



Apellidos, Nombre: Amado Cibreiro, Andrés



1. Proyecto de desarrollo de componentes de Odoo en implantación cliente/servidor. Incluir documentación y capturas de pantalla que sean necesarias.

Crea un componente o módulo que gestione una agenda telefónica con las siguientes características:

- Modelo y controlador:
 - Objeto en la aplicación cuya nomenclatura siga el patrón `agendalInicialesNombreCompleto` (por ejemplo, `agendaFGF`)
 - La agenda debe proporcionar al menos dos campos: `name` y `telephone`.
 - Tabla en la base de datos con los datos del objeto.
- Vista:
 - Menú en la aplicación que enlace al objeto. El nombre del mismo debe seguir el patrón `AgendalIniciales NombreCompleto` (por ejemplo, `Agenda FGF`).
 - Vista árbol con los datos del objeto.
 - Vista formulario con los datos del objeto.

Módulo de ejemplo:

1. Vista en forma de árbol:

Agenda FGF		Opciones	YourCompany		Mitchell Admin (odoo)
Agenda FGF		Mostrar	Buscar...		Q
Crear			Filtros Agrupar por Favoritos		1-1 / 1 < >
<input type="checkbox"/>	Nombre	Teléfono			
<input type="checkbox"/>	Fernando	5555555555			

2. Vista de entrada de datos:

Agenda FGF Opciones

Agenda FGF / Nuevo

Guardar Descartar

Nombre

Teléfono

3. Descripción del módulo:

Aplicaciones / Agenda FGF

Guardar Descartar 2 / 3

Agenda FGF

Por Fernando Gómez Folgar

Actualizar Desinstalar

Información Datos técnicos Características instaladas

Sitio web	http://www.ieslosada.com	Nombre técnico	agenda_fgf
Categoría	Sin categoría	Licencia	LGPL versión 3
Resumen	Módulo de gestión de Agenda	Última versión	15.0.0.1

Agenda FGF permite gestionar los nombres y los números de teléfono de los contactos

Para implementar el módulo hay que tener en cuenta las instrucciones siguientes:

(CA5.1 Recoñecéronse as sentenzas da linguaxe propia do sistema ERP-CRM. 15% ME)

1. Crear la plantilla del módulo mediante la opción scaffold de Odoo.

(CA5.2 Utilizáronse os elementos de programación da linguaxe para crear compoñentes de manipulación de datos. 20% ME)

2. Implementar el módulo empleando los elementos del lenguaje de programación (python, xml, csv).

(CA5.3 Modificáronse compoñentes de software para lle engadir funcionalidade nova ao sistema. 20% ME)

3. Modificar el componente creado definir el icono del módulo y su información personalizada en relación al autor del módulo, sitio web, resumen corto e información ampliada indicativa del propósito del mismo.

(CA5.4 Integráronse os novos compoñentes de software no sistema ERP-CRM. 20% ME)

4. Instalar el módulo en una instalación cliente-servidor de Odoo.

(CA5.5 Verifícase o correcto funcionamento dos compoñentes creados. 15%)

5. Diseñar, documentar y realizar pruebas para verificar el correcto funcionamiento del componente.

(CA5.6 Documentáronse todos os compoñentes creados ou modificados 10%)

6. Documentar el código fuente que se ha implementado tanto para la vista como el modelo y el controlador.

Forma de entrega:

- Entrega el proyecto comprimido en formato zip empleando la nomenclatura UD5_Apellido1_Apellido2_Nombre.zip con la siguiente estructura interna:
 - Módulo distribuible en un archivo comprimido con la nomenclatura UD5_Apellido1_Apellido2_Nombre.tar.gz
 - Documentación del proyecto en formato pdf con la nomenclatura UD5_Apellido1_Apellido2_Nombre.pdf

Sumario

1 Creación de nuevo módulo.....	4
1.1 Comando scaffold.....	4
1.2 Creación de modelo y vistas.....	5
1.3 Actualizar lista de aplicaciones.....	8
1.4 Modificación de información del módulo.....	12
1.5 Modificación de icono.....	14
2 Instalación del módulo en una instalación cliente-servidor.....	16
3 Diseño y documentación de pruebas para verificar el funcionamiento correcto del módulo.....	22
4 4. Documentación del código fuente.....	24

Índice de figuras

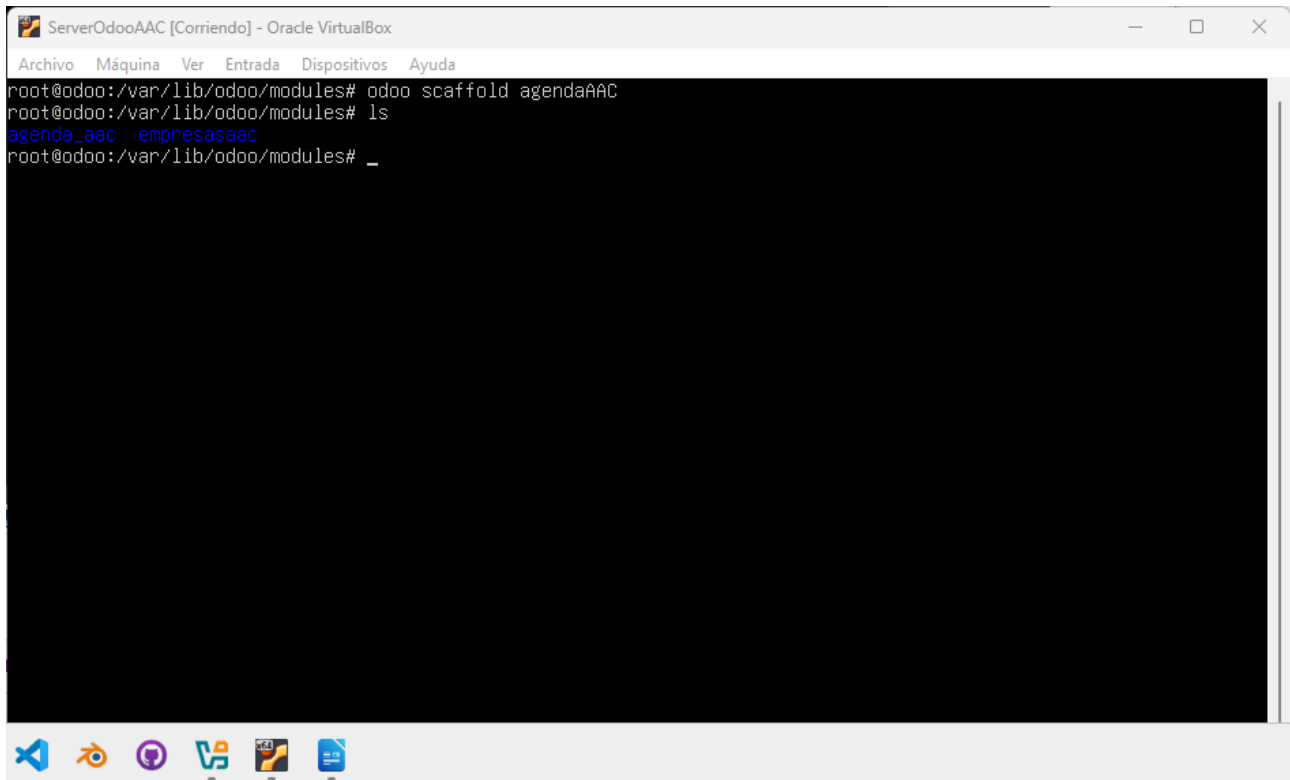
Figura 1: Comando scaffold.....	5
Figura 2: Nuevo modelo.....	6
Figura 3: Nueva vista.....	8
Figura 4: Descomentar línea security.....	9
Figura 5: Actualizar lista de aplicaciones.....	10
Figura 6: Actualizamos el módulo.....	11
Figura 7: Modificación de archivo <code>__manifest__.py</code>	13
Figura 8: Contenido modificado.....	14
Figura 9: Imagen en <code>/static/description</code>	15
Figura 10: Ejemplo icono.....	16
Figura 11: Archivo a exportar.....	17
Figura 12: Importación de carpeta <code>agenda_rac</code>	18
Figura 13: Exportación de archivo <code>.zip</code>	19
Figura 14: Colocación de archivo en ruta <code>/etc/odoo/addons</code>	20
Figura 15: Actualización del nuevo módulo.....	21
Figura 16: Nuevo módulo externo instalado correctamente.....	22
Figura 17: Mejora de diseño y seguridad del módulo.....	23
Figura 18: Prueba de la aplicación.....	24
Figura 19: <code>view.xml</code>	25
Figura 20: <code>model.py</code>	26

