

# SISTEMAS DE GESTIÓN EMPRESARIAL

## TEMA 5, Tarea

### Desarrollo de componentes

#### Contenido

- [Enunciado](#)
- [Notas](#)
- [Previo](#)
- [Ejercicio](#)
  - [Estructura](#)
    - [Estructura completa](#)
    - [Estructura mínima](#)
  - [Modelo](#)
  - [Vista](#)
  - [Instalación del módulo](#)
- [Permisos](#)
- [Archivos](#)
  - [\\_\\_init\\_\\_.py](#)
  - [\\_\\_manifes\\_\\_.py](#)
  - [agenda.py](#)
  - [agenda\\_agenda\\_view.xml / views.xml](#)
  - [ir.model.access.csv](#)
  - [agenda\\_security.xml](#)



**Notas:** 📝

La instalación se hace mediante *scaffold* para organizar el contenido del módulo y porque, además, así lo recomienda Odoo a pesar del texto del temario.

- **Error** `ModuleNotFoundError: No module named 'pypdf2'`

En alguna instalación, al ejecutar *scaffold*, aparece este error porque falta el módulo PyPdf2 de Python.

<https://programminghistorian.org/es/lecciones/instalar-modulos-python-pip#instrucciones-para-windows>

También falla 'win32service' y al ejecutar `pip install win32service` sigue dando error.

```
C:\Odoo\server>pip install win32service
ERROR: Could not find a version that satisfies the requirement win32service (from versions: none)
ERROR: No matching distribution found for win32service
```

Finalmente uso el intérprete que Odoo integra en la carpeta Python.

```
C:\Odoo\python>python.exe C:/odoo/server/odoo-bin scaffold C:/odoo/server/odoo/addons/agenda
```

Creamos los grupos para los permisos para probar.

El archivo 'security/ir.model.access.csv' está comentado por defecto al crear la estructura con *scaffold*.

Solamente se ha probado en el modo de la estructura completa, no en la mínima.

## Enunciado

En esta unidad hemos aprendido cómo crear nuevos componentes utilizando sentencias del lenguaje propio del sistema ERP-CRM. Hemos creado componentes de manipulación de datos mediante módulos que crean a su vez tablas en la base de datos. También hemos añadido módulos al sistema y comprobado que funcionan. Además, hemos conocido herramientas adicionales para la creación de formularios e informes.

Ahora es el momento que pongamos en práctica estos conocimientos. La tarea consiste en crear un componente o módulo que gestione una **agenda telefónica** con las siguientes características:

### Modelo y Controlador.

- Objeto en la aplicación llamado `agenda` con, al menos, dos campos: `Nombre` y `Teléfono`.
- Tabla en la base de datos con los datos del objeto.

### Vista.

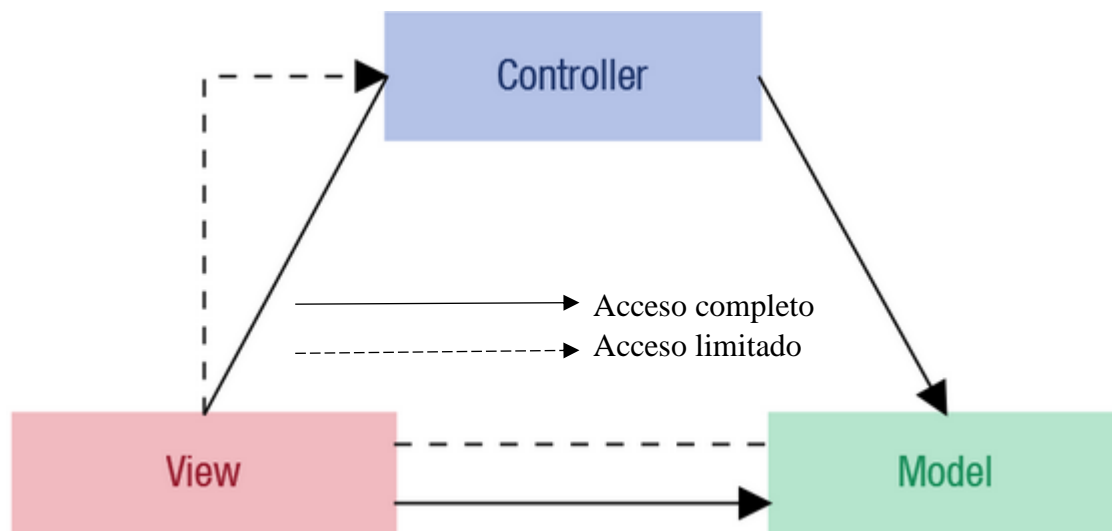
- Menú en la aplicación que enlace al objeto.
- Vista formulario con los datos del objeto.
- Vista árbol con los datos del objeto.

