UT5 DESARROLLO DE MÓDULOS EN ODOO
PARTE 1

CONCEPTOS BÁSICOS

Tabla de contenido

Actualización: 24 de noviembre de 2025

1. TU PRIMER MÓDULO EN ODOO

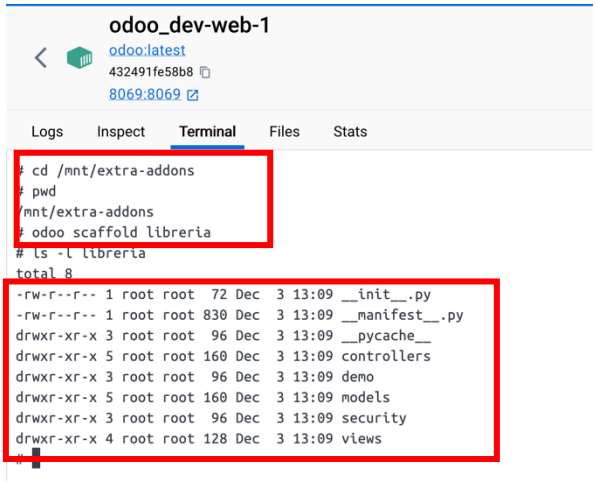
Odoo nos permite programar con paquetes. Odoo se basa en un **framework** que se llaman **OpenObject** que permite programar aplicaciones de tipo RAD (Rapid Application Development) sobre una capa ORM (Object Relational Mapping o Mapeo Objeto-Relacional).

ORM es una herramienta que nos permite mapear, o lo que es lo mismo, **convertir los objetos de tu aplicación a un formato adecuado para ser almacenados en cualquier base de datos**.

Odoo usa una arquitectura Modelo, vista, controlador (MVC) ya que estas partes las tiene muy bien diferenciadas. Define estas 3 partes claramente.

Los **modelos son las clases que se traducirán en tablas**. Las vistas serán archivos xml y los **controladores permiten manipular los modelos**.

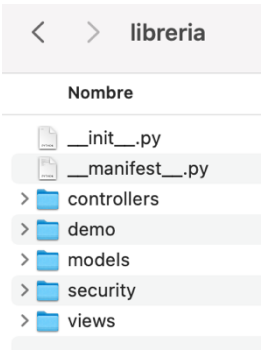
Vamos al lío! Vamos a crear nuestro primer módulo. Dentro del contenedor de Odoo abrimos el terminal y ejecutamos el comando siguiente:



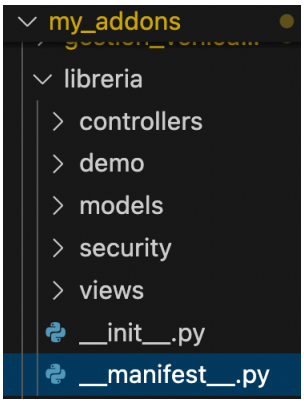
El comando `odoo scaffold libreria` es una herramienta que proporciona Odoo para generar la estructura básica de un módulo personalizado.

Con `scaffold`, Odoo crea automáticamente la estructura y archivos iniciales necesarios para desarrollar un nuevo módulo, en este caso, llamado librería.

Este comando es especialmente útil para desarrolladores que deseen crear un módulo desde cero sin tener que configurar manualmente la estructura y los archivos básicos.



Esto generará un nuevo módulo llamado `libreria` en el directorio `/mnt/extra-addons` que es el directorio que tenemos mapeado en nuestro directorio local `my_addons`



En este momento tendremos en nuestro directorio de desarrollo local la estructura de un módulo en Odoo (Python).

Y en nuestro editor de código (yo uso Visual Studio Code) tendremos la estructura creada para el nuevo módulo.

2. ARCHIVOS Y CARPETAS DE TU PRIMER MÓDULO

Cuando ejecutas el comando `odoo scaffold nombreMódulo`, Odoo genera la siguiente estructura básica de carpetas y archivos en la carpeta `libreria`:

```
libreria/
├── __init__.py
├── __manifest__.py
├── controllers/
│   └── __init__.py
├── models/
│   └── __init__.py
├── views/
│   └── views.xml
└── security/
    └── ir.model.access.csv
```

Explicación de cada archivo y carpeta

`__init__.py`: Este archivo **permite que Python reconozca esta carpeta como un paquete**. Es obligatorio en cada directorio que contiene código Python dentro de un módulo de Odoo. Su **función** principal es **importar las submódulos** de `models` y `controllers`.