1 Creación de nuevo módulo

1.1 Comando scaffold

Para crear el módulo utilizaremos el comando scaffold para crear la plantilla

```
odoo scaffold agendaAAC
```



The screenshot shows a terminal window titled "ServerOdooAAC [Corriendo] - Oracle VirtualBox". The terminal output is as follows:

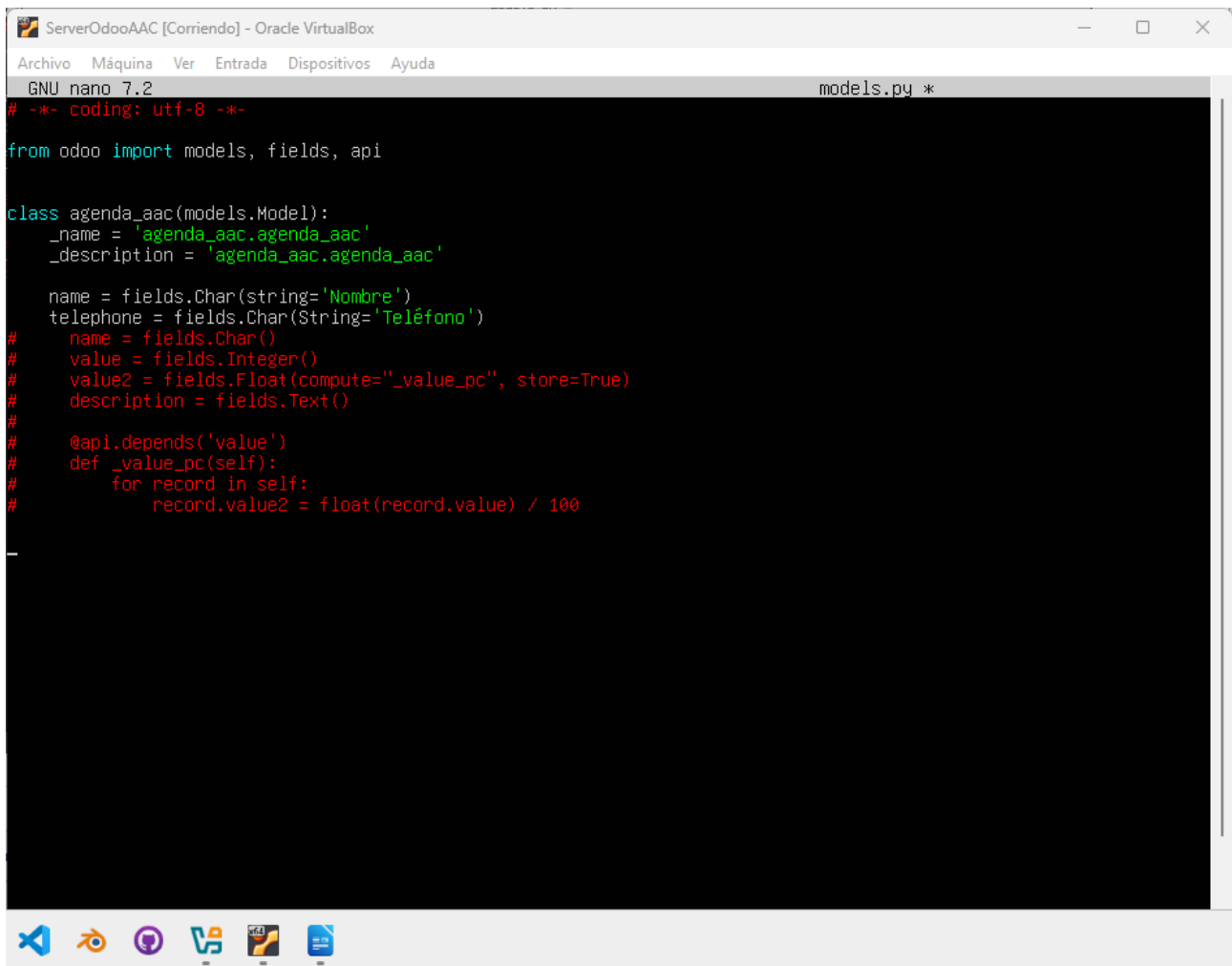
```
root@odoo:/var/lib/odoo/modules# odoo scaffold agendaAAC
root@odoo:/var/lib/odoo/modules# ls
agenda_aac  empresasaac
root@odoo:/var/lib/odoo/modules# _
```

The terminal window has a menu bar with "Archivo", "Máquina", "Ver", "Entrada", "Dispositivos", and "Ayuda". The bottom of the window shows a taskbar with icons for VS Code, Docker, a terminal, a file manager, and a web browser.