Además del módulo deberás escribir también un informe con todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea.

## Previo

### El paradigma MVC en Odoo

- Modelo:
  - Es el SGBD.
  - **Odoo**: las tablas PostgreSQL.
- Vista:
  - Es la interfaz de usuario y el código para generarla.
  - **Odoo**: las vistas están definidas en los archivos XML.
- Controlador:
  - Se encarga de procesar las peticiones.
    - Consulta los datos al modelo.
    - Realiza los cálculos necesarios.
    - Solicita la vista.
  - **Odoo**: Los objetos de OpenERP creados en Python.



### Especificaciones funcionales para el desarrollo del componente **Agenda**.

- Para cada módulo existe una carpeta con el nombre del módulo en el directorio **addons** del servidor.

## Ejercicio

### Estructura

#### Crear el directorio y los archivos con *scaffold*

Para cada módulo existe una carpeta con el nombre del módulo en el directorio **addons** del servidor.

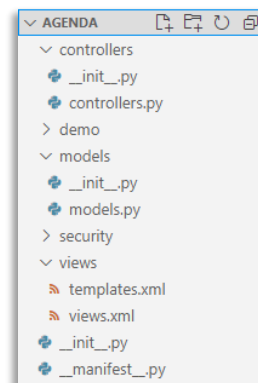
Directorios y archivos mínimos que Odoo crea automáticamente a través de *scaffold*

Comando: `path_toPython path_to_odoo_bin scaffold directorio`

*Python.exe* está en *Odoo/python*.

*Odoo-bin* se encuentra en *Odoo/server*.

El directorio que vamos a crear es *agenda*.



Nos situamos en el directorio *Odoo/server*

```
C:\Odoo\python>python.exe C:/odoo/server/odoo-bin scaffold agenda C:/odoo/server/odoo/addons
```

Tenemos todos los archivos creados.

La instalación del módulo se realiza de la misma forma que en el modo manual.

Las capturas de pantalla para la instalación con *scaffold* son las mismas que la instalación manual.

Diferencias con los archivos de la instalación mínima:

#### `__manifest__.py`

Al archivo creado automáticamente le añadimos las líneas que indican que el módulo es una aplicación y la que especifica la licencia.

Estas líneas también se añaden en el archivo creado manualmente.

Se mantiene la declaración del directorio *demo* y se modifica el contenido de *data*.

Con scaffold	Manual
<pre># always loaded 'data': [     # 'security/ir.model.access.csv',     'views/views.xml',     'views/templates.xml', ], # only loaded in demonstration mode 'demo': [     'demo/demo.xml', ],  #Indicar que es una aplicación 'application': True,  #Especificamos la licencia 'license': 'LGPL-3',</pre>	<pre>#En data se indican los archivos que siempre serán cargados, entre #ellos los xml 'data' : [     'agenda_agenda_view.xml' ],  #Indicar que es una aplicación 'application': True,  #Especificamos la licencia 'license': 'LGPL-3',</pre>

#### `__init__.py`

Se deja el archivo creado por defecto.

#### Vista:

En lugar de usar el archivo *agenda/agenda\_agenda\_view.xml* se define la vista en *views/views.xml*.

El contenido es el mismo.

#### Modelo:

El modelo se declara en *models/models.py* y es el mismo código que en la instalación mínima.