`__manifest__.py`: Este archivo es el descriptor del módulo. Contiene metadatos del módulo, como su nombre, versión, dependencias, y otras configuraciones importantes. Odoo usa este archivo para saber cómo instalar y cargar el módulo. Algunos elementos clave en el archivo `__manifest__.py`:

- `name`: Nombre legible del módulo.
- `description`: Descripción del módulo. Las 3 comillas sirven para poder añadir varias líneas.

- `depends`: Lista de módulos de los que depende este módulo.
- `data`: Lista de archivos de datos XML que se cargarán al instalar el módulo.

`controllers/`: Esta carpeta contiene los controladores de Python para el módulo, utilizados para manejar las rutas y la lógica de la interfaz web. Inicialmente solo contiene `__init__.py`, ya que los controladores suelen crearse una vez que es necesario implementar una interfaz o API personalizada.

`models/`: Aquí se define la lógica de negocio y las estructuras de datos del módulo. Los modelos en Odoo son clases de Python que representan tablas en la base de datos. El archivo `__init__.py` se usa para cargar las clases de modelos que se crearán.

`views/`: Contiene archivos XML que definen la interfaz de usuario para los modelos del módulo. El archivo `views.xml` es un ejemplo de archivo de vista y suele contener formularios, listas y otros elementos de interfaz.

`security/`: Contiene configuraciones de seguridad, como permisos de acceso para los modelos del módulo. Odoo crea automáticamente un archivo `ir.model.access.csv` que es un archivo de control de acceso de modelo, donde puedes definir qué grupos de usuarios tienen permisos de lectura, escritura, creación y eliminación en cada modelo.

3. MODIFICACIÓN DE LA ESTRUCTURA BASE

¿Qué tenemos que cambiar?

`__manifest__.py`

```
# -*- coding: utf-8 -*-
{
    'name': "Gestión de Libreria Bosco",

    'summary': "Aplicación de gestión de librería",

    'description': """
Aplicación de gestión de librería para el instituto Juan Bosco de Alcázar de San Juan
    """,

    'author': "Fernando del Fresno",
    'website': "https://www.miwebsituviera.com",

    # Categories can be used to filter modules in modules listing
    # Check https://github.com/odoo/odoo/blob/15.0/odoo/addons/base/data/ir_module_category_data.xml
    # for the full list
    'category': 'Uncategorized',
    'version': '0.1',

    # any module necessary for this one to work correctly
    'depends': ['base'],

    # always loaded
    'data': [
        'security/ir.model.access.csv',     #esta linea temenos que descomentarla
        'views/views.xml',
        'views/templates.xml',
```

```
    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
}
```

models/models.py

```
# -*- coding: utf-8 -*-

from odoo import models, fields, api


class libro(models.Model):
     _name = 'libreria.libro'
     _description = 'libreria.libro'

     name = fields.Char()
     numPaginas = fields.Integer()
#      value2 = fields.Float(compute="_value_pc", store=True)
#      description = fields.Text(string='Descripción del libro', required=True)
#
#      @api.depends('value')
#      def _value_pc(self):
#          for record in self:
#              record.value2 = float(record.value) / 100
```

security/ir.model.access.csv

Controles de acceso al menú. Modificamos el id, el name y el model_id

views.xml | __manifest__.py .../libreria | models.py .../libreria/... 1 | ir.model.access

my_addons > libreria > security > ir.model.access.csv

```
1  id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2  access_libreria_libro,libreria.libro,model_libreria_libro,base.group_user,1,1,1,1
3
```

views/views.xml

```
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="libreria.libro_list">
      <field name="name">Libros list</field>
      <field name="model">libreria.libro</field>
      <field name="arch" type="xml">
        <list>
          <field name="name"/>
          <field name="numPaginas"/>
        </list>
      </field>
    </record>

    <!-- actions opening views on models -->
    <record model="ir.actions.act_window" id="libreria.libro_action_window">
      <field name="name">Listado de Libros </field>
      <field name="res_model">libreria.libro</field>
      <field name="view_mode">list,form</field>
    </record>
```

