

PROGRAMACIÓN DE OPENERP/ODOO

1.3. VISTAS

José Zambudio Bernabeu
jose.zambudio@diagram.net

INTRODUCCIÓN VISTAS

□ Tipos de elementos

1. Business object

- Clases de python las cuales extienden de la clase `orm.Model`. La gestión en base de datos de dichos módulos la mantiene el ORM.

2. Data

- Vienen dados por ficheros XML o CSV, y nos proporcionan todos aquellos datos que serán introducidos en base de datos. Entre estos tenemos: la declaración de vistas, workflows, datos demo, datos de configuración...

3. Reports

- Ficheros RML, HTML/MAKO o templates de OpenOffice con los que junto con la información de nuestro módulo, utilizaremos para generar informes PDF, HTML o ODT.

CREANDO MENUS

- ❑ Los menús son registros para el modelo `ir.ui.menu`
- ❑ Como cualquier registro podemos definir un record en un xml:

```
<record id="record_menu_ejemplo" model="ir.ui.menu">
  <field name="name">Mi menu</field>
  <field name="action" ref="id_del_action_a_ejecutar"/>
  <field name="sequence" eval="5"/>
  <field name="parent_id" ref="id_del_record_del_menu_padre"/>
</record>
```

- ❑ Pero es recomendable utilizar el shortcut que OpenERP nos habilita:

```
<menuitem id="record_menu_ejemplo"
  name="Mi menu"
  action="id_del_action_a_ejecutar"
  sequence="5"
  parent="id_del_record_del_menu_padre"
  groups="listado_de_grupos_con_permiso"/>
```

CREANDO MENUS

- ❑ @id: Obligatorio
 - ➔ Especificamos el id (debe ser único)
- ❑ @name:
 - ➔ Nombre del menú a mostrar en la interfaz del usuario. Por defecto aplica el nombre del @action.
- ❑ @action:
 - ➔ Especificamos el id del action a ejecutar (ir.actions). No es obligatorio.
- ❑ @groups:
 - ➔ Especificamos cada uno de los grupos de usuarios que pueden ver el menu. Los separamos por ‘,’.
 - ➔ Si no lo especificamos indicamos que todos los grupos pueden verlo.
- ❑ @sequence:
 - ➔ Integer. Se utiliza para ordenar los menús.
 - ➔ En caso de no especificarse, OpenERP establece por defecto 10. En caso de tener el mismo sequence OpenERP los ordena por id (orden de creación)
- ❑ @parent:
 - ➔ Especificamos la jerarquía, el id del menu padre del que vamos a depender.

CREANDO ACTIONS

- ❑ Los actions definen el comportamiento del programa para responder ante las acciones de un usuario, por ejemplo accionar un menuitem, “Imprimir”, “Más”...
- ❑ Podemos definir varios tipos de actions:

<i>Ventanas</i>	<i>Informes</i>	<i>Ejecución</i>
<i>ir.actions.act_window</i>	<i>ir.actions.report.xml</i>	<i>ir.actions.act_url</i>
<i>ir.actions.wizard</i>		<i>ir.actions.server</i>
<i>ir.actions.act_window_close</i>		<i>ir.actions.client</i>

CREANDO ACTIONS - WINDOW

❑ ir.actions.act_window

```
<record model="ir.actions.act_window" id="id_de_nuestro_action">
  ... <field name="name">nombre.de.nuestro.action</field>
  ... <field name="res_model">modelo</field>
  ... <field name="view_type">form|tree</field>
  ... <field name="view_mode">tree,form,calendar...</field>
  ... <field name="view_id" ref="id_vista_por_defecto"/>
  ... <field name="domain">[('total', '=', '0')]</field>
  ... <field name="context">{'nuestro_flag': True}</field>
</record>
```

CREANDO ACTIONS - WINDOW

- ❑ `ir.actions.act_window`
 - ➔ **name:** Nombre del action.
 - ➔ **view_id:** ID de la vista (`ir.ui.view`) a mostrar cuando el `act_window` se active.
 - ➔ **domain:** Lista de constraints para filtrar los registros a mostrar.
 - ➔ **context:** Diccionario (Por defectos dinamicos, flag...). `{}`
 - ➔ **res_model:** Modelo al que vincular el action.
 - ➔ **target:** [current, new, inline, inlineview]
 - ➔ **view_type:** [tree, form]
 - ➔ **view_mode:** Sólo si `view_type = form`, establecemos los tipos de vistas que tendrá nuestro modelo. tree, form
 - ➔ **limit:** 80
 - ➔ **auto_refresh:** 0
 - ➔ **search_view:** ID de la vista search (`ir.ui.view`)

