Práctica 3 - Primeros módulos Odoo

Sistemas de Gestión Empresarial

Cristian Fernández

16 de noviembre de 2025

Índice

1. Introducción

El objetivo de esta práctica consiste en implantar y configurar los módulos de Odoo indicados en esta misma documentación, demostrando su funcionamiento mediante capturas de pantalla. Además, se realizarán las modificaciones sobre el modelo y la vista del módulo *Lista de tareas* con las mejoras que se consideren oportunas, documentando todo el proceso con capturas.

Se reutilizará el repositorio creado en la práctica anterior, con objeto de tener todo el trabajo bien organizado y disponible para una posible visualización en cualquier momento.

La finalidad de la práctica en detalle es la siguiente:

- **Implantación de Módulo básico.**
  - Desarrollar e instalar el módulo *hola mundo* propuesto en el temario.
- **Implantación de Primer módulo.**
  - Desarrollar e instalar el módulo *lista de tareas* propuesto en el temario.
- **Modificación de Primer módulo.**
  - Modificar el módulo *lista de tareas* con las mejoras que se consideren oportunas y documentar dichas mejoras.

Cada punto tendrá una explicación breve del proceso de creación y se demostrará su funcionamiento mediante capturas. Si fuese necesario también se comentarán los problemas encontrados y las soluciones empleadas.

2. Implantación de módulo básico *hola mundo*

Odoo permite la creación de módulos personalizados al gusto del usuario en los que puede extender funcionalidades según se requiera. Para demostrarlo crearemos un módulo de ejemplo básico muy sencillo cuya única finalidad será aparecer listado en la lista de módulos de nuestro contenedor Odoo. El propósito es que el sistema detecte y muestre correctamente el módulo, lo cual nos permitirá avanzar al siguiente punto.

Asentada la premisa comenzaremos la tarea creando una carpeta dentro de nuestro proyecto llamada *Ejemplo01-HolaMundo*, en el interior del directorio en el que hayamos configurado el alojamiento de nuestros módulos personalizados (en mi caso la carpeta *addons*).

La carpeta recién creada contendrá los siguientes ficheros:

- **Fichero *__init__.py*:** Este fichero estará vacío.
- **Fichero *__manifest__.py*:** Este fichero contendrá una línea de código (se mostrará en la siguiente captura).

Posteriormente levantaremos nuestros contenedores con docker compose, accederemos a Odoo en el navegador y buscaremos el módulo recién creado.

Vamos a ver las capturas de todo el proceso en la siguiente página.

