulario a usar, a elegir entre form, table, list, grid, etc.
- *button_insert*: presencia o no del botón de inserción.
- *button_delete*: presencia o no del botón de borrado.
- *button_update*: presencia o no del botón de actualización.
- *source_filter_field*: campo por el cual se filtra el formulario.
- *source_filter_value*: valor del campo anterior.
- *source_filter_where*: regla de filtrado con la instrucción where propia de SQL.
- *Navigator_field*: campo filtro del navegador en el caso de usar navegador.
- *Navigator_position*: posición del navegador dentro del formulario.

2.2.4 FORMULARIOS

En este apartado se indican los parámetros para definir los formularios que son la base de la aplicación.

Para definir el formulario, se utilizan las etiquetas <form> donde tenemos que indicar la base de datos de la cual extraemos los datos o como en el caso especial del registro de usuarios indicar que utilizaremos la MDB, los campos que componen el formulario, el tipo, etc.

Ilustramos esto último con un ejemplo donde veremos distintos tipos de campos y las dos variantes de origen de datos, BD y MDB.

La primera trata sobre el registro de usuarios del sistema donde indicamos que utilizamos la MDB en el parámetro <datasource>mdb.cfg</datasource> y la tabla _user en el parámetro source="_user".

La segunda es la definición de un formulario tipo donde recurrimos a la base de datos de la aplicación.

```
<form id="form_reg_usuario" source="_user" caption="Usuarios">
  <datasource>mdb.cfg</datasource>
    <fields>
        <key source="uid"/>
        <int source="gid" caption="Grupo" len="30" maxlen="50"
    msg="Falta el grupo" canbenull="N">
            <search>form_reg_grupo</search>
              <searchfield>name</searchfield>
        </int>
        <string source="descr" caption="Nombre" len="50" maxlen="70"
        msg="No hay nombre"/>
        <string source="name" caption="Usuario SSO" len="30" maxlen="50"
        msg="No hay nombre de usuario"/>
        <string source="passwd" caption="Password SSO" len="30"
        maxlen="50" msg="Falta el password" canbenull="N"/>
        <string source="mainid" caption="main" len="50" maxlen="70"
        msg="No hay main">
            <search>;main_admin:administrador;main_usuario1:usuario_
            normal;main_usuario2:usuario_avanzado; </search>
        </string>
    </fields>
</form>
```

```
<form id="form_apps" source="Aplicaciones" caption="Registro de aplicaciones generales" >

    <orderby> nombre_ap </orderby>

        <fields>

            <key source="id_app"/>

            <string source="nombre_ap" caption="Nombre " len="20" maxlen="40" canbenull="N" msg="Nombre no existente"/>

            <text source="descr" caption="Descripcion " width="40" height="3" canbenull="Y"/>

            <image-file source="imagen" caption="Icono" canbenull="Y">

                <width>60</width>

                <height>60</height>

            </image-file>

            <string source="url" caption="Hipervinculo" len="60" maxlen="80" canbenull="N" msg="Debes introducir una url valida"/>

            <string source="tipo" caption="Tipo" len="15" maxlen="20" canbenull="N" msg=" Introduzca un tipo">

                <search>;general;especifica;</search>

            </string>

        </fields>

<form>
```

Ahora pasamos a explicar los campos mostrados en los formularios.

En todo formulario debemos indicar el identificador del formulario que las estructuras y submenús necesitan, la clave primaria de la tabla referenciada y los campos que lo componen.

- id: identificador del formulario.
- source: origen de la fuente de datos, es decir, el nombre de la tabla.
- caption: mensaje o texto asociado al formulario que se mostrará como título.
- datasource: este campo se utiliza para el caso de la MDB como se muestra en el primer ejemplo.
- filter: en el caso de querer emplear un filtro para el formulario se debe indicar aquí su valor.
- orderby: indica el campo por el cual se ordenaran los resultados del formulario.
- fields: etiqueta donde se empiezan a definir los campos del formulario que a continuación se detallan.
    - source: nombre del campo en la base de datos al cual hace referencia.
    - type: tipo de datos del campo a elegir entre int, string, text, float, date, time, timestamp, blob-file e image-file. Dentro de estos tipos de datos especificar que el tipo blob se utiliza como hipervínculo al archivo que se añade. Comentar que antes de definir los campos hay que definir la clave primaria mediante la instrucción key source.
    - caption: texto asociado al campo.
    - len: longitud del campo que se muestra en el formulario.
    - maxlen: máxima longitud del campo que se muestra.
    - attr: atributo asociado al campo, ya sea de solo lectura (R) u oculto (H).
    - msg: mensaje de error que aparece si no se introducen correctamente los campos en el formulario.
    - canbenull: indica si el campo asociado puede ser nulo o no.
    - format: especifica el formato de presentación de los datos.
    - picture: especifica el formato de los datos introducidos.