## Desarrollo manual. Contenido mínimo. ↗

<https://emperove.gitbooks.io/fundamentos-de-desarrollo-odoo-10/content/capitulo-2.html>

<https://www.odoo.com/documentation/14.0/developer/howtos/backend.html>

- La capa del **modelo**, que define la estructura de los datos de la aplicación
- La capa de **\*vista**, que describe la interfaz de usuario
- La capa del **controlador**, que soporta la lógica de negocio de la aplicación

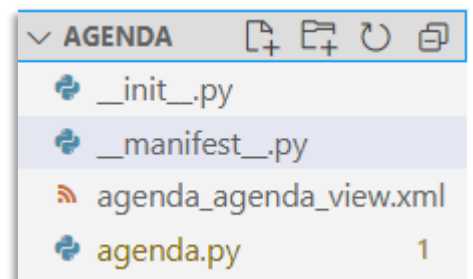
Un directorio que contiene un archivo de manifiesto `__manifest__.py` es un paquete.

Creamos el directorio *agenda*.

Dependiendo del directorio de instalación de Odoo, puede cambiar dependiendo de la máquina en la que se ha instalado.

La ruta parte de `Odoo*/server/odoo`.

- `__init__.py`:
  - Archivo necesario para que Python trate el directorio como un paquete.
  - Contiene los import de cada archivo que contenga código Python.
- `__manifest__.py`: archivo de descripción del módulo.
  - Valores que debe contener:
    - `name`: nombre del módulo.
    - `version`: versión del módulo.
    - `description`: una descripción del módulo.
    - `author`: persona o entidad que ha desarrollado el módulo.
    - `website`: sitio web del módulo.
    - `license`: tipo de licencia del módulo (por defecto GPL).
    - `depends`: lista de módulos de los que depende el módulo.
    - `init_xml`: lista de los archivos XML que se cargarán con la instalación del módulo.
    - `installable`: determina si el módulo es instalable o no.
- `agenda.py`:
  - Se define la clase con los campos que tiene.
  - Al crear la clase, se crea el modelo (tabla en la base de datos) y el controlador, ya que definimos el comportamiento.
- `agenda_agenda_view.xml`:
  - Vista del objeto (`nombreModulo_nombreObjeto.xml`).



*Archivos mínimo del módulo "agenda"*

## Modelo

El modelo tiene un listado de atributos y funciones que especifican su comportamiento.

Los modelos, que son objetos Python, que heredan de `Model.models`, y que son traducidos a objetos de bases de datos mediante ORM.

Nuestro modelo se describe en [agenda.py](#)

El modelo tendrá dos campos:

- Nombre del contacto, cadena de texto de un máximo de 60 caracteres, obligatorio
- Teléfono, cadena de texto de un máximo de 9 caracteres, obligatorio

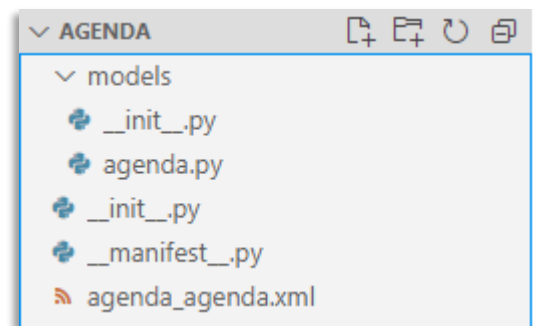
### Crear un directorio `models`

Odoo recomienda colocar los modelos dentro del directorio `models`, así que vamos a modificar la estructura del directorio `agenda` para adaptarlo.

Creamos el directorio `models` y movemos a él el archivo `agenda.py`.

Dentro de `models` hay que crear otro archivo `__init__.py` para importar los archivos Python que haya en él.

A su vez, debemos modificar en `agenda/__init__.py` la línea que importaba `agenda.py` para que ahora importe `models`.



Directorio modificado para agrupar los modelos

Seguimos sin el directorio `models`

El atributo más importante del modelo es `_name`, que es, además, obligatorio. El nombre de la clase hace referencia al `nombreDelMódulo.nombreDeLaClase`. Contenido mínimo de un modelo:

```
from odoo import models, fields
class AgendaModel(models.Model):
    _name = 'agenda.AgendaModel'
```

Declaramos los campos necesarios para el objeto `agenda`.

`name` es un nombre especial que Odoo usará para referenciar el objeto.

```
name = fields.Char('Nombre', required = True, help='Nombre del contacto')
telefono = fields.Char('Teléfono', required = True, size=9, help='9 dígitos')
```

El modelo no tiene, de momento, ninguna función.