INTRODUCCIÓN VISTAS

- ❑ Las vistas describen la forma por la que se va a presentar los registros de los modelos a los usuarios.
- ❑ Más específicamente... Podemos definir un (o muchos) vistas, para un mismo modelo.
- ❑ Estas vistas pueden ser o no del mismo tipo.
- ❑ Tipos de vistas básicas:
 - ➔ Form
 - ➔ Tree
- ❑ Más tipos de vistas:
 - ➔ Calendar
 - ➔ Graph
 - ➔ Diagram
 - ➔ Gantt
 - ➔ Kanban
 - ➔ Search

INTRODUCCIÓN VISTAS

- ❑ @model: Para definir una vista debemos realizar un record al modelo “ir.ui.view”
- ❑ @id: Por convención “view_*_`[tree|form...]`”
 - ❑ `<field name=“name”>...</field>`
 - ❑ Indicamos un name para la vista. Por convención mismo que @id cambiando ‘_’ por ‘.’
 - ❑ `<field name=“inherit_id” ref=“id_vista”/>`
 - ❑ Heredamos de una vista...
 - ❑ `<field name=“model”>...</field>`
 - ❑ Modelo de los records que va a mostrar la vista.
 - ❑ `<field name=“field_parent”>...</field>`
 - ❑ Indicamos a la vista que el contenido debe representar el contenido siguiendo su jerarquía.

INTRODUCCIÓN VISTAS

- ❑ `<field name="groups_id" eval="[(6, 0, [ref('record_group_id')])]" />`
 - ❑ ACLs...
- ❑ `<field name="priority">...</field>`
 - ❑ Orden para seleccionar las vistas
 - ❑ Por defecto 16.
 - ❑ `_order = "priority,name"`
- ❑ `<field name="arch" type="xml">...</field>`
 - ❑ Contenido de la vista. Aquí establecemos el tipo y el estilo.
 - ❑ `<tree...></tree>`
 - ❑ `<form...></form>`
 - ❑ ...

INTRODUCCIÓN VISTAS

- ❑ Atributos generales
 - ❑ **create**
 - ❑ Determinamos si se pueden crear nuevos registros desde dicha vista.
 - ❑ default = 1 ~ true
 - ❑ **delete**
 - ❑ Determinamos si se pueden borrar registros desde dicha vista.
 - ❑ default = 1 ~ true
 - ❑ **edit**
 - ❑ Determinamos si se pueden modificar registros desde dicha vista.
 - ❑ default = 1 ~ true
 - ❑ **string**
 - ❑ Nombre de la vista.

INTRODUCCIÓN VISTAS

❑ Elementos generales

❑ Field tag: <field [atributos]/>

- ❑ Definimos la columna del modelo a mostrar en la vista.
- ❑ Atributos:

<i>attrs</i>	<i>Modificaciones al resto de atributos vía una condición previa.</i>
<i>string</i>	<i>Modificamos el label definido para el field.</i>
<i>width</i>	<i>No funciona en los tree. Para el form indica el ancho del campo.</i>
<i>on_change</i>	<i>Trigger que ejecuta un método del servidor cuando el valor del field es modificado.</i>
<i>filter_domain</i>	<i>Search. Permite modificar completamente el domain, sobre-escribiendo al del action.</i>

INTRODUCCIÓN VISTAS

<i>invisible</i>	<i>Indica que el field va a estar oculto en el navegador.</i>
<i>password</i>	<i>Reemplaza el contenido del valor del field por “*”</i>
<i>sum</i>	<i>Tree. Indicamos que el campo contiene el sumario de todas las líneas. Lo pinta en “left bottom”</i>
<i>avg</i>	<i>Mismo que sum pero “average” (promedio, media...)</i>
<i>select</i>	<i>1 -> búsqueda básica. 2 -> búsqueda avanzada. Por defecto 1.</i>
<i>groups</i>	<i>Indicamos para que grupos de usuarios será visible.</i>
<i>operator</i>	<i>Reemplaza el operador por defecto para el field.</i>
<i>colspan</i>	<i>Número de columnas que debe utilizar el field. Por defecto: 4.</i>

