

OBJETIVOS

Mostrar las capacidades y usos relacionados con la manipulación de la tabla `config_form_fields` con la intención de personalizar los formularios mostrados en QGIS para los elementos de red (node, connec, gully, arc). Se podrán traducir las etiquetas y los tooltips de todos los widgets, así como moverlos o esconderlos en caso que nos interese.

INTRODUCCIÓN

Dado que una manipulación incorrecta puede producir daños importantes en la tabla de configuración, existe una función de backup de cualquier tabla de la base de datos que puede resultar muy útil para este proceso. Su uso consiste en:

CREACIÓN DEL BACKUP:

```
SELECT gw_fct_admin_manage_backup ($${"data":{"action":"CREATE",
"backupName":"test6", "table":"config_form_fields"}}$)
```

RECUPERACIÓN DEL BACKUP:

```
SELECT gw_fct_admin_manage_backup ($${"data":{"action":"RESTORE", "backupName":"test6",
"newBackupName":"test5", "table":"config_form_fields"}}$)
```

BORRADO DEL BACKUP:

```
SELECT gw_fct_admin_manage_backup ($${"data":{"action":"DELETE",
"backupName":"test4", "table":"config_form_fields" }}$)
```

COMENTARIOS:

El proceso de creación almacena toda la información en la tabla `temp_table` con `fprocesscat_id = 111`

Para evitar posibles malas manipulaciones se usa una estrategia de doble factor de seguridad. En este sentido:

- Es obligado poner las dos keys siempre (backupName y table)
- Antes de restaurar el backup, un backup de los datos existentes en la tabla son guardados con el nombre de newBackupName.

DESCRIPCIÓN

Para modificar **config_form_fields** para personalizar los formularios.

Notas a tener presente. La clave primaria de la tabla es:

formname & formtype & tabname & columnname

Las únicas filas que deben manipularse para personalizar los formularios de los elementos son las que en la columna *formname* tienen el prefijo:

```
ve_node_*      ve_arc_*
ve_connec_*ve_gully_*
```

De forma normal deben existir registros para todos los campos de los elementos que encontramos en la tabla *cat_feature* (también deben coincidir con las vistas *child* que tenemos creadas en nuestro esquema).

Entonces, en la tabla *config_form_fields*, tendremos para cada vista *child* un numero de filas que coincide con todos los campos que tenga esta vista, para poder así, configurarlos uno a uno.

De las muchas columnas que hay en *config_form_fields*, se describen todas y cada una de ellas para lograr el grado de configuración que se desee:

[POSICIÓN]

tabname: Para administrar los formularios de los diferentes elementos de la red, se gestionan los widgets en función de la pestaña (tab) donde se encuentren. Como vemos en la imagen, hay diferentes tabs, y cada cual va a tener el nombre que le corresponda: **data**, element, document, etc.

