

1. Introducción

Haciendo uso de lo visto en prácticas anteriores:

- creamos una base de datos para una empresa que comercializará artículos náuticos.
- modificamos el perfil principal para que datos como el nombre o el email se visualicen correctamente en los informes, más adelante.
- instalamos los siguientes módulos: Gestión de inventario, Contabilidad y finanzas, Gestión de compras y Ventas.

Create Database

Odoo is up and running!
Create a new database by filling out the form, you'll be able to install your first app in a minute.

Database Name

Email

Password

Language **Country**

Spanish / Español Spain

☐ Load demonstration data (Check this box to evaluate Odoo)

Continue

 **Nombre de la compañía**
Náutica S.L.

Lema de la compañía
Adaptables al mar

Información General

Dirección Astilleros, nave 15
Calle 2...
Gijón Asturias 33211
España

Sitio web http://www.empresanautica.com

Teléfono 904390011

Fax

Correo electrónico info@empresanautica.com

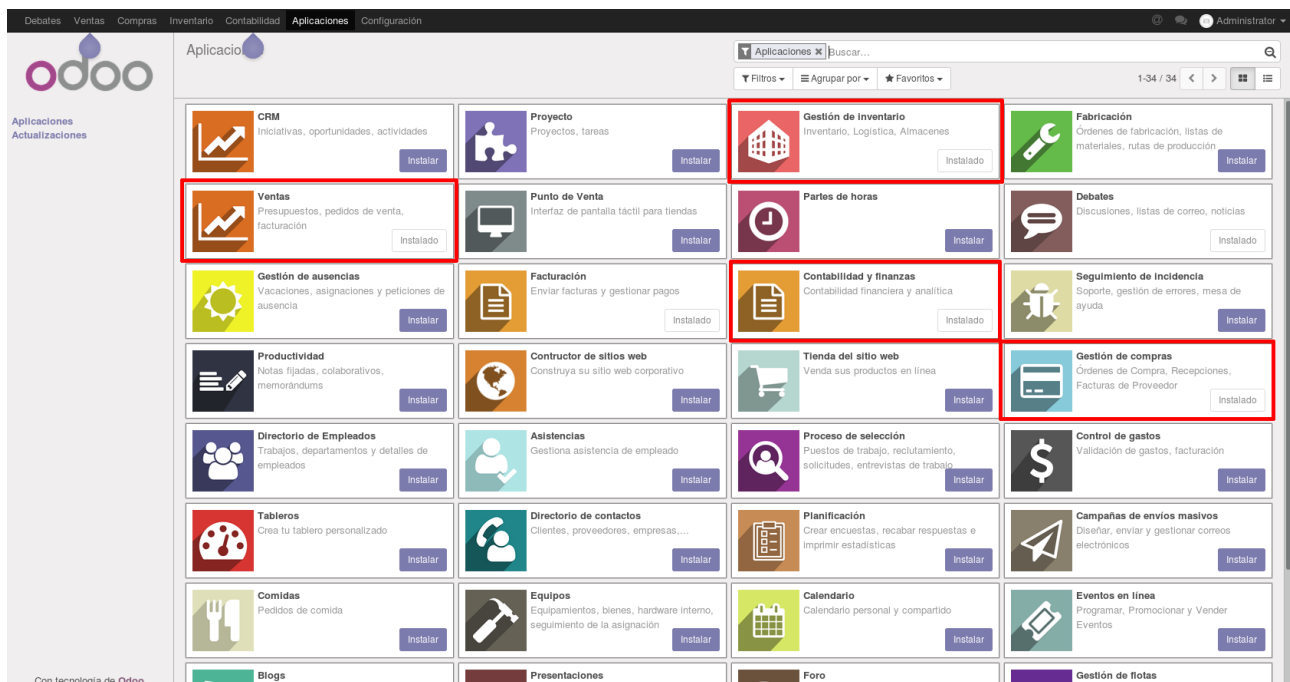
NIF ESW3302035E *

Registro de compañía

Moneda EUR

Activar aquí una nueva moneda antes de crear una nueva compañía.

*Recuerda que el CIF requiere el prefijo del país



Durante la instalación de los módulos Odoo añade automáticamente otros como Debates y Facturación.

2. Modelo de datos y vistas

2.0 Qué son los modelos y las vistas

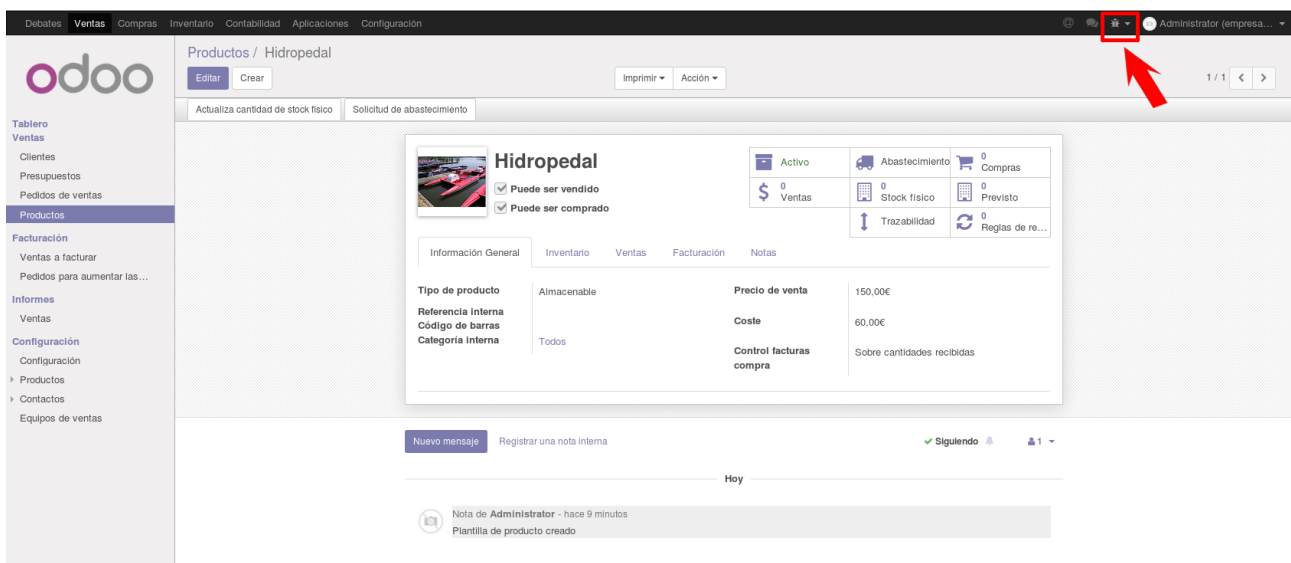
Los modelos recuerdan a las clases en programación. Por ejemplo, tenemos un concesionario y queremos registrar cada coche como un producto pero pronto observamos que el formulario que viene por defecto no nos permite añadir ni quitar campos, y nosotros necesitamos algo más específico para las características técnicas del vehículo. Lo que haríamos en programación sería crear directamente nuestra propia clase, lo cual además nos permite entendernos con otras clases. En Odoo lo que creamos es un modelo u objeto (e usa más el término objeto para referirse a algo ya creado, con identidad propia). Por tanto, cuando hablamos de objeto nos referimos sólo a la definición de los campos, no a lo que se muestra en pantalla. Para eso están las vistas.

Una vista define cómo mostrar en pantalla un objeto. Es decir, de entre todos esos campos que dispongo cuáles quiero que se vean, en qué parte, en qué orden, etc. A partir de un objeto se pueden generar diferentes vistas. La conclusión más valiosa que sacamos de esto es que absolutamente todo lo que se ve en el navegador será siempre algún tipo de vista y que los objetos en sí permanecen ocultos a nosotros pero en constante uso. Existen varios tipos de vista estándar, siendo los más habituales las vistas formulario, kanban y tree. Las veremos más adelante.

2.1 Modificando un objeto y una vista existentes

Para entender mejor cómo funciona lo anterior vamos a tomar un objeto ya creado, ver qué campos tiene e incluir alguno más. Después comprobaremos cómo hay que indicarle también a su vista o vistas que tengan en cuenta nuestros campos extra.

