

Sistemas de Gestión Empresarial

UD04 - Implantación de sistemas ERP/CRM

0. Consideraciones previas	2
1. Configurar carpeta para nuestros módulos	2
2. Crear estructura del módulo	3
3. Entendiendo la estructura de carpetas del módulo	4
4. Activar el modo desarrollo en la interfaz web de Odoo	5
5. Los modelos	5
6. Seguridad	5
7. Vistas	6
8. Acciones	7

0. Consideraciones previas

Como en unidades anteriores, para realizar esta tarea emplearemos la máquina virtual configurada durante la UD2. Sin embargo, a diferencia de la UD3, donde ejecutábamos el archivo de configuración, en esta unidad emplearemos el servicio `odoo16` creado. Una vez iniciado y habilitado, podemos comprobar su estado.

```
odoo@odoo-VirtualBox:~$ sudo systemctl start odoo16.service
odoo@odoo-VirtualBox:~$ sudo systemctl enable odoo16.service
Created symlink /etc/systemd/system/multi-user.target.wants/odoo16.service → /etc/systemd/system/odoo16.service.

odoo@odoo-VirtualBox:~$ sudo systemctl status odoo16
● odoo16.service - Odoo como servicio
   Loaded: loaded (/etc/systemd/system/odoo16.service; enabled; vendor preset:
   Active: active (running) since Fri 2023-03-10 18:57:33 CET; 2min 34s ago
     Main PID: 3482 (python3)
       Tasks: 3 (limit: 9457)
      Memory: 91.6M
         CPU: 3.901s
      CGroup: /system.slice/odoo16.service
              └─3482 python3 /opt/odoo/odoo-bin
```

1. Configurar carpeta para nuestros módulos

Los módulos instalados a través de Odoo se almacenan en la carpeta `addons`. Para crear nuestro módulo optamos por usar una carpeta propia, de forma que futuras reinstalaciones de Odoo no sobrescriban nuestros ficheros. Además, también nos facilita poder llevarlos a otro equipo.

Desde la terminal crearemos la nueva carpeta (`mkdir /opt/odoo/modulos_extra`). Con ella creada, editaremos el archivo de configuración (`.odoorc`) para indicarle que, además de la ruta por defecto, también habrá módulos en nuestra ruta.

```
GNU nano 6.2 .odoorc *
[options]
addons_path = /opt/odoo/addons,/opt/odoo/modulos_extra
```

Para aplicar los cambios, reiniciaremos el servicio.

```
odoo@odoo-VirtualBox:~$ sudo systemctl restart odoo16
```

2. Crear estructura del módulo

Los módulos de Odoo tienen ciertos requisitos mínimos (ciertas carpetas que están presentes en todos ellos). Esta estructura se podría crear a mano, pero Odoo nos facilita la creación de dicha estructura mediante el siguiente comando:

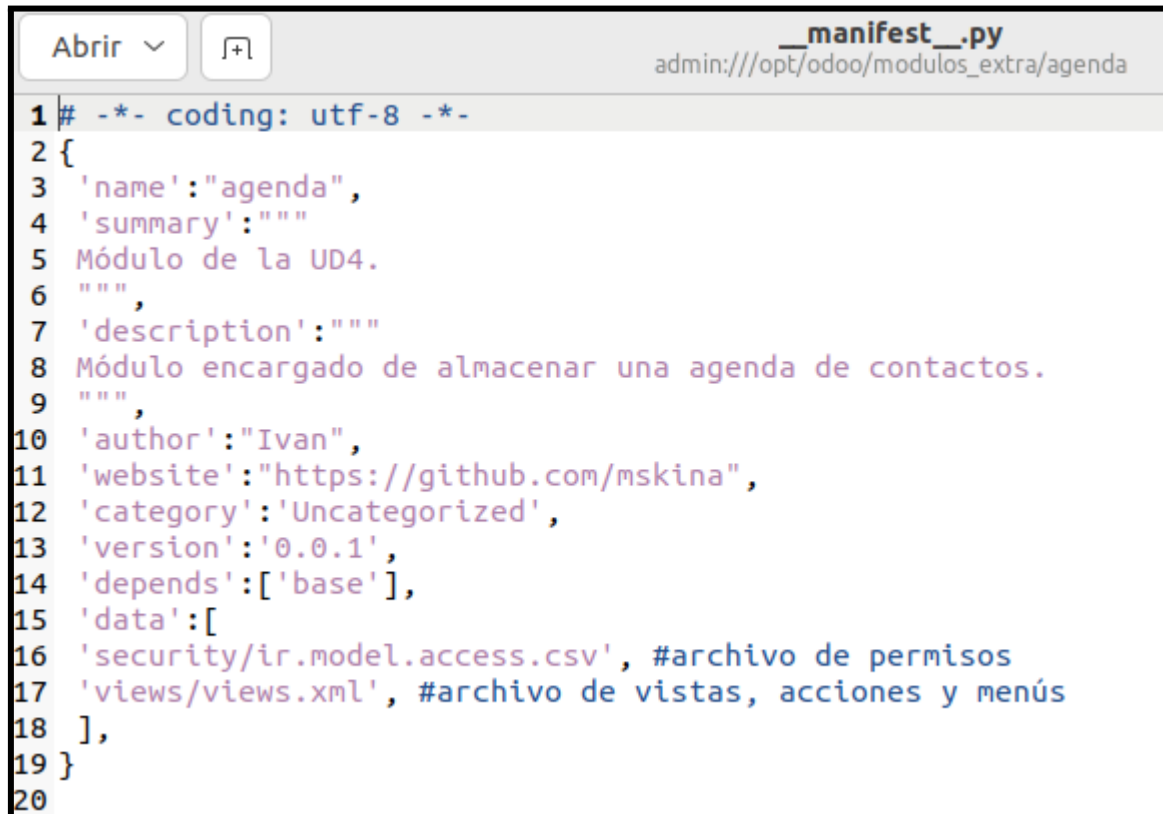
```
operador_odoo@odoo-VirtualBox:/opt/odoo$ /opt/odoo/odoo-bin scaffold agenda /opt/odoo/modulos_extra
```