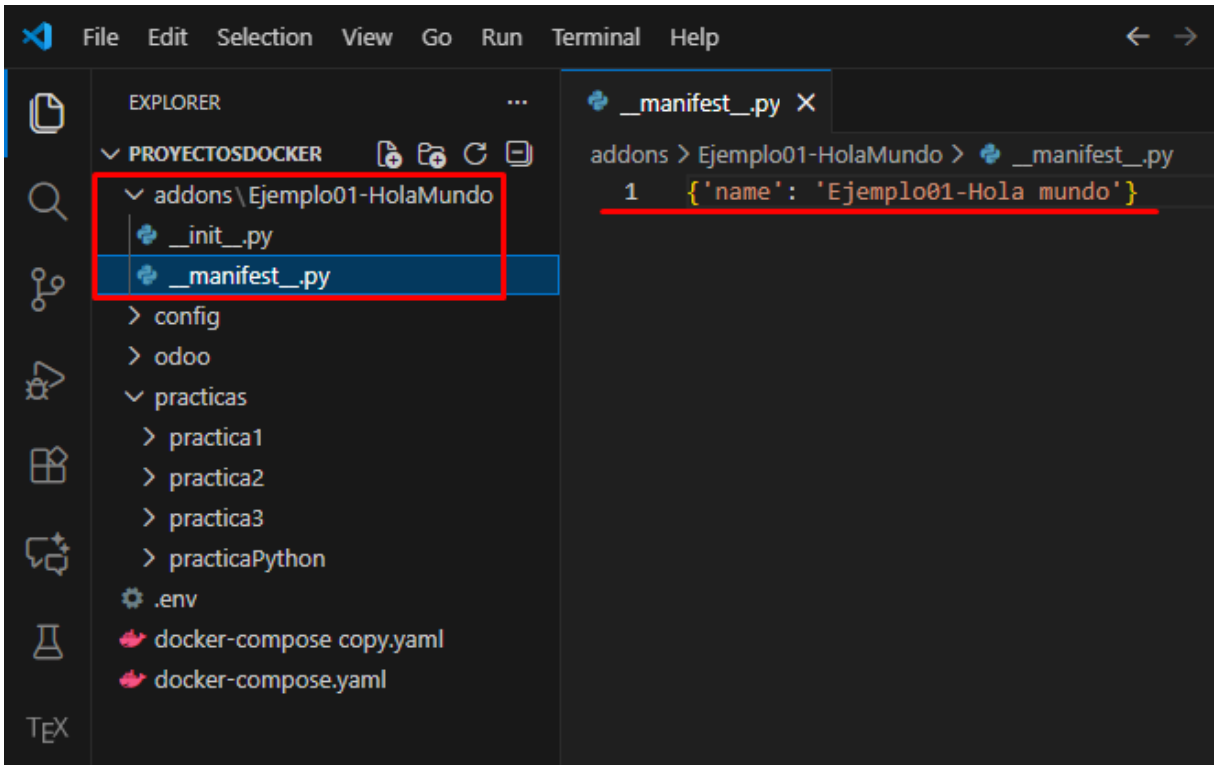


Figura 1: Carpeta y archivos creados junto con la línea de código

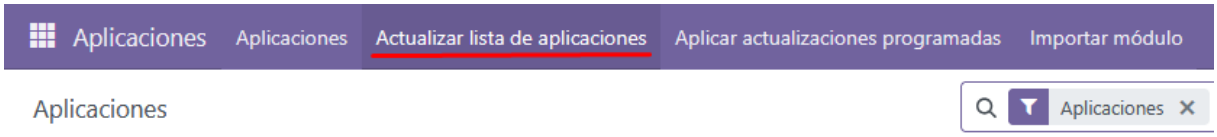


Figura 2: Dentro de Odoo pulsamos en *Actualizar lista de aplicaciones*

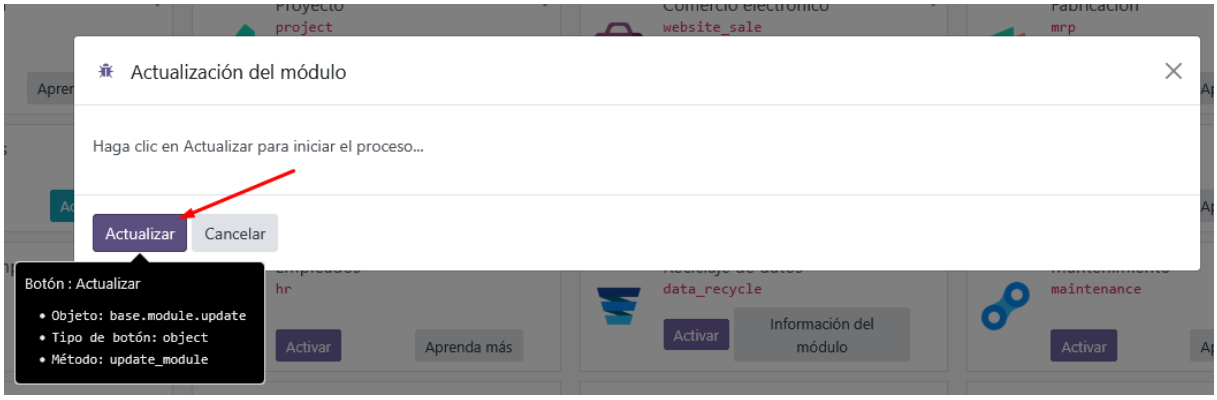


Figura 3: Nos saldrá este modal. Pulsamos en *Actualizar*

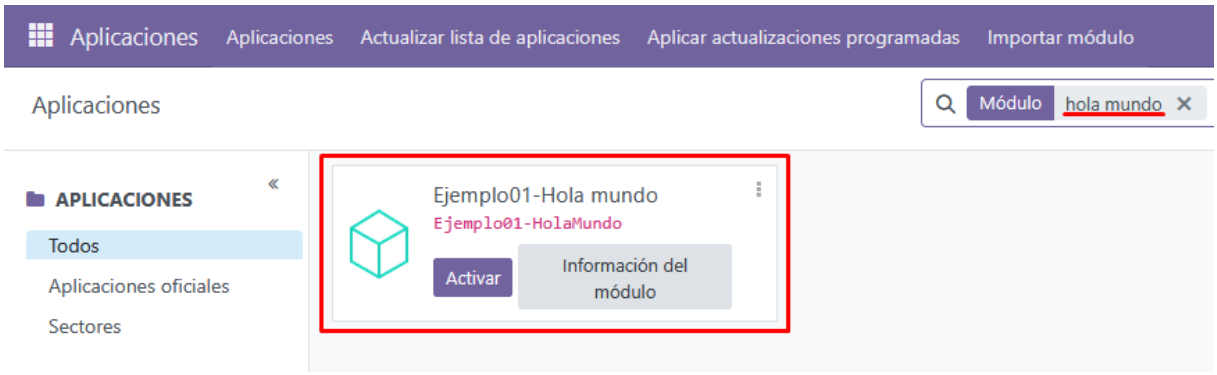


Figura 4: Eliminamos los filtros y buscamos "hola mundo". Si se visualiza nuestro módulo es que hemos tenido éxito y podemos seguir avanzando hasta el siguiente punto