## Instalación del módulo

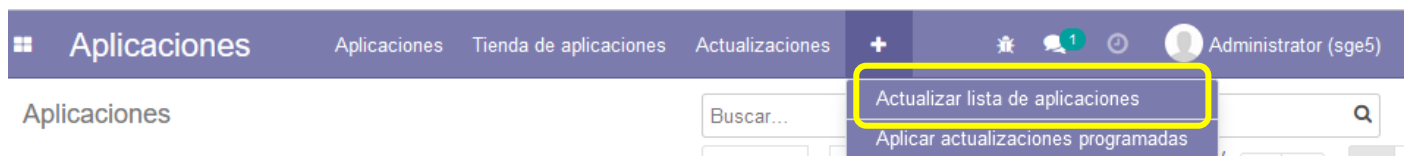
El archivo `agenda_agenda_view.xml`, en el cual se define la vista, no puede estar vacío

```
File "c:\odoo\server\odoo\addons\agenda\agenda_agenda_view.xml", line 1  
lxml.etree.XMLSyntaxError: Document is empty, line 1, column 1
```

Le damos un contenido mínimo

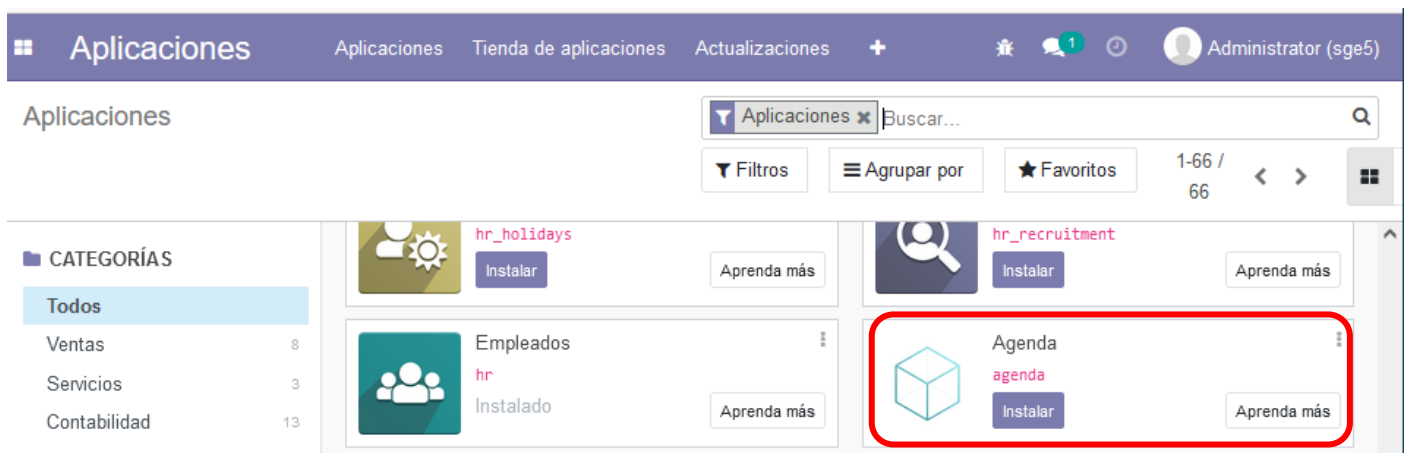
```
<odoo>  
  <data>  
  </data>  
</odoo>
```

Desde *Aplicaciones*, con el modo desarrollador activado, actualizamos el listado de aplicaciones disponibles.