INTRODUCCIÓN VISTAS

<i>nolabel</i>	<i>No muestra el label del field. Por defecto = 1 a no ser que indiquemos el field dentro de un group.</i>
<i>required</i>	<i>Indica que para poder modificar/crear un registro, dicho campo debe estar introducido.</i>
<i>readonly</i>	<i>Indica que el field es de sólo lectura.</i>
<i>domain</i>	<i>Restringe vía un filtro el contenido que va a mostrar el field.</i>
<i>context</i>	<i>“flags”, “default_”, “form_view_ref”, “tree_view_ref”, “group_by”...</i>
<i>states</i>	<i>Estados en los que va a ser visible el field.</i>
<i>digits</i>	<i>Mismo uso que en la declaración del field en el model.</i>
<i>icon</i>	<i><code>openerp/tools/misc.py:608 - 644</code> <code>addons/web/src/img/icons</code></i>

INTRODUCCIÓN VISTAS

<i>mode</i>	<i>Sequence de las vistas cuando se can a ver.</i>
<i>filename</i>	<i>Para binary, nombre por defecto del fichero a descargar.</i>
<i>height</i>	<i>No funciona en los tree. Para el form indica el alto del campo.</i>
<i>default_focus</i>	<i>Al inicio de un formulario, se hace focus a aquel field con dicho campo activo.</i>
<i>statusbar_visible</i>	<i>Mismo caso que states. Indicamos los estados en los que la barra será visible.</i>
<i>statusbar_colors</i>	<i>Especificamos los colores que tendrá para cada estado: “blue” “red”</i>
<i>options</i>	<i>Indicamos ciertas opciones para el cliente web. +info: addons/web/static/src/js/*.js</i>
<i>placeholder</i>	<i>Mismo uso que en html...</i>

INTRODUCCIÓN VISTAS

group	<i>Indicamos que grupos tienen acceso al field.</i>
widget	<p><i>Indica el widget que va a utilizar el field para mostrar el contenido:</i></p> <ul style="list-style-type: none"> • <i>one2many_list</i> • <i>many2many_tags</i> • <i>many2many_kanban</i> • <i>many2many_binary</i> • <i>many2onebutton</i> • <i>selection</i> • <i>statusbar</i> • <i>handle</i> • <i>monetary</i> • <i>mail_followers</i> • <i>mail_thread</i> • <i>many2many_tags_email</i> • <i>char</i> • <i>float_time</i> • <i>html</i> • <i>url</i> • <i>one2many</i> • <i>email</i> • <i>integer</i> • <i>progressbar</i> • <i>image</i> • <i>pad</i> • <i>date</i>

INTRODUCCIÓN VISTAS

❑ Elementos generales

❑ Button tag: <button [atributos]/>

- ❑ Definimos un botón a mostrar en la vista. Dicho botón realizará una determinada acción.
- ❑ Atributos:

<i>attrs</i>	<i>Modificaciones al resto de atributos vía una condición previa.</i>
<i>invisible</i>	<i>Indica que el botón va a estar oculto en el navegador.</i>
<i>name</i>	<i>Id del action - nombre del método - nombre del signal -> Ver @type.</i>
<i>icon</i>	<i>Muestra un icono sobre el botón. Si no se indica el botón es de sólo texto.</i>
<i>string</i>	<i>En vistas tree los botones no tienen label. En los form indica el label a mostrar.</i>

INTRODUCCIÓN VISTAS

<i>states</i>	<i>Estados en los que va a ser visible el botón.</i>
<i>confirm</i>	<i>Salta un alert del navegador: js: confirm("Press a button!");</i>
<i>special</i>	<i>Único valor disponible: "cancel". Indica que el popup debe cerrarse sin realizar ninguna acción.</i>
<i>default_focus</i>	<i>Al inicio de un formulario, se hace focus a aquel field con dicho campo activo.</i>
<i>colspan</i>	<i>Número de columnas que debe utilizar el botón.</i>
<i>target</i>	<i>En caso de @type="action" define el target del action definido.</i>
<i>readonly</i>	<i>Muestra el botón como disable.</i>
<i>context</i>	<i>"flags", "default_", "form_view_ref", "tree_view_ref", "group_by"...</i>

INTRODUCCIÓN VISTAS

<i>help</i>	<i>Al igual que el field en la definición del modelo, podemos indicar al usuario una ayuda del funcionamiento del botón.</i>
<i>class</i>	<i>Añadimos una clase css.</i>
<i>type</i>	<p><i>Define el tipo de acción a realizar:</i></p> <ul style="list-style-type: none"> • workflow (default): Manda una señal (signal) para el modelo actual utilizando el @name del botón como nombre de esta. • object: El botón ejecuta un método del modelo actual utilizando el @name del botón como nombre de este. • action: El botón ejecuta un action (ir.actions) con id indicado en el @name de este.