Figura 1: Comando scaffold

1.2 Creación de modelo y vistas

Modificaremos el archivo “agenda_aac/models/models.py” para crear un nuevo modelo.



```
ServerOdooAAC [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 7.2 models.py *
# -*- coding: utf-8 -*-

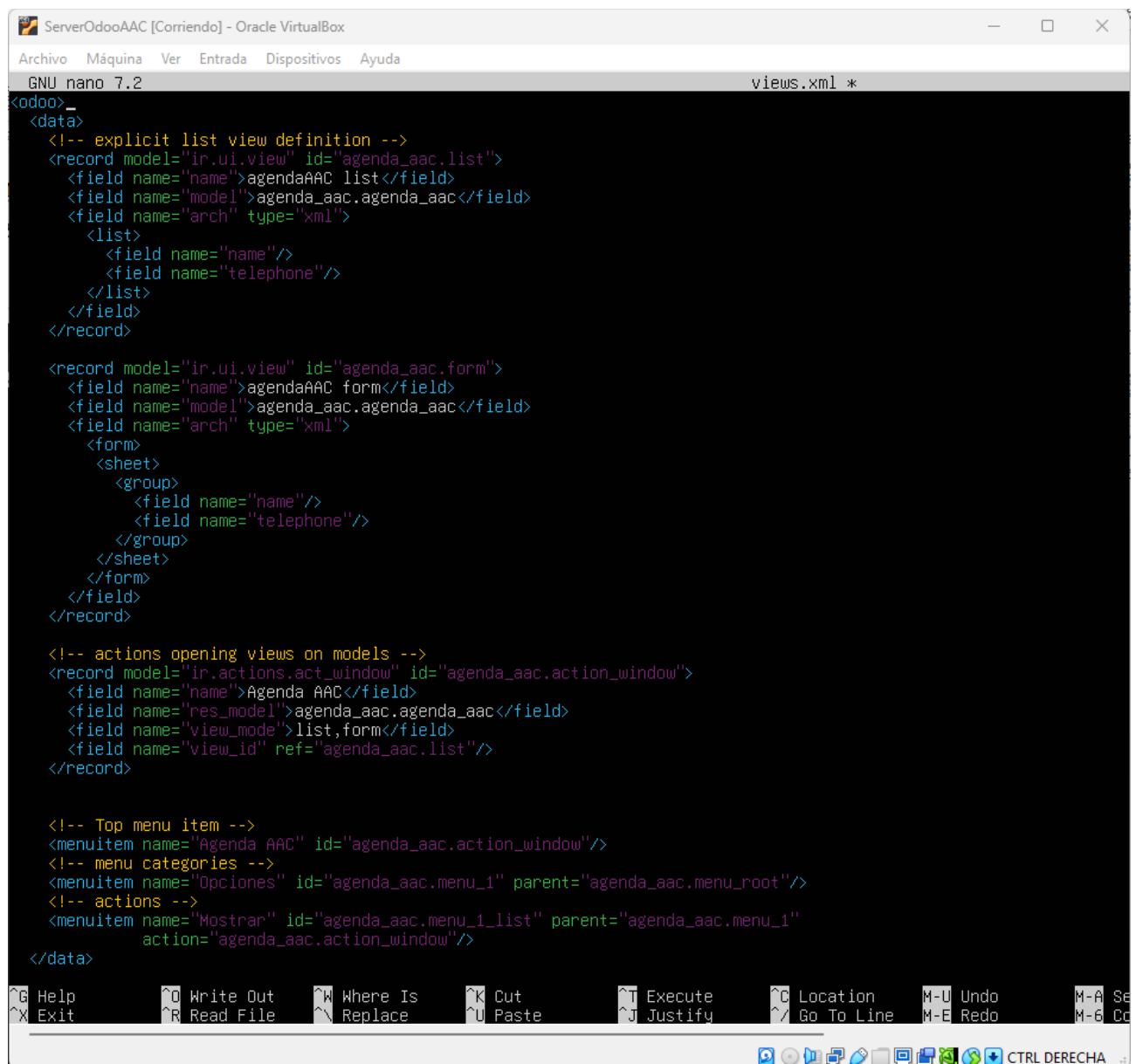
from odoo import models, fields, api

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

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

Figura 2: Nuevo modelo

El siguiente paso será modificar el archivo “views.xml” que estará en la carpeta “views”. En el archivo definiremos tanto el tipo de vista que será (lista y formulario), sus campos, los menús y asignarlo al modelo que hemos creado antes.



```
GNU nano 7.2 views.xml *
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="agenda_aac.list">
      <field name="name">agendaAAC list</field>
      <field name="model">agenda_aac.agenda_aac</field>
      <field name="arch" type="xml">
        <list>
          <field name="name"/>
          <field name="telephone"/>
        </list>
      </field>
    </record>

    <record model="ir.ui.view" id="agenda_aac.form">
      <field name="name">agendaAAC form</field>
      <field name="model">agenda_aac.agenda_aac</field>
      <field name="arch" type="xml">
        <form>
          <sheet>
            <group>
              <field name="name"/>
              <field name="telephone"/>
            </group>
          </sheet>
        </form>
      </field>
    </record>

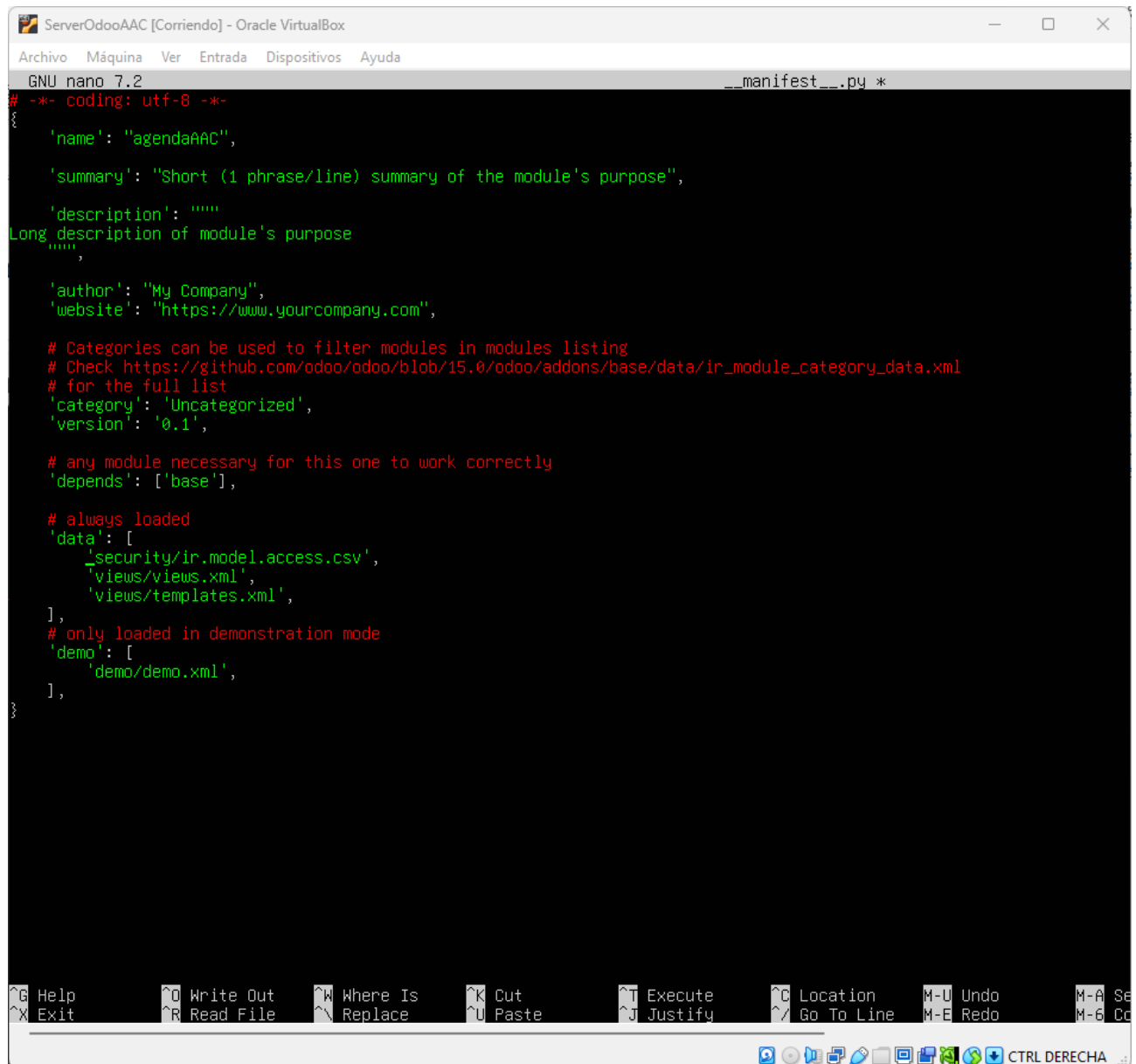
    <!-- actions opening views on models -->
    <record model="ir.actions.act_window" id="agenda_aac.action_window">
      <field name="name">Agenda AAC</field>
      <field name="res_model">agenda_aac.agenda_aac</field>
      <field name="view_mode">list,form</field>
      <field name="view_id" ref="agenda_aac.list"/>
    </record>

