Novedades en gvHIDRA

Versión 5.0.0

Tabla de contenidos

Lista de figuras

Apéndice A. Novedades de la versión

En este punto se destaca aspectos más relevantes que incorpora la versión.

1. **Herencia en plantillas, plugins y ficheros de idiomas.** [1]
2. **Mejora de rendimiento en la visualización de resultados.** [2]
3. **Nuevos customs.** [2]
4. **Menú de la aplicación.** [3]
5. **Configuración aspectos generales de las ventanas.** [5]
   a. Icono "modificado" en barra superior [5]
   b. Ubicación botonera: Nueva búsqueda, listado, edición. [6]
   c. Redimiensionar paneles [7]
   d. Conteo de registros en las solapas de los detalles [8]
6. **Configuración nuevas funcionalidades en paneles tabulares.** [9]
   a. Opciones de selección de registros en un tabular [9]
   b. Selección directamente en la fila. [10]
   c. Mostrar número total de registros en un tabular [11]
   d. Parámetro *column_align* para campos en un panel tabular. [12]
7. **Nuevas operaciones y parámetros en plugins.** [12]
   a. cwinfocontenedor. Operaciones getValue() y setValue() [12]
   b. cwinformation. Operaciones getValue() y setValue(). Parámetros nuevos. [13]
   c. cwbotontooltip con funcionamiento independiente del estado del panel. [14]
   d. Visibilidad de etiquetas que acompañan componentes. [14]
   e. Parámetro confirm en botones tooltip. [14]
8. **Debug en desarrollo.** [15]
   a. Debug de plantillas en tiempo de ejecución. [15]
   b. Debug de JavaScript. [15]

A.1. Herencia en plantillas, plugins y ficheros de idiomas.

Con esta herencia se abre la posibilidad de poder particularizar ciertos comportamientos para una aplicación en concreto sin tener que depender del framework.

Se han añadido tres niveles de herencia a la hora de trabajar en GVHIDRA con plantillas, plugins y ficheros de idiomas de Smarty. De este modo, se permite tanto en **aplicación como en custom personalizar** los distintos elementos Smarty disponibles en GVHIDRA sin tener que depender de cambios en el framework, permitiendo enriquecer el comportamiento de las aplicaciones. La precedencia de la **herencia es App > Custom > Framework**, esto es,

cualquier elemento Smarty (plantilla, plugin, fichero idioma) se buscará primero en la aplicación, si no está disponible, se buscará entonces en Custom, y si tampoco está disponible se buscará entonces en el código del framework GVHIDRA.

Las ubicaciones de los distintos elementos Smarty son:

- Ficheros idiomas --> **/lang/**
- Plantillas tpl --> **/plantillas/**
- Plugins --> **/smarty/plugins/**

Por tanto, si dentro de la aplicación se creara un fichero `function.cwlista.php` dentro de la carpeta `app/smarty/plugins/`, el comportamiento del plugin `cwlista` pasaría a ser el definido por dicho fichero, ignorando el comportamiento estándar GVHIDRA que se venía usando desde `app/igep/smarty/gvhplugins/function.cwlista.php`, permitiendo así modificar/sobreescribir comportamientos sin depender del núcleo GVHIDRA, y también añadir nuevas funcionalidades mediante nuevos plugins, etc. De forma análoga se podría sobreescribir o extender las plantillas (.tpl) usadas por los plugins, y lo mismo con los ficheros de idiomas (.conf).

A.2. Mejora de rendimiento en la visualización de resultados.

En esta versión cuando se realiza una consulta, y ésta devuelve más de un registro, no se "dibujan" todas las capas (...) correspondientes a cada página como se hacía antes. Hasta ahora, cuando se paginaba lo que se hacía era hacer visible la capa de la página correspondiente, ocultando la anterior. Crear tal cantidad de capas, cuando la consulta devolvía un número alto de registros, podía resultar pesado.

En esta versión se ha cambiado la comunicación entre presentación y negocio utilizando ahora llamadas Ajax, y con ello se crea una estructura JSON con toda la información del resultado de la consulta. Por lo tanto, respecto al código HTML ahora sólo se crea una página (capa (...), teniendo toda la información del resultado de la consulta en el objeto JSON, la paginación se realiza obteniendo los valores correspondientes del objeto JSON y asignándolos a cada uno de los campos de la página dibujada.

Por lo tanto el peso de la página que se obtiene para cada pantalla ha disminuido de forma contundente, haciendo que el tiempo de acceso a cada pantalla en caso de una gran cantidad de registros ha disminuido.

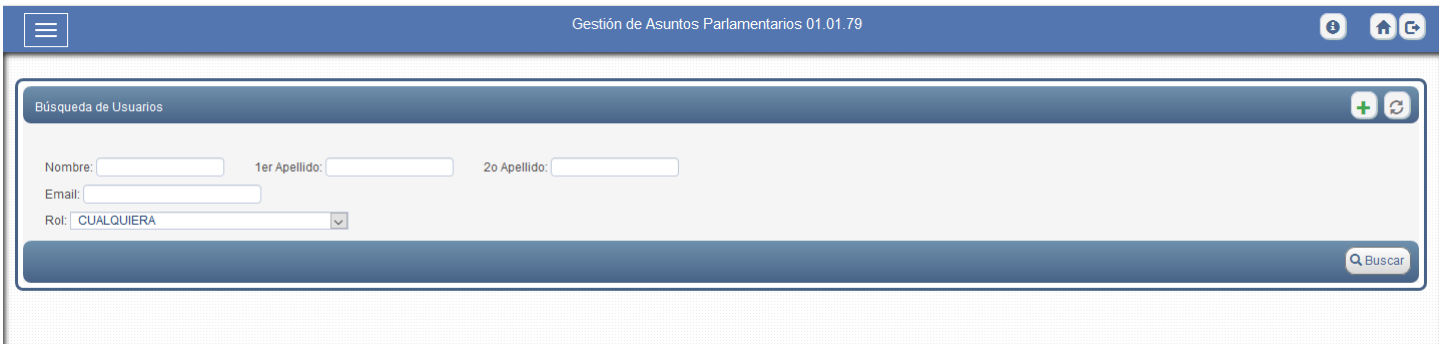
A.3. Nuevos customs.

En el paquete que se distribuye del framework de esta versión se incluyen varios customs para poder utilizar al desarrollar aplicaciones.