Partiendo de la base de que nuestro proyecto está perfectamente configurado gracias a las prácticas anteriores, el único problema con el que me he topado es la necesidad de que nuestro Odoo debe estar configurado en modo desarrollador y para ello tenemos que hacer una pequeña configuración previa.

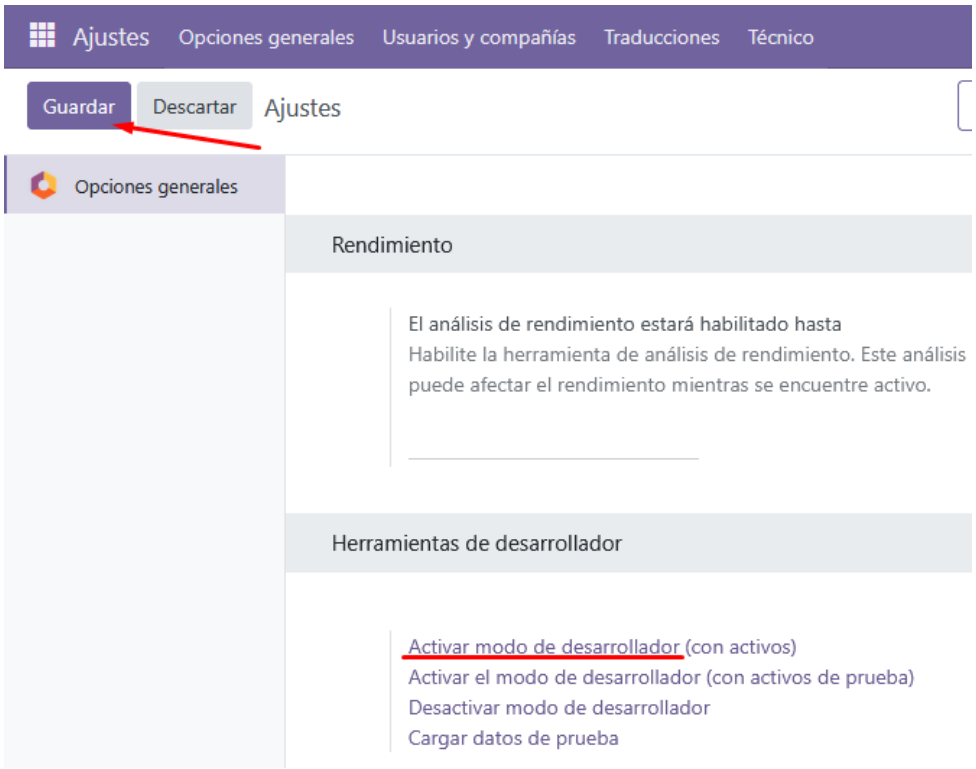


Figura 5: Menú Ajustes - *Activar modo de desarrollador* y pulsar en *Guardar*

3. Implantación de primer módulo *lista de tareas*

El siguiente paso consistirá en crear un tipo de módulo fácil de entender: uno que cree nuevos modelos de datos (ficheros maestros) y permita que se observen estos modelos a través de un nuevo menú.

Pero antes una reflexión.