INTRODUCCIÓN VISTAS

❑ Elementos generales

❑ Group tag: <group [atributos]/>

- ❑ Utilizado tanto en vistas “form,search”
- ❑ Agrupa los fields que contiene.
- ❑ Atributos:

<i>Expand</i>	<i>En las vistas search, podemos agrupar fields bajo un icono de agrupado. 1 (expandido - default), 0 (agrupado)</i>
<i>String</i>	<i>Label para el grupo.</i>
<i>attrs</i>	<i>Modificaciones al resto de atributos vía una condición previa.</i>
<i>colspan</i>	<i>Número a columnas a usar.</i>
<i>col</i>	<i>Número de columnas a generar (internas)</i>

INTRODUCCIÓN VISTAS

<i>rowspan</i>	<i>Número de filas a utilizar.</i>
<i>states</i>	<i>Estados en los que va a ser visible el grupo.</i>
<i>height</i>	<i>Indica el alto del grupo.</i>
<i>width</i>	<i>Indica el ancho del grupo.</i>
<i>name</i>	<i>Agilizar herencia.</i>
<i>invisible</i>	<i>Indica que el grupo va a estar oculto en el navegador.</i>

INTRODUCCIÓN VISTAS

❑ Elementos generales

❑ Separator tag: <separator [atributos]/>

- ❑ Añade un título que se utiliza para separar campos en una vista.

❑ Atributos:

<i>invisible</i>	<i>Indica que el separador va a estar oculto en el navegador.</i>
<i>name</i>	<i>Agilizar herencia.</i>
<i>colspan</i>	<i>Número de columnas que debe utilizar el separador.</i>
<i>rowspan</i>	<i>Número de filas que debe utilizar el separador.</i>
<i>string</i>	<i>Indicamos texto a mostrar para el separador.</i>

❑ Elementos generales

❑ Label tag: <label [atributos]/>

❑ Especificamos manualmente la posición del label de un field.

❑ Atributos:

<i>string</i>	<i>Especificamos el texto a mostrar.</i>
<i>for</i>	<i>Muestra el label del field con @name que se indica en este campo.</i>
<i>attrs</i>	<i>Modificaciones al resto de atributos vía una condición previa.</i>
<i>invisible</i>	<i>Indica que el label va a estar oculto en el navegador.</i>
<i>name</i>	<i>Agilizar herencia.</i>

INTRODUCCIÓN VISTAS

<i>colspan</i>	<i>Número de columnas que debe utilizar el label.</i>
<i>help</i>	<i>Al igual que el field en la definición del modelo, podemos indicar al usuario una ayuda del funcionamiento del field (al que corresponde nuestro label).</i>
<i>width</i>	<i>Indica el ancho del label.</i>
<i>name</i>	<i>Agilizar herencia.</i>

INTRODUCCIÓN VISTAS

- ❑ Elementos generales
 - ❑ Newline tag: `<newline/>`
 - ❑ Fuerza un salto de línea incluso si no se han rellenado todas las columnas.

ON_CHANGE ATTRIBUTE

- ❑ Con el atributo `on_change` podemos declarar un método el cuál será invocado cuando cambie el valor del field en una vista.

`<field`

`name="nombre_del_field"`

`on_change="nombre_del_método(
nombre_del_field,
otro_field,`

`...,`

`context)"/>`

`def nombre_del_método(self, cr, uid, ids,
nombre_del_field, otro_field, ..., context=None):`

ON_CHANGE ATTRIBUTE

- ❑ Cuando modifiquemos el valor del field “nombre_del_field” el cliente web ejecutará el método “nombre_del_método” del servidor.
- ❑ Dicho método debe devolver un diccionario, con el que podemos:
 - ➔ Modificar el valor de cualquier field de la vista.
 - ➔ Modificar el domain de cualquier field de la vista.
 - ➔ Mostrar al usuario un mensaje de error.