- Custom **greyStyle**

  Es el estilo clásico de gvHidra, el que se venía distribuyendo hasta ahora.

Figura A.1. Custom greyStyle



- Custom **lightStyle**

  Se ha creado una versión del custom con colores más claros, elementos más grandes y espaciados entre sí. Se incluye una versión compact de este estilo: **cpLightStyle**

Figura A.2. Custom lightStyle

- Custom **darkStyle**

  El darkStyle es una versión, como su propio nombre indica, más oscura del custom con tonalidades grises y negro, elementos más grandes y espaciados entre sí. Se incluye una versión compact de este estilo: **cpDarkStyle**

Figura A.3. Custom darkStyle

A.4. Menú de la aplicación.

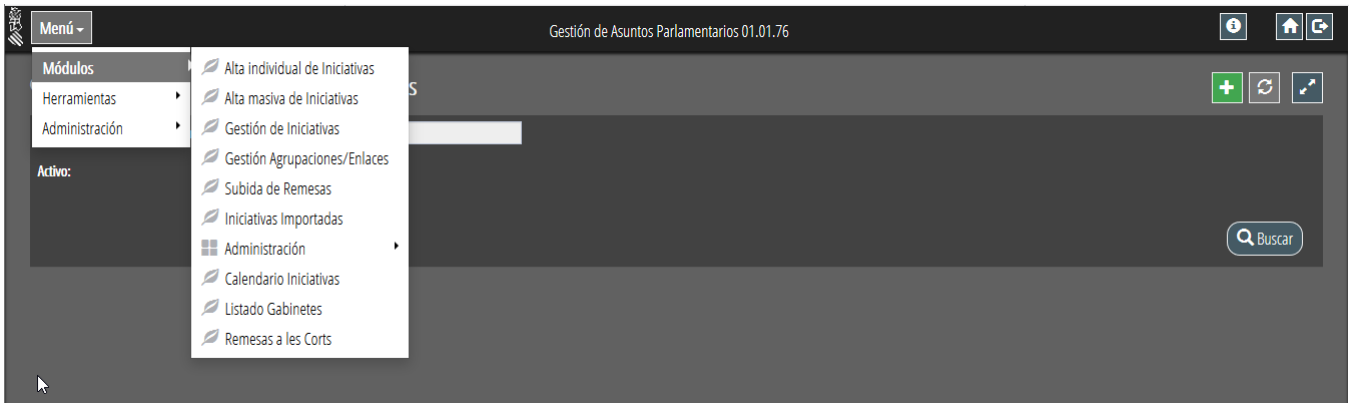
Un aspecto destacable de esta versión es la posibilidad de elegir la posición del menú. El plugin **`cwventana`** tiene el parámetro ***`layout`*** que puede contener una serie de valores (selectores css) que determinarán cierta visualización de algún elemento. Estos valores son los que permitirán elegir las características y posición del menú.

- **Menú en la barra superior de la ventana.**

  Es la posición clásica de gvHIDRA, por lo tanto, es la opción por defecto del framework, lo que implica que no es obligatorio ponerlo. En este caso el parámetro ***`layout`*** podría contener el valor **`classic-menu`**.

```
{cwventana layout="classic-menu" titulo="APL"
tipoAviso=$smty_tipoAviso codAviso=$smty_codError descBreve=$smty_descBreve
 textoAviso=$smty_textoAviso onLoad=$smty_jsOnLoad}
```

Figura A.4. Menú barra superior (custom darkStyle).



- **Menú en una barra lateral a la izquierda.** El menú se ubicará en una barra lateral en la parte de la izquierda de la pantalla. Inicialmente, al entrar a una pantalla con este menú aparecerá escondido, para visualizarlo se tendrá que hacer click en el botón de despliegue del menú.

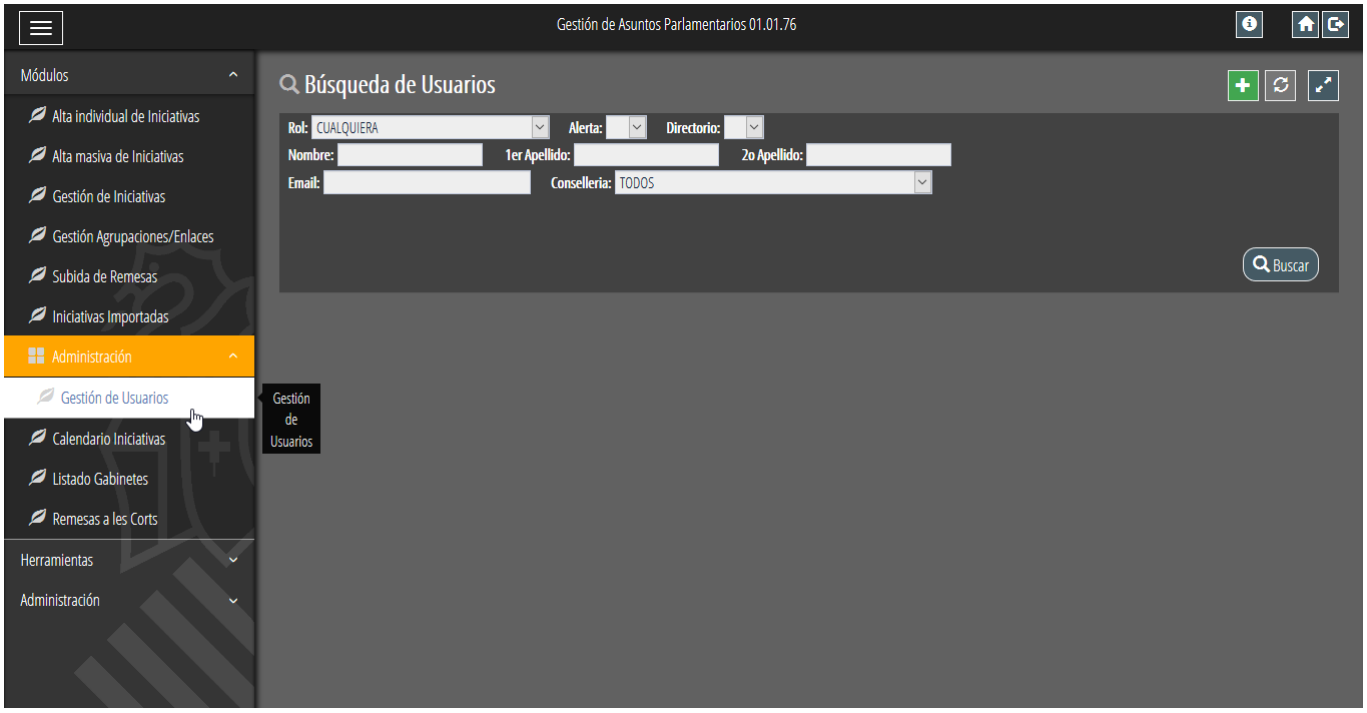
Para este funcionamiento, el parámetro **layout** deberá contener el valor **vertical-menu**.

```
{cwventana layout="vertical-menu" titulo="APL"
tipoAviso=$smty_tipoAviso  codAviso=$smty_codError  descBreve =
 $smty_descBreve  textoAviso=$smty_textoAviso onLoad=$smty_jsOnLoad}
```

