

## Pràctica 4. Estructura i creació d'un mòdul

---

A Odoo podem treballar amb mòduls o aplicacions, però que son?

Els **mòduls** són blocs per a la construcció de les aplicacions en Odoo. Un mòdul pot afegir o modificar característiques en Odoo. De vegades, els mòduls poden ser anomenats "add-ons" (complements).

Les **aplicacions** no són diferents dels mòduls regulars, però funcionalment, aquestes proporcionen una característica central, al voltant de la qual altres mòduls afegeixen característiques o opcions. Aquestes proveeixen els elements base per a una àrea funcional, com comptabilitat o RRHH, sobre les quals altres mòduls afegeixen característiques. Per això són ressaltades al menú Apps de Odoo.

En el nostre cas, anem a crear un mòdul simple per veure la seva estructura i començar a entendre com hem de treballar-los:

### Crear un mòdul

En primer lloc, modifiquem la ruta del fitxer de configuració per indicar on podrà trobar els nostres "addons". Heu de cercar el directori de la instal·lació d'Odoo, en el meu cas es troba a `/etc/odoo/odoo.conf`. Podeu executar la comanda **service odoo status** per esbrinar la seva ubicació.

En aquest directori hi ha un fitxer de configuració `odoo.conf`, en el que afegirem al final la següent línia per indicar on es trobaran els nostre mòduls (Heu de crear les carpetes al vostre directori personal):

```
addons_path = /usr/lib/python3/dist-packages/odoo/addons,/home/el_teu_usuari/odoo-dev-13/addons
```

#### Exemple:

```
addons_path = /usr/lib/python3/dist-packages/odoo/addons,/home/dmorenoar/odoo-dev-13/addons
```

I a continuació reiniciem el servei:

```
sudo /etc/init.d/odoo restart
```

La **estructura mínima d'un mòdul** es la següent, en aquest cas, estem creant el mòdul "Todo App":

```
root@dmorenoar-VirtualBox: /home/dmorenoar/odoo-dev/addons/todo_app# tree -L 2
.
├── __init__.py
├── __manifest__.py
├── models
│   ├── __init__.py
│   └── todo_app.py
└── views
    └── template.xml
```

Si no volem crear l'estructura, Odoo ofereix un mecanisme per ajudar a començar un nou mòdul, la comanda següents ens permet crear un mòdul, en aquest cas ho crearem a la nostra carpeta personal:

```
> odoo scaffold /home/dmorenoar/odoo-dev/addons/modulAutomatic
```

```
root@dmorenoar-VirtualBox: /home/dmorenoar/odoo-dev/addons/modul_automatic# tree -L 2
.
├── controllers
│   ├── controllers.py
│   └── __init__.py
├── demo
│   └── demo.xml
├── __init__.py
├── __manifest__.py
├── models
│   ├── __init__.py
│   └── models.py
├── security
│   └── ir.model.access.csv
└── views
    ├── templates.xml
    └── views.xml
```

Hem de tenir present que els mòduls a Odoo treballen fent servir l'arquitectura MVC (Model-Vista-Controlador).

Aquesta arquitectura es defineix per:

- El **model (M)**, defineix l'estructura de les dades.
- La **vista (V)**, descriu la interfície amb l'usuari o la usuària.
- El **controlador (C)**, suporta la lògica de negoci de l'aplicació.