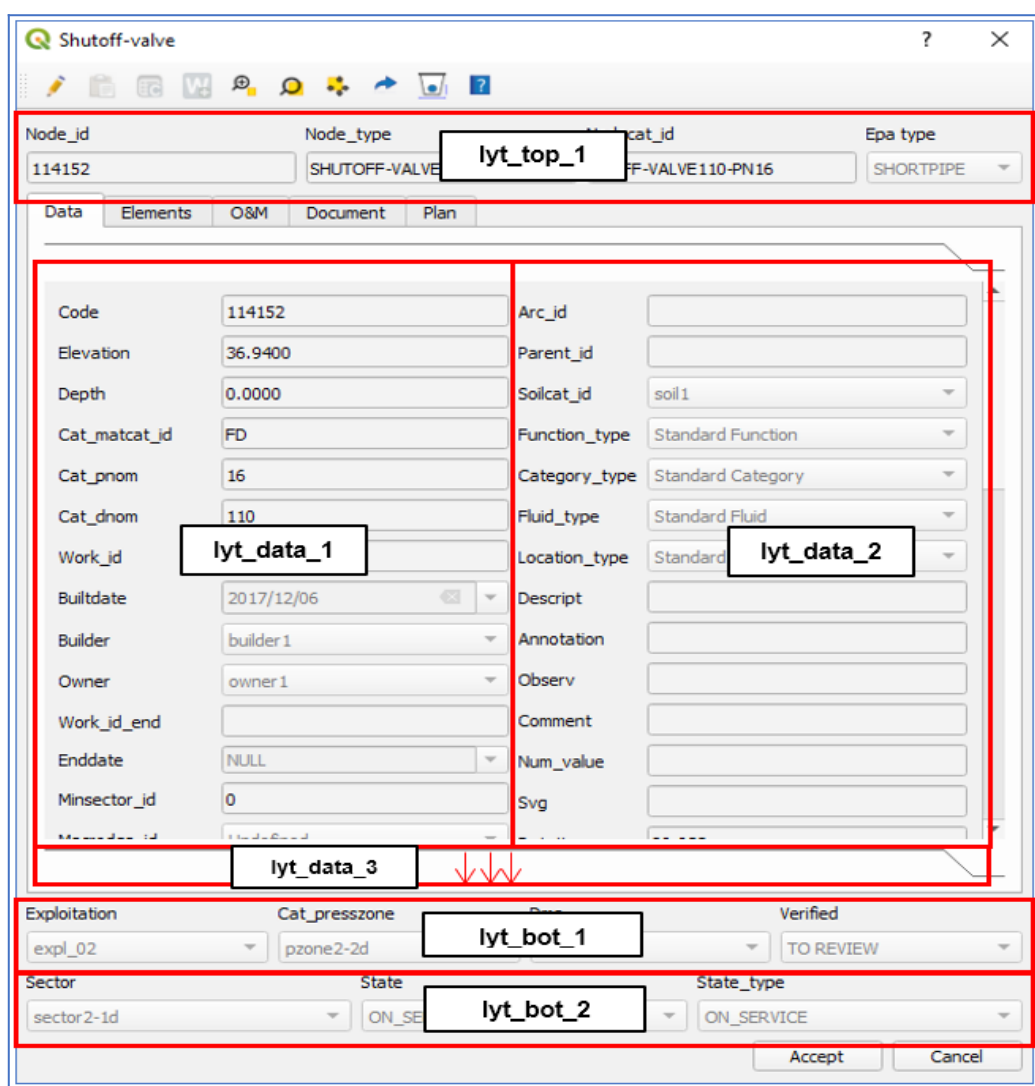
Existen dos tipos de tabs, los diferencia la disposición de los layouts. La mayoría tienen un o dos layouts, pero los de la feature_info tienen todos 3 layouts verticales (excepto el tab data)

Cuando hay valores en config_form_fields que hacen referencia a un formulario sin tabs, el valor para esta columna será: **main**.

layoutname: Para todos los tabs existen tres layouts (1,2,3). El nombre sigue la regla pneumotécnica de **lyt _ tabname _ (1,2,3)**

Adicionalmente tenemos los layouts **lyt_top_1**, **lyt_bot_1** (fila de arriba), **lyt_bot_2** (fila de abajo). Ver imagen.