En el caso de que interese mantener el menú de forma fija en el lado izquierdo, que no se esconda cada vez que se pulse en una opción, al parámetro **layout** se le debe añadir el valor **vertical-menu-visible**.

```
{cwventana layout="vertical-menu vertical-menu-visible" titulo="APL"
tipoAviso=$smty_tipoAviso  codAviso=$smty_codError  descBreve =
 $smty_descBreve  textoAviso=$smty_textoAviso onLoad=$smty_jsOnLoad}
```

Figura A.5. Menú barra lateral (custom darkStyle).



A.5. Configuración aspectos generales de las ventanas.

A.5.1. Icono "modificado" en barra superior.

Tanto en un panel tabular como en una ficha, cuando se está modificando algún dato o insertando un registro nuevo aparece en la barra inferior un icono que indica que el panel contiene modificaciones. En el caso de que el panel sea más extenso que el tamaño del monitor, y por lo tanto no se vea la barra inferior, existe la posibilidad de mostrar el icono en la barra superior del panel. Esta opción se activará añadiendo al parámetro ***`layout`*** del plugin *`cwbarrasuppanel`* el siguiente selector de css **`show-modified`**

```
{cwbarrasuppanel layout="show-modified" titulo="Panel listado"}
```

Figura A.6. Panel modificado, icono barra superior.

A.5.2. Ubicación botonera: Nueva búsqueda, listado, edición.

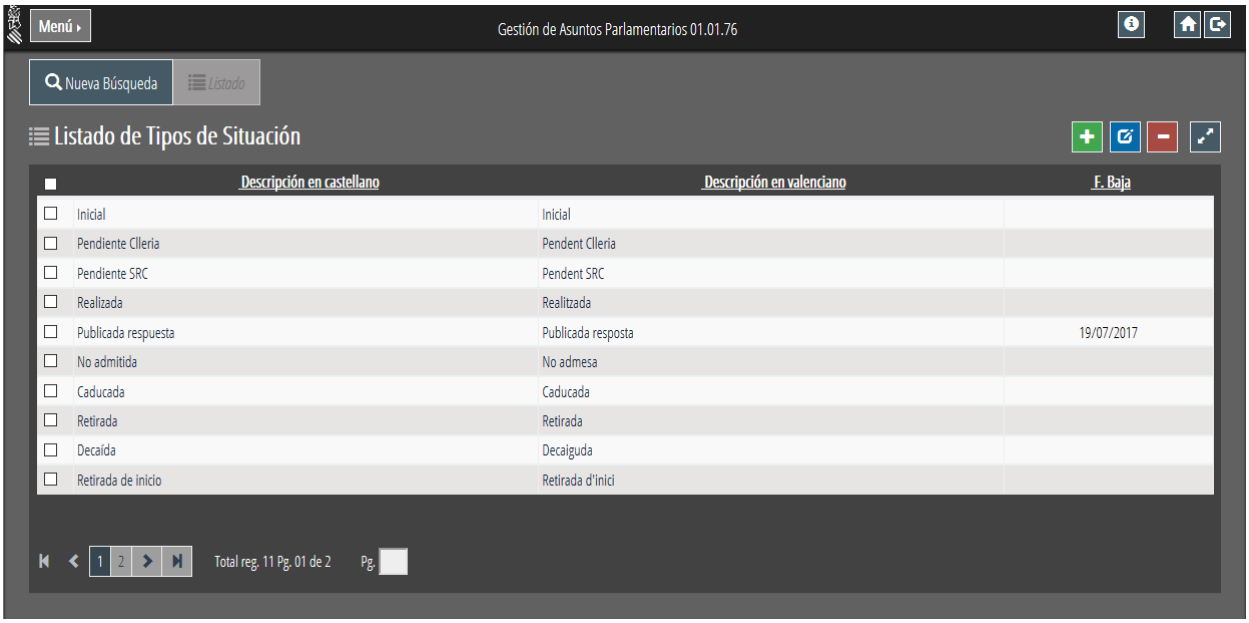
La botonera que aparece justo debajo de la barra superior de la ventana (Nueva búsqueda, listado, edición) puede visualizarse en formato maximizado y minimizado.

- **Formato maximizado botonera.**

  Es la visualización clásica de gvHIDRA, por lo tanto, es la opción por defecto del framework. En este caso el parámetro **layout** deberá contener el valor **botonera-classic**

```
{cwventana layout="botonera-classic" titulo="APL"
 tipoAviso=$smty_tipoAviso codAviso=$smty_codError descBreve=
$smty_descBreve textoAviso=$smty_textoAviso onLoad=$smty_jsOnLoad}
```

Figura A.7. Formato maximizado botonera (custom darkStyle).