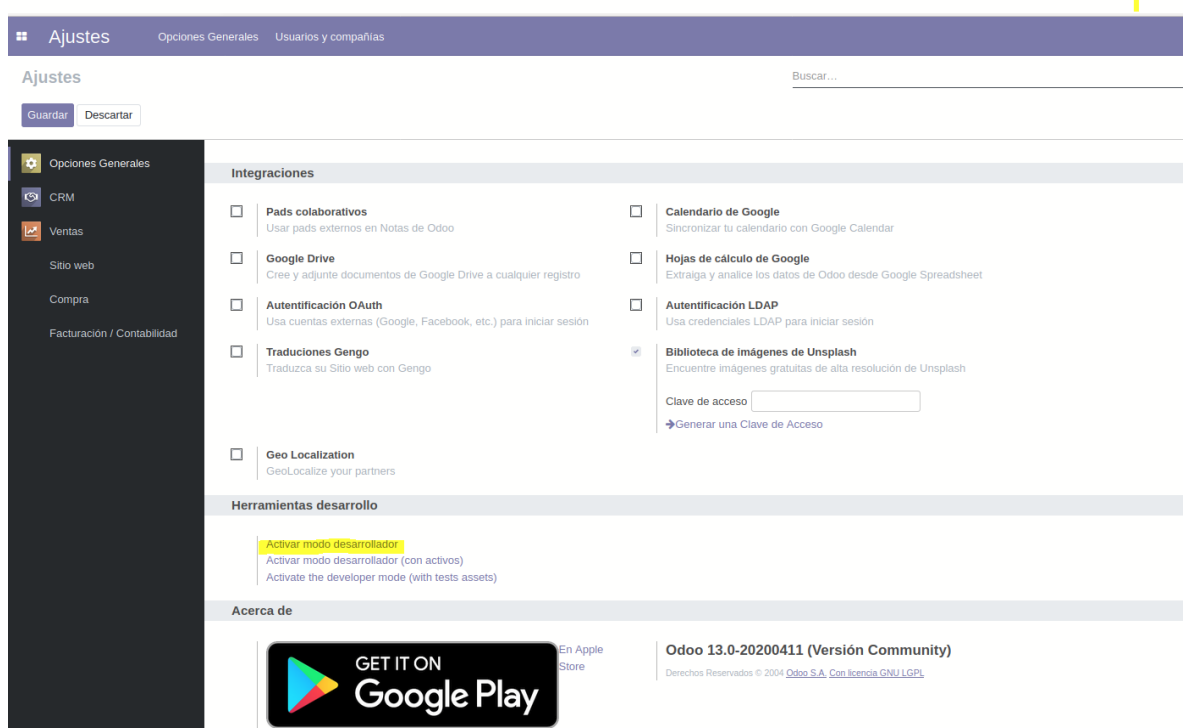
Presentada l'arquitectura, s'identifica cada part de l'estructura.

- **\_\_manifest\_\_.py**: Trobarem un diccionari amb les dades principals del nostre mòdul.
- **\_\_init\_\_.py**: Importem tots els mòdels python que necessita el mòdul.
- **models**: Declarem tots els models a nivel python que seran representats a la base de dades. El mapatge de la base de dades és gestionat automàticament per Odoo, i el mecanisme responsable per això és el model objecte relacional, (ORM - object relational model).

- **views:** Incluirem totes les vistes per al nostre mòdul. Les vistes són definides usant XML, les quals són usades pel marc de treball (framework) de el client web per generar vistes HTML de dades.
- **controllers:** Els controllers interpreten les peticions del navegador i envia dades de tornada.
- **security:** Definirem les regles de seguretat del nostre mòdul.
- **demo:** Conté contingut de demostració per al model.

## Afegir el mòdul

A Odoo hem d'activar el mode de desenvolupador per començar a fer la màgia:



Seguint la estructura que ens ha definit la comanda scaffold podem anar al llistat d'aplicacions i actualitzar la llista per veure que ens ha identificat el nostre mòdul:

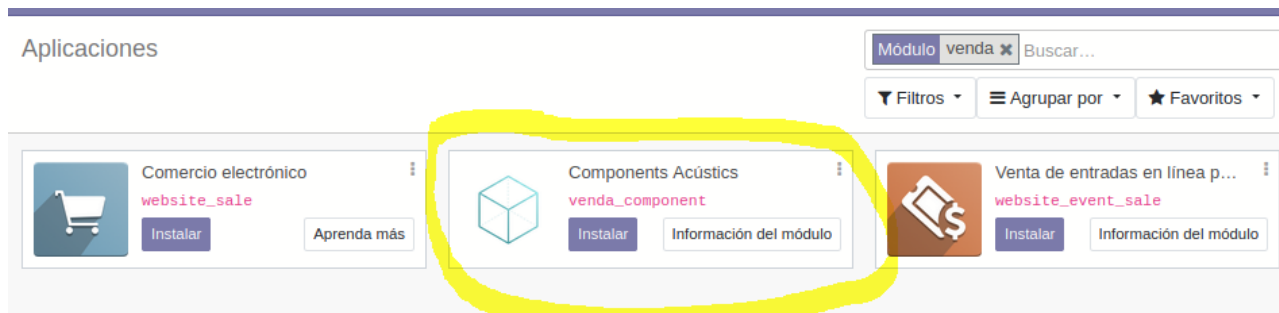


Aquest procés és el que hem de repetir quan dissenyem el nostre mòdul i el volem instal·lar. Així que ara ens posarem a crear un mòdul per tal d'entendre tot el seu funcionament.