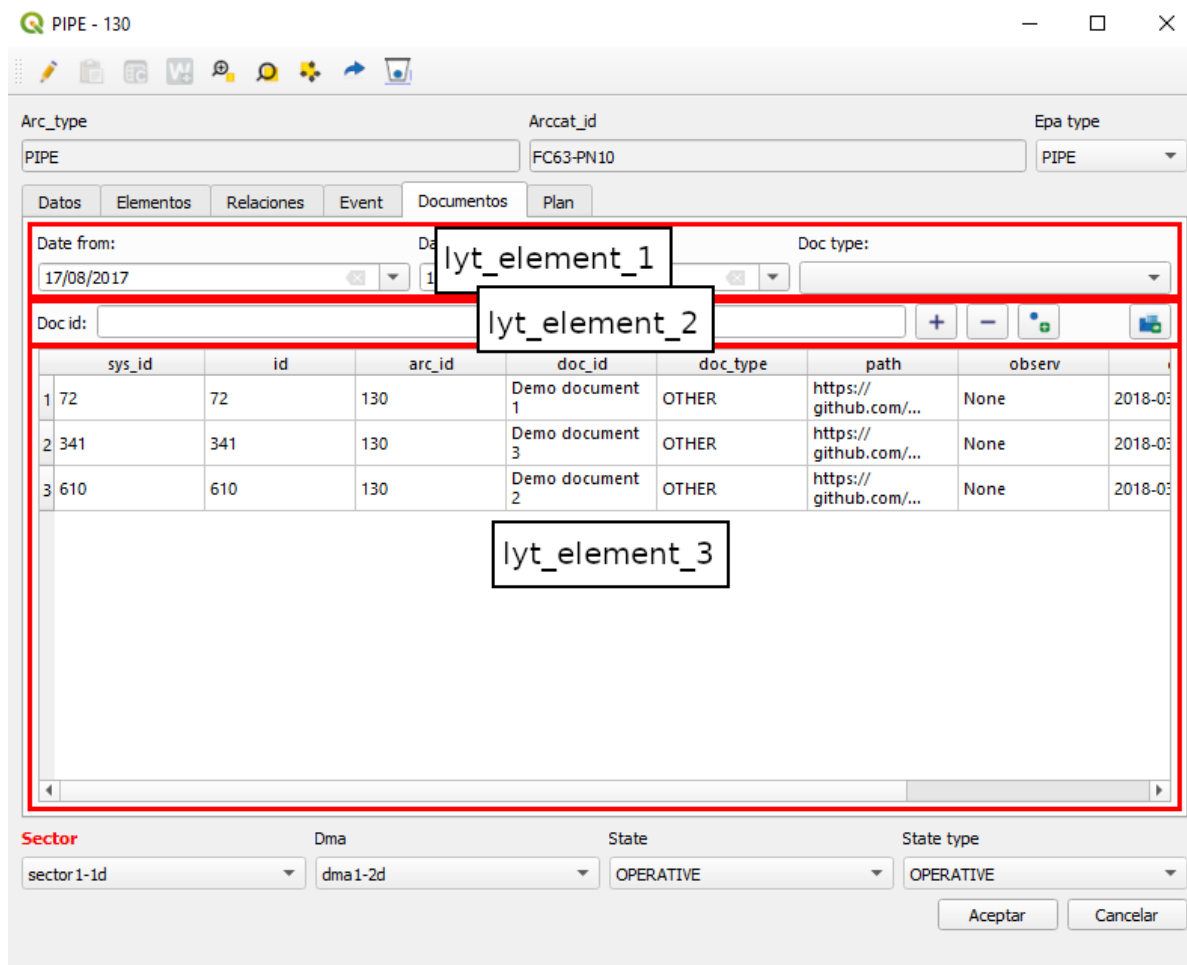
layoutorder: orden del campo dentro de su layout correspondiente. Se irán ordenando de forma ascendente usando el valor que tengan en este campo. Si se encuentran dos campos con el mismo valor de **layoutname** y **layout_order**, evidentemente se solaparan dentro del formulari.



The screenshot shows a web-based form for a 'Shutoff-valve'. The form has a top navigation bar with tabs: Data, Elements, O&M, Document, and Plan. The 'Data' tab is active. The form is organized into several sections, each with a red box and a label indicating its layout name:

- lyt_top_1:** A section at the top containing fields for Node_id (114152), Node_type (SHUTOFF-VALVE), Cat_id (FF-VALVE110-PN16), and Epa type (SHORTPIPE).
- lyt_data_1:** A large section on the left containing fields for Code (114152), Elevation (36.9400), Depth (0.0000), Cat_matcat_id (FD), Cat_pnom (16), Cat_dnom (110), Work_id, Builtdate (2017/12/06), Builder (builder1), Owner (owner1), Work_id_end, Enddate (NULL), and Minsector_id (0).
- lyt_data_2:** A section on the right containing fields for Arc_id, Parent_id, Soilcat_id (soil1), Function_type (Standard Function), Category_type (Standard Category), Fluid_type (Standard Fluid), Location_type (Standard), Descript, Annotation, Observ, Comment, Num_value, and Svg.
- lyt_data_3:** A section at the bottom of the main data area containing fields for Exploitation (expl_02), Cat_presszone (pzone2-2d), Verified (TO REVIEW), Sector (sector2-1d), State (ON_SERVICE), and State_type (ON_SERVICE).
- lyt_bot_1:** A section at the bottom containing fields for Exploitation, Cat_presszone, and Verified.
- lyt_bot_2:** A section at the bottom containing fields for Sector, State, and State_type.

At the bottom right of the form, there are 'Accept' and 'Cancel' buttons.