En la siguiente imagen se muestra el resultado de definir el formulario de la página anterior.

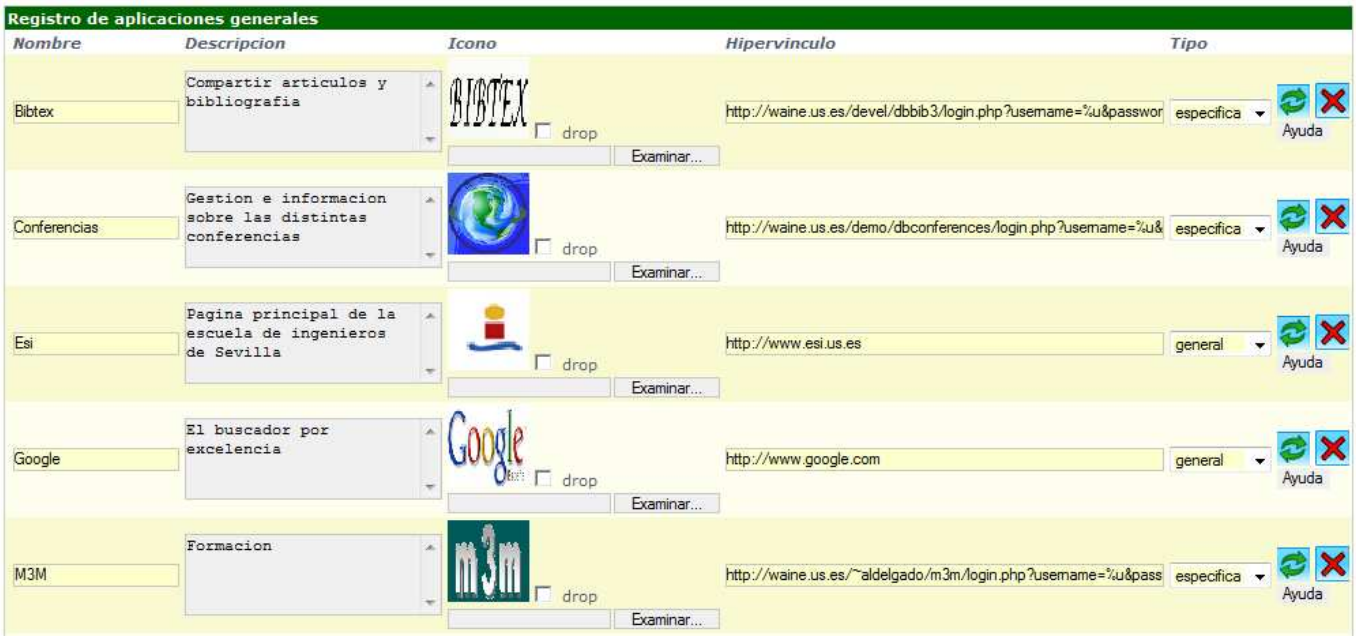

Imagen 8 - Formulario

2.2.5 ACCIONES

Dentro de un formulario podemos especificar otros botones distintos a los habituales (inserción, borrado y actualización) que realicen acciones definidas por el usuario.

Para ello debemos definirlos después del formulario entre las etiquetas <buttons> especificando seguidamente el tipo de acción a realizar entre las siguientes:

- System: esta acción se define para poder ejecutar algún comando de sistema.
- Web: esta acción se define para hacer una llamada a un hipervínculo web.
- Execute: esta acción se define para realizar llamadas a la base de datos.

En el siguiente ejemplo se muestra como definir un botón con acción.

```
<buttons>
        <action type="web" caption="Ayuda">
                <code>ayuda.php</code>
        </action>
</buttons>
```

Este es el resultado de la definición del modelo anterior, un botón de ayuda a la hora de rellenar el formulario que abre una página web donde se explica el procedimiento.

