DIAGRAM SOFTWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

PROGRAMACIÓN DE OPENERP/ODOO

1.3. VISTAS

José Zambudio Bernabeu
jose.zambudio@diagram.net

Nubeaerp global cloud solutions

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 shortcout 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:

Ventanas	*Informes*	*Ejecución*
ir.actions.act_window	*ir.actions.report.xml*	*ir.actions.act_url*
~~*ir.actions.wizard*~~		*ir.actions.server*
ir.actions.act_window_close		*ir.actions.client*

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

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 del la vista vamos añadiendo los fields a mostrar. Estos fields se irán mostrando por columnas.

TREE VIEWS

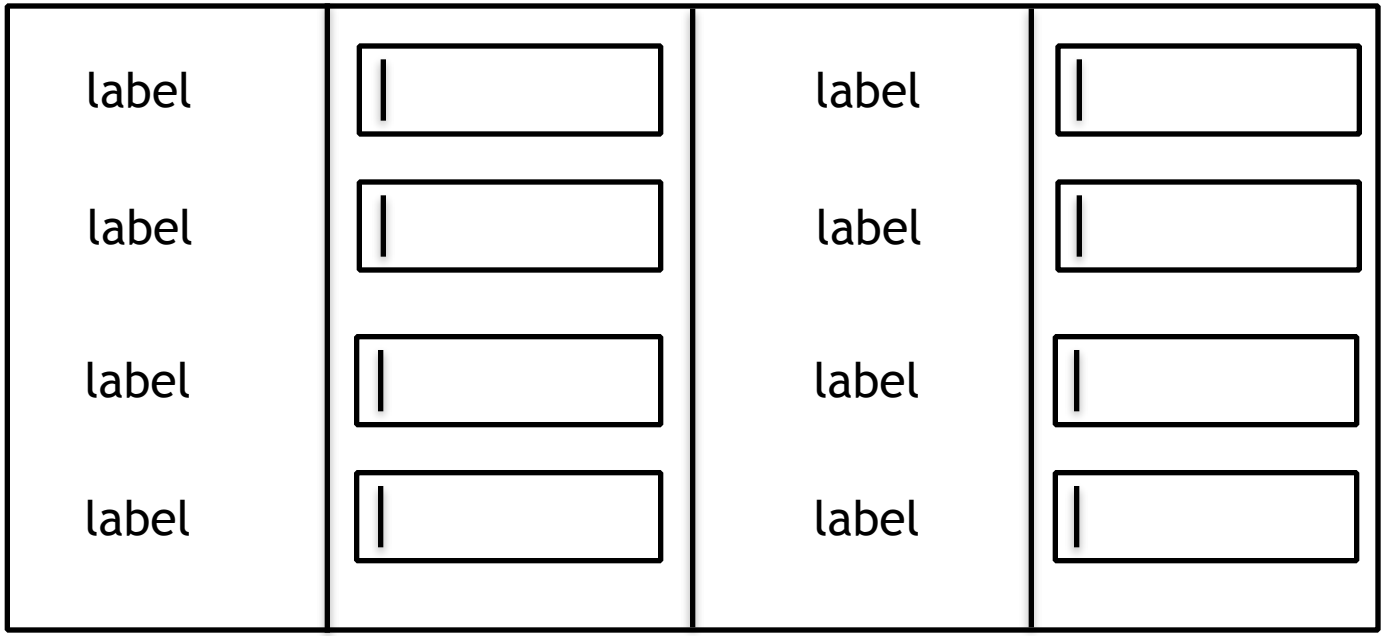
- Atributos:
  - colors:

```
<record id="invoice_tree" model="ir.ui.view">
    <field name="name">account.invoice.tree</field>
    <field name="model">account.invoice</field>
    <field name="arch" type="xml">
        <tree colors="blue:state == 'draft';black:state in ('proforma','proforma2','open');gray:state == 'cancel'" string="Invoice">
            <field name="partner_id" groups="base.group_user"/>
            <field name="date_invoice"/>
```

    - colour:field_name [operator] [condition];
  - toolbar:
    - toolbar="1" —> agrupamos las líneas de forma que al hacer click sobre un registro nos muestra

FORM VIEWS

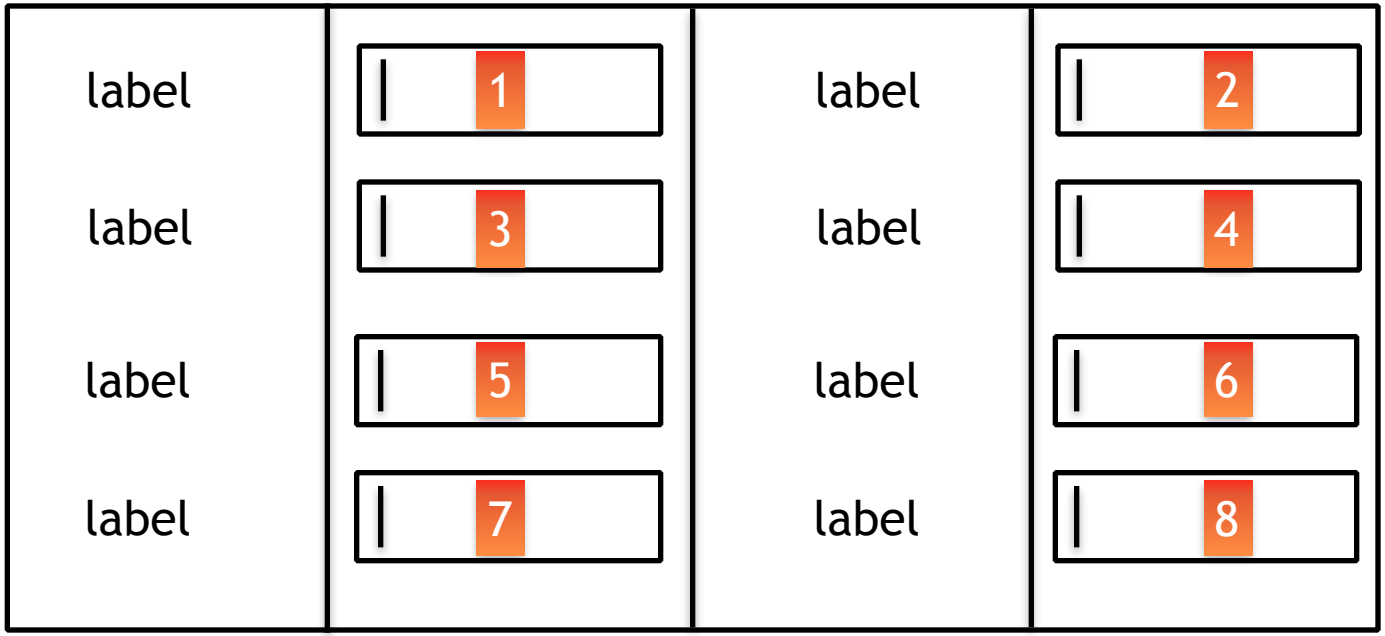
- Disposición:
  - Por defecto…



  - 4 columnas, label - input + label - input

FORM VIEWS

- Disposición:
  - Por defecto…



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