Seguint la següent estructura farem un mòdul per a la venda d'un component de cotxe (Podeu escollir el producte que vulgueu) :

```
root@unassigned-hostname:/home/alumne/odoo-dev/addons/venda_component# tree -L 2
.
├── controllers
│   ├── controllers.py
│   └── __init__.py
├── __init__.py
├── __manifest__.py
├── models
│   ├── __init__.py
│   └── venda_component.py
└── views
    ├── templates.xml
    └── views.xml
```

El contingut de cada fitxer s'ha d'adaptar a les nostres necessitats. Pot seguir la [documentació](#) per omplir els fitxers. Podeu consultar que aneu per bon camí després de configurar el fitxer `__init__.py` i `__manifest__.py`. Si heu configurat correctament els fitxers, actualitzeu la llista i podreu veure el mòdul.



**Observació:** Si intentem instal·lar el nostre mòdul ens sortirà un error ja que no hem definit el

model ni la vista.

Anem a continuar amb la [documentació](#) per crear el model i la vista. Per crear el model farem la vinculació del mapeig d'Objecte-Relacional (Object-Relational Mapping). Per conèixer els atributs que podem representar al model podeu consultar els apunts o la documentació.

L'únic **atribut obligatori** es `_name`.

Encara que s'acostuma a utilitzar alguns comuns:

**Atributs estructurals**, defineixen el comportament:

- **`_rec_name`** — marca el nom utilitzat per les vistes per referenciar el model (records).
- **`_order`** — permet marcar en funció de quins atributs s'ordenarà el model.
- **`_description`** — permet marcar una descripció pel mòdul.
- **`_inherit`** — crea una herència de la classe referenciada.
- **`_inherits`** — permet fixar quins atributs i/o mètodes s'hereten de cada classe
- **`_sql_constraints`** — permet fixar les restriccions de SQL
- **`_constraints`** — permet fixar les restriccions de l'objecte

**Atributs simples**, permeten guardar valors concrets:

- **`char`** — Permet emmagatzemar una cadena de text.
- **`text`** — Permet emmagatzemar una cadena de text que inclogui múltiples línies.
- **`selection`** — Permet emmagatzemar les dades per un menú desplegable.
- **`integer`** — Permet emmagatzemar un enter
- **`float`** — Permet emmagatzemar un real
- **`date/datetime`** — Permet emmagatzemar una data / data incorporant hora.
- **`boolean`** — Permet emmagatzemar un valor lògic.
- **`string`** — serà el títol del camps (nom que veurem en la UI).
- **`default`** — valor per defecte que volem donar a aquest camp.

**Atributs reservats**, generats automàticament per Odoo:

- **`id`** — número únic que identifica cada entrada.
- **`create_uid`** — usuari que ha creat la entrada
- **`create_date`** — data i hora de creació de l'entrada.
- **`write_uid`** — últim usuari que ha modificat l'entrada
- **`write_date`** — data i hora de l'última modificació.

Es pot consultat tots els atributs a la [documentació](#).

A continuació, pots veure un exemple de l'estructura del model per la venda d'un component amb diferents tipus d'atributs:

```
venda_component.py x  __init__.py — venda_component x  __init__.py — venda_
4
5  COLOR = [
6      ('negre', 'Negre'),
7      ('plata', 'Plata'),
8      ('gris', 'Gris'),
9  ]
10
11  class VendaComponent(models.Model):
12      _name = 'venda.component'
13      _description = 'Gestió de la venda de components'
14      name = fields.CharField(string='Nom del component', required=True)
15      marca = fields.CharField(string='Marca', required=True)
16      color = fields.Selection(COLOR, default='negre')
17      is_offer = fields.BooleanField('Té descompte?')
18
```

```
__init__.py — ver  __init__.py — venda_component/models x
# -*- coding: utf-8 -*-

from . import venda_component
```

I també de la vista associada:

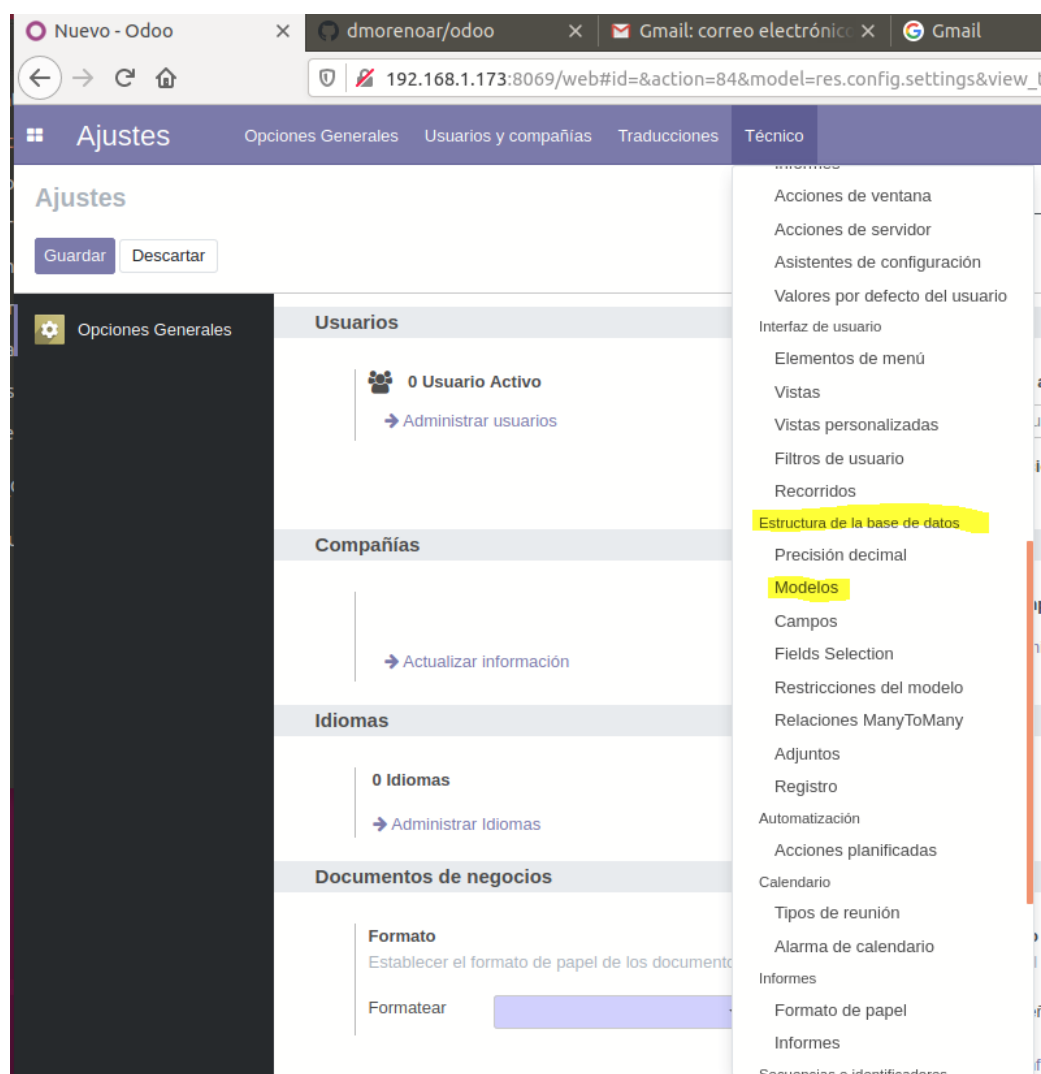
```
venda_component.py x templates.xml x untitled
1 <?xml version="1.0" encoding="utf-8" ?>
2 <odoo>
3 <!-- FORM VIEW - Vista del formulari per afegir els registres -->
4 <record id="venda_component view_form" model="ir.ui.view">
5   <field name="name">venda.component.view.form</field>
6   <field name="model">venda.component</field>
7   <field name="type">form</field>
8   <field name="arch" type="xml">
9     <search>
10       <field name="name"></field>
11     </search>
12     <form string="Formulari de venda">
13       <group colspan="2" col="2">
14         <field name="name"></field>
15         <field name="marca"></field>
16         <field name="color"></field>
17         <field name="is_offer"></field>
18       </group>
19     </form>
20   </field>
21 </record>
22 <!-- TREE VIEW Llistat del registres -->
23 <record id="venda_component view_tree" model="ir.ui.view">
24   <field name="name">venda.component.view.tree</field>
25   <field name="model">venda.component</field>
26   <field name="arch" type="xml">
27     <tree string="Tree Component">
28       <field name="name"></field>
29       <field name="marca"></field>
30       <field name="color"></field>
31       <field name="is_offer"></field>
32     </tree>
33   </field>
34 </record>
35 <!-- ACTION Accions que pot fer el mòdul-->
36 <record id="venda_component action" model="ir.actions.act_window">
37   <field name="name">venda.component.action</field>
38   <field name="type">ir.actions.act_window</field>
39   <field name="res_model">venda.component</field>
40   <field name="view_mode">tree,form</field>
41 </record>
42 <!-- MENU Defineix un element de menu a la part superior-->
43 <menuitem id="menu_root" name="Components" sequence="1"></menuitem>
44 <menuitem id="sub_menu" name="Acústics" sequence="1" parent="menu_root" action="venda_component_action"></menuitem>
45
46 </odoo>
47
```

Pots consultar el projecte base a l'enllaç de github:

<https://github.com/dmorenoar/odoo.git>

Una vegada hagis creat el mòdul prova a realitzar la seva instal·lació o actualització.