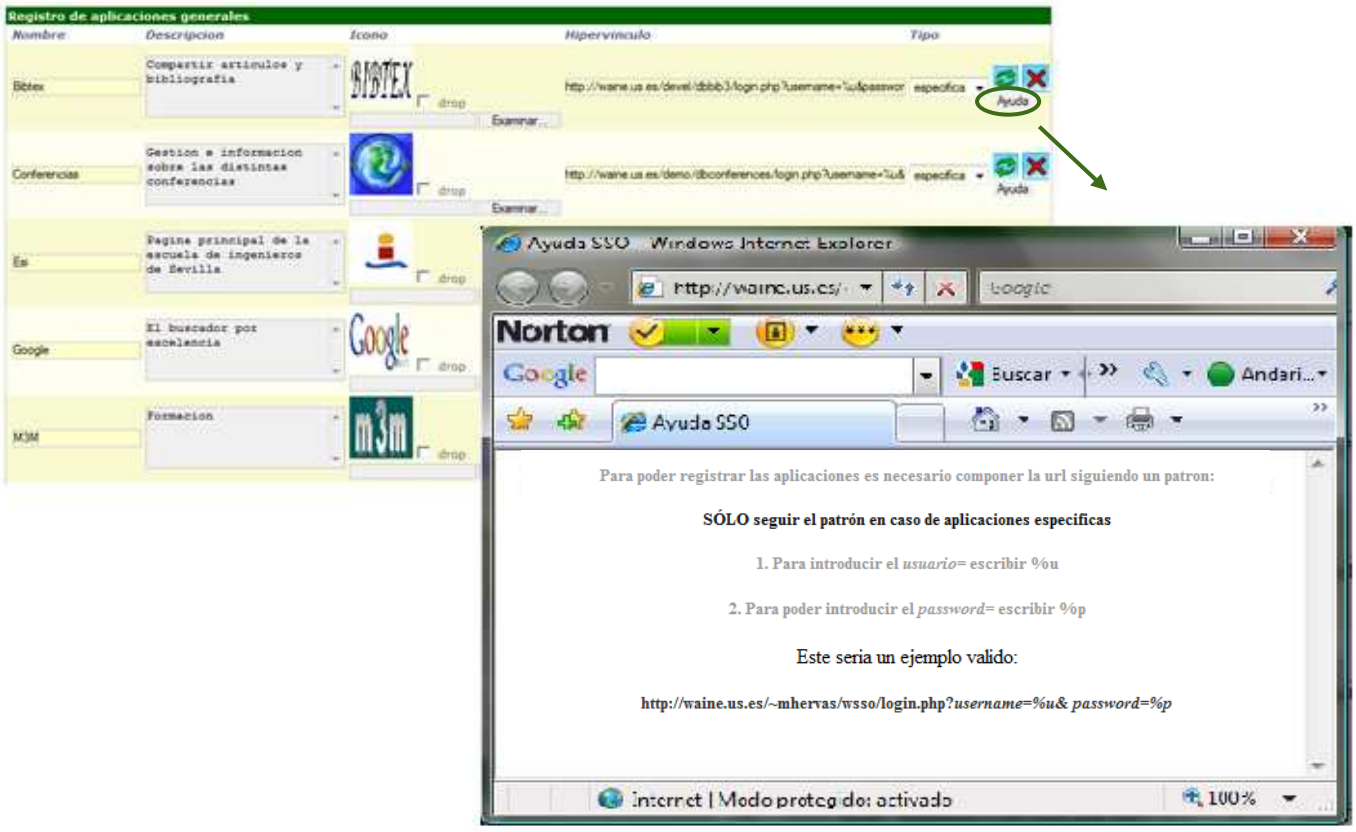

Imagen 9 - Botón con acción

2.2.6 EVENTOS

Por último comentar los eventos que pueden ir asociados a un formulario.

Al igual que los botones, se definen después del formulario entre las etiquetas <events>. Seguidamente hay que especificar cuando se producirá el evento, a elegir entre:

- onload.
- onunload.
- beforeinsert.
- afterinsert.
- beforeupdate.
- afterupdate.
- beforedelete.
- afterdelete.

Después de elegir el momento, hay que indicar la acción que se realizará seguida del código o cuerpo de la misma y parámetros si fueran necesarios.

En el siguiente ejemplo se ilustra lo anteriormente indicado. En este caso se define un evento en el formulario de registro de usuarios para que al eliminar uno de ellos se borre su referencia de las tablas de la base de datos.

```
<events>
        <afterdelete>
                <action type="source">
                        <code>
                        DELETE FROM rel_u_a WHERE uid_SSO_rel=%pk;
                        DELETE FROM gr_us WHERE uid=%pk;
                        </code>
                </action>
        </afterdelete>
</events>
```