Lo primero que debemos hacer es activar el **Modo desarrollador** en el menú de Configuración. Una vez hecho nos vamos a “Ventas > Productos” y pulsamos en “Crear” para dar de alta un producto que nos sirva de ejemplo. En mi caso:



Ahora mismo estamos observando una vista. Esta vista está asociada al objeto Producto, del cual de momento no sabemos demasiado. ¿Cómo podemos llegar a esa información? Arriba a la derecha, junto al nombre de usuario con el que nos hemos conectado, se ve un pequeño icono similar a un insecto. Si hacemos click sobre él desplegará las herramientas de desarrollador. De entre sus

opciones, la que nos interesa es “Editar FormVer”, situada justo al final. Nos debe dibujar esta ventana:

Editar FormVer

Nombre de la vista: product.template.product.form

Tipo de vista: Formulario

Modelo: product.template

Secuencia: 8

Activo: ☒

Campo hijo:

Vista heredada: product.template.common.form

Ver modo heredado: Vista base

Modelo de datos: product.template.product.form

ID externo: product.product_template_only_form_view

Estructura | Permisos de acceso | Vistas heredadas

Editar traducciones

```
1 <?xml version="1.0"?>
2 <data><xpath expr="//form" position="attributes">
3   <attribute name="name">Product Template</attribute>
4 </xpath>
5 <field name="type" position="after">
6   <field name="default_code" attrs="{ 'invisible': [ ('product_variant_count', '>', 1)] }"/>
7   <field name="barcode" attrs="{ 'invisible': [ ('product_variant_count', '>', 1)] }"/>
8 </field>
9
10 <div name="button box" position="inside">
11   <button name="108" type="action" icon="fa-sitemap" class="oe_stat_button" attrs="{ 'invisible': [ ('product_vari
12   <field string="Variants" name="product_variant_count" widget="statinfo"/>
13 </button>
14 </div>
15
16 <xpath expr="//page[@name='sales']" position="after">
17   <page name="variants" string="Variants" groups="product.group_product_variant">
18     <field name="attribute_line_ids" widget="one2many_list" context="{ 'show_attribute': False }">
19       <tree string="Variants" editable="bottom">
20         <field name="attribute_id"/>
21         <field name="value_ids" widget="many2many_tags" options="{ 'no_create_edit': True }" domain="[ ('attr
22       </tree>
23     </field>
24     <p class="oe_grey">
25       <strong>Warning</strong>: adding or deleting attributes
26       will delete and recreate existing variants and lead
27       to the loss of their possible customizations.
28     </p>
29   </page>
```

Guardar Descartar

La información relevante está al principio. De momento nos quedamos sólo con el primer y el tercer campo, que son aquellos archivos que debemos buscar y retocar a mano, así que anotamos sus nombres para poder realizar la búsqueda y nos vamos a otra parte. Concretamente cambiamos a “Configuración > Estructura de la base de datos > Modelos”

De primeras se nos mostrará una lista con todos los objetos existentes en la base de datos actual. Para filtrar, simplemente escribimos el nombre del objeto que habíamos anotado hace un momento en el TextField del buscador. Luego hacemos click en el nombre para abrirlo.

Modelos

Crear Importar

Modelo product.template

Filtros Agrupar por Favoritos

Modelo	Descripción del modelo	Tipo	Modelo transitorio
product.template	Plantilla de producto	Objeto base	<input type="checkbox"/>

Modelos

Campos

Restricciones del modelo

Relaciones many2many

Adjuntos

Registro

Modelos referenciables

Precisión decimal

La estructura de un objeto es muy parecida a la de una clase en programación. Vamos con el encabezado: “Descripción del modelo” es de carácter aclaratorio para el desarrollador. “Modelo” es el identificador único que tendremos que usar siempre para referirnos a este objeto, muy importante. “Modelo transitorio” es el transient de la programación (si la base de datos se exporta o se guarda, cualquier objeto con este checkbox activado sería excluido de la operación). “Tipo” nos informa de que es un objeto creado por Odoo, no por nosotros. “En las aplicaciones” se detalla en qué aplicaciones o módulos se está utilizando este objeto, para que procedamos con cabeza.

Descripción del modelo	Plantilla de producto	Tipo	Objeto base
Modelo	product.template	En las aplicaciones	account, product, purchase, sale, stock, stock_account
Modelo transitorio	<input type="checkbox"/>		

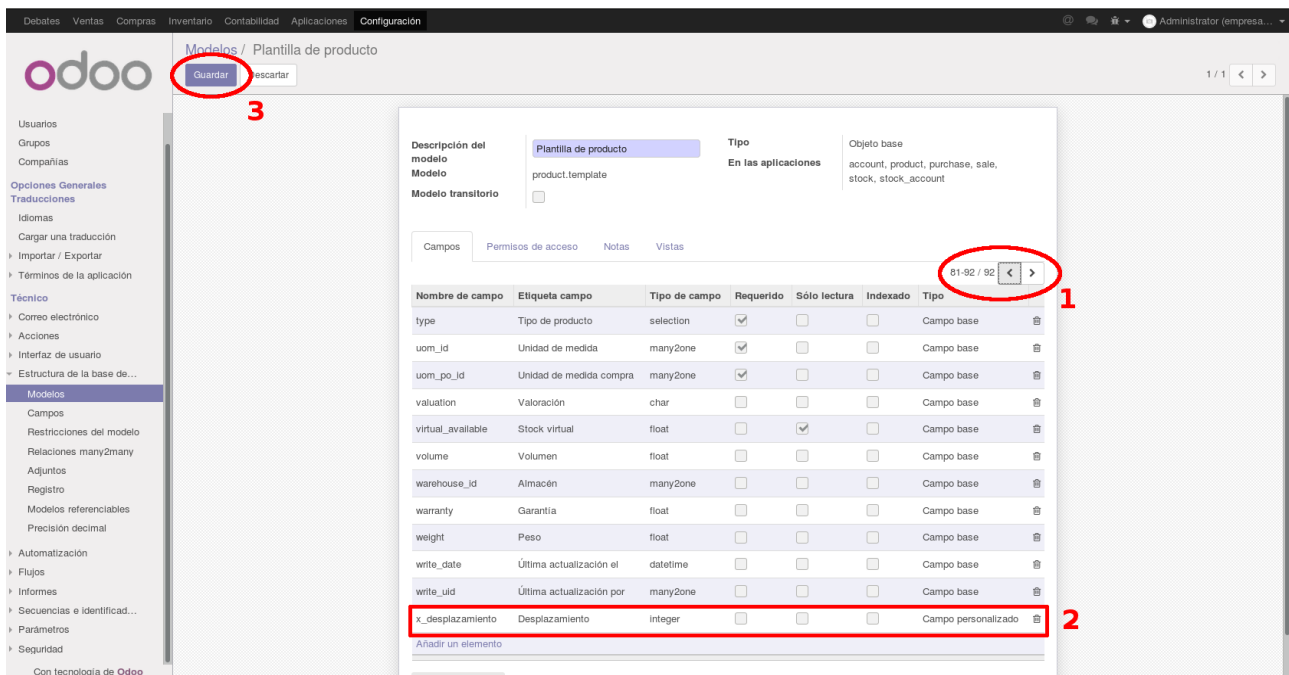
Siguiendo con la parte de abajo tenemos: los campos o datos de este objeto con sus tipos y modificadores. Los permisos de acceso a esos datos por parte de otros usuarios. Las notas para desarrolladores. Las diferentes vistas que están funcionando a partir del objeto (entre ellas, lógicamente, product.template.product.form; la vista formulario del producto).

Para añadir un campo más, es tan sencillo como pulsar sobre el botón “Editar” (arriba a la izquierda) y luego bajar hasta el final de la página. Justo encima de donde se lee “Crear Menú” aparece, algo escondida, la opción “Añadir un elemento”

supplier_taxes_id	Impuestos de proveedor	many2many	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo base	
taxes_id	Impuestos cliente	many2many	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo base	
track_service	Seguimiento del Servicio	selection	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo base	
tracking	Seguimiento	selection	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo base	
Añadir un elemento							
Crear un menú							

Lo ahora tenemos en pantalla es la creación de un Campo exclusivo para el objeto Producto. “Nombre del campo” es el identificador de ese campo, por tanto, lo que tendremos que utilizar más adelante. “Etiqueta Campo” es el nombre que se leerá junto a la casilla donde podamos escribir algo, cuando el campo aparezca ya en una vista; es decir, esto no funciona como en HTML o Swing donde añadimos un TextField y después un label o etiqueta al lado que aclare qué se debe escribir en él, sino que Odoo nos pide que se lo indiquemos directamente aquí y él ya lo mostrará todo junto en la vista. “Tipo de campo” representa qué clase de dato es aceptado por ese campo. “Campo de ayuda” es el texto flotante que aparece automáticamente cuando dejamos el ratón encima de un elemento y se genera un pop-up de ayuda.