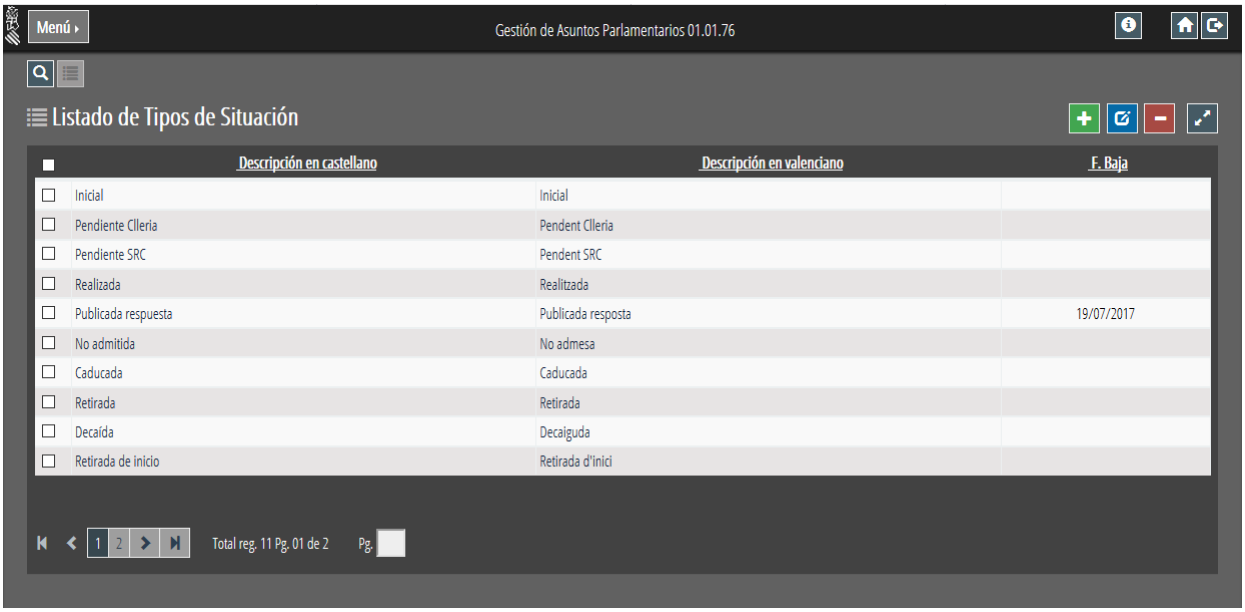


- **Formato minimizado botonera.**

  En este caso la botonera aparecerá minimizada, es decir, los botones solamente contendrán los iconos que acompañan al texto en la versión anterior. Para conseguirlo, el parámetro **layout** deberá contener el valor **botonera-tabs**

```
{cwventana layout="botonera-tabs" titulo="APL"
 tipoAviso=$smty_tipoAviso codAviso=$smty_codError descBreve=
$smty_descBreve textoAviso=$smty_textoAviso onLoad=$smty_jsOnLoad}
```

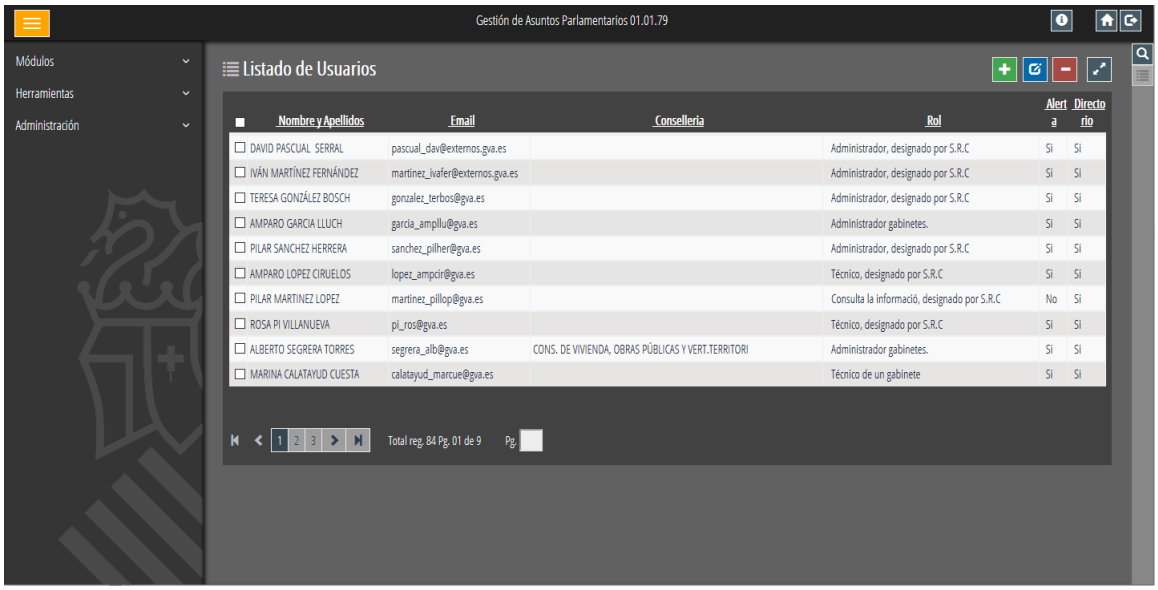
Figura A.8. Formato minimizado botonera (custom darkStyle).



Si se combina la botonera minimizada con el menú vertical, la botonera se colocará en el lado derecho de la pantalla de forma vertical.

```
{cwventana layout="vertical-menu botonera-tabs" titulo="APL"
 tipoAviso=$smty_tipoAviso codAviso=$smty_codError descBreve=
$smty_descBreve textoAviso=$smty_textoAviso onLoad=$smty_jsOnLoad}
```

Figura A.9. Formato minimizado botonera con menú vertical (custom darkStyle).



A.5.3. Redimensionar paneles.

En la barra superior de los paneles aparecerá, por defecto, un botón para maximizar/minimizar el panel.