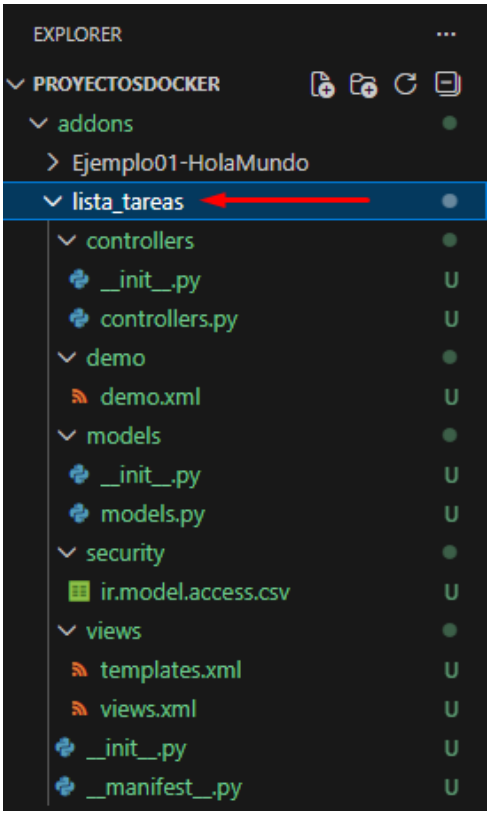
Crear la estructura de un nuevo módulo de Odoo puede resultar engorroso y tedioso si se hace con cierta frecuencia (y muy probablemente en el contexto de un desarrollador así sea) por lo que usaremos una funcionalidad que incorpora el propio Odoo llamada **scaffold**, que creará automáticamente toda la estructura de un nuevo módulo consiguiendo entre otras cosas aliviar el tedio de tener que hacerlo manualmente y mejorando nuestra calidad de vida como desarrolladores (quizás esté exagerando pero ciertamente se agradece).

Desde un terminal accederemos al interior de nuestro contenedor de Odoo con el comando "***docker compose exec*** *nombreServicio* ***/bin/bash***" y una vez dentro introducimos el comando "***odoo scaffold*** *nombreNuevoModulo /ruta/volumen-addons/*".

Deberemos sustituir la parte coloreada en azul por los valores correctos según nuestra configuración, tal como se puede observar en la siguiente captura:

```
PS C:\Users\Abueloncho\Documents\proyectos\code\proyectosDocker> docker compose ps
NAME          IMAGE                COMMAND                  SERVICE             CREATED       STATUS
d-adminer     adminer:standalone   "entrypoint.sh docke…"   web-db-management   2 hours ago   Up 2 hours
d-odoo        odoo:18              "/entrypoint.sh --de…"   web                 2 hours ago   Up 2 hours
d-postgres    postgres:17.6        "docker-entrypoint.s…"   db                  2 hours ago   Up 2 hours
PS C:\Users\Abueloncho\Documents\proyectos\code\proyectosDocker> docker compose exec web /bin/bash
odoo@f727b9575294:/$ ls
bin  boot  dev  entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys
odoo@f727b9575294:/$ odoo scaffold lista_tareas /mnt/extra-addons/
odoo@f727b9575294:/$
```

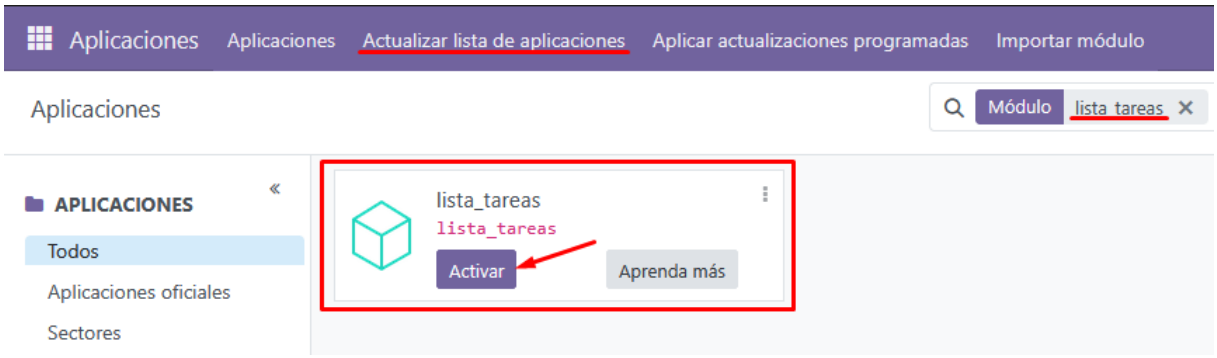
Si todo ha ido bien se deberá crear el módulo con toda su estructura tal que así:



Para evitar futuros problemas le daremos permisos de escritura al módulo dentro del contenedor con el comando "*chmod 777 -R /mnt/extra-addons/nombreNuevoModulo*" tal como muestra la captura:

```
odoo@f727b9575294:/$ ls -lah /mnt/extra-addons/
total 4.0K
drwxrwxrwx 1 root root 4.0K Nov 15 21:48 .
drwxr-xr-x 1 root root 4.0K Oct  9 21:23 ..
drwxrwxrwx 1 root root 4.0K Nov 15 19:34 Ejemplo01-HolaMundo
drwxr-xr-x 1 odoo odoo 4.0K Nov 15 21:48 lista_tareas
odoo@f727b9575294:/$ chmod 777 -R /mnt/extra-addons/lista_tareas/
odoo@f727b9575294:/$ ls -lah /mnt/extra-addons/
total 4.0K
drwxrwxrwx 1 root root 4.0K Nov 15 21:48 .
drwxr-xr-x 1 root root 4.0K Oct  9 21:23 ..
drwxrwxrwx 1 root root 4.0K Nov 15 19:34 Ejemplo01-HolaMundo
drwxrwxrwx 1 odoo odoo 4.0K Nov 15 21:48 lista_tareas
odoo@f727b9575294:/$
```

Ahora podemos comprobar que Odoo reconoce y muestra el módulo así que, como en el apartado anterior, actualizamos y búscamos el módulo nuevo:



Y justo aquí nos topamos con el primer problema ya que, si pulsamos en *Activar* nos encontraremos con el siguiente error:

Por su forma de funcionar intermanente Odoo necesita reiniciar el servicio y actualizar el módulo para que se actualicen los cambios realizados en el mismo, así que vamos a arreglar el error anterior **descomentando todo el código que contengan los archivos** y modificando el ***__manifest__.py***, reiniciando el servicio y activando/actualizando el contenedor.

__manifest__.py quedaría tal que así:

_manifest__.py U

addons > lista_tareas > __manifest__.py

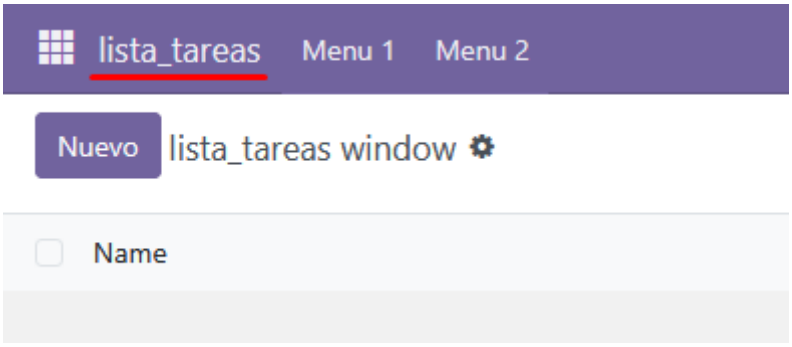
```
# -*- coding: utf-8 -*-
{
    'name': "Lista de Tareas",

    'summary': "Short (1 phrase/line) summary of the module's purpose",

    'description': """
Long description of module's purpose
    """,

    'author': "My Company",
    'website': "https://www.yourcompany.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
    # 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',
        'views/views.xml',
        'views/templates.xml',
    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
}
```