PIPE - 130

Arc_type: PIPE, Arccat_id: FC63-PN10, Epa type: PIPE

Tabs: Datos, Elementos, Relaciones, Event, Documentos, Plan

Date from: 17/08/2017, Doc id: , Doc type:

	sys_id	id	arc_id	doc_id	doc_type	path	observ	
1	72	72	130	Demo document 1	OTHER	https://github.com/...	None	2018-03
2	341	341	130	Demo document 3	OTHER	https://github.com/...	None	2018-03
3	610	610	130	Demo document 2	OTHER	https://github.com/...	None	2018-03

Sector: sector1-1d, Dma: dma1-2d, State: OPERATIVE, State type: OPERATIVE

Buttons: Aceptar, Cancelar

[CARACTERÍSTICAS BASICAS]

Datatype. Tipo de dato. No aplica para elementos tipo combo. Los posibles valores son: ["string", "double", "date", "bytea", "boolean", "text", "integer", "numeric"]

Widgettype Tipo de widget. Los posibles valores son: ["datetime", "label", "nowidget", "text", "image", "typeahead", "button", "check", "combo", "hyperlink", "divider", "list", "spinbox", "hspacer", "tableview"]

typeahead & combo: Se rellenan usando los [GESTIÓN DE LOS DOMINIOS DE VALORES] con alguna limitación para los typeahead. El combo además puede usar el key enableWhenParent para estar activo solo en determinados valores de parent

list: Se usa la lista definida en el campo linkedobject tomada de la tabla config_form_list

spinbox: Se define el numero de decimales en el key 'spinboxDecimals' de widgetcontrols.

image: Se usa la imagen definida en el campo linkedobject tomada de la tabla sys_image

button: Se asigna la función de front-end definida en widgetfunction

link: El elemento tendrá formato linkable. Se debe asignar funcion de front-end en widgetfuncion.

hspacer: El elemeto será un separador para dar espacio entre otros widgets.

tableview: El elemento será una tabla con los datos definidos en la tabla config_form_list

label: Etiqueta del campo en el formulario y la tabla de atributos. Totalmente personalizable.

hidden: TRUE / FALSE – se muestra / no se muestra en el formulario y la tabla de atributos

tooltip: texto que se muestra en caso de situarte encima de la etiqueta del campo. Totalmente personalizable.

(*) **RECOMENDAMOS:** en caso de traducción de la etiqueta, poner al principio del tooltip el nombre real del campo (english) para poder saberlo fácilmente

placeholder: valor de ejemplo a mostrar cuando el campo está vacío.

iseditable: TRUE / FALSE - se puede / no se puede editar el campo en el formulario y la tabla de atributos.

ismandatory: TRUE / FALSE - en caso de TRUE, este campo deberá tener valor obligatoriamente.

isparent TRUE / FALSE Cuando un widget es padre de otro, permite recargar combos de los hijos que tienen identificado a este widget como su padre (dv_parent_id)

isautoupdate TRUE / FALSE Dispara el update del formulario sin esperar al ok del usuario. Valido para campos que se precisa recalculan cosas como profundidades o demás: No es posible esta opción para widgets tipo typeahead.

isfilter TRUE / FALSE Cuando tengamos un widget tipo list, puede ser filtrado por widgets que se encuentran en su mismo tab. Estos widgets pueden ser cualquiera, pero tendrán el atributo isfilter=true. Para ellos son de especial interes los keys 'vdefault' y 'listFilterSign' de widgetcontrols.

[GESTIÓN DE LOS DOMINIOS DE VALORES PARA COMBO & TYPEAHEAD]

En combinación con widgets tipo typeahead o combo.

dv_querytext: Query text siempre con el mismo criterio de id / idval. Si el widgettype es "typeahead", es obligatorio que id & idval para querytext deberan ser el mismo campo

dv_orderby_id: TRUE / FALSE Permite ordenar por id en lugar de por idval

dv_isnullvalue: TRUE / FALSE Permitir valores nulos en el querytext

dv_parent_id: Para aquellos widgets que tengan un parent

dv_querytext_filterc: Query text adicional filtrando por el valor del parent id

[CARACTERÍSTICAS AVANZADAS]

stylesheet: campo tipo json que permite una personalización gráfica de la etiqueta. Ver FAQs para ejemplos de este campo.