    <!-- Top menu item -->
    <menuitem name="Agenda AAC" id="agenda_aac.action_window"/>
    <!-- menu categories -->
    <menuitem name="Opciones" id="agenda_aac.menu_1" parent="agenda_aac.menu_root"/>
    <!-- actions -->
    <menuitem name="Mostrar" id="agenda_aac.menu_1_list" parent="agenda_aac.menu_1"
      action="agenda_aac.action_window"/>
  </data>
</odoo>
```

Figura 3: Nueva vista

El siguiente paso será acceder al archivo “__manifest__.py” y descomentar la línea de “security”.

Esta línea permitirá acceder al archivo que se encuentra en “security”. En ella podremos gestionar los permisos relacionados con el módulo.



```
GNU nano 7.2 __manifest__.py *
# -*- coding: utf-8 -*-
{
    'name': 'agendaAAC',
    '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_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',
        'views/views.xml',
        'views/templates.xml',
    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
}
```

Figura 4: Descomentar línea security

1.3 Actualizar lista de aplicaciones

Lo siguiente que haremos será reiniciar odoo con `systemctl restart odoo` y luego lo abriremos en nuestro equipo cliente.

Una vez dentro, activamos el modo desarrollador, nos dirigimos a “Aplicaciones” y luego pulsamos en “Actualizar lista de aplicaciones”.