Podem consultar l'estructura del model de la base de dades i els atributs que hem configurat a **Ajustes -> Técnico -> Modelos**



**Ajustes**

Guardar Descartar

**Opciones Generales**

**Usuarios**

0 Usuario Activo  
→ Administrar usuarios

**Compañías**

→ Actualizar información

**Idiomas**

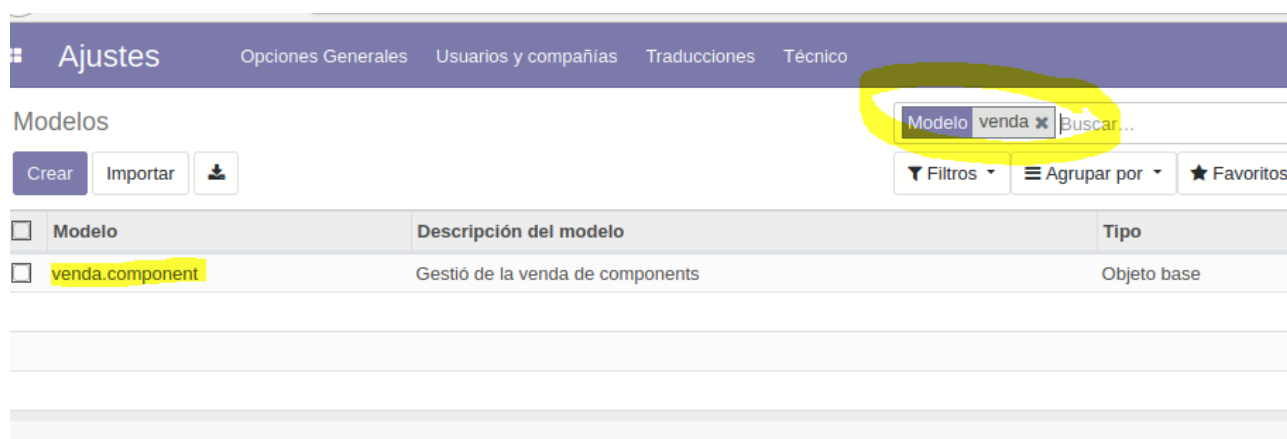
0 Idiomas  
→ Administrar Idiomas

**Documentos de negocios**

**Formato**  
Establecer el formato de papel de los documentos  
Formatear

**Técnico**

- Acciones de ventana
- Acciones de servidor
- Asistentes de configuración
- Valores por defecto del usuario
- Interfaz de usuario
- Elementos de menú
- Vistas
- Vistas personalizadas
- Filtros de usuario
- Recorridos
- Estructura de la base de datos**
- Precisión decimal
- Modelos**
- Campos
- Fields Selection
- Restricciones del modelo
- Relaciones ManyToMany
- Adjuntos
- Registro
- Automatización
- Acciones planificadas
- Calendario
- Tipos de reunión
- Alarma de calendario
- Informes
- Formato de papel
- Informes
- Secuencias e identificadores



**Ajustes**

Opciones Generales Usuarios y compañías Traducciones Técnico

**Modelos**

Crear Importar

Modelo venta Buscar...

Filtros Agrupar por Favoritos

Modelo	Descripción del modelo	Tipo
venta.component	Gestió de la venda de components	Objeto base





Ajustes

Opciones GeneralesUsuarios y compañíasTraduccionesTécnico

Modelos / Gestió de la venda de components

EditarCrear

ImprimirAcción

Descripción del modelo

Modelo

Modelo transitorio

Hilo de mensajes

Actividad de correo

Lista negra de correo

Gestió de la venda de components

venda.component

☐

☐

☐

☐

Tipo

En las aplicaciones

Objeto base

venda\_component

Campos

Permisos de accesoReglas de registroNotasVistas

Nombre de campo	Etiqueta de campo	Tipo de campo	Requerido	Sólo lectura	Index
__last_update	Last Modified on	Fecha y hora	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
color	Color	Selección	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
create_date	Created on	Fecha y hora	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
create_uid	Created by	many2one	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
display_name	Display Name	Carácter	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
id	ID	entero	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
is_offer	Té descompte?	booleano	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
marca	Marca	Carácter	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
name	Nom del component	Carácter	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
write_date	Last Updated on	Fecha y hora	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
write_uid	Last Updated by	many2one	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

## Suport:

- [Documentació creació d'un mòdul a d'Odo.](#)
- [Manual de Python.](#)
- [Capítulo 2 - Primera aplicación](#)