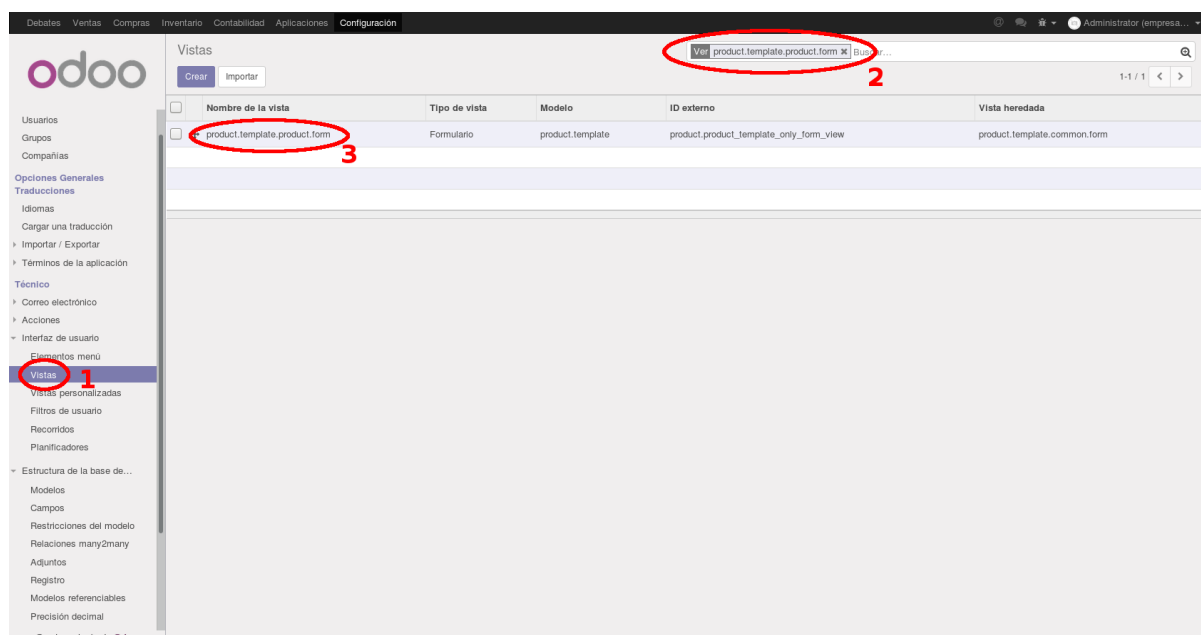
Lo que más llama la atención es el prefijo “x_” que se genera solo en el identificador. Se trata de una convención para poder distinguir los campos originales del objeto respecto a los que va añadiendo el desarrollador. De modo que conservaremos esta práctica. Con todo hecho, pulsamos el botón “Guardar & Cerrar” y el campo ya estaría creado.



Para encontrar el campo creado, es posible que tengamos que usar los cursores de paginado en la lista. Una vez que hayamos terminado con todos los campos que necesitamos, pulsamos sobre "Guardar". En mi caso he añadido dos más, quedando al final:

x_desplazamiento	Desplazamiento	integer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo personalizado	
x_max_pasajeros	Máximo de pasajeros	integer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo personalizado	
x_matricula	Matricula	char	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo personalizado	
Añadir un elemento							

Ahora todas las vistas que utilicen el objeto Producto pueden hacer uso de nuestros campos mostrándolos en pantalla como se les indique. Arriba habíamos apuntado la vista asociada a nuestro producto. El proceso de buscarla es idéntico al del objeto, pero en la categoría de arriba, es decir, "Interfaz de Usuario > Vistas".



Al entrar en la vista que muestra nuestro producto nos encontramos con una complicación extra: si nos fijamos en la pestaña “estructura”, el código de esta vista sólo representa pequeñas añadiduras que se hacen con respecto a otro código que hereda de una vista padre. No siempre ocurre así y lo normal es que de primeras observemos un código íntegro y legible, no de una vista heredada.

Por una parte este sistema es beneficioso porque de esta manera se va incluyendo contenido sin tener que manipular el original, pero por el otro nos obliga a tener que consultar la vista padre para saber dónde existe algún fragmento de código más claro y donde sea fácil incrustar los campos que hemos añadido al objeto. La vista padre se localiza en el campo “Vista heredada” y basta con clickar encima para viajar a su ficha.

Nombre de la vista	product.template.product.form
Tipo de vista	Formulario
Modelo	product.template
Secuencia	8
Activo	<input checked="" type="checkbox"/>
Campo hijo	
Vista heredada	product.template.common.form
Ver modo heredado	Vista base
Modelo de datos	product.template.product.form
ID externo	product.product_template_only_form_view

Estructura
Permisos de acceso
Vistas heredadas

El código de la vista padre ya es más intuitivo. A partir de aquí lo único que tenemos que hacer es abrir una segunda pestaña que muestre la vista normal de nuestro producto y comparar lo que vemos con lo que se puede deducir del código. En este caso, localizamos el código de la pestaña “Información general” encerrado por etiquetas <page>

Productos / Hidropedal

Editar

Crear

Imprimir

Acción

Actualizar cantidad de stock físico

Solicitud de abastecimiento

Hidropedal

Activo

Abastecimiento

Compras

0 Ventas

0 Stock físico

0 Previsto

Trazabilidad

Pagos de re...

Información General

Inventario

Ventas

Facturación

Notas

Tipo de producto

Referencia interna

Código de barras

Categoría interna

Almacenable

Todos

Precio de venta

150,00€

Coste

60,00€

Control facturas compra

Sobre cantidades recibidas

Nuevo mensaje

Registrar una nota interna

Siguiente

Nota de Administrador - hace 4 horas

Plantilla de producto creada

```

29 *
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *

```

```

<page string="General Information" name="general_information">
  <group>
    <group name="group_general">
      <field name="type"/>
      <field name="categ_id" string="Internal Category"/>
    </group>
    <group name="group_standard_price">
      <field name="list_price" widget="monetary" options='{ "currency_field": "cur
      <label for="standard_price" groups="base.group.user"/>
      <div name="standard_price uom" groups="base.group.user">
        <field name="standard_price" widget="monetary" options='{ "currency_fiel
      </div>
      <field name="company_id" groups="base.group_multi_company" options='{ "no.cr
      <field name="uom_id" groups="product.group.uom" options='{ "no.create": True
      <field name="uom_po_id" groups="product.group.uom" options='{ "no.create": 1
      <field name="currency_id" invisible="1"/>
    </group>
  </group>
</page>
<page string="Sales" attrs='{ "invisible": ("!sale.ok", "<!--</div>
<div name="pricelist item" groups="product.group.product.pricelist">
  <separator string="Pricing"/>
  <field name="item_id" nolabel="1" context='{ "default_base": "list_price", "defe
  <tree string="Pricelist Items" editable="bottom">
    <field name="pricelist_id" string="Pricelist" required="1"/>
    <field name="fixed_price" string="Price" required="1"/>
    <field name="min_quantity"/>
    <field name="date_start"/>
    <field name="date_end"/>
    <field name="applied_on" invisible="1"/>
  </tree>
</div>
<group name="sale">
  <group name="email_template_and_project" attrs='{ "invisible": ("!type", "!<!--</group>
</page>
<page string="Notes" name="notes">
  <group name="description">
    <separator string="Description for Quotations" colspan="4"/>
    <field name="description sale" colspan="4" nolabel="1" placeholder="This note

```

Para incluir los campos que hemos creado, podemos optar por dos soluciones. Añadirlos directamente a la pestaña “Información General” como un nuevo grupo, y con cada campo en una línea <field> tal que:

```

<group>
  <field name="x_desplazamiento"/>
  <field name="x_max_pasajeros"/>
  <field name="x_matricula"/>
</group>

```

O bien añadir una pestaña propia, a continuación de la otra:

```
30 <page string="General Information" name="general_information">
31 <group>
32 <group name="group_general">
33 <field name="type"/>
34 <field name="categ_id" string="Internal Category"/>
35 </group>
36 <group name="group_standard_price">
37 <field name="list_price" widget="monetary" options="{ 'currency_field': 'cur
38 <label for="standard_price" groups="base.group_user"/>
39 <div name="standard_price_uom" groups="base.group_user">
40 <field name="standard_price" widget="monetary" options="{ 'currency_fiel
41 </div>
42 <field name="company_id" groups="base.group_multi_company" options="{ 'no_cr
43 <field name="uom_id" groups="product.group_uom" options="{ 'no_create': True
44 <field name="uom_po_id" groups="product.group_uom" options="{ 'no_create': 1
45 <field name="currency_id" invisible="1"/>
46 </group>
47 </group>
48 </page>
49 <page string="Embarcación" name="boat">
50 <group>
51 <field name="x_desplazamiento"/>
52 <field name="x_max_pasajeros"/>
53 <field name="x_matricula"/>
54 </group>
55 </page>
```

Muy importante: en una vista formulario, `<field>` SIEMPRE debe ir dentro de un `<group>`

Guardamos y estaría listo.