Figura 6: Cambiamos nombre y descomentamos la línea del csv

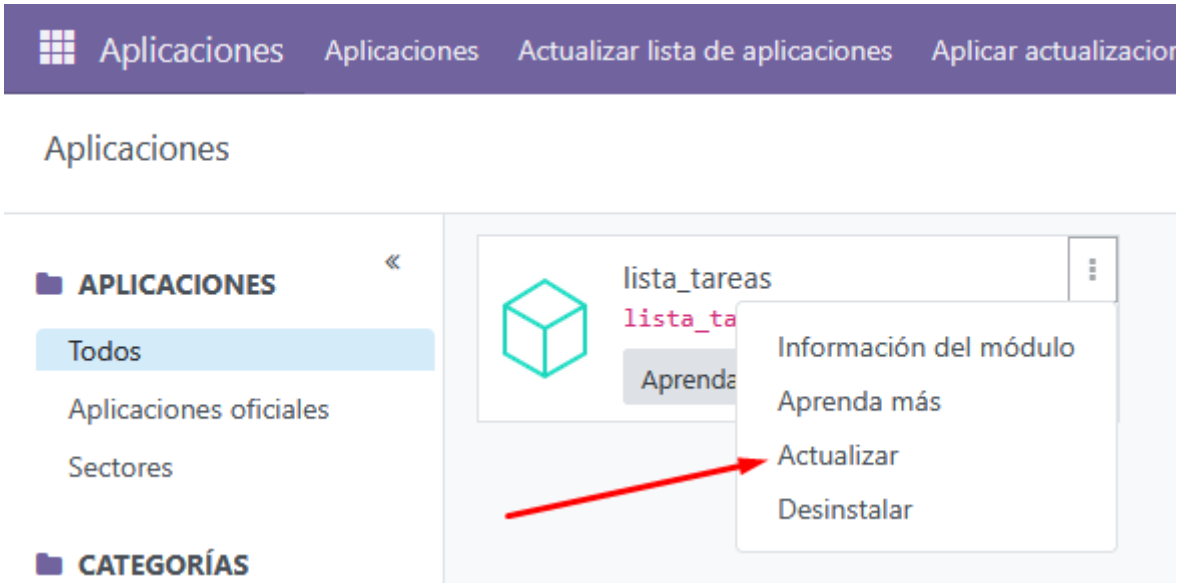
Y, desde un terminal, con docker compose reiniciaremos el contenedor con el comando "***docker compose restart*** *nombreServicio*"

```
PS C:\Users\Abueloncho\Documents\proyectos\code\proyectosDocker> docker compose ps
NAME          IMAGE                COMMAND                  SERVICE             CREATED
d-adminer     adminer:standalone   "entrypoint.sh docke…"   web-db-management   2 hours ago
d-odoo        odoo:18              "/entrypoint.sh --de…"   web                 2 hours ago
d-postgres    postgres:17.6        "docker-entrypoint.s…"   db                  2 hours ago
PS C:\Users\Abueloncho\Documents\proyectos\code\proyectosDocker> docker compose restart web
[+] Restarting 1/1
 ✓ Container d-odoo  Started
PS C:\Users\Abueloncho\Documents\proyectos\code\proyectosDocker>
```

Si volvemos a intentar activar el contenedor esta vez sí procederá correctamente y nos mostrará este resultado:



También nos dejará actualizar el módulo.



4. Modificación del primer módulo *lista de tareas*

Vamos a modificar nuestro módulo de forma liviana para comprobar si se actualiza correctamente. Básicamente crearemos unos cuantos campos en el modelo que sean descriptivos y los añadiremos a la vista para poder visualizarlos correctamente desde Odoo.
Las siguientes capturas mostrarán la creación de una nueva tarea desde nuestro módulo en la que se visualizará claramente el proceso de creación y el resultado y, posteriormente, mostraré unas capturas con las modificaciones que he hecho en el código.

Desde la vista principal del módulo clicamos en el botón *New*.