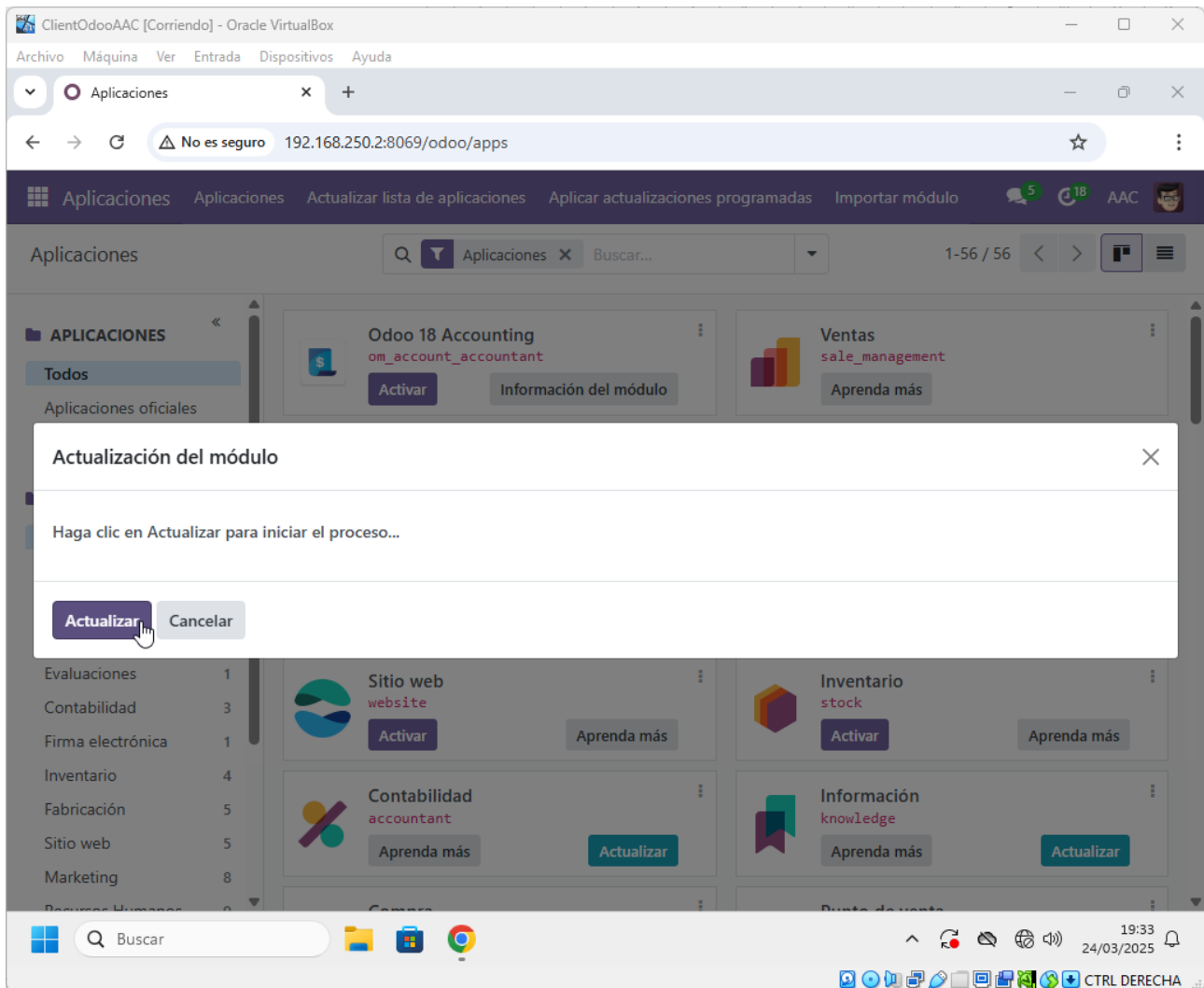


Figura 5: Actualizar lista de aplicaciones

Si lo buscamos nos debería ya de aparecer. Pulsamos en “Actualizar” y ya tendríamos nuestro módulo instalado.

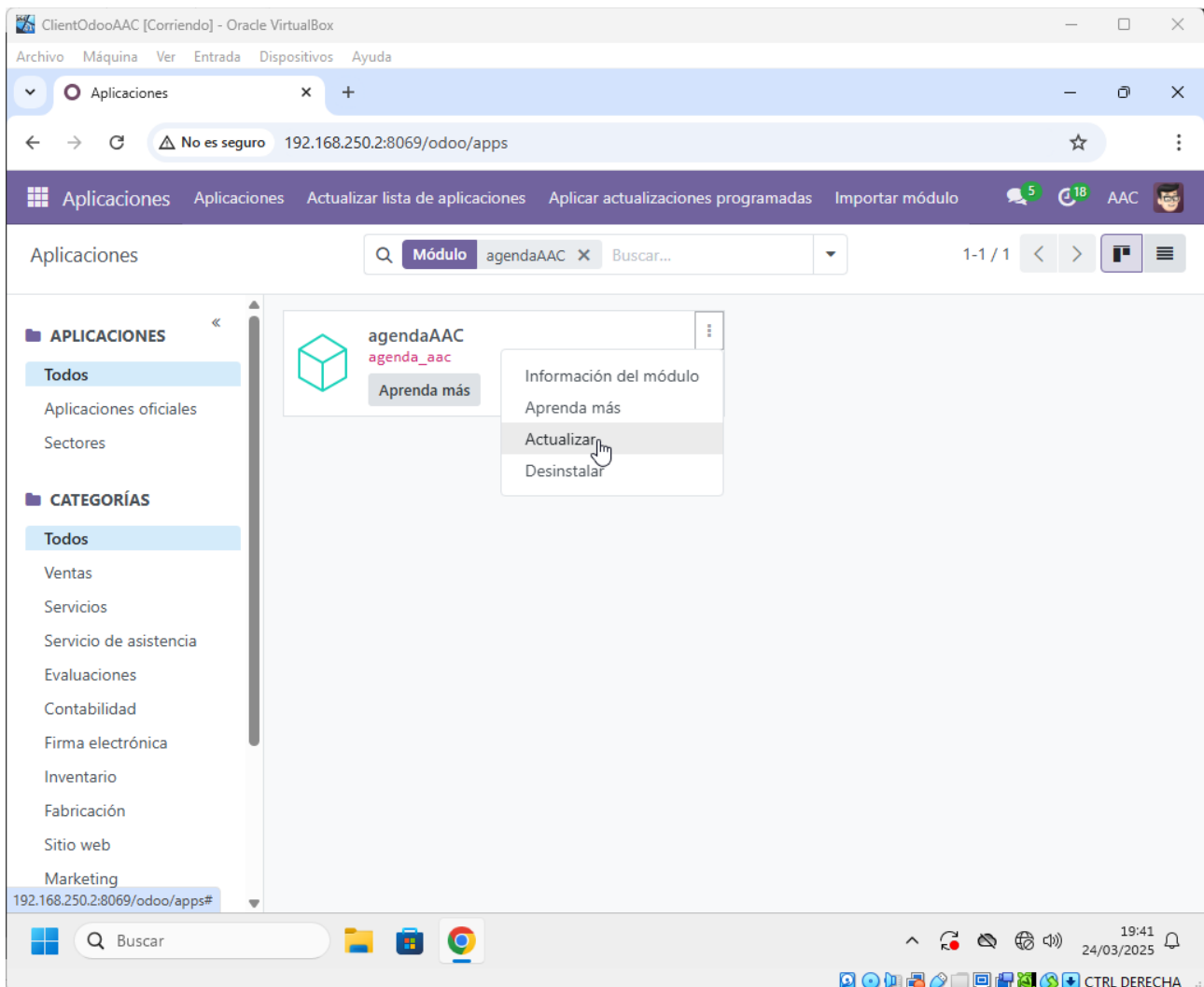
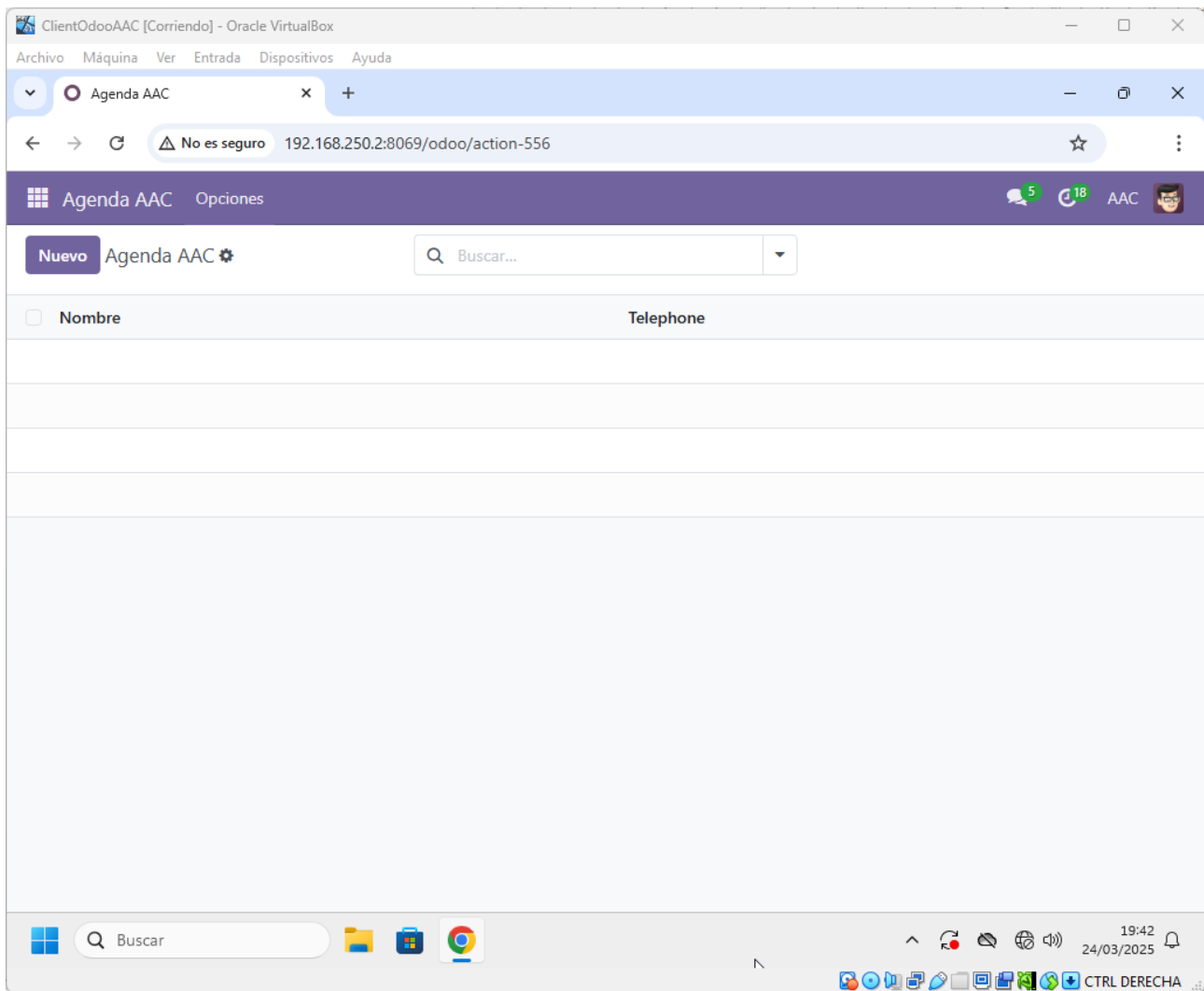


Figura 6: Actualizamos el módulo

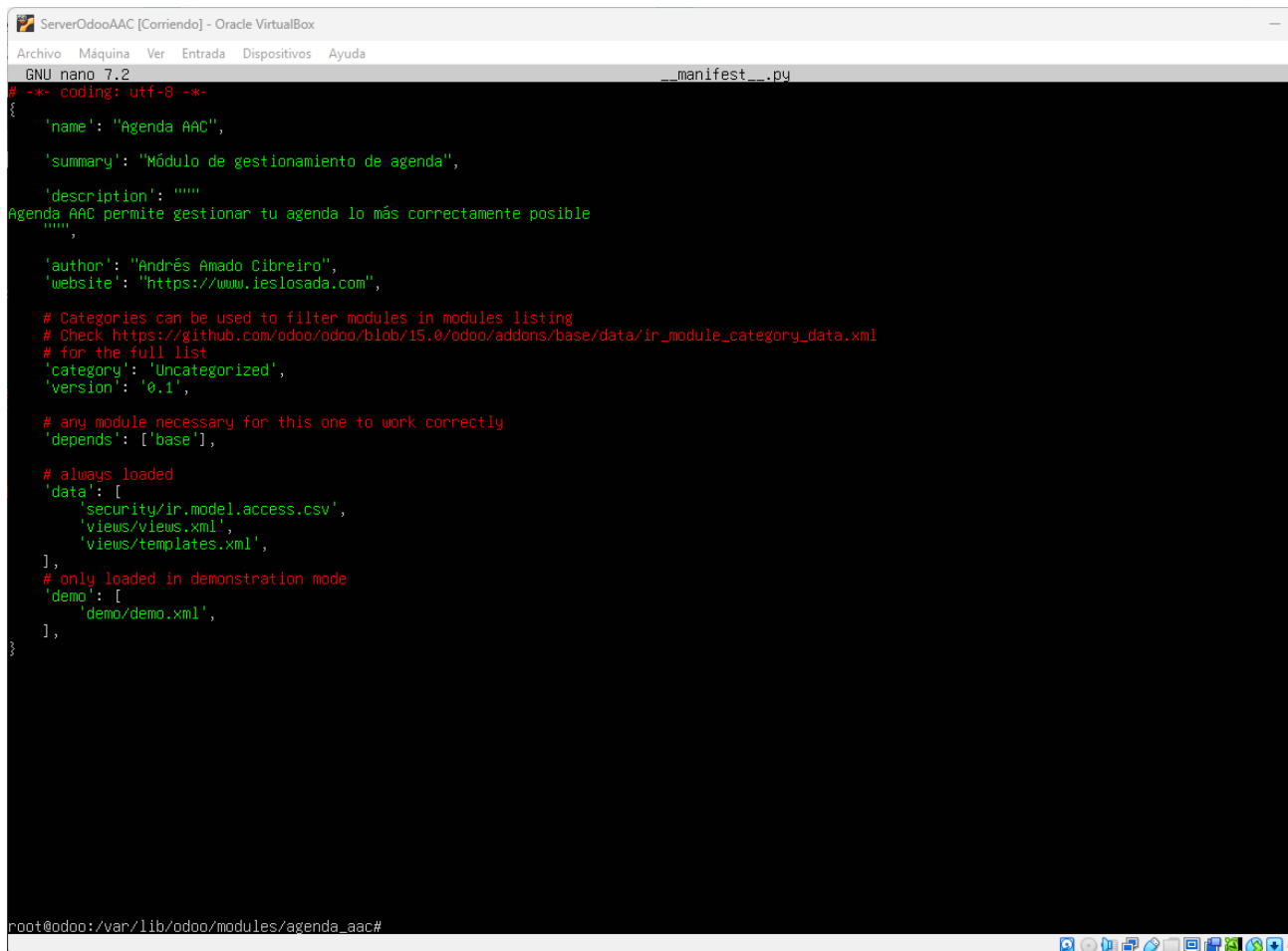
Se tendría que ver de la siguiente forma:



1.4 Modificación de información del módulo

El siguiente paso será cambiar la información relacionada con nuestro módulo.

Para poder cambiar esto, deberemos modificar el archivo “__manifest__.py” y agregar los campos que queramos.