ON_CHANGE ATTRIBUTE

```

□ {
  'value': {
    'nombre_del_field': 'valor',
    'otro_field': 'valor',
    ...,
  },
  'domain':{
    'nombre_del_field': [('field','=', 'valor')]
  },
  'warning':{
    'title': _('Warning...'),
    'message': _('Message...'),
  }
}

```

- ❑ Disposición:
 - ❑ Por defecto...

label	<input type="text"/>	label	<input type="text"/>
label	<input type="text"/>	label	<input type="text"/>
label	<input type="text"/>	label	<input type="text"/>
label	<input type="text"/>	label	<input type="text"/>

- ❑ 4 columnas, label - input + label - input

- ❑ Disposición:
 - ❑ Por defecto...

label	<input type="text" value="1"/>	label	<input type="text" value="2"/>
label	<input type="text" value="3"/>	label	<input type="text" value="4"/>
label	<input type="text" value="5"/>	label	<input type="text" value="6"/>
label	<input type="text" value="7"/>	label	<input type="text" value="8"/>

- ❑ Los fields se van añadiendo desde la izquierda a la derecha, y de arriba a abajo

- ❑ Disposición:
 - ❑ Por defecto...

label	1	
2		
label	3	label 4
label	5	label 6

- ❑ Un field puede utilizar varias columnas.

□ Atributos:

→ De los generales...

- ▶ create, delete, edit, string

→ type

→ link

→ col:

- ▶ Establecemos el número de columnas interno. Por defecto: 4.

❑ Elementos:

- ❑ <notebook [atributos]/>
- ❑ <field [atributos]/>
- ❑ <group [atributos]/>
- ❑ <button [atributos]/>
- ❑ <label [atributos]/>
- ❑ <separator [atributos]/>
- ❑ <image [atributos]/>
- ❑ <newline [atributos]/>
- ❑ <header>...</header>
- ❑ <footer>...</footer>
- ❑ <form [atributos]/>
- ❑ <dashboard [atributos]/>
- ❑ <search [atributos]/>
- ❑ <filter [atributos]/>
- ❑ <html [atributos]/> → <div>...</div>, <p>...</p>
- ❑ <properties [atributos]/>

- ❑ **<notebook [atributos]/>**
 - ➔ Atributos:
 - colspan
 - tabpos: Establece la posición de las pestañas del notebook
 - tabpos=“[up,down,left,right]”
 - ➔ Elementos:
 - <page [atributos]/>

- ❑ **<page [atributos]/>**
 - ❑ Atributos:
 - ❑ De los generales...
 - ❑ string
 - ❑ name
 - ❑ attrs
 - ❑ invisible
 - ❑ Elementos:
 - ❑ <newline [atributos]/>
 - ❑ <label [atributos]/>
 - ❑ <field [atributos]/>
 - ❑ <group [atributos]/>
 - ❑ <properties [atributos]/>

- ❑ **<page [atributos]/>**
 - ❑ Elementos:
 - ❑ **<separator [atributos]/>**
 - ❑ **<button [atributos]/>**
 - ❑ **<notebook [atributos]/>**
 - ❑ **<page [atributos]/>**
 - ❑ ~~**<dashboard [atributos]/>**~~
 - ❑ ~~**<filter [atributos]/>**~~
 - ❑ ~~**<html [atributos]/>**~~
 - ❑ ~~**<form [atributos]/>**~~

TREE VIEWS

- ❑ Las utilizamos para mostrar información en forma de lista.
- ❑ Son las vistas más simples.

```
<record id="view_invoice_line_tree" model="ir.ui.view">
  <field name="name">account.invoice.line.tree</field>
  <field name="model">account.invoice.line</field>
  <field name="arch" type="xml">
    <tree string="Invoice Line">
      <field name="name"/>
      <field name="account_id" groups="account.group_account_user"/>
      <field name="quantity"/>
      <field name="uos_id" groups="product.group_uom"/>
      <field name="price_unit"/>
      <field name="discount" groups="sale.group_discount_per_so_line"/>
      <field name="price_subtotal"/>
    </tree>
  </field>
</record>
```

- ❑ En el contenido de la vista vamos añadiendo los fields a mostrar. Estos fields se irán mostrando por columnas.

■ Atributos:

■ De los generales...