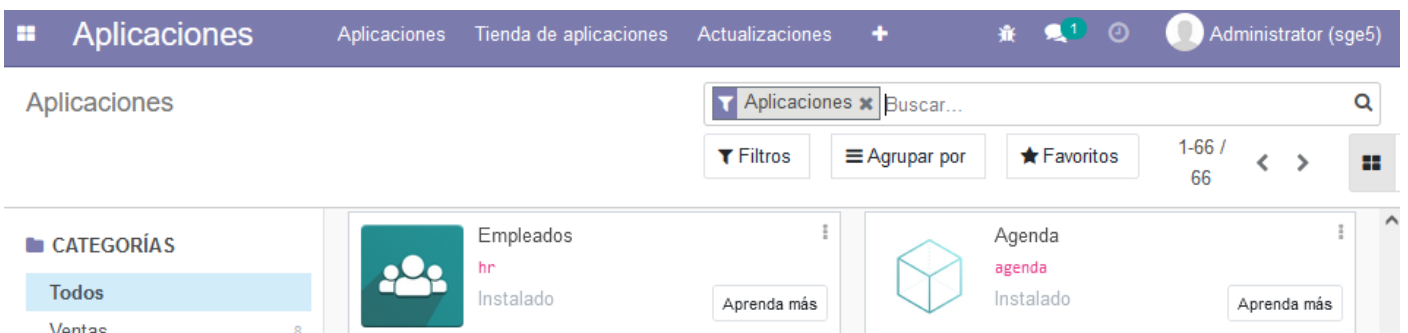


La aplicación aparece disponible para su instalación.




En caso de no aparecer (podríamos haberla no marcado como “aplicación” en `__manifest__.py`), habría que buscarla desde la caja de búsqueda.





La instalación se realiza correctamente.






Comprobamos el modelo desde los ajustes.




**Ajustes** Opciones generales Usuarios y compañías Traducciones Técnico    Administrator (SGE\_5)

Modelos  Buscar... 

[Crear](#) 

 Filtros  Agrupar por  Favoritos 1-1 / 1 < >

<input type="checkbox"/> Modelo	Descripción del modelo	Tipo	Modelo transitorio
<input type="checkbox"/> agenda.agenda	Representa cada uno de los registros de la agenda	Objeto base	<input type="checkbox"/>

**Ajustes** Opciones generales Usuarios y compañías Traducciones Técnico    Administrator (SGE\_5)

Modelos  
/ Representa cada uno de los registros de la age...

[Editar](#) [Crear](#) [Imprimir](#) [Acción](#) 1 / 1 < >






<b>Descripción del modelo</b>	Representa cada uno de los registros de la agenda	<b>Tipo</b>	Objeto base
<b>Modelo</b>	agenda.agenda	<b>En las aplicaciones</b>	agenda
<b>Pedido</b>	name		
<b>Modelo transitorio</b>	<input type="checkbox"/>		
<b>Hilo de mensajes</b>	<input type="checkbox"/>		
<b>Actividad de correo</b>	<input type="checkbox"/>		
<b>Lista negra de correo</b>	<input type="checkbox"/>		

[Campos](#) [Permisos de acceso](#) [Reglas de registro](#) [Notas](#) [Vistas](#)

Nombre de campo	Etiqueta de campo	Tipo de campo	Requerido	Sólo lectura	Indexado	Tipo
__last_update	Last Modified on	Fecha y hora	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Campo base
create_date	Created on	Fecha y hora	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Campo base
create_uid	Created by	many2one	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Campo base
display_name	Display Name	Carácter	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Campo base
id	ID	entero	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Campo base
name	Nombre	Carácter	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo base
telefono	Teléfono	Carácter	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Campo base
write_date	Last Updated on	Fecha y hora	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Campo base
write_uid	Last Updated by	many2one	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Campo base

pgAdmin 4

**pgAdmin** File Object Tools Help

Browser      Dashboard Properties SQL Statistics Dependencies

> account\_tour\_upload\_bill\_em  
 > account\_unreconcile  
 > agenda\_agenda  
     Columns (7)  
         id  
         name  
         telefono  
         create\_uid  
         create\_date  
         write\_uid  
         write\_date  
 > Constraints  
 > Indexes  
 > RLS Policies

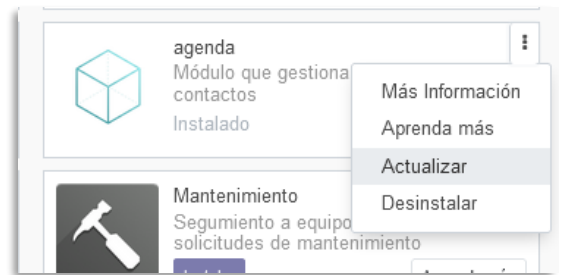
<input type="checkbox"/> Name	Comment
<input type="checkbox"/> id	
<input type="checkbox"/> name	Nombre
<input type="checkbox"/> telefono	Teléfono
<input type="checkbox"/> create_uid	Created by
<input type="checkbox"/> create_date	Created on
<input type="checkbox"/> write_uid	Last Updated by
<input type="checkbox"/> write_date	Last Updated on