A la hora de hacer pruebas y comprobar el resultado, se recomienda mantener las dos pestañas abiertas: la del código que estamos editando, y la que muestra el resultado, es decir, la vista normal refrescando su código. La causa es que si metemos la pata con el código y tratamos de volver al editor de la vista, Odoo puede generar un error impidiéndonos acceder de nuevo a él. Por esta razón, lo recomendable es evitar directamente modificar el código original como acabamos de hacer. En su lugar se puede crear una vista nueva que añada sólo una pestaña con nuestros campos de forma separada y por herencia. No obstante, si queremos comprobar que la pestaña con nuestros campos tal y como lo tenemos ahora, ha sido correctamente implementada, el resultado sería éste:

The screenshot shows the Odoo product form for 'Hidropedal'. The 'Embarcación' tab is selected and highlighted with a red box. This tab contains three fields: 'Desplazamiento' (Displacement) with a value of 0, 'Máximo de pasajeros' (Maximum passengers) with a value of 0, and 'Matricula' (Registration). The left sidebar shows the navigation menu with 'Productos' selected. The top bar shows the user 'Administrator (empresa)'. The bottom bar shows a message from the Administrator: 'Plantilla de producto creado'.

2.2 Utilizando una vista heredada de tipo Formulario

Para crear una vista heredada nos vamos de nuevo a “Configuración > Interfaz de usuario > Vistas”. Arriba a la izquierda, seleccionamos el botón “Crear”. Aquí haremos un alto para poner en claro qué tenemos que hacer y sobre todo el porqué.

Ahora mismo la vista que muestra nuestro producto se llama *product.template.product.form*. Esta vista apenas muestra código, sino que hereda de otra vista llamada *product.template.common.form*. Lo que nosotros debemos hacer es crear una nueva vista que herede sólo de la más reciente. ¿Por qué? Porque al heredar de ella recogemos también todo lo que le obtiene por ascendencia, es decir, su código y el de la vista padre. Por explicarlo de un modo más visual:

- Abuelo: *product.template.common.form*, tiene un código propio “A”
- Padre: *product.template.product.form*, hereda el código “A” y además añade su parte “B”
- Hijo: *nuestra_futura_vista*, hereda los códigos “A” y “B”, y además añade su parte “C”

El segundo punto que tenemos que aclarar es el de Secuencia, que choca un poco con lo explicado arriba. La secuencia es una especie de prioridad, pero no entendida como dos vistas hijo que compiten entre sí por ver quién muestra nuestro producto. Afecta también a la vista padre y a la vista abuelo, de modo que cualquier vista hijo con un valor de secuencia más bajo que el padre y el abuelo se coloca antes que ellos. Por esta razón va de 0 a más. En la siguiente captura podemos ver la secuencia de *product.template.product.form*, es decir, en la nueva vista a crear debemos rebajar ese valor para colocarnos por delante de él:

Nombre de la vista	<input type="text" value="product.template.product.form"/>	Campo hijo	<input type="text"/>
Tipo de vista	<input type="text" value="Formulario"/>	Vista heredada	<input type="text" value="product.template.common.form"/>
Modelo	<input type="text" value="product.template"/>	Ver modo heredado	<input type="text" value="Vista base"/>
Secuencia	<input type="text" value="8"/>	Modelo de datos	<input type="text" value="product.template.product.form"/>
Activo	<input checked="" type="checkbox"/>	ID externo	<input type="text" value="product.product_template_only_form_view"/>

El tercer aspecto, es el código que hay que añadir. Se realiza con XPath y el resultado sería:

```
1 <xpath expr="//page[@name='general_information']" position="after">
2 <page string="Embarcación" name="boat">
3 <group>
4 <field name="x_desplazamiento"/>
5 <field name="x_max_pasajeros"/>
6 <field name="x_matricula"/>
7 </group>
8 </page>
9 </xpath>
```

La primera línea ordena esto: *busca una pestaña (page) con la siguiente característica: tiene un atributo ‘name’ cuyo valor es explícitamente ‘general_information’*. Cuando encuentres esa pestaña, mueve el curso a la línea inmediatamente detrás. A continuación se añade el código de nuestra pestaña allí donde esté situado el cursor.

Mucho ojo con el uso de las comillas: *general_information* va entre comillas simples porque se encuentra ya a su vez en el interior de otras comillas, en este caso dobles.

Con todo, nuestra nueva vista heredada debe presentar este aspecto:

The screenshot shows the Odoo configuration interface for a view. The top navigation bar includes 'Debates', 'Ventas', 'Compras', 'Inventario', 'Contabilidad', 'Aplicaciones', and 'Configuración'. The left sidebar contains a menu with 'Tablero', 'Usuarios', 'Grupos', 'Compañías', 'Opciones Generales', 'Traducciones', 'Idiomas', 'Cargar una traducción', 'Importar / Exportar', 'Términos de la aplicación', 'Técnico', 'Correo electrónico', 'Acciones', 'Interfaz de usuario', 'Elementos menú', 'Vistas', 'Vistas personalizadas', 'Filtros de usuario', 'Recorridos', 'Planificadores', 'Estructura de la base de...', 'Modelos', and 'Campos'. The main content area is titled 'Vistas / product.template.common.form.boat' and contains a form with the following fields: 'Nombre de la vista' (product.template.common.form.boat), 'Tipo de vista' (Formulario), 'Modelo' (product.template), 'Secuencia' (7), 'Activo' (checked), 'Campo hijo' (empty), 'Vista heredada' (product.template.product.form), 'Ver modo heredado' (Vista de extensión), and 'Modelo de datos ID externo' (empty). Below the form are tabs for 'Estructura', 'Permisos de acceso', and 'Vistas heredadas'. The 'Estructura' tab is active, showing an XML snippet:

```
1- <xpath expr="//page[@name='general_information']" position="after">
2- <page string="Embarcación" name="boat">
3-   <group>
4-     <field name="x_desplazamiento"/>
5-     <field name="x_max_pasajeros"/>
6-     <field name="x_matricula"/>
7-   </group>
8- </page>
9- </xpath>
```

A la hora de elegir el nombre hemos añadido una palabra al final para que se continúe leyendo en él la herencia que sigue nuestra vista. El tipo de vista formulario es capital que coincida con el original. El modelo continúa siendo el mismo, product.template, porque de él nos vienen los campos a mostrar. La secuencia ha sido rebajada para adelantar al padre, que a su vez adelantaba ya al abuelo. Vista heredada apuntará a la vista padre para heredar de éste todo su código y el de sus ascendientes.

Las acciones de ventana y las vistas

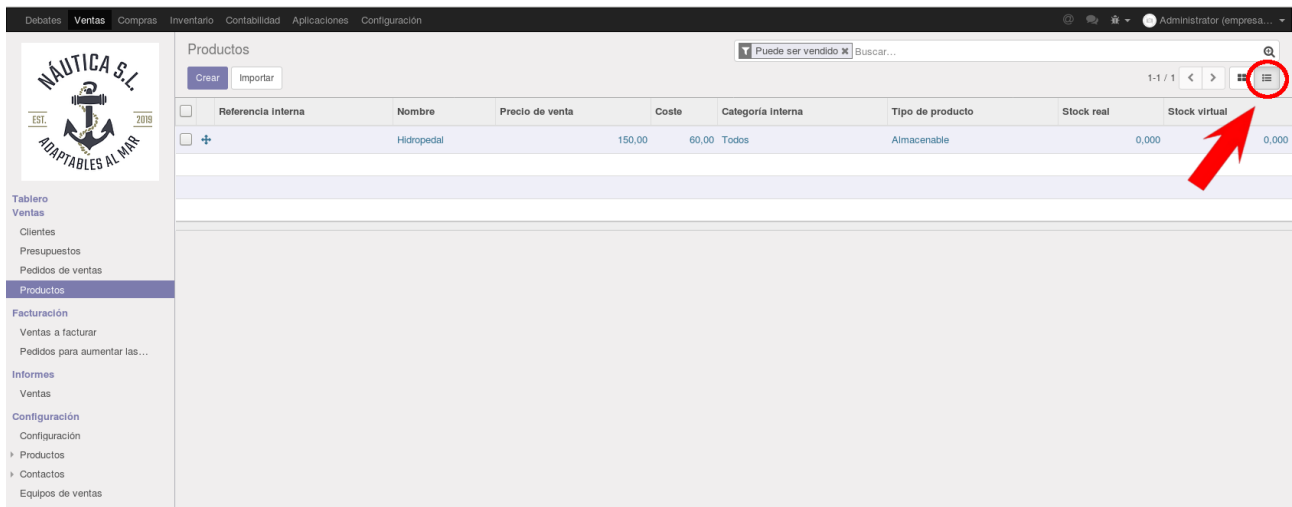
¿Y si todo lo anterior no funciona? Entonces es probable que entre en juego un cuarto factor: las acciones de ventana. Son las encargadas de establecer qué ocurre cuando hacemos click sobre un submenú del panel izquierdo, por ejemplo, “Ventas > Ventas > Producto”. Entre las opciones que podemos personalizar está la de definir que una vista se muestre de manera fija para un submenú, independientemente del valor de secuencia que posea. Para corregir esta configuración que Odoo trae de inicio con determinadas vistas, hemos de ir a “Configuración > Interfaz de usuario > Elementos menú”.