Aquí le indicamos a Odoo que en nuestra carpeta cree nuestro módulo (agenda) y la estructura mínima



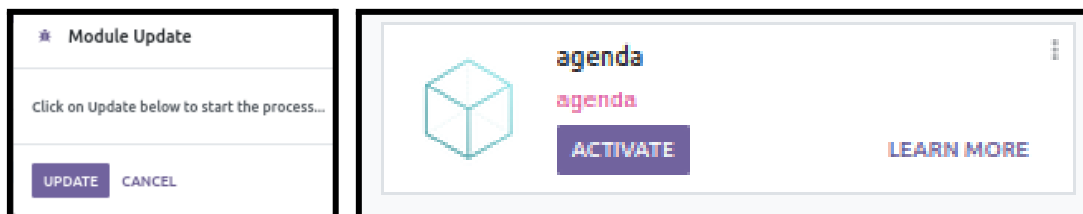
3. Entendiendo la estructura de carpetas del módulo

Como vemos en la captura anterior, en la carpeta hay dos archivos de Python: `init` y `manifest`, ambos con «`_`» antes y después del nombre. `init` podríamos resumirlo como un documento de inicialización del módulo, mientras que `manifest` se emplea para mostrar metadatos del módulo, como la descripción, el autor, la versión, etc, junto con cierta parte técnica.




```
1 # -*- coding: utf-8 -*-
2 {
3     'name': "agenda",
4     'summary': ""
5     Módulo de la UD4.
6     "",
7     'description': ""
8     Módulo encargado de almacenar una agenda de contactos.
9     "",
10    'author': "Ivan",
11    'website': "https://github.com/mskina",
12    'category': 'Uncategorized',
13    'version': '0.0.1',
14    'depends': ['base'],
15    'data': [
16        'security/ir.model.access.csv', #archivo de permisos
17        'views/views.xml', #archivo de vistas, acciones y menús
18    ],
19 }
20
```

Editamos dicho archivo incluyendo una pequeña información mínima sobre nuestra aplicación, eliminando ciertos comentarios así como dependencias que no emplearemos. Una vez guardados los cambios, reiniciamos Odoo (`sudo systemctl restart odoo16`).



4. Activar el modo desarrollo en la interfaz web de Odoo

En mi caso concreto, al conservar la instalación de todas estas unidades, no he necesitado activarlo ya que conservo dicha configuración. Lo puedo comprobar, entre otras formas, porque en todo momento tengo visible el icono  en la barra superior de Odoo.