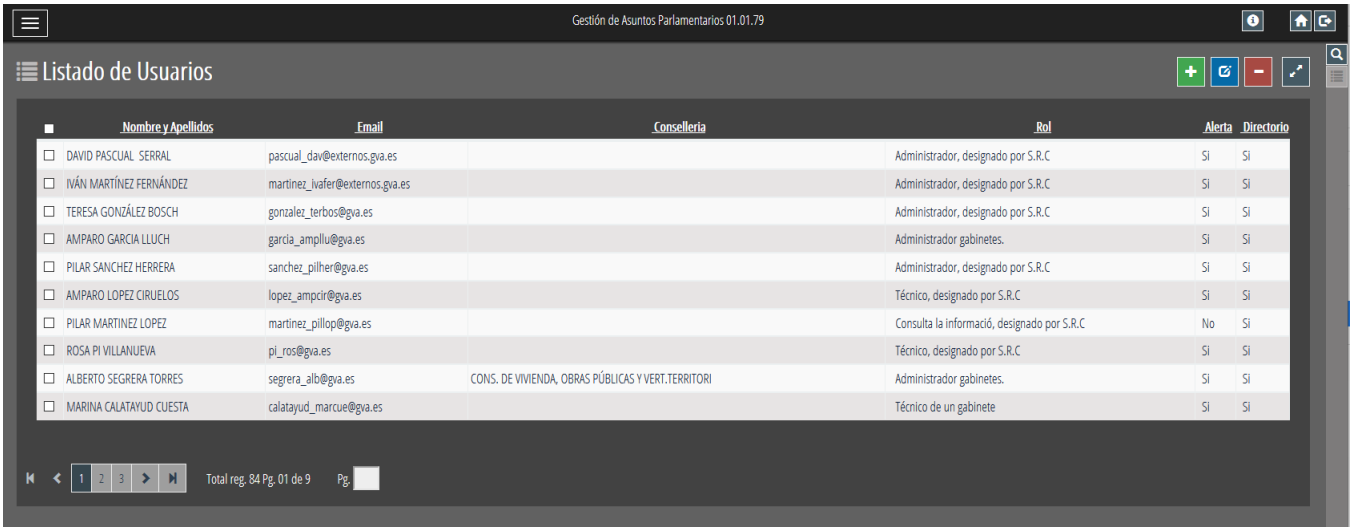
Figura A.10. Botón redimensionar.

Si no se quiere que aparezca este botón en la barra superior únicamente se tendrá que añadir al parámetro ***layout*** el valor **`no-maximize`**.

```
{cwbarrasuppanel layout="no-maximize" titulo="Panel listado"}
```

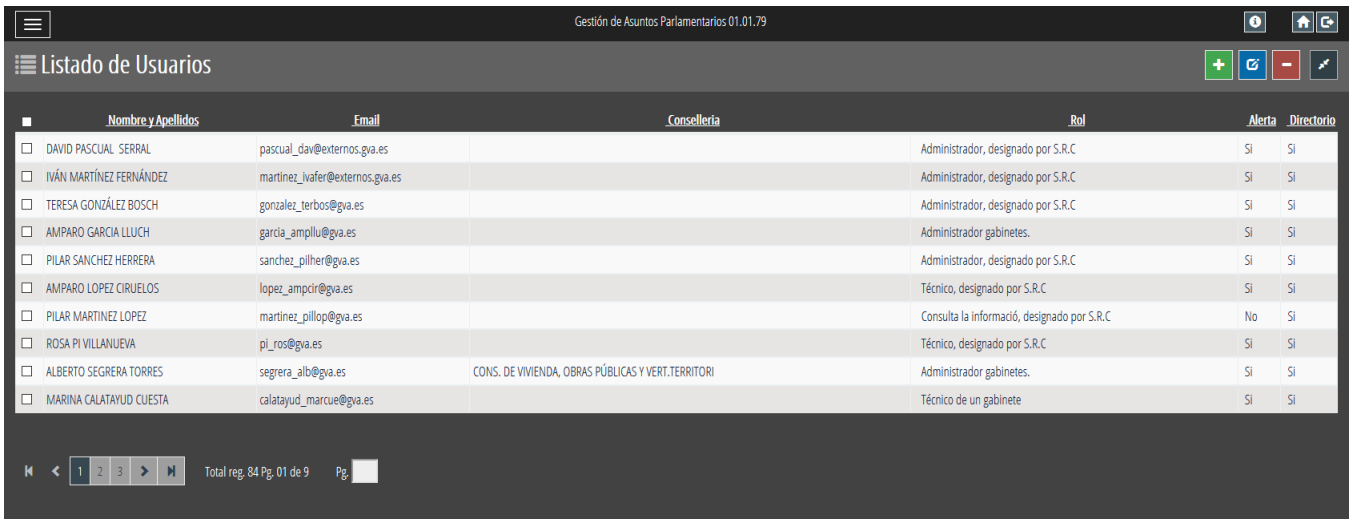
- **Panel minimizado**, estado por defecto.

Figura A.11. Panel minimizado (estado por defecto).



- **Panel maximizado.**

Figura A.12. Panel maximizado.



A.5.4. Conteo de registros en las solapas de los detalles.

Las pestañas de los detalles pueden ahora mostrar el n.º de tuplas que devolvería la consulta al pinchar en cada pestaña. Para mostrar el conteo en las pestañas, se ha añadido un parámetro booleano al método ‘addSlave’ que se invoca en la clase manejadora que actúa como maestra, cuyo valor por defecto es false y que puede cambiarse a true para activar el conteo de tuplas en dicha relación:

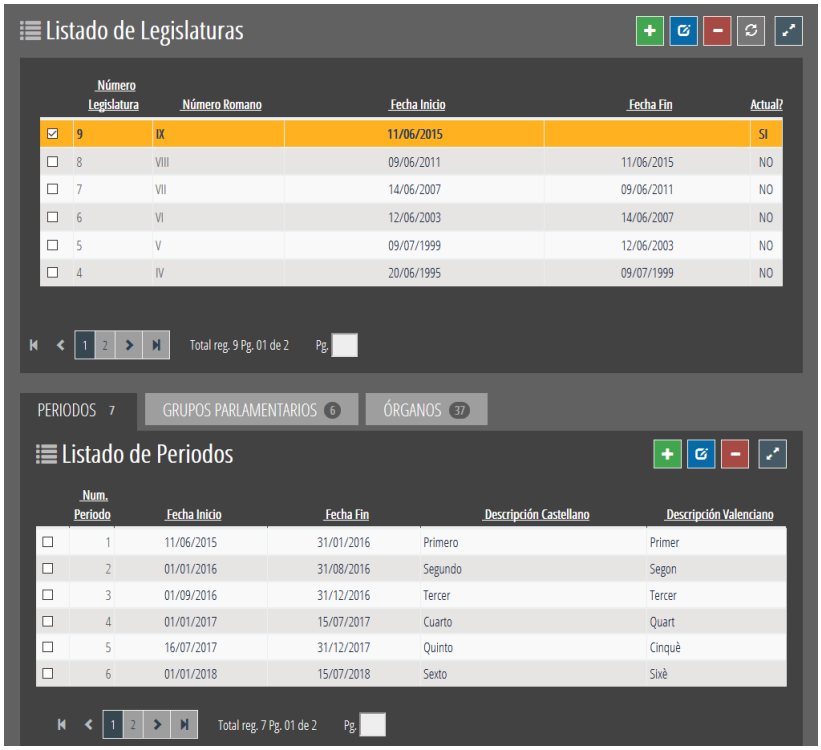
```
public function addSlave( $nombreClaseManejadora, $listasCamposPadre,
 $listasCamposHijo, $contarTotalTuplas=false, $agruparConteo=true )
```

- **$contarTotalTuplas**: Campo booleano opcional con el que se indicará si se quiere el conteo o no en las pestañas del detalle . Por defecto es false.