Allí buscaremos el menú donde se nos presenta el problema. Por ejemplo, si no se mostrara la pestaña Embarcación en mi producto, buscaría el menú “Ventas/Ventas” y en la ventana que se abre “Productos” dentro de la pestaña submenús. Al hacer click sobre “Productos” se muestra una nueva ventana donde aparece la acción asociada. De nuevo clickamos sobre ella. En este paso puede ocurrir o no que se muestre la propia configuración de la acción. Si no sucede, simplemente cerramos la ventana actual, y la acción se habrá cargado por detrás.

En la configuración de la acción buscamos su última categoría “Vistas”. En ella debe aparecer una tabla que relaciona los tipos de vista con las que cargan por defecto. Ni siquiera es necesario que figure alguna por defecto, de modo que podemos dejar todos los campos vacíos. De no encontrar nada, los tipos vistas consultarán cuál es la primera en valor de secuencia dentro de la herencia actual.

2.3 Utilizando una vista heredada de tipo Tree o Lista

Siguiendo con nuestro producto, observamos qué sucede si nos vamos a “Ventas > Productos” y seleccionamos un tipo de vista Tree.



Los campos que habíamos añadido no aparecen por ninguna parte.

De nuevo repetimos el proceso visto con anterioridad y pulsamos sobre el icono de herramientas de desarrollador, y en “Editar TreeVer” para examinar qué vista se mostrando ahora mismo.

✱ Editar TreeVer

Nombre de la vista

product.template.product.tree

Tipo de vista

Árbol

Modelo

product.template

Secuencia

16

Activo

☒

Campo hijo

Vista heredada

Ver modo heredado

Vista base

Modelo de datos

product.template.product.tree

ID externo

product.product_template_tree_view

Estructura

Permisos de acceso

Vistas heredadas

Editar traducciones

1

<?xml version="1.0"?>

2

<tree string="Product">

3

<field name="sequence" widget="handle"/>

4

<field name="default_code"/>

5

<field name="name"/>

6

<field name="list_price"/>

7

<field name="standard_price"/>

8

<field name="categ_id"/>

9

<field name="type"/>

10

<field name="uom_id" options="{ 'no_open': True, 'no_create': True }" groups="product.group_uom"/>

11

<field name="active" invisible="1"/>

12

</tree>

13

Guardar

Descartar

No hereda de ninguna vista padre y el código es legible a la perfección, por lo que sería muy sencillo añadir a mano los campos. Pero para aplicar lo aprendido, crearemos nuestra propia vista heredada de tipo Tree, disminuirémos la secuencia a 15 y adicionaremos sólo los campos nuevos, por ejemplo a continuación del campo Nombre.

Regresamos a “Configuración / Interfaz de usuario / Vistas” y pulsamos en “Crear”, y aquí volvemos a insistir en utilizar dos pestañas distintas en el navegador para realizar las pruebas.

La nueva vista resulta en:

Nombre de la vista	product.template.product.tree.boat	Campo hijo	
Tipo de vista	Árbol	Vista heredada	product.template.product.tree
Modelo	product.template	Ver modo heredado	Vista base
Secuencia	15	Modelo de datos	
Activo	<input checked="" type="checkbox"/>	ID externo	

Estructura

Permisos de acceso

Vistas heredadas

Editar traducciones

```
1 <xpath expr="//field[@name='name']" position="after">
2   <field name="x_desplazamiento"/>
3   <field name="x_max_pasajeros"/>
4   <field name="x_matricula"/>
5 </xpath>
```

3. Creación de Menús

Hasta ahora hemos visto que los objetos definen la estructura de los datos en campos, y que estos campos se pueden mostrar de múltiples formas mediante las vistas. Pero cuando hacemos click sobre un menú o submenú del panel izquierdo también se está cargando un objeto y la vista que dispone sus campos. Esa relación objeto-menú/submenú se puede configurar libremente.

Para hacer la prueba vamos a crear un objeto de cero que nos servirá para almacenar los campos relativos al alquiler de un cobertizo privado donde el cliente puede reparar o descansar su embarcación.

Debates Ventas Compras Inventario Contabilidad Aplicaciones Configuración

Modelos / Nuevo

Guardar Descartar

NAÚTICA S.L.

EST. 2018

ADAPTABLES AL MAR

Configurado

Opciones Generales

Traducciones

Idiomas

Cargar una traducción

Importar / Exportar

Términos de la aplicación

Técnico

Correo electrónico

Acciones

Interfaz de usuario

Elementos menú

Vistas

Vistas personalizadas

Filtros de usuario

Recorridos

Plantificadores

Estructura de la base de...

Modelos

Campos

Restricciones del modelo

Relaciones many2many

Adjuntos

Registro

Modelos referenciables

Con tecnología de Odoo

Descripción del modelo

Alquiler de un embarcadero

Tipo

Objeto personalizado

Modelo

x_boathouse.rent

En las aplicaciones

Modelo transitorio

☐

Campos

Permisos de acceso

Notas

Vistas

Nombre de campo	Etiqueta campo	Tipo de campo	Requerido	Sólo lectura	Indexado	Tipo
x_identificador	Identificador	char	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo personalizado
x_metros_cuadrados	Metros cuadrados	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo personalizado
x_ocupado_libre	Ocupado/libre	boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo personalizado
x_precio_dia	Precio por día	float	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo personalizado
Añadir un elemento						
Crear un menú						

Al pulsar sobre “Guardar” se nos generarán de forma automática otros campos que no hemos especificado (este proceso es normal). Ahora nos falta un quinto campo llamado “Cliente” para el cual necesitamos antes crear uno. Es posible que al hacerlo nos salte un error relacionado con la falta de datos sobre la contabilidad que debe aplicársele. Para solucionarlo habría que ir a “Contabilidad > Configuración > Configuración” y seleccionar algunos de los planes de contabilidad que se sugiere por defecto. Una vez listo, creamos el campo Cliente y establecemos para dicho campo un tipo especial llamado relación.

Al principio de este documento, cuando se explicaba por primera vez qué eran un objeto y una vista, se decía que los objetos podían relacionarse entre sí igual que lo hacen las clases en la programación. En este caso una relación trata precisamente de eso. Nosotros pretendemos que este campo albergue no un dato, sino un objeto en sí que es el cliente. ¿Y cuál es el objeto de un cliente? Basta consultar cualquiera de las vistas formulario, kanban o árbol del submenú Cliente para comprobar que el modelo es *res.partner*

Abrir: Campos

Nombre de campo: x_cliente

Etiqueta campo: Cliente

Tipo de campo: many2one

Campo ayuda: Cliente que tiene contratado el alquiler

Propiedades base

Requerido: ☐

Sólo lectura: ☐

Almacenado: ☒

Indexado: ☐

Copiado: ☒

Relación del objeto: res.partner

Al eliminar: Establecer a NULL

Dominio: []

En cuanto el tipo de relación se lee de fuera para dentro, entendido esto como desde el objeto con el que nos queremos relacionar hacia nosotros. Es decir, un cliente puede alquilar varios cobertizos. Como un cobertizo no puede ser alquilado por más de un cliente a la vez, la relación queda así. El siguiente paso es crear las vistas formulario y árbol para visualizar los campos de nuestro nuevo objeto. Son vistas que no heredan de ninguna otra, con lo que hay que prestar atención a la cabecera del código xml. También conviene recordar que las vistas Form necesitan agrupar sus campos field dentro de algún group, no siendo así necesario en una vista Tree.