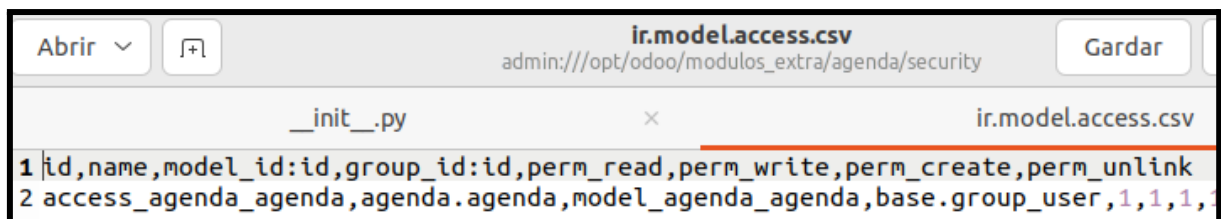
5. Los modelos

En la carpeta models editamos el archivo `models.py`, descomentando las primeras líneas y añadiendo los campos de nombre y teléfono. También aprovecho para cambiar la descripción. Con esto estamos creando una tabla en la base de datos con el nombre indicado en name, donde almacenaremos nombres y teléfonos en formato char.

```
# -*- coding: utf-8 -*-  
  
from odoo import models, fields, api  
  
class agenda(models.Model):  
    _name = 'agenda.agenda'  
    _description = 'Agenda UD4'  
  
    nombre = fields.Char()  
    telefono = fields.Char()
```

6. Seguridad

En la carpeta security se encuentra el archivo `ir.model.access.csv`, que será donde vengán definidas las reglas de seguridad. En este ejemplo no necesitamos modificarlo, pero en caso de hacerlo, nos permitiría crear tantas reglas como deseásemos, declarando cada una de ellas en una nueva línea.



```
__init__.py  
1 # -*- coding: utf-8 -*-  
2  
3 from . import controllers  
4 from . import models
```

En `init` podemos confirmar que tanto `models` como `controllers` están recogidos, de forma que aseguremos que ambos son iniciados correctamente.

7. Vistas

En la subcarpeta de `views` contamos con el archivo `views.xml`, el cual contiene una plantilla que procederemos a editar para tener nuestra aplicación funcional.

Por cada vista que creemos, se generará un registro con un id único. En nuestro caso, completaremos los datos en función de las elecciones previas (nombre y teléfono).

```
<record model="ir.ui.view" id="agenda.list">
  <field name="name">Lista de registros</field>
  <field name="model">agenda.agenda</field>
  <field name="arch" type="xml">
    <tree>
      <field name="nombre"/>
      <field name="telefono"/>
    </tree>
  </field>
</record>
```

Las acciones se guardan en una tabla de postgres. Todas ellas tienen un id (normalmente, `nombredelmodulo.algoquequeramos`)

```
<record model="ir.actions.act_window" id="agenda.action_window">
  <field name="name">Ventana del directorio</field>
  <field name="res_model">agenda.agenda</field>
  <field name="view_mode">tree,form</field>
</record>
```

En la parte final del archivo tenemos la posibilidad de editar el nombre que aparece en el menú superior, en el de categorías así como un elemento que realiza una acción, a la que enlazamos desde el atributo `action`.

```
<menuitem name="Agenda" id="agenda.menu_root"/>

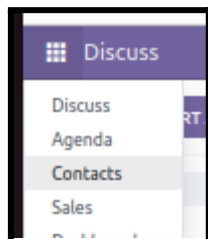
<!-- menu categories -->

<menuitem name="Menu principal" id="agenda.menu_1" parent="agenda.menu_root"/>

<!-- actions -->

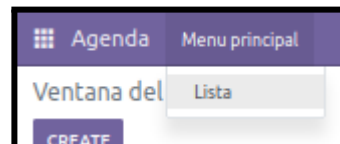
<menuitem name="Lista" id="agenda.menu_1_list" parent="agenda.menu_1"
  action="agenda.action_window"/>
```

8. Acciones



Una vez guardados los cambios, reiniciamos Odoo. Tras ello, activamos nuestra aplicación, que podemos ver en el menú con el nombre que definimos.

Al acceder, podemos comprobar que también aparece el elemento de **Menú principal** definido, así como el elemento **Lista**.



Al crear entradas, podemos comprobar que tanto nombre como teléfono están visibles en el lugar esperado.

Nombre	Telefono
Iván	654321987
Alejandro	987654321

Desde pgAdmin, podemos comprobar que la tabla incorpora tanto nuestras columnas como algunas adicionales de trazabilidad (id, creación fechas, etc).

	id [PK] integer	create_uid integer	write_uid integer	nombre character varying	telefono character varying	create_date timestamp without time zone
1	1	2	2	Iván	654321987	2023-03-10 23:31:17.514769
2	2	2	2	Alejandro	987654321	2023-03-10 23:31:46.407712

De esta forma, podemos comprobar que el módulo creado funciona correctamente, tanto por la parte de los datos guardados como por su visualización en la aplicación.