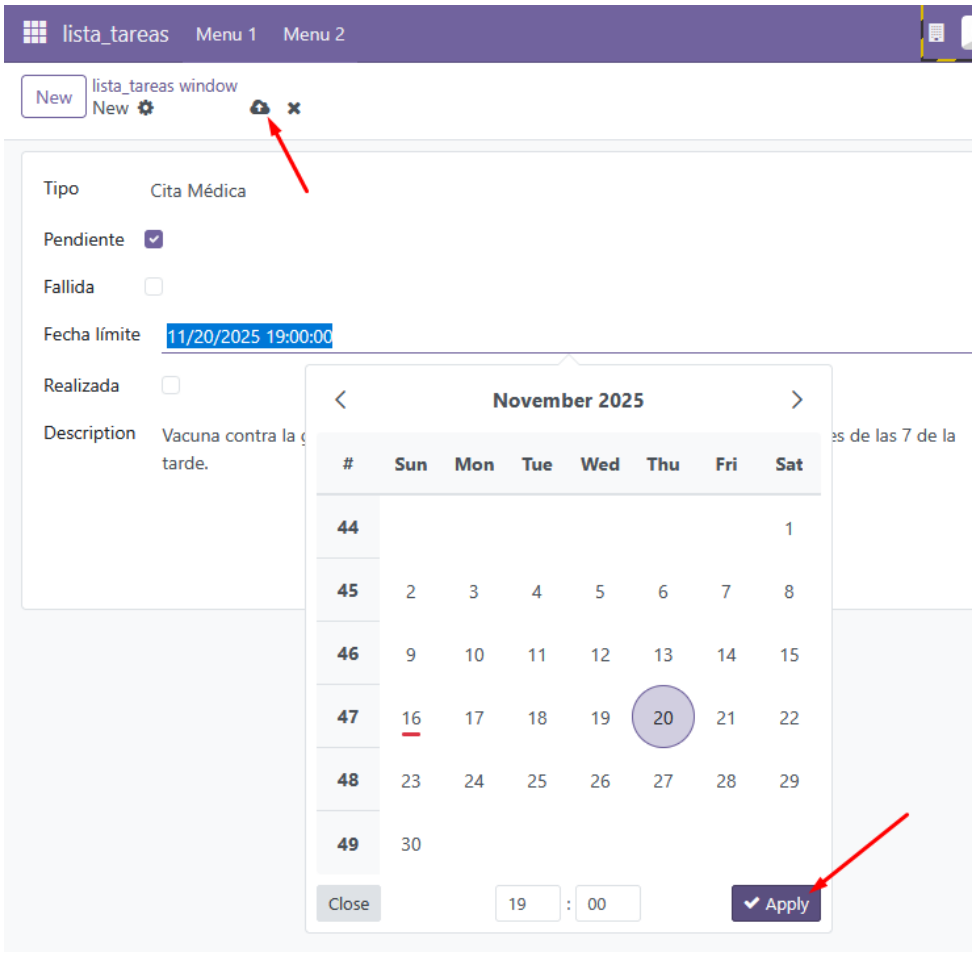


Figura 7: Ejemplo de una tarea nueva

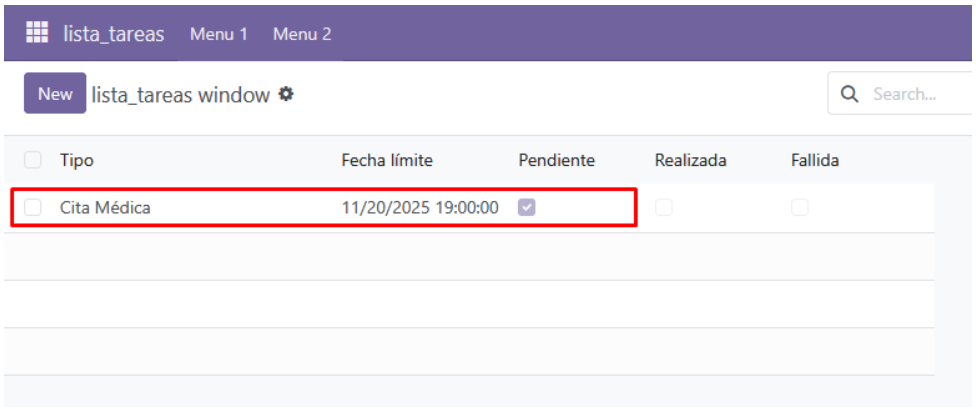


Figura 8: Una vez creada podremos visualizarla en la vista principal

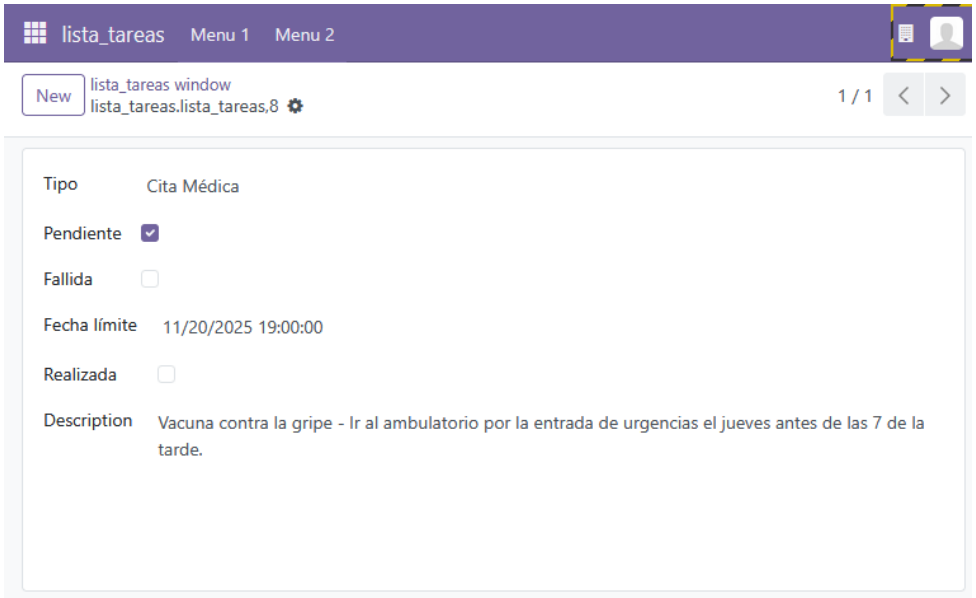


Figura 9: Si hacemos clic en ella podremos visualizarla con más detalle

Cuando todo está configurado y preparado resulta muy simple utilizar el módulo. Al final comentaré una problemática muy incómoda que me he encontrado.

Pero antes las capturas del código en la siguiente página:

models.py U views.xml U

addons > lista_tareas > models > models.py > ...

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

from odoo import models, fields, api


class lista_tareas(models.Model):
    _name = 'lista_tareas.lista_tareas'
    _description = 'lista_tareas.lista_tareas'

    tipo = fields.Char()
    fecha_limite = fields.Datetime(string='Fecha límite')
    pendiente = fields.Boolean()
    realizada = fields.Boolean()
    fallida = fields.Boolean()
    description = fields.Text()

    @api.depends('value')
    def _value_pc(self):
        for record in self:
            record.value2 = float(record.value) / 100

```

Figura 10: Captura del modelo

models.py U views.xml U

addons > lista_tareas > views > views.xml

```xml
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="lista_tareas.list">
      <field name="name">lista_tareas list</field>
      <field name="model">lista_tareas.lista_tareas</field>
      <field name="arch" type="xml">
        <list>
          <field name="tipo"/>
          <field name="fecha_limite"/>
          <field name="pendiente"/>
          <field name="realizada"/>
          <field name="fallida"/>
        </list>
      </field>
    </record>
    <!-- actions opening views on models -->