  Para el conteo se construirá una consulta que por defecto utilizará las searchQueries() definidas en las distintas clases detalle/hija (definidas en el constructor de cada clase manejadora con los métodos 'setSelectForSearchQuery' y 'setWhereForSearchQuery'). Si quisiera optimizarse la consulta de conteo en cualquiera de los detalles, se puede utilizar un nuevo método 'setSelectForDetailCount', de funcionamiento similar a 'setSelectForSearchQuery' pero que se usará solo para el conteo de tuplas al actuar como detalle/hija. Si no se especifica nada en 'setSelectForDetailCount', se usará los especificado en 'setSelectForSearchQuery'. En cuanto a las condiciones de filtro de la búsqueda (aka. sección sql WHERE), se utilizarán las mismas que para la consulta de búsqueda normal, es decir, se usarán las especificadas en 'setWhereForSearchQuery' y en los filtros. De ese modo, garantizaremos que se filtran siempre las mismas tuplas en el conteo que en la búsqueda.

- **$agruparConteo**: Campo booleano opcional que agrupa la consulta de conteo de tuplas junto con la consulta de otros detalles. Por defecto es true.

Figura A.13. Maestro/nDetalles con conteo de registro en cada detalle.



A.6. Configuración nuevas funcionalidades en paneles tabulares.

A.6.1. Opciones de selección de registros en un tabular.

Cuando nos encontramos en un panel tabular, hasta ahora, se podía trabajar con más de un registro a la vez (parámetro *conCheckTodos=true* del plugin `cwtabla`) o en exclusividad con uno solo (parámetro *seleccionUnica=true* del plugin `cwtabla`).

En esta versión se amplian las posibilidades de selección de registros a los siguientes conjuntos: "Todos", "Ninguno", "Página actual", "Invertir selección". Para activar esta posibilidad en un panel tabular el valor del parámetro *con-*

CheckTodos debe tener el valor `multiSelect`. De este modo, junto al checkbox de selección de todas las tuplas de la tabla (checkAll) aparecerá un desplegable con opciones de selección múltiple sobre las tuplas

```
{cwtabla conCheckTodos="multiSelect" ... }
```

Figura A.14. Panel tabular con opciones de selección de registros.

A.6.2. Selección directamente en la fila.

Tradicionalmente, en gvHIDRA, en un panel tabular para seleccionar un registro sobre el que trabajar (editar, eliminar) se selecciona dicho registro marcando el checkbox de la fila y pulsando el botón de la operación a realizar. En esta versión se ha incorporado la funcionalidad de poder realizar estos dos pasos en uno solo, haciendo click o doble click en cualquier parte de la fila. Para ello existen dos nuevos parámetros para el plugin **`cwfila`** que son *`onRowClick`* y *`onRowDblClick`*, respectivamente.

El valor de estos parámetros será el *`id`* del botón que queramos que se ejecute al hacer click sobre la fila.

- **Un solo click.**

  Como se deduce de su nombre, con un solo click en la fila se realizará la acción que se haya indicado en el parámetro *`onRowClick`*.

```
...{cwbotontooltip id="btn_Modificar" iconCSS="glyphicon glyphicon-
edit" titulo=#gvhlang_modificar# funcion="modificar" accion="modificar"
 actuaSobre="ficha" action="editar"}
{cwtabla ...}
  {cwfila tipoListado="false" onRowClick="btn_Modificar"}
 {* ... *}
  {/cwfila}
{/cwtabla}
```

- **Doble click.**

  Como se deduce de su nombre, con un doble click en la fila se realizará la acción que se haya indicado en el parámetro *`onRowDblClick`*.

```
...{cwbotontooltip id="btn_Modificar" iconCSS="glyphicon glyphicon-
edit" titulo=#gvhlang_modificar# funcion="modificar" accion="modificar"
 actuaSobre="ficha" action="editar"}
{cwtabla ...}
  {cwfila tipoListado="false" onRowDblClick="btn_Modificar"}
 {* ... *}
```

```
  {/cwfila}
{/cwtabla}
```

Figura A.15. Panel tabular con selección en fila activado.

A.6.3. Mostrar número total de registros en un tabular.

En determinadas ocasiones puede interesar conocer el número total de registros aunque no se visualicen todos en una sola página. Para ello existe un nuevo parámetro para los plugins `cwtabla` y para `cwpaginador` llamado ***`conTotalRegistros`*** al que se le pueden asignar los siguientes valores:

- **always**: Mostrará siempre el número total de registros en el bloque al que afecte.
- **never**: No mostrará nunca el número total de registros en el bloque al que afecte.
- **only-multipage**: Sólo mostrará el número total cuando haya más de una página de datos.
- **only-singlepage**: Sólo mostrará el número total cuando haya una única página de datos.

El valor por defecto de *`conTotalRegistros`* para `cwtabla` es `never`, mientras que para `cwpaginador` es `only-multipage`

Figura A.16. Panel tabular con total de registros en cwtabla.

```
{cwtabla conCheck="true" conCheckTodos="true" numFilasPantalla="6" datos=
$smty_datosTabla conTotalRegistros="always"}
  {cwfila tipoListado="false"}
    {* ... *}
  {/cwfila}
  {cwpaginador iconCSS="true" conTotalRegistros="never"}
{/cwtabla}
```

Figura A.17. Panel tabular con total de registros en cwpaginador.