- create, delete, edit, string

■ colors:

- colors="[colour]:[condition];"

- default: colors="black:True"

- ejemplo: colors="red:state == 'cancelled';blue: state != 'cancelled' and total > 0;"

■ fonts:

- fonts="bold:message_unread==True"

■ editable:

- Opciones: "top" | "bottom"

■ toolbar:

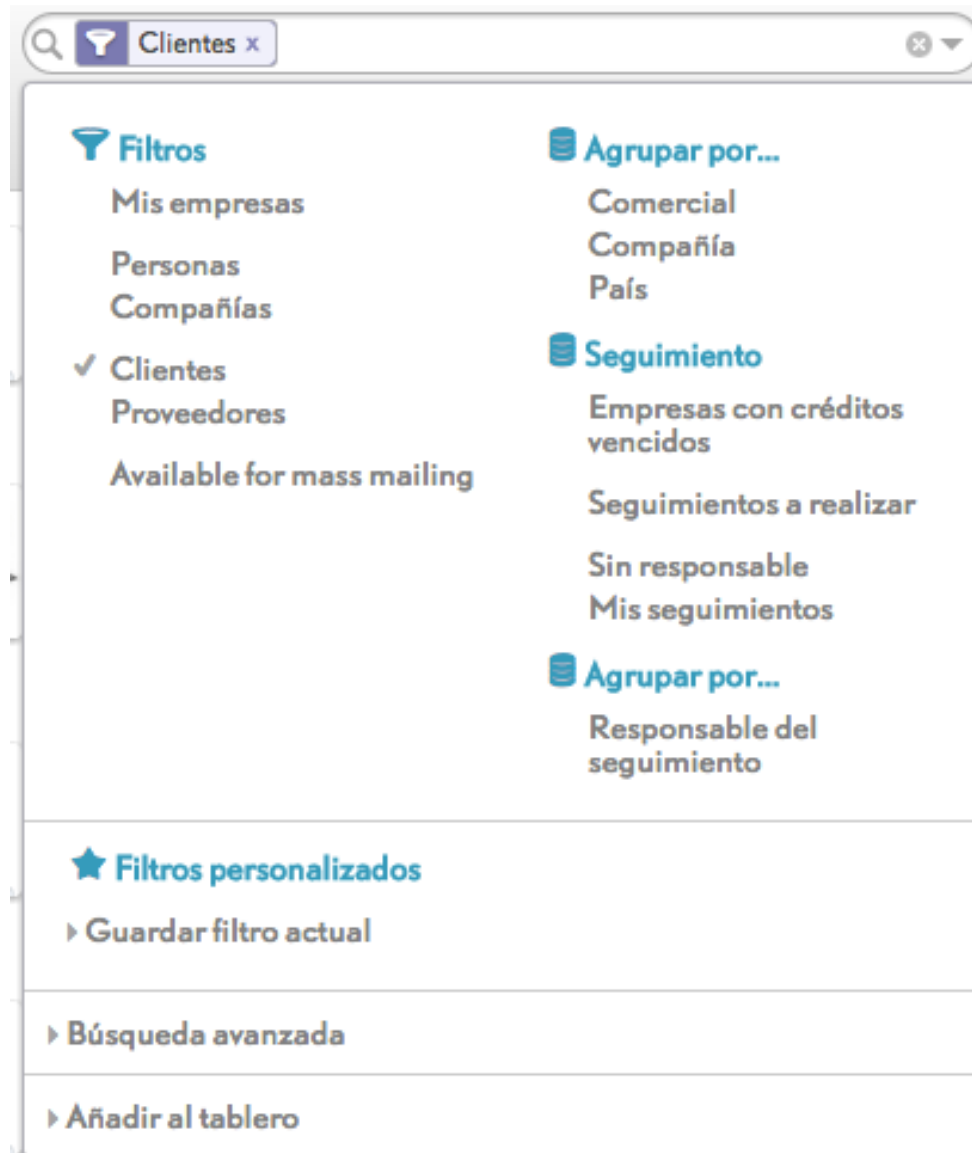
- Opciones: "1" | "True"

- Mostrar acciones "Print", "Más"

❑ Elementos:

- ❑ `<field [atributos]/>`
- ❑ `<button [atributos]/>`
- ❑ ~~`<separator [atributos]/>`~~
- ❑ ~~`<tree [atributos]/>`~~
- ❑ ~~`<group [atributos]/>`~~
- ❑ ~~`<filter [atributos]/>`~~
- ❑ ~~`<html [atributos]/>`~~ → `<div>...</div>`, `<p>...</p>`
- ❑ ~~`<newline [atributos]/>`~~

SEARCH VIEWS



SEARCH VIEWS

- ❑ Muy parecido a la vista form...
- ❑ Se aplica únicamente a la vista tree...
- ❑ Podemos definir:
 - ❑ Fields por los que se realizará la búsqueda del string introducido.
 - ❑ Filtros predeterminados.
 - ❑ Agrupaciones de los registros de la vista.
- ❑ Además el usuario puede definir filtros personalizados y compartidos con el resto de usuarios.