widgetcontrols: Permiten un control avanzado del widget con las siguientes opciones:

autoupdateReloadFields – recarga al momento otros campos en caso de que uno sea modificado. Actua en combinación con isautoupdate

```
UPDATE config_form_fields SET widgettype = 'combo' , isreload=true, widgetcontrols =
gw_fct_json_object_set_key(widgetcontrols, 'autoupdateReloadFields' ,["cat_matcat_id",
"cat_dnom", "cat_pnom"]::json) WHERE column_id IN ('arccat_id', 'nodecat_id', 'connecat_id')
```

enableWhenParent – habilita un combo solo en caso que el campo parent tenga ciertos valores

```
UPDATE config_form_fields SET widgetcontrols = gw_fct_json_object_set_key
(widgetcontrols,'enableWhenParent',['1, 2']::json) WHERE column_id IN ('state_type')
```

regexpControl – Control de lo que puede escribir usuario mediante expresion regular en widgets tipo texto libre

```
UPDATE config_form_fields SET hidden=false, datatype = 'text', widgetcontrols =
gw_fct_json_object_set_key(widgetcontrols,'regexpControl','[\d]+:[0-5][0-9]:[0-5][0-9]::text)
WHERE column_id = 'observ'
```

Nota adicional: Dado que el carácter '\' es reservado de sistema para PostgreSQL deberá hacerse el update con un \' \' para que en la row aparezcan dos, de manera que la sintaxis almacenada y con la que se va a trabajar será [\d]+:[0-5][0-9]:[0-5][0-9]

maxMinValues – establece un valor máximo para campos numéricos en widgets de texto libre

```
UPDATE config_form_fields SET widgetcontrols = gw_fct_json_object_set_key
(widgetcontrols,'maxMinValues',{'min':0.001, "max":100}::json) WHERE column_id = 'descript'
```

setMultiline – establece la posibilidad de campos multilinea para escritura con enter

spinboxDecimals – establece numero decimales concretos para el widget spinbox (vdef 2)

```
UPDATE config_form_fields SET widgetcontrols = gw_fct_json_object_set_key (widgetcontrols,
'spinboxDecimals', '3') WHERE column_id = 'descript'
```

widgetdim – Dimensiones para el widget

vdefault – Valor por defecto del widget. Tiene sentido para aquellos widgets que no pertenecen a datos de un feature, puesto que los valores por defecto se definen en los que el usuario ya tiene establecidos en config_param_user. De especial interés para los widgets filtro.

listFilterSign – Signo (LIKE, ILIKE, =, >, <) para los campos tipo filtro. En caso de omisión se usará ILIKE para listas tipo tableview e = para listas tipo tab

skipSaveValue – Si se define este valor como true, no se guardaran los cambios realizados en el widget correspondiente. Por defecto no hace falta poner nada porque se sobreentiende true.

labelPosition:- Si se define este valor [top, left, none], el label ocupará la posición relativa respecto al widget. Por defecto se sobreentiende left. Si el campo label está vacío, labelPosition se omite.

widgetfunction: Se define el nombre de la función de python que se ejecutará, y si los hubiera las características de los parametros adicionales. Se puede definir el fichero a utilizar con la clave "module", por defecto se entiende el fichero core/utills/tools_backend_calls.py. Para utilizar un fichero diferente a tools_backend_calls.py se tendrá que importar en tools_gw.py.

```
{"functionName":"add_document","module":"info", "parameters":{"sourcwidget", "targetwidget"}}
```

DOCUMENTO PROTOCOLO

P-09

Configuración de form_fields (3.5)

linkedobject:

Para widgettype list: Nombre de la lista radicada en tabla config_form_list para ser vinculada. En esta tabla se configura la query a ser usada (querytext) y el cliente con el que se va a llamar. Hay dos campos en la tabla que no tienen código asociado de momento, como son

listtype: Hace referencia a como se muestra la lista: tab (elementos en vertical para un tab estrecho) o en attributetable (elementos en tableview para un ancho mayor)

listclass: Clase de elementos mostrados en la lista (icon, iconos tipo galería o list).

Recomendado que las listas tengan el nombre list_* en la definición de la tabla donde son creados.

Para widgettype image: Nombre de la imagen radicada en tabla sys_image para ser vinculada. Recomendable que las imágenes tengan el nombre img_*

Para widgettype [text/check/combo/typeahead]: action (optativa) vinculada con el widget (getcatalog p.e.) que se encuentre disponible en el dialogo, configurada en config_form_tabs. Recomendable que las actions tengan el nombre action_*

Para widgettype button: Nombre de un icon (optativo) para setear en el button con la imagen asociada que se encuentra en la carpeta de plugin icons/backend/20x20. Recomendable que los nombres de los iconos sean simples numeros.png

FAQS

¿Algunas *queries* útiles para hacer reemplazos masivos?