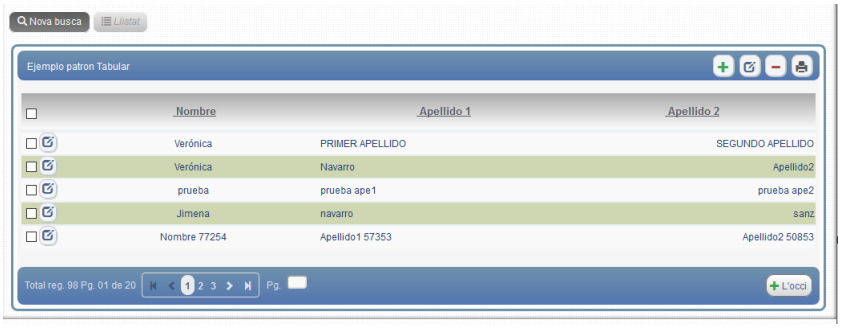
```
{cwtabla conCheck="true" conCheckTodos="true" numFilasPantalla="6" datos=
$smty_datosTabla}
  {cwfila tipoListado="false"}
    {* ... *}
  {/cwfila}
  {cwpaginador iconCSS="true" conTotalRegistros="only-multipage"}
{/cwtabla}
```

A.6.4. Alineación de campos en un panel tabular.

En esta versión se incluyen unos estilos para que podamos alinear los campos por columnas en un panel tabular. Para utilizarlo se ha de incluir el estilo en el parámetro ***class*** con alguno de los siguientes valores: **['column-align-left', 'column-align-right', 'column-align-center']**

Figura A.18. Alineación campos en panel tabular.

```
{cwcampotexto nombre="nombre" class="column-align-left" editable="true"
 size="10" label="Nombre" dataType=$dataType_Tabular.nombre}
{cwcampotexto nombre="ape1" class="column-align-center" editable="true"
 size="10" label="Apellido 1" dataType=$dataType_Tabular.ape1}
{cwcampotexto nombre="ape2" class="column-align-right" editable="true"
 size="10" label="Apellido 2" dataType=$dataType_Tabular.ape2}
```



A.7. Nuevas operaciones y parámetros en plugins.

A.7.1. cwinfocontenedor. Operaciones getValue() y setValue().

Este plugin pasa de ser meramente informativo sobre el que no se podía actuar a poder tener un contenido dinámico. Desde la clase manejadora se podrá acceder a su contenido con los métodos **`getValue()`** y **`setValue()`**. El

único requisito es añadirle como segundo parámetro **"external"**, esto es porque se trata como un campo externo, no del conjunto de datos resultado de la consulta.

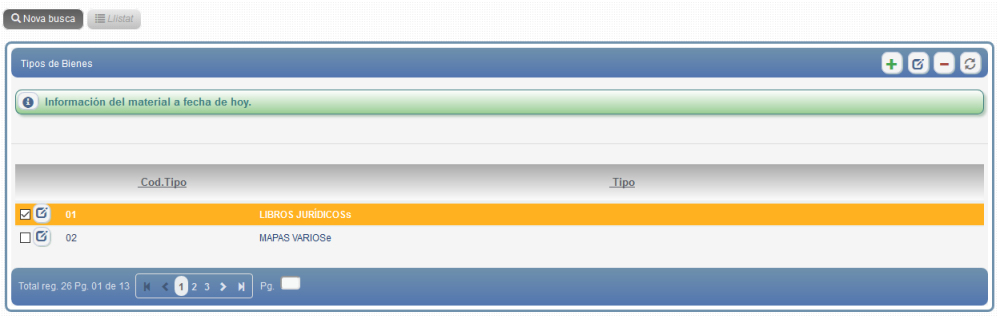
```
$contenedor = $objDatos->getValue ('infoContenedor1', 'external');

$objDatos->setValue (
  'infoContenedor2' ,
  'Se muesta la información correspondiente al día de hoy'
);
```

Por otra parte se añade el parámetro ***"class"*** para poder personalizar con css el plugin.

Figura A.19. Plugin cwinfocontenedor.

```
{cwinfocontenedor id="infoFicha" class="information" value="Información del
 material a fecha de hoy."}{/cwinfocontenedor}
```



A.7.2. cwinformation. Operaciones getValue() y setValue(). Parámetros nuevos.

Este plugin pasa de ser meramente informativo sobre el que no se podía actuar a poder tener un contenido dinámico. Desde la clase manejadora se podrá acceder a su contenido con los métodos **`getValue()`** y **`setValue()`**, en este caso el valor sí puede pertenecer al conjunto de datos del resultado de la consulta.

El plugin también ofrece más posibilidades de diseño asociadas a los siguientes parámetros que se le han añadido:

- **posicion**: Establecerá la posición de la capa, donde se mostrará el contenido de "value", respecto al botón tooltip.

  Los valores son: **["top", "right", "bottom", "left"]**

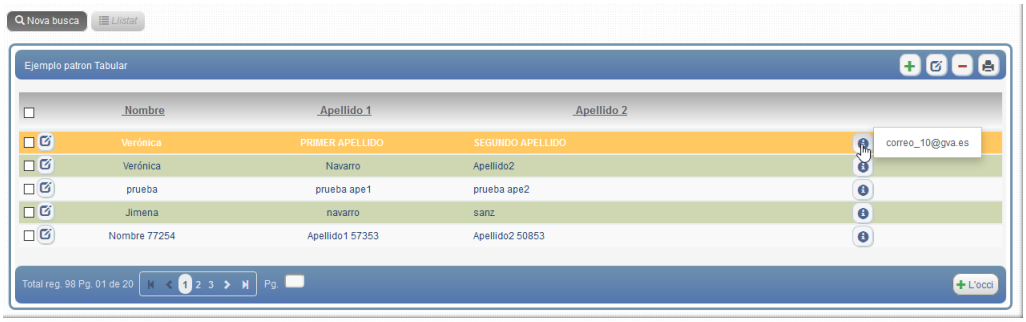
- **size**: Valor para poder asignar tamaño a la columna en el caso de estar en un panel tabular.

- **trigger**: Momento en el que se mostrará el texto informativo.

  Los valores son: **["focus", "hover"]**

Figura A.20. Plugin cwinformation.

```
{cwinformation iconCSS="glyphicon glyphicon-info-sign" posicion="right"
 nombre="mail" value=$defaultData_Tabular.mail
 dataType=$dataType_Tabular.mail}
```



A.7.3. cwbotontooltip con funcionamiento independiente del estado del panel.