## Vista

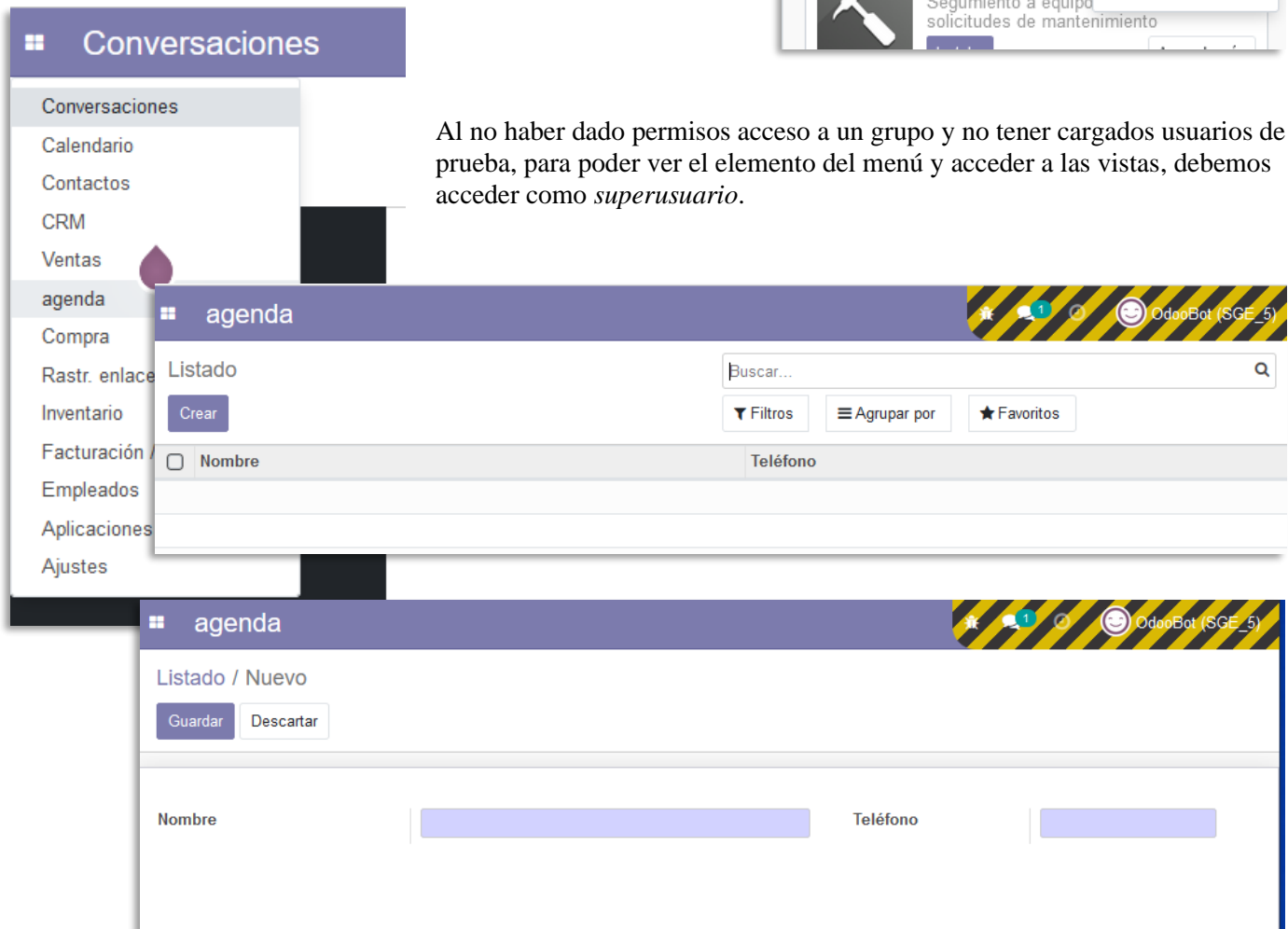
El archivo en el que se definen las vistas es

agenda\_agenda\_view.xml.

Después de definir la vista tree y form, las acciones de menú y el elemento del menú principal, hay que actualizar la aplicación para que se recargue el archivo xml.