Nombre de la vista: x_boathouse.rent.form

Tipo de vista: Formulario

Modelo: x_boathouse.rent

Secuencia: 16

Activo: ☒

Campo hijo:

Vista heredada:

Ver modo heredado: Vista base

Modelo de datos: ID externo

Estructura

Permisos de acceso

Vistas heredadas

Editar traducciones

```
1 <?xml version='1.0'?>
2 <form string='Alquiler'>
3   <group>
4     <field name='x_cliente'/'>
5     <field name='x_identificador'/'>
6     <field name='x_metros_cuadrados'/'>
7     <field name='x_ocupado_libre'/'>
8     <field name='x_precio_dia'/'>
9   </group>
10 </form>
```

Nombre de la vista: x_boathouse.rent.tree

Tipo de vista: Árbol

Modelo: x_boathouse.rent

Secuencia: 16

Activo: ☒

Campo hijo:

Vista heredada:

Ver modo heredado: Vista base

Modelo de datos: ID externo

Estructura

Permisos de acceso

Vistas heredadas

Editar traducciones

```
1 <?xml version='1.0'?>
2 <tree string='Alquiler'>
3   <field name='x_cliente'/'>
4   <field name='x_identificador'/'>
5   <field name='x_metros_cuadrados'/'>
6   <field name='x_ocupado_libre'/'>
7   <field name='x_precio_dia'/'>
8 </tree>
```

Cuando las vistas estén listas toca aprender cómo acceder a ellas a partir de un componente menú.
¿Qué tenemos que tener en cuenta?

- La jerarquía: los menús se anidan unos dentro de otros. Y en caso de que un menú no tenga elemento padre, se visualiza directamente en la barra superior.
- Las acciones de ventana: son quienes enlazan un menú o submenú con el objeto cuyos vistas se mostrarán de inmediato. Es decir, debemos crear una acción para nuestro nuevo menú.
- Los permisos: por defecto, si no establecemos nada, nuestro menú/submenú estará a la vista de todos los usuarios. Basta con asignárselo a un grupo de usuarios para que automáticamente se restrinja la visibilidad al resto.

Lo que haremos será crear el siguiente menú: Espacios / Alquiler. Esto implica crear un menú llamado Espacios y otro en su interior Alquiler, que es el que asociaremos al objeto `x_boathouse.rent` y sus dos vistas form y tree. Para crear un menú nos vamos a “Configuración / Técnico / Interfaz de Usuario / Elementos menú” y pulsamos en “Crear”. Crearemos un menú “Espacios” que se pueda ver en la barra superior, es decir, sin elementos padre; y otro anidado en él llamado “Alquiler”.

El paso siguiente es crear nuestra acción de ventana para que Alquiler quede ligado al objeto `x_boathouse.rent` y a sus dos vistas. De hecho, hasta que no liguemos “Alquiler” a una acción de ventana, ni “Espacios” ni “Alquiler” serán visibles para nosotros.

Las vistas tree y form son detectadas automáticamente.

Por último sólo nos queda enlazar el submenú “Alquiler” a esta acción y comenzará a ser visible e interactuable. Es necesario que el tipo de acción sea *ir.actions.act_window*

Menú	Alquiler	Ruta completa	Espacios/Alquiler
Menú padre	Espacios	Acción	ir.actions.act_w Alquiler
Secuencia	10	Archivo icono web	

Permisos de acceso	Submenús
Nombre del grupo	
Añadir un elemento	

Al acceder por primera vez a “Alquiler” se nos habilita la opción de crear un primer objeto. Al hacerlo, si hemos configurado bien la relación, nos debería salir el único cliente creado como opción seleccionable.



Ahora, si consultamos la ficha de este cliente, no encontraremos ningún campo relacionado con el alquiler que tiene contratado. Para solucionarlo se ha de crear un campo extra en el objeto *res.partner* (clientes) de tipo relación, y crear una vista heredada (por ejemplo, de su formulario *res.partner.form*) para volverlo visible. Este objeto también tiene una vista fija asociada a su acción de ventana con lo que no nos queda otra que modificar ese valor a mano.

Nombre de la vista: *res.partner.form.rent*

Tipo de vista: Formulario

Modelo: *res.partner*

Secuencia: 0

Activo: ☒

Campo hijo: []

Vista heredada: *res.partner.form*

Ver modo heredado: Vista de extensión

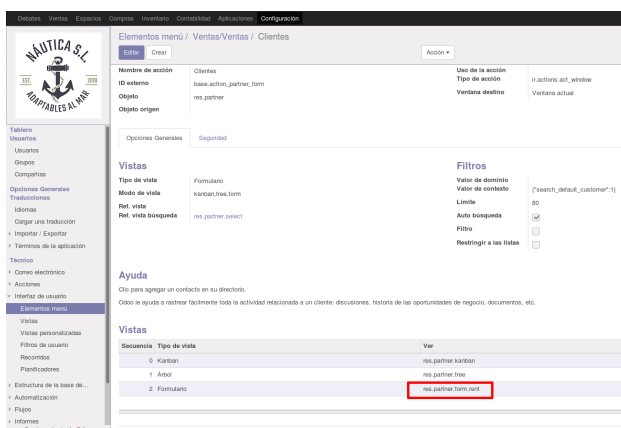
Modelo de datos: ID externo

Editar traducciones

```

1- <xpath expr="//page[@name='internal_notes']" position="before">
2- <page string="Alquiler">
3- <group>
4- <field name="x_alquiler"/>
5- </group>
6- </page>
7- </xpath>

```



4. Gestión de usuarios y grupos

Se ubica en “Configuración > Usuarios > Grupos”, siempre y cuando esté activo el modo desarrollador. Lo primero que se muestra al entrar en el submenú Grupos es una lista de todos ellos (serán más o menos dependiendo de cuántas aplicaciones hayamos instalado), y estructurados mediante categorías jerárquicas como sucede con los roles clásicos de las bases de datos. Así “Empleados / Empleado” es el grupo más sencillo al que va a parar por defecto cualquier usuario al que no le hayamos asignado ningún rol asociado a un módulo y permite el acceso al canal General de Debate.

Para comenzar a trabajar con grupos, lo primero que haremos será sumar tres usuarios nuevos además de nuestro Administrador. La intención es asociarlos a tres grupos distintos que a su vez guardan una relación de herencia entre sí en orden ascendente, es decir, un *usuario_normal* gozará de unos permisos de acceso mínimos a los módulos instalados; otro *usuario_gente* los heredará incorporando algún privilegio extra, y por último un *usuario_administrador* se situará por encima de los anteriores, en lo alto de la cadena.

Usuarios			Usuarios internos x Buscar...
<input type="button" value="Crear"/> <input type="button" value="Importar"/>			
<input type="checkbox"/>	Nombre	Usuario	Idioma
<input type="checkbox"/>	Administrator	admin@empresanautica.com	Spanish / Español
<input type="checkbox"/>	usuario_administrador	usuario_administrador@gmail.com	Spanish / Español
<input type="checkbox"/>	usuario_gente	usuario_gente@gmail.es	Spanish / Español
<input type="checkbox"/>	usuario_normal	usuario_normal@gmail.es	Spanish / Español

Es importante recordar dos cosas durante la creación de usuarios: debemos dotar de una contraseña a cualquier usuario recién creado ya que Odoo no regula este paso mediante ningún asistente. La segunda es que todos los usuario deben contar al menos con un permiso de acceso vigente (ya sea por parte de las aplicaciones instaladas, o de las categorías “Empleados” y “Administración” que vienen por defecto en toda base de datos). De no cumplirse lo último nos encontraremos siempre con un **Internal Server Error** al realizar el login, puesto que Odoo no sabe a qué módulo ni vista derivar el usuario conectado.

Una vez creados, regresamos a Grupos y pulsamos “Crear” para abrir la ventana que genera la configuración de un nuevo grupo. La pestaña activa de inicio es “Usuarios”. Aquí elegiremos el nombre del grupo y en el campo Aplicación podremos asociarle directamente una categoría padre, no obstante, esto no invoca permiso alguno sino que se realiza con vistas a organizar nuestros grupos por categorías/jerarquías. Por ejemplo, podría interesarnos crear una supercategoría llamada “Práctica” dentro de la cual anidásemos las otras tres de forma que los grupos se mostraran así: “Práctica / Grupo normal”, “Práctica / Grupo gerente”, “Práctica / Grupo administrador”. Pero “Práctica” como tal no tendría por sí sola ningún permiso inherente. Para la práctica, llamaremos al grupo “Grupo Normal”, y agregaremos en él a *usuario_normal*.