Reemplazar para **todos los tipos de elemento** el **label** de un **campo concreto**. En este caso se reemplaza el *label* del campo *presszonecat_id* a Zona de presión para todos los elementos, ya sean *node*, *connec*, *gully* o *arc*.

```
UPDATE config_form_fields SET label='Zona de presión' WHERE columnname = 'presszonecat_id' AND formtype='form_feature' AND formname LIKE 've_%';
```

Ocultar para **todos los tipos de elemento** un **campo concreto**. En este caso se oculta el campo *verified* para todos los elementos.

```
UPDATE config_form_fields SET isenabled=FALSE WHERE columnname = 'verified' AND formtype='form_feature' AND formname LIKE 've_%';
```

Ocultar un **campo concreto** solo para los **elementos tipo node**. En este caso se oculta el campo *code* para los elementos tipo *node*.

```
UPDATE config_form_fields SET isenabled=FALSE WHERE columnname = 'code' AND formtype='form_feature' AND formname LIKE 've_node_%';
```

Ocultar un **campo concreto** solo para algún tipo de **elemento específico**. En este caso se oculta el campo *parent_id* para los tipos de *nodo*: *pump*, *reduction* y *tank*.

```
UPDATE config_form_fields SET isenabled=FALSE WHERE columnname = 'parent_id' AND formtype='form_feature' AND formname IN ('ve_node_pump', 've_node_reduction', 've_node_tank');
```

¿Se puede añadir un campo que yo quiera en el formulario?

Si se puede añadir el campo que uno quiera SIEMPRE Y CUANDO SEA SOLO CONSULTA y no se trate de un campo adicional (*addfield*). Para ello hay que:

Modificar la *ve_node_** / *ve_arc_** / *ve_connec_** / *ve_gully_** añadiendo todos aquellos campos que se quiera visualizar (solo consulta. Edición no permitida puesto que el *trigger* no funcionará).

La modificación puede ser con *DROP* o *CREATE OR REPLACE*, nosotros aconsejamos este último método. Mientras que para *DROP* hay que conservar el disparador y rehacerlo, si lo hacemos con *CREATER OR REPLACE* debemos añadir los campos nuevos OBLIGATORIAMENTE en el final de la vista, no pueden estar intermedios.

Añadir en la *config_form_fields* una nueva fila referente al campo creado.

EJEMPLO: Quiero añadir el descriptor del material en el formulario de *ve_arc_ownpipes*, puesto que en el formulario y en la vista correspondiente solo está el id:

Modifico la *ve_arc_ownpipe* haciendo un *JOIN* a *cat_mat* y añadiendo el campo *cat_mat.descriptor*.

Añadir en la *config_form_fields* una nueva fila referente al campo creado, en este caso *descriptor*. Si he usado un alias para el nombre deberé poner el alias.

Este es un entorno complejo ¿Se puede gestionar copias de seguridad?

Ver INTRODUCCIÓN del protocolo.

DOCUMENTO PROTOCOLO

P-09

Configuración de form_fields (3.5)

Que valores puedo poner en el campo stylesheet para modificar los colores de la etiqueta?

Poner color y negrita en un campo.

- Valor en *stylesheet* - {"label":"color:blue; font-weight:bold"}
- Resultado

Code	1092
Elevation	23.2100

Color en HTML y cambiar tamaño de letra.

- Valor en *stylesheet* - {"label":"color:#ff3300; font-size: 15px"}
- Resultado

Code	1092
Elevation	23.2100

Poner color en el background.

- Valor en *stylesheet* - {"label":"color:yellow; font-weight:bold; background-color:#9999ff"}
- Resultado

Code	1092
Elevation	23.2100

REFERENCIAS

REVISIONES:

Acción	Usuario	Fecha
Creado	Albert	11/11/2019
Modificado	Xavier T.	13/11/2019
Modificado	Albert B.	07/04/2020
Modificado	Xavier T.	07/07/2020
Modificado	Xavier T.	21/10/2020
Modificado	Xavier T.	08/04/2021
Modificado	Néstor	09/04/2021
Modificado	Albert B.	12/04/2021
Modificado	Albert B	03/09/2021
Modificado	Sergi Maspi	17/09/2021
Modificado	Xavier Torret	01/12/2021