Al no haber dado permisos acceso a un grupo y no tener cargados usuarios de prueba, para poder ver el elemento del menú y acceder a las vistas, debemos acceder como *superusuario*.



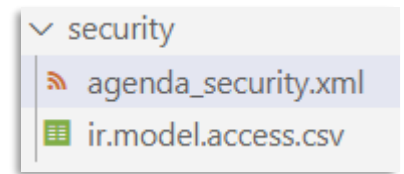
Retocamos la vista form y el nombre dado al ítem del menú para que aparezca con mayúscula.



## Permisos

Definimos los grupos que van a tener permiso de acceso al módulo.

Los grupos los definimos en el archivo security/agenda\_security.xml.



Definimos una categoría de seguridad para el módulo.

El campo sequence indica la prioridad del módulo que, al no depender de otros, podríamos omitirla. En lugar de omitir el campo, lo especificamos con un valor aleatorio y alto.

```
<record model="ir.module.category" id="agenda.module_category_agenda">
  <field name="name">Agenda</field>
  <field name="description">Gestión de números telefónicos</field>
  <field name="sequence">100</field>
</record>
```

Definimos los grupos de seguridad.

Tan sólo creamos un grupo de seguridad “Usuario” con acceso a la agenda.

implied\_ids amplía el acceso a grupos ya existentes. El valor 4 indica que el grupo se agrega.

```
<!-- Grupos de seguridad -->
<record model="res.groups" id="group_agenda_usuario">
  <field name="name">Usuario</field>
  <field name="category_id" ref="agenda.module_category_agenda" />
  <field name="comment">Usuario con acceso a la agenda</field>
  <field name="implied_ids" eval="[(4,ref('base.user_root')), (4,ref('base.group_user')), (4,ref('base.user_admin'))]"/>
</record>
```

ir.model.access.csv

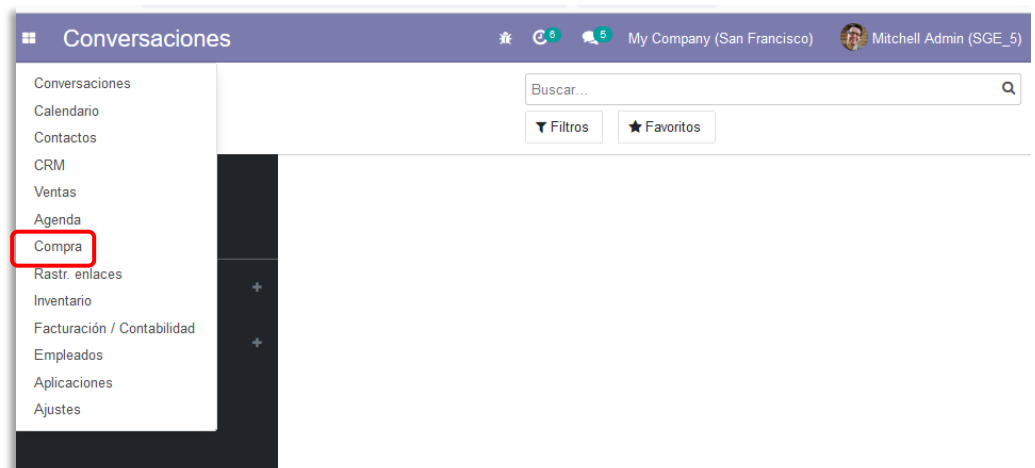
Añadimos la línea para dar permisos al grupo creado

access\_agenda\_agenda\_usuario,agenda.Usuario,model\_agenda\_agenda,agenda.group\_agenda\_usuario,1,1,1,1

Indicamos las modificaciones en el archivo \_\_manifest\_\_.py

```
'data': [
    'security/agenda_security.xml',
    'security/ir.model.access.csv',
    'views/views.xml',
    'views/templates.xml',
],
```

Una vez que hemos dado los permisos, se muestra el módulo correctamente.