La segunda pestaña “Heredado” sí nos permitirá agregar o configurar los permisos de acceso. Puede realizarse de dos maneras: la más habitual mediante “paquetes” ya hechos que incorpora Odoo y que por normal general son descriptivos respecto de cada módulo instalado. Por ejemplo, si pulsamos sobre “añadir un elemento”, y en la ventana que se despliega escribimos “Ventas” en el campo de búsqueda obtendremos:

Añadir: Hereda

Grupo ventas x Buscar...

1-5 / 5

Nombre del grupo
<input type="checkbox"/> Configuración técnica / Contabilidad analítica para las ventas
<input type="checkbox"/> Configuración técnica / Mostrar incoterms en los Pedidos de Ventas y en facturas relacionadas
<input type="checkbox"/> Ventas / Responsable
<input checked="" type="checkbox"/> Ventas / Usuario: Solo mostrar documentos propios
<input type="checkbox"/> Ventas / Usuario: Todos los documentos

Seleccionar Crear Cancelar

En las tres últimas líneas se aprecia mejor cómo podemos seleccionar entre tres paquetes de permisos relacionados con el módulo Ventas, siendo Responsable el más completo. Nosotros escogeremos el más sencillo (recuadrado). La operación se puede repetir con los otros módulos instalados (Compras e Inventario), logrando al final:

Aplicación: Portal

Nombre:

Compartir grupo: ☐

Usuarios Heredado Menús Vistas Permisos de acceso Reglas Notas

Los usuarios agregados a este grupo se añaden automáticamente a los siguientes grupos.

Nombre del grupo
Inventario / Usuario
Compras / Usuario
Ventas / Usuario: Todos los documentos

Añadir un elemento

Siguiendo con las pestañas, “Menús” y “Vistas” habilitan permisos especiales sobre estos dos tipos de componentes de manera particular y no tienen mayor misterio. En cambio sí puede haber confusión entre las dos siguientes, “Permisos de acceso” y “Reglas”. La primera nos sirve para establecer libremente la modificación de un permiso en concreto (es decir, no un paquete de varios permisos) y que denominamos regla. La pestaña “Reglas”, en cambio, incorpora la carga de reglas ya creadas. En otras palabras, “Permisos de acceso” crea reglas desde cero, y “Reglas” sólo invoca aquellas ya existentes para que sean tenidas en cuenta.

Recapitulando: tras agregar *usuario_normal* al grupo “Grupo Normal”, el resultado es:

Debates Ventas Compras Inventario Aplicaciones

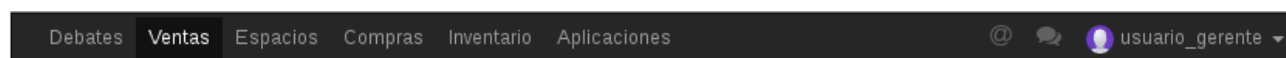
usuario_normal

Debates y Aplicaciones le vienen de “Empleado” aunque sobre Aplicaciones no cuenta con ningún derecho; el resto de módulos visibles procede de los paquetes que hemos añadido a mano. Pero por ejemplo, este usuario no contaría con acceso al menú “Espacios” creado por nosotros donde se alquilan embarcaderos, ni tampoco al menú de “Configuración”.

Ojo, si en cualquier momento se nos ocurre retirar a un usuario de un grupo activo, hemos de saber que Odoo no le retira por ello los permisos comunes a dicho grupo; el usuario continúa en posesión de estos como si le hubieran pertenecido originalmente, es decir, el ERP no tiene forma de comprobar cómo llegaron esos permisos a manos del usuario y por tanto los deja tal y como están.

Para el segundo grupo, “Grupo Gerente”, heredamos del grupo anterior en la pestaña “Heredado”. Para poder visualizar el menú “Espacios” creamos una regla sobre el objeto `x_boat_rent`, es decir, aquel que contiene los campos del submenú Alquiler y en el que se apoyan las vistas de “Espacios > Alquiler”.

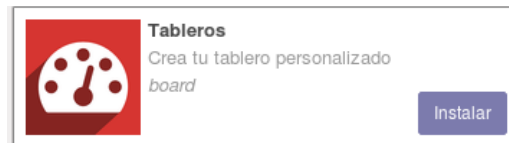
El resultado final para `usuario_gerente` debería ser éste:



Al último usuario le añadiremos los permisos de configuración para un control total sobre Odoo. Además de ello hereda también del grupo Gerente los permisos del Grupo Normal.

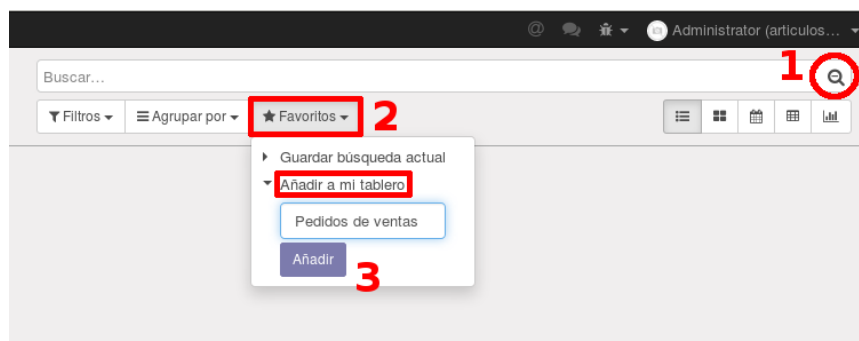
5. Tableros

Es la herramienta para confeccionar un menú rápido de vistas ya creadas, de modo que no tengamos que pasar antes por las aplicaciones en las que están anidadas por norma. De hecho el tablero puede abarcar una visión plural y simultánea. Para habilitar esta opción puede instalarse la aplicación relacionada, del mismo nombre:

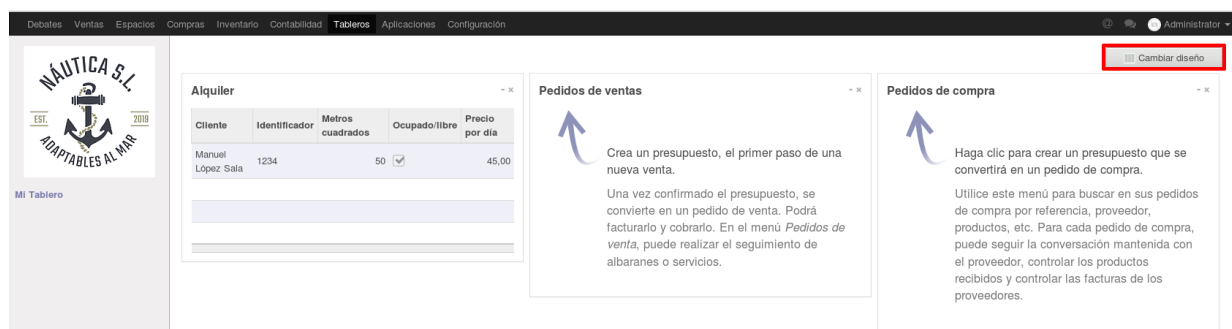


A continuación debemos activar el modo búsqueda avanzada, ya que muestra una botonera de opciones asociadas entre las que está incluir nuestra búsqueda como elemento de un tablero. Por ejemplo, si queremos incrustar la vista “Ventas > Pedidos de Ventas”, una vez desplegada esta vista clickamos sobre la lupa de búsqueda avanza y seleccionamos Favoritos donde ya se activa la adición al tablero. El procedimiento es siempre el mismo: localizar la vista que queremos incluir y buscar el botón Favoritos.

Es importante tener en cuenta que los tableros van ligados a cada sesión en particular, de modo que el tablero personalizado bajo la cuenta de un usuario será diferente al de otro empleado distinto.