❑ Atributos:

❑ De los generales...

❑ String

❑ Elementos:

❑ <field [atributos]/>

❑ <group [atributos]/>

❑ expand, string.

❑ <separator [atributos]/>

❑ <filter [atributos]/>

❑ <newline [atributos]/>

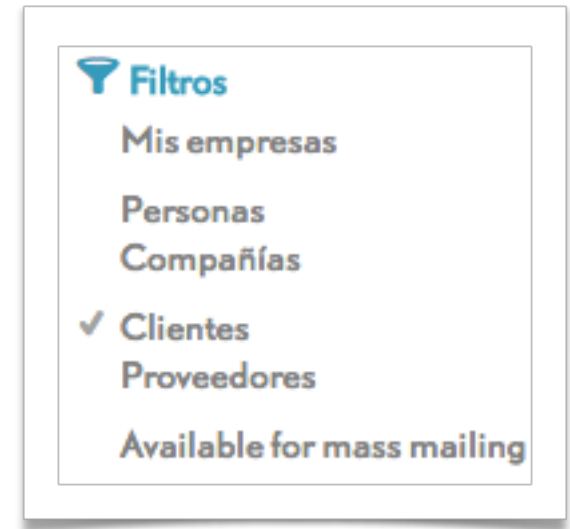
❑ <search [atributos]/>

❑ <properties [atributos]/>

SEARCH VIEWS

❑ Elementos:

- ❑ <filter [atributos]/>
 - ❑ Atributos:
 - ❑ string:
 - ❑ domain
 - ❑ help



September 2013						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
01	02	03	04	05	06	07
08	09	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

- ☐ Folio
- ☐ Draft

CALENDAR VIEWS

❑ Atributos:

- ❑ De los generales...
 - ❑ string
- ❑ **date_start** -> Fecha inicial. Required.
- ❑ **date_stop** -> Fecha fin. Ignorado si especificamos date_delay
- ❑ **date_delay** -> Campo numérico que indica el tiempo en horas para el registro.
- ❑ **day_length** -> Campo entero para indicar las horas laborables en un día, por defecto 8 horas.
- ❑ **color** -> Field, normalmente many2one, el cuál la interfaz utilizará para distribuir los colores por registros.
- ❑ **mode** -> String. Determina el zoom de la vista:
 - ❑ day, week, month

- ❑ **Elementos:**
 - ❑ `<field [atributos]/>`
 - ❑ ~~`<html [atributos]/>`~~

❑ Atributos:

❑ De los generales...

- ❑ create, delete, edit
- ❑ quick_create
- ❑ default_group_by

- ❑ **Elementos:**
 - ❑ <field [atributos]/>
 - ❑ <templates [atributos]/>

- ❑ **Atributos:**
 - ❑ De los generales...
 - ❑ string
 - ❑ orientation
 - ❑ type
 - ❑ color

- ❑ **Elementos:**
 - ❑ `<field [atributos]/>`
 - ❑ `<html [atributos]/>`

❑ Atributos:

❑ De los generales...

- ❑ create, delete, edit, string

❑ mode:

- ❑ day, 3days, week, 3weeks, month, 3months, year, 3years, 5years

- ❑ color

- ❑ date_delay

- ❑ day_length

- ❑ date_start

- ❑ date_stop

- ❑ date_string

- ❑ default_group_by

- ❑ **Elementos:**
 - ❑ `<level [atributos]/>`
 - ❑ `<field [atributos]/>`
 - ❑ `<html [atributos]/>`

- ❑ **Atributos:**
 - ❑ De los generales...
 - ❑ create, string

❑ Elementos:

- ❑ `<node [atributos]/>`
- ❑ `<arrow [atributos]/>`
- ❑ `<label [atributos]/>`