```
ServerOdooAAC [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 7.2 __manifest__.py
# -*- coding: utf-8 -*-
{
    'name': 'Agenda AAC',
    'summary': 'Módulo de gestionamiento de agenda',
    'description': """
Agenda AAC permite gestionar tu agenda lo más correctamente posible
""",
    'author': "Andrés Amado Cibreiro",
    'website': "https://www.ieslosada.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',
        'views/views.xml',
        'views/templates.xml',
    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
}
```

root@odoo:/var/lib/odoo/modules/agenda_aac#

Figura 7: Modificación de archivo __manifest__.py

Si accedemos a más información sobre el módulo, veremos que el contenido ha sido modificado.

Si esto no ocurre, deberemos borrar la caché del módulo. Para hacer esto, deberemos desinstalar el módulo (antes hacer una copia) y volver a instalarlo.

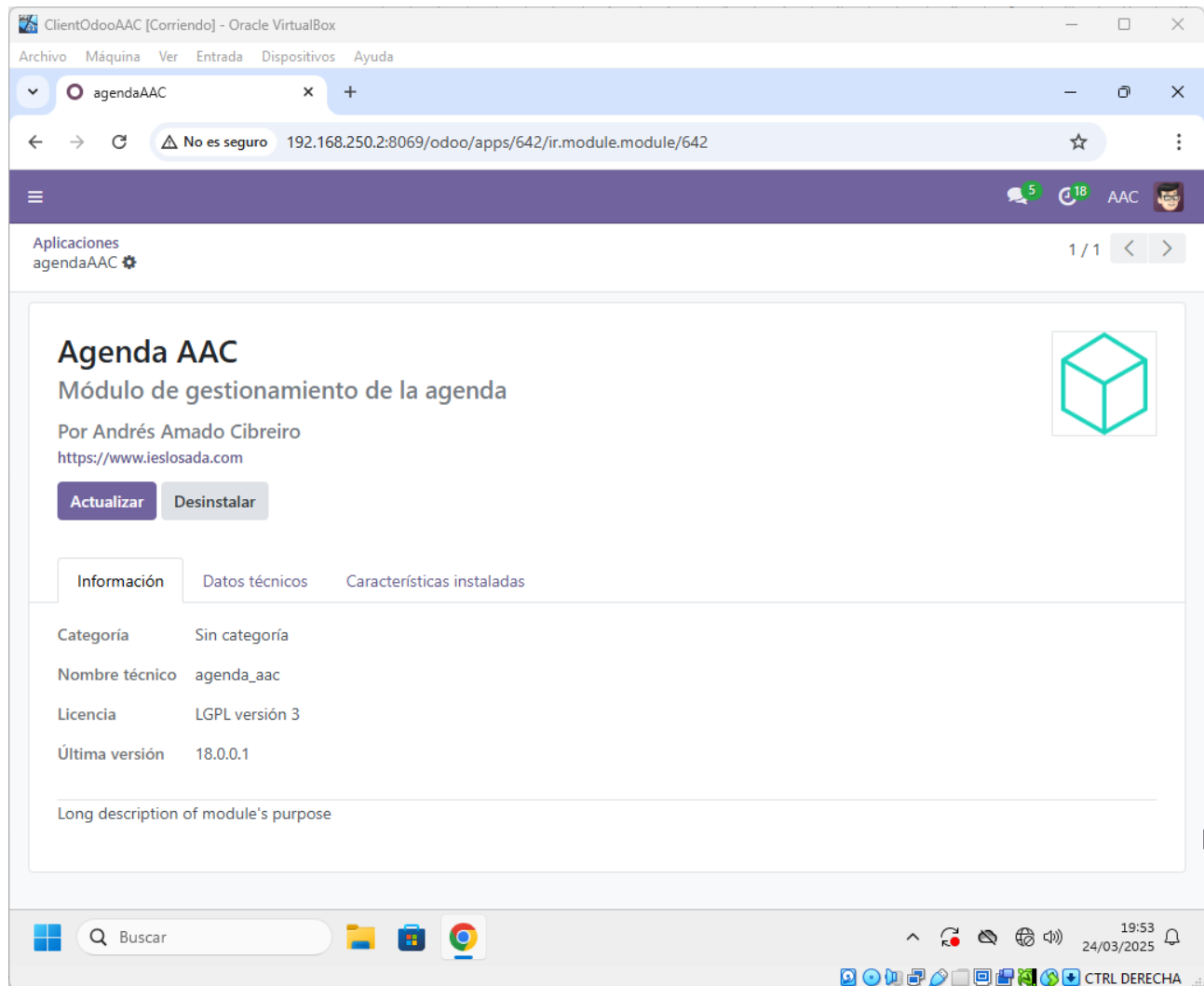


Figura 8: Contenido modificado

1.5 Modificación de icono

Si quisiéramos cambiar el icono de nuestro módulo, tendríamos que hacer lo siguiente:

En la máquina del servidor, crearemos una nueva carpeta llamada “static” y dentro de ella otra llamada “description” en el directorio donde está nuestro módulo.

En ella, colocaremos el archivo png que queremos que actúe como icono.

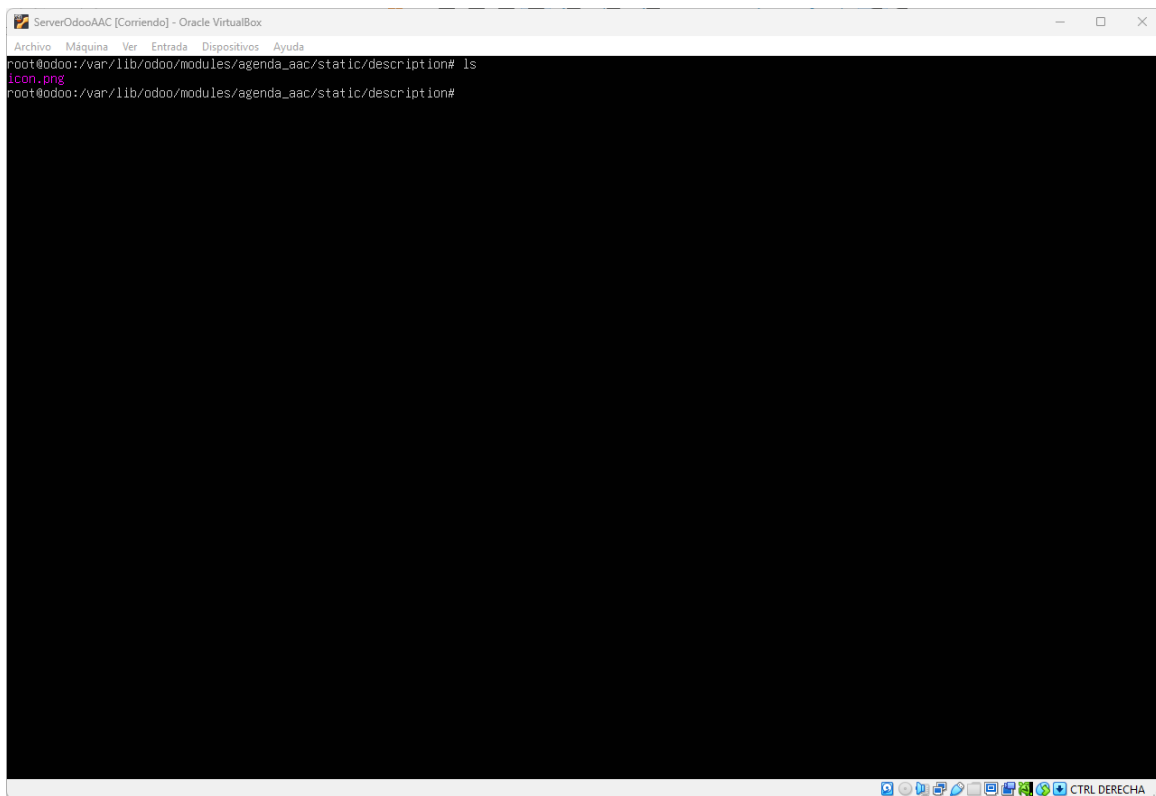


Figura 9: Imagen en /static/description

De esta forma si actualizamos la lista de aplicaciones, veremos que el icono se ha cambiado de forma correcta.

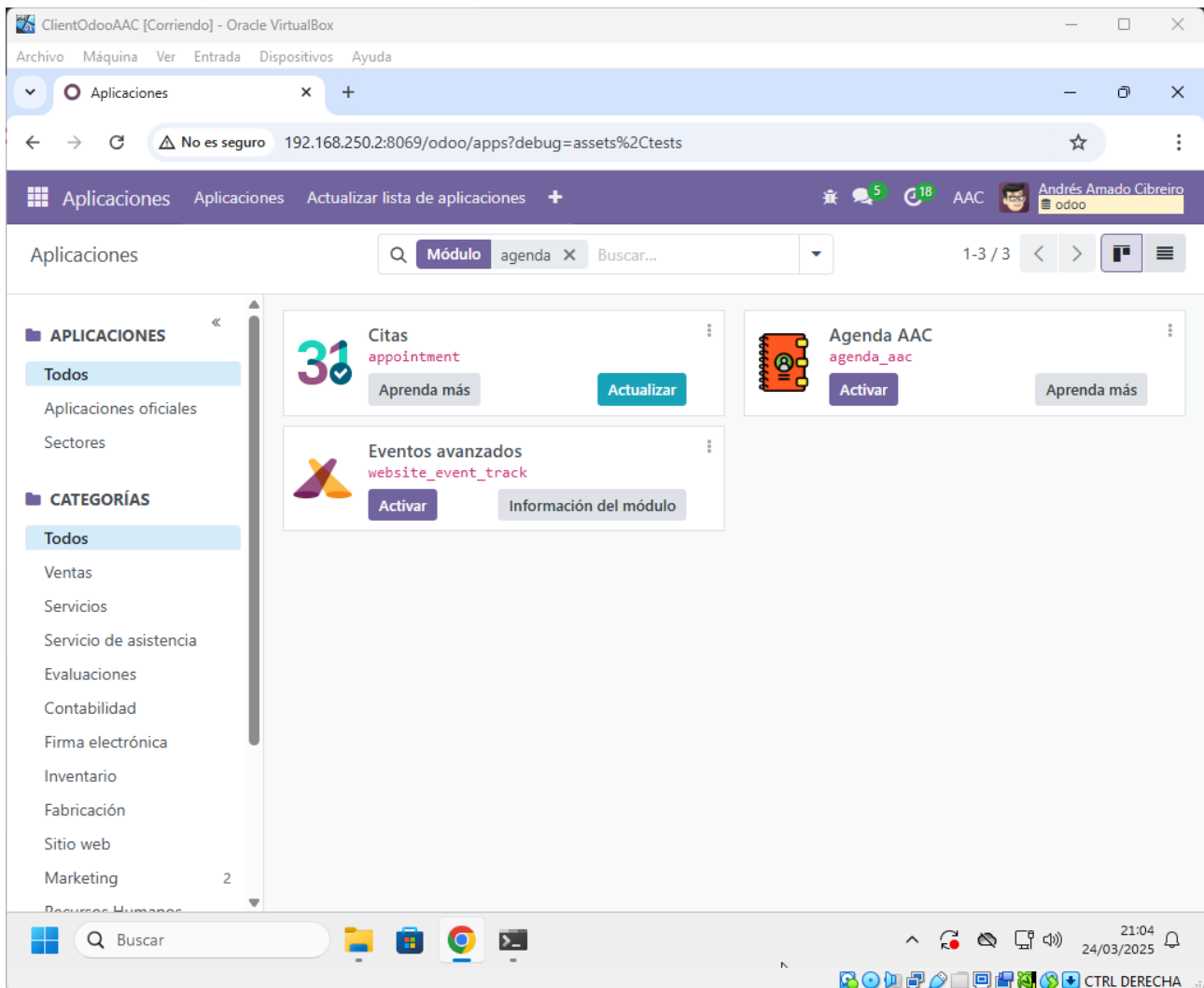


Figura 10: Ejemplo icono

2 Instalación del módulo en una instalación cliente-servidor

Lo primero que deberemos hacer será exportar la carpeta de nuestro módulo al equipo cliente con scp.

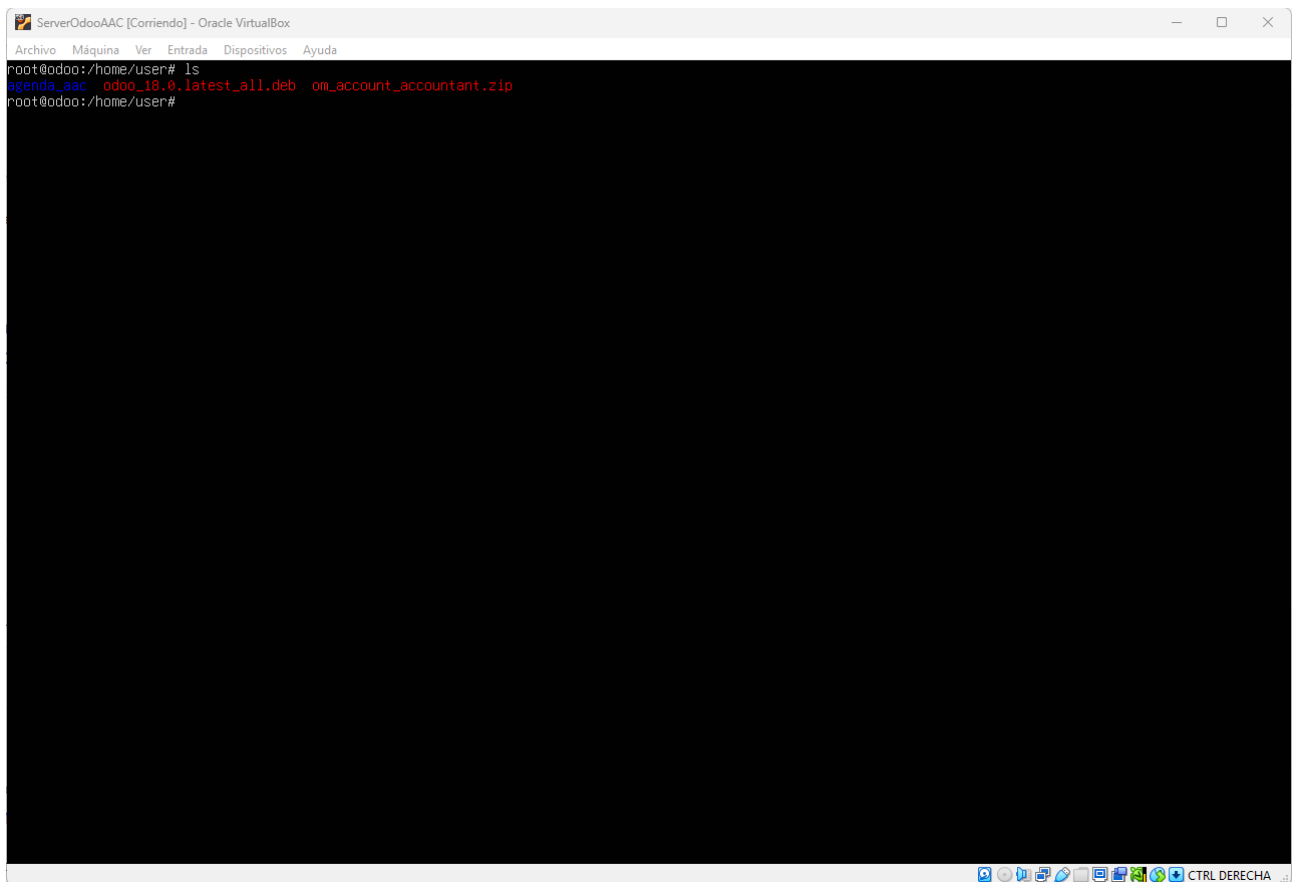