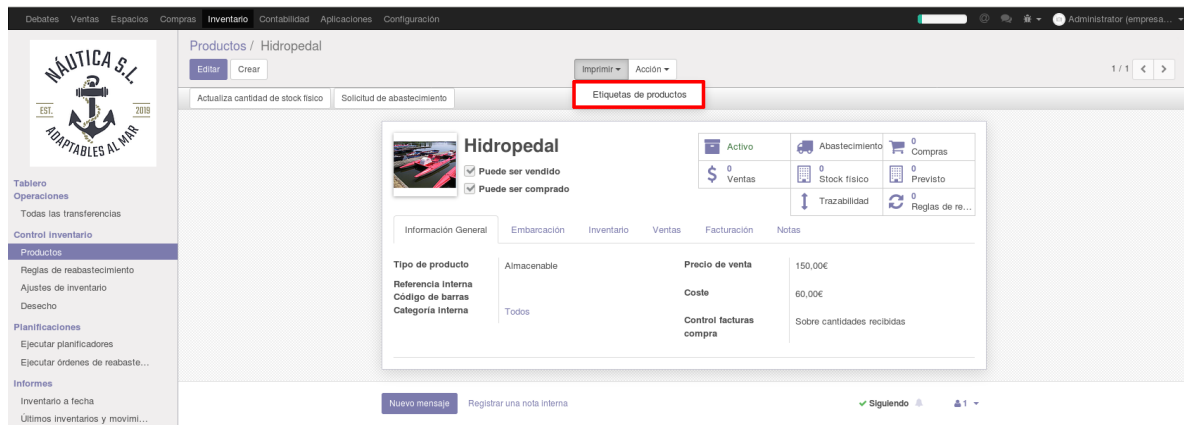


En cuanto regresemos al tablero, podremos configurar también la disposición visual seleccionando la única opción disponible en el vértice superior derecho. En este ejemplo se ve una vista repartida en tres columnas.



6. Informes

Los informes son las impresiones que hacemos de una vista en un formato documental que pueda intercambiarse o moverse a otro dispositivo. El ejemplo más ilustrado es el de los documentos PDF. Siguiendo con la tónica general de la práctica es razonable presuponer que dichos informes necesitarán de un objeto respaldando la información y también que la vista implicará un código detrás para la disposición de los elementos y la presentación. Si echamos un vistazo al producto que habíamos creado, podremos encontrar una opción de generación de informe en el menú Imprimir.



En cambio, si quisiéramos encontrar un informe asociado al submenú “Espacios > Alquiler” no existiría tal opción. Tanto en un caso como en otro crearemos dos informes de cero. En el primero, simplemente se añadirá como una opción más, y en el segundo será la única disponible.

Para visualizar o crear un informe hemos de ir a “Configuración > Técnico > Informes > Informes”. Si curioseamos en alguno de los pocos informes que se muestran, estos no hacen más que ligar un objeto existente con una vista de tipo QWeb que es realmente la que presenta los elementos en pantalla. Por tanto, se podría decir que un informe consta de dos partes: el elemento de relación entre el objeto y la vista QWeb, y la propia vista QWeb analizada al detalle. Si ahora pulsamos en “Crear” iremos a parar a la ventana de generación de informe.

Nombre: Informe de embarcación

Tipo de informe: HTML

Modelo: product.template

Nombre de la plantilla: embarcacion.informe

Nombre del reporte impreso:

Seguridad | Propiedades avanzadas

Nombre del grupo:

Añadir un elemento

El campo nombre es aquel que se desplegará en Odoo en cuanto pulsemos sobre el botón Imprimir. Tipo de informe admite varios tipos, siendo los habituales PDF y HTML, este último el que escogeremos para poder hacer pruebas rápidas sin tener que generar un PDF por cada vez. Modelo es el objeto que nos suministrará los campos (recordamos que embarcación era un producto con nuevos campos añadidos en una pestaña aparte en la vista formulario heredada). Nombre de la plantilla es la vista Qweb que se ocupa del apartado gráfico. Como todavía no la hemos creado podemos, o bien dejarlo en blanco, o bien adelantar el nombre que tendrá, en este caso “embarcacion.informe”. En la botonera de la esquina superior derecha tenemos la opción de hacer visible ya la generación del informe mediante el menú imprimir y de consultar la vista QWeb asociada. Tanto una cosa como la otra están en construcción. Empezamos por la segunda: al clicar sobre QWeb y no existir vista alguna, vamos a parar a la creación de una vista.

Nombre de la vista: **embarcacion.informe**

Tipo de vista: QWeb

Modelo: product.template

Secuencia: 16

Activo: ☒

Campo hijo:

Vista heredada:

Ver modo heredado: Vista base

Modelo de datos:

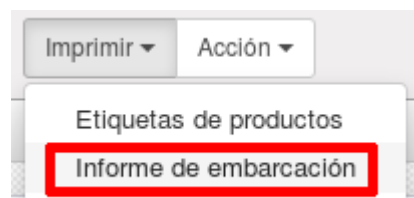
ID externo:

Estructura | Permisos de acceso | Vistas heredadas

Editar traducciones

```
1 <?xml version="1.0"?>
2 <t t-name="embarcacion.informe.titulo">
3
4 </t>
```

A la hora de crear nuestra vista QWeb el aspecto más delicado es el del nombre de la vista. Si habíamos dejado en blanco el campo “Plantilla” durante la creación del informe no hay problema, pero si habíamos escrito algo, ahora debe coincidir aquí. También hemos cambiado el Tipo de vista a QWeb tras haber visto ya Form, Kanban y Tree. El código xml de la estructura se genera en blanco de modo que hemos añadido la cabecera mínima y un título para el documento. Hecho esto, si ahora regresáramos a la ventana de creación de Informe y activásemos “Añadir al menú imprimir”, como mínimo deberíamos poder contemplar una segunda opción de informe en la ficha de nuestro producto:



Sin embargo, de ejecutar el informe obtendríamos un error (y no porque el código xml continúe estando vacío). Con error nos referimos a una particularidad de las interfaces de usuario, las reglas definidas por nosotros o los propios informes: es el ID externo. Resulta que nuestras creaciones y modificaciones en una base de datos van siendo almacenadas en está como un registro más, y que al momento de cargarse juegan en desventaja ya que los datos de la base se cargan en último lugar. Esto tiene especial importancia a la hora de encontrar elementos ya que Odoo necesita asegurarse de que algo está creado o no en el momento en que otro elemento hace referencia o apunta a ello.

Para crear un identificador externo, vamos a la siguiente categoría en “Configuración > Técnico”, es decir, “Secuencias e Identificadores > Identificadores externos”.

Dentro del primer recuadro es obligatorio especificar un módulo y un ID externo, de tal modo que ambos actúen como un ID completo e inconfundible. Es fácil dividir el nombre de la vista para obtener ambos. En cuanto al modelo a mostrar y el valor numérico del ID de registro se obtienen como atributos de la URL en la ventana donde definimos la vista QWeb:

Al guardar el ID externo, automáticamente el campo Registro debería actualizarse. Ahora sí, el error al visualizar el formulario que hemos creado debería haber desaparecido. Ya podemos realizar una prueba sencilla con el código xml de la vista QWeb para observar los datos en pantalla de nuestra embarcación.

Para sacar a informe el espacio alquilado por un usuario seguiríamos el proceso análogo: crear un informe que lleva asociada una vista Qweb, aún por definir. A la hora de establecer esta vista generar un identificador externo cuyos parámetros van implícitos en la URL de la ventana durante la creación de la propia vista. Modificar el código xml asociado al objeto creado para los alquileres.

espacios.alquiler.informe

Módulo	espacios	Nombre a mostrar	espacios.alquiler.informe
Identificador externo	alquiler.informe	Nombre del modelo	ir.ui.view
No actualizable	<input type="checkbox"/>	ID de registro	724
Fecha de actualización	30/01/2019 19:24:40	Registro	espacios.alquiler.informe
Fecha inicial	30/01/2019 19:24:40		

A la hora de establecer la condición del checkbox Ocupado/libre, cuidado con las dobles comillas para englobar toda la sentencia como valor del atributo.

Estructura
Permisos de acceso
Vistas heredadas

[Editar traducciones](#)

```

1 <?xml version="1.0"?>
2 <t t-name="espacios.alquiler.informe">
3   <div>
4     <t t-set="alquiler" t-value="docs"/>
5     <h2>Datos del alquiler</h2>
6     <p><strong>Nombre: </strong><span t-field="alquiler.x_cliente"/></p>
7     <p><strong>Identificador: </strong><span t-field="alquiler.x_identificador"/></p>
8     <p><strong>Metros cuadrados: </strong><span t-field="alquiler.x_metros_cuadrados"/></p>
9     <p><strong>Ocupado/Libre: </strong>
10       <span t-if="alquiler.x_ocupado_libre==true">Ocupado</span>
11       <span t-if="alquiler.x_ocupado_libre==false">Libre</span>
12     </p>
13     <p><strong>Precio por día: </strong><span t-field="alquiler.x_precio_dia"/></p>
14   </div>
15 </t>
16

```