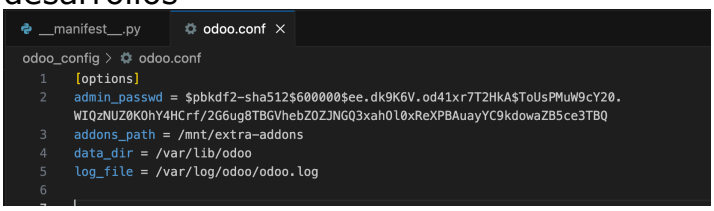
```
    <!-- server action to the one above -->
<!--
    <record model="ir.actions.server" id="libreria.action_server">
      <field name="name">libreria server</field>
      <field name="model_id" ref="model_libreria_libreria"/>
      <field name="state">code</field>
      <field name="code">
        action = {
          "type": "ir.actions.act_window",
          "view_mode": "list,form",
          "res_model": model._name,
        }
      </field>
    </record>
-->

    <!-- Top menu item -->
    <menuitem name="Gestión de Libreria" id="libreria.menu_root"/>
    <!-- menu categories -->
    <menuitem name="Gestión" id="libreria.menu_1" parent="libreria.menu_root"/>
    <!-- actions -->
    <menuitem name="Libros" id="libreria.menu_1_list" parent="libreria.menu_1"
              action="libreria.libro_action_window"/>
  </data>
</odoo>
```

4. AÑADIR EL MÓDULO DESARROLLADO A LA LISTA DE APLICACIONES DE ODOO

Para añadir el módulo a la lista de aplicaciones del Dashboard de Odoo debemos asegurarnos de lo siguiente:

- Que tenemos el módulo en la carpeta `my_addons` o donde mapeamos en el archivo docker-compose.yml la carpeta /mnt/extra-addons
- Debemos asegurarnos que el archivo de configuración de Odoo tiene indicado donde almacenaremos nuestros propios desarrollos



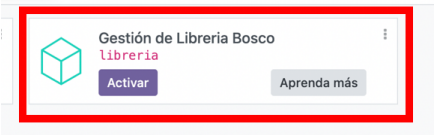
- Tenemos activado el modo desarrollador.

Actualizar la lista de aplicaciones de Odoo

1. En el menú de la izquierda, ve a **Aplicaciones**.
2. En la esquina superior derecha, haz clic en **Actualizar lista de aplicaciones**.
3. Después de actualizar, Odoo debería reconocer tu módulo personalizado en la lista de aplicaciones.

Los nombres que aparecen pertenecen:

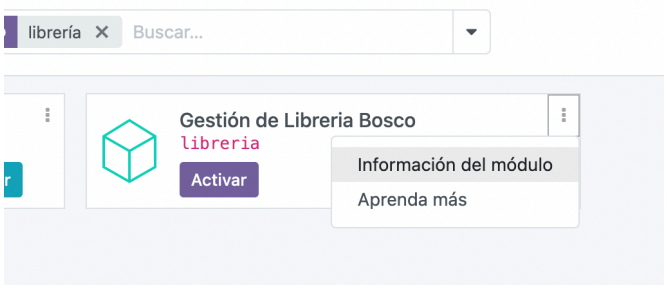
- Gestión de Librería Bosco. Campo 'name' en el archivo manifest.py
- Librería. Nombre de la carpeta donde está el módulo en nuestro directorio local.

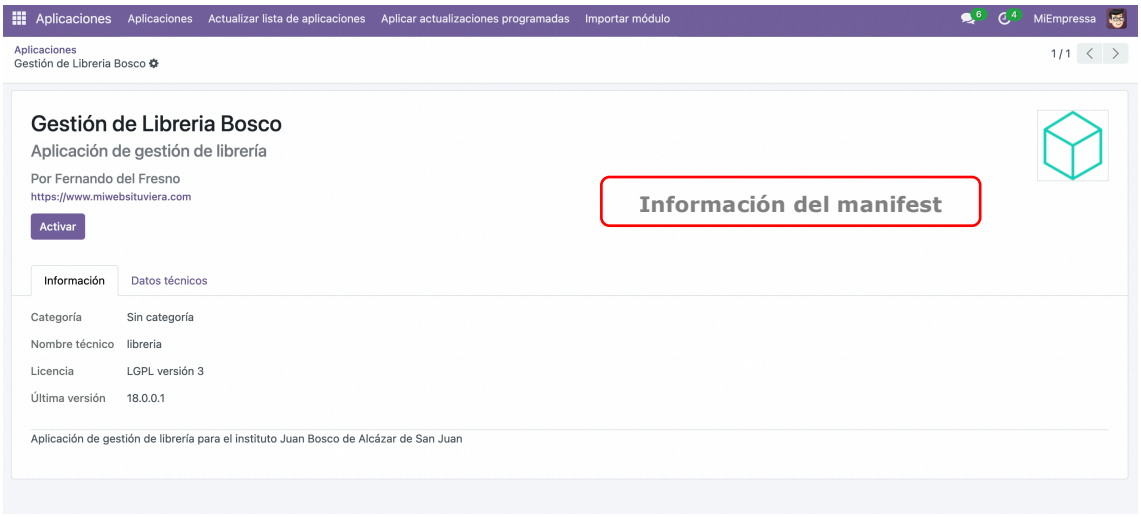


Ahora toca comprobar si se ha instalado correctamente. Una vez instalado, puedes acceder a las funcionalidades del módulo desde el menú o configuraciones específicas del módulo.

Si encuentras algún error al instalar el módulo, revisa los logs de Odoo para detalles adicionales, lo cual puede ayudarte a identificar si algún archivo o configuración necesita ajustes.

Pulsamos sobre la información del módulo





Ahora toca activar el módulo

Ahora es posible que puedan aparecer errores también. Consulta los errores y busca la solución. Se aprende mucho solucionando errores.

5. APLICAR CAMBIOS AL MÓDULO

Cada vez que hagamos cambios en el módulo deberemos actualizar la lista de aplicaciones y luego actualizar el módulo. Ese será el método más habitual pero también podremos hacerlo de 2 maneras diferentes:

- **Desde la terminal de nuestro S.O.**

```
docker exec -it odoo_dev-web-1 odoo -u libreria -d pruebas --
dev=reload
```

Esto lanzará en nuestro contenedor `odoo_dev-web-1` la actualización del módulo `libreria` en nuestra base de datos `-d pruebas` aplicándose los cambios y limpiando la caché con `--dev=reload`.

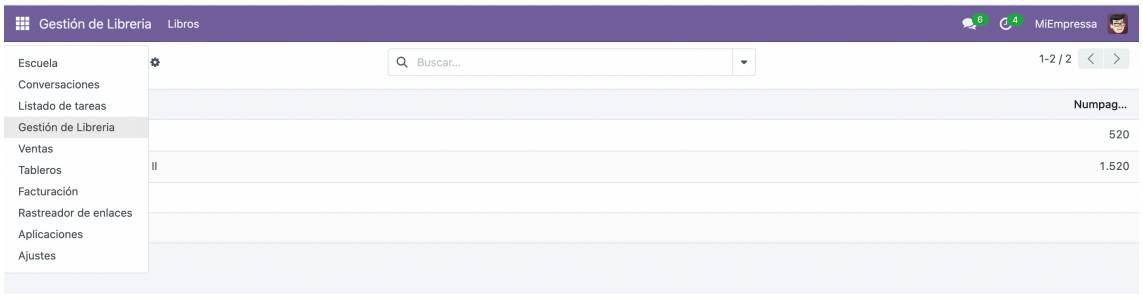
NOTA: Si le pusiste a BD otro nombre deberás sustituir `pruebas` por tu nombre de BD.

- **Desde la terminal del contenedor.**

Puedes usar este comando que hará exactamente lo mismo pero desde dentro de la terminal del propio contenedor de Odoo.

```
odoo -u libreria -d pruebas --dev=reload
```

Ahora sólo queda que pruebes el módulo



Cuando hayas insertado algunos libros puedes consultarlos en la BD con esta consulta. Puedes identificar todas las entidades porque comenzarán con el nombre del módulo.

```
SELECT *
FROM libreria_libro;
```