    <record model="ir.actions.act_window" id="lista_tareas.action_window">
      <field name="name">lista_tareas window</field>
      <field name="res_model">lista_tareas.lista_tareas</field>
      <field name="view_mode">list,form</field>
    </record>
    <!-- server action to the one above -->
    <record model="ir.actions.server" id="lista_tareas.action_server">
      <field name="name">lista_tareas server</field>
      <field name="model_id" ref="model_lista_tareas_lista_tareas"/>
      <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="lista_tareas" id="lista_tareas.menu_root"/>
    <!-- menu categories -->
    <menuitem name="Menu 1" id="lista_tareas.menu_1" parent="lista_tareas.menu_root"/>
    <menuitem name="Menu 2" id="lista_tareas.menu_2" parent="lista_tareas.menu_root"/>
    <!-- actions -->
    <menuitem name="List" id="lista_tareas.menu_1_list" parent="lista_tareas.menu_1"
              action="lista_tareas.action_window"/>
    <menuitem name="Server to list" id="lista_tareas" parent="lista_tareas.menu_2"
              action="lista_tareas.action_server"/>

  </data>
</odoo>
```

Figura 11: Captura de la vista

En esta ocasión me he topado con el problema más tedioso y farragoso de la práctica. Y es que Odoo da muchos problemas a la hora de actualizar los módulos. En primer lugar, el modo desarrollador no se queda habilitado entre sesiones por lo que hay que activarlo manualmente cada vez que levantemos el contenedor o iniciemos sesión.

Por si fuera poco entre accesos al módulo puede deshabilitarse el modo desarrollador sin venir a cuento teniendo que volver a activarlo manualmente lo cual es muy molesto.

La solución que empleé para trabajar con comodidad ha sido, una vez se activa el modo desarrollador, convertirse en superusuario tal como se ve a continuación.