Para permitir más versatilidad al plugin cwbotontooltip, cuando éste se encuentra dentro de un panel tabular o ficha, se le añade el parámetro ***dependPanel***, este parámetro permitirá definir si el funcionamiento del botón es independiente o no del estado del panel. Es decir, si estará o no activo a pesar de que el panel no esté en modo de modificación o de inserción. Si *dependPanel* tiene el valor **"false"**, el botón estará **activo siempre**, es decir no dependerá de que el panel esté en modo inserción o edición.

Este parámetro también se añade para el plugin **cwcampotexto** pero solamente tendrá utilidad cuando el campo sea de tipo fecha, ya que el funcionamiento de si está activo o no recaerá sobre el botón tooltip del calendario que aparecerá al lado del campo.

```
{cwbotontooltip id="verInformation" dependPanel="false" iconCSS="glyphicon
 glyphicon-refresh" accion="particular" action="verInformation"
 actuaSobre="ficha" titulo="ver información"}
```

A.7.4. Visibilidad de etiquetas que acompañan componentes.

No siempre queremos que se muestre la etiqueta o texto asociado a un campo, aunque puede interesarnos que dicho texto exista en realidad para utilizarlo en mensajes con el usuario. La solución en utilizar el parámetro y fijar su valor a false, así, evitaremos que se muestre el texto asociado.

Para ello se ha añadido un parámetro **"showLabel"** a todos los componentes que pueden aparecer dentro de un panel (cwareatexto, cwcampotexto, cwbotontooltip...), con él se podrá indicar si se quiere o no mostrar la etiqueta asociada al componente.

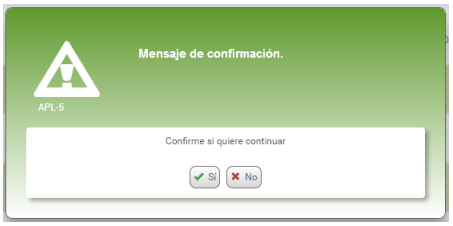
```
{cwcampotexto nombre="ape1" editable="true" size="10" label="Apellido
 1" showLabel="false" dataType=$dataType_Tabular.ape1}
```

A.7.5. Parámetro confirm en botones tooltip.

El parámetro **"confirm"**hasta ahora solo se podía utilizar en el plugin cwboton, pero a partir de esta versión se ha incluido en el plugin **cwbotontooltip**. A este parámetro se le debe pasar el código del mensaje que se quiere mostrar (p.ej "APL-5"), por lo tanto al pulsarlo saltará la ventana con el mensaje, será un mensaje de solicitud de confirmación para ejecutar o no la acción del panel.

Figura A.21. Mensaje de confirmación.

```
{cwbotontooltip iconCSS="glyphicon glyphicon-plus" texto="Particular"
 class="button" accion="particular" action="MsgConfirm" confirm="APL-5"}
```



A.8. Debug en desarrollo.

A.8.1. Debug de plantillas en tiempo de ejecución en entorno de desarrollo.

Debug en tiempo de ejecución de plantillas Smarty en entorno local y de desarrollo (NO EN ENTORNO DE PRODUCCIÓN): Para poder ver, en tiempo de ejecución, la ventana de debug de un formulario Smarty basta con añadir el parámetro 'SMARTY_DEBUG' a la ruta URL en la barra del navegador.

Por ejemplo, si la url de la aplicación fuera...

```
http://localhost/app/index.php?view=views/Vista.php
```

para mostrar la ventana de debug habría que añadirle '&SMARTY_DEBUG' :

```
http://localhost/app/index.php?view=views/Vista.php&SMARTY_DEBUG
```

Debido a que se muestran detalles internos de la aplicación, por seguridad este modo de depuración solo está disponible en los entornos local y de desarrollo, y por tanto NO ESTÁ DISPONIBLE EN ENTORNO DE PRODUCCIÓN. Esto es, solo está activado en los casos en que, en el fichero de configuración de la aplicación ('gvHidraConfig.inc.xml' o similar), la propiedad 'p_entorno' tenga el valor 'LOCALHOST' o 'DESARROLLO':

```
<property id="p_entorno">LOCALHOST</property>
 <property id="p_entorno">DESARROLLO</property>
```

A.8.2. Debug de JavaScript.

Si el navegador soporta el debugger se puede ultilizar *console.log()* para mostrar valores de JavaScript en la ventana de debugger. Cuando estamos desarrollando nos puede ser útil mostrar mensajes de este tipo. En esta versión para depurar mientras se desarrolla se pueden activar los mensajes de consola que por defecto tiene el framework con la directiva **logJSSettings** en el fichero de configuración de la aplicación **externo.gvHidraConfig.inc.xml**.

Activar el log:

```
<logJSSettings status='LOG_ALL'>
```

Desactivar el log:

```
<logJSSettings status='LOG_NONE'>
```

Si se quiere hacer uso en la consola JavaScript en el JavaScript que se incluya en alguna plantilla, es aconsejable utilizar la función gvh.showConsoleMsg(type,msg). Ya que al hacer uso de este método funcionará el activar o no del log con la variable logJSSettings. Esta función necesita de dos parámetros:

- **type**:Será el tipo de mensaje que se mostrará en la consola.

  Los valores posibles son los siguientes:

  - log: Mensajes generales de información.
  - error: Cuando queramos destacar que se ha producido un error, el mensaje en consola aparecerá en rojo.
  - warn: Muestra un mensaje de advertencia.
  - dir: Muestra un listado interactivo de las propiedades de un objeto JavaScript específico.
  - table: Muestra datos tabulares en forma de una tabla.
  - beginGroup: Crea un grupo en el que indenta todos los mensajes que se encuentren hasta que se indique cierre del grupo.
  - endGroup: Finaliza el grupo abierto con "beginGroup".
- **msg**:Mensaje que queremos que se visualice en la consola.

Ejemplos:

Error:

```
gvh.showConsoleMsg('error',' ¡¡Falta la indicación del campo destino de la
 ventana de selección!!');
```

Grupo:

```
gvh.showConsoleMsg('beginGroup',' ** gvh_panel.js - changeField() ');
...
gvh.showConsoleMsg('log', ' FIELD: '+idCampo );
...
gvh.showConsoleMsg('error', '¡¡¡ changeField() - '+campoJSON+'no existe este
 campo en el registro '+regJSON+' !!!' );
...
gvh.showConsoleMsg('endGroup');
```