## Archivos de instalación mínima

### `__init__.py`

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

#Importamos los archivos Python que se incluyen en el módulo
#agenda.py es la clase que define el modelo.
from . import agenda
```

### `__manifest__.py`

```
# -*- coding: utf-8 -*-
#name: nombre del módulo
#summary: subtítulo
#versión: versión del módulo
#description: descripción del módulo
#author: persona que desarrolla el módulo
#website: sitio web del módulo
#license: tipo de licencia (por defecto GPL)
#depends: lista de módulos de los que depende el módulo.
#init_xml: lista de archivos xml que se cargarán con la instalación del módulo.
#installable: determina si el módulo es instalable o no

{
    'name': "agenda",

    'summary': """
        Módulo que gestiona los teléfonos de los contactos""",

    'description': """
        Módulo de pruebas para gestionar una agenda telefónica
        """,

    'author': "Roberto Rodríguez",
    'website': "http://www.robtorodriguez.net",

    # Categories can be used to filter modules in modules listing
    # Check https://github.com/odoo/odoo/blob/14.0/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'],

    #En data se indican los archivos que siempre serán cargados, entre
    #ellos los xml
    'data' : [
        'agenda_agenda_view.xml'
    ],

    #Indicar que es una aplicación
    'application': True,

    #Especificamos la licencia
    'license': 'LGPL-3',
}
```

**agenda.py**

```
# -*- coding: utf-8 -*-
from odoo import models, fields
class agenda(models.Model):

    #_name es el nombre de referencia del modelo: nombreDelModulo.nombreDeLaClase
    _name = 'agenda.agenda'
    _description = 'Representa cada uno de los registros de la agenda'
    _order = 'name'

    #Los campos indican qué almacena el objeto y dónde
    #name es un nombre especial que Odoo usará para referenciar el objeto
    name = fields.Char('Nombre', required = True, help='Nombre del contacto')
    telefono = fields.Char('Teléfono', required = True, size=9, help='9 dígitos')
```

**agenda\_agenda\_view.xml / views.xml**

```
<odoo>
  <data>

  <!-- Vista árbol -->
  <record model="ir.ui.view" id="agenda.agenda_list_view">
    <field name="name">agenda.view.tree</field>
    <field name="model">agenda.agenda</field>
    <field name="arch" type="xml">
      <tree>
        <field name="name"/>
        <field name="telefono"/>
      </tree>
    </field>
  </record>

  <!-- Vista de formulario -->
  <record model="ir.ui.view" id="agenda.agenda_form_view">
    <field name="name">agenda.view.form</field>
    <field name="model">agenda.agenda</field>
    <field name="arch" type="xml">
      <form string="Nuevo registro">
        <!-- Insertamos el formulario en una hoja para que no ocupe toda la pantalla -->
        <sheet>
          <!-- Cabecera -->
          <div class="oe_title">
            <h1>Nuevo contacto</h1>
          </div>
          <group>
            <field name="name" placeholder="Nombre del contacto"/>
            <field name="telefono" placeholder="000000000"/>
          </group>
        </sheet>
      </form>
    </field>
  </record>

  <!-- Acciones de menú -->
  <record model="ir.actions.act_window" id="agenda.action_window">
    <field name="name">Listado</field>
    <field name="res_model">agenda.agenda</field>
    <field name="view_mode">tree,form</field>
  </record>

  <!-- Menú raíz -->
  <menuitem name="Agenda" id="agenda.menu_root" action="agenda.action_window"/>

</data>
</odoo>
```

**ir.model.access.csv**

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_agenda_agenda,agenda.agenda,model_agenda_agenda,base.group_user,1,1,1,1
access_agenda_agenda_usuario,agenda.Usuario,model_agenda_agenda,agenda.group_agenda_usuario,1,1,1,1
```

**agenda\_security.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <data>

    <!-- Categoría de seguridad del módulo -->
    <record model="ir.module.category" id="agenda.module_category_agenda">
      <field name="name">Agenda</field>
      <field name="description">Gestión de números telefónicos</field>
      <field name="sequence">100</field>
    </record>

    <!-- Grupos de seguridad -->
    <record model="res.groups" id="group_agenda_usuario">
      <field name="name">Usuario</field>
      <field name="category_id" ref="agenda.module_category_agenda" />
      <field name="comment">Usuario con acceso a la agenda</field>
      <field name="implied_ids"
eval="[(4,ref('base.user_root')), (4,ref('base.group_user')), (4,ref('base.user_admin'))]"/>
    </record>

  </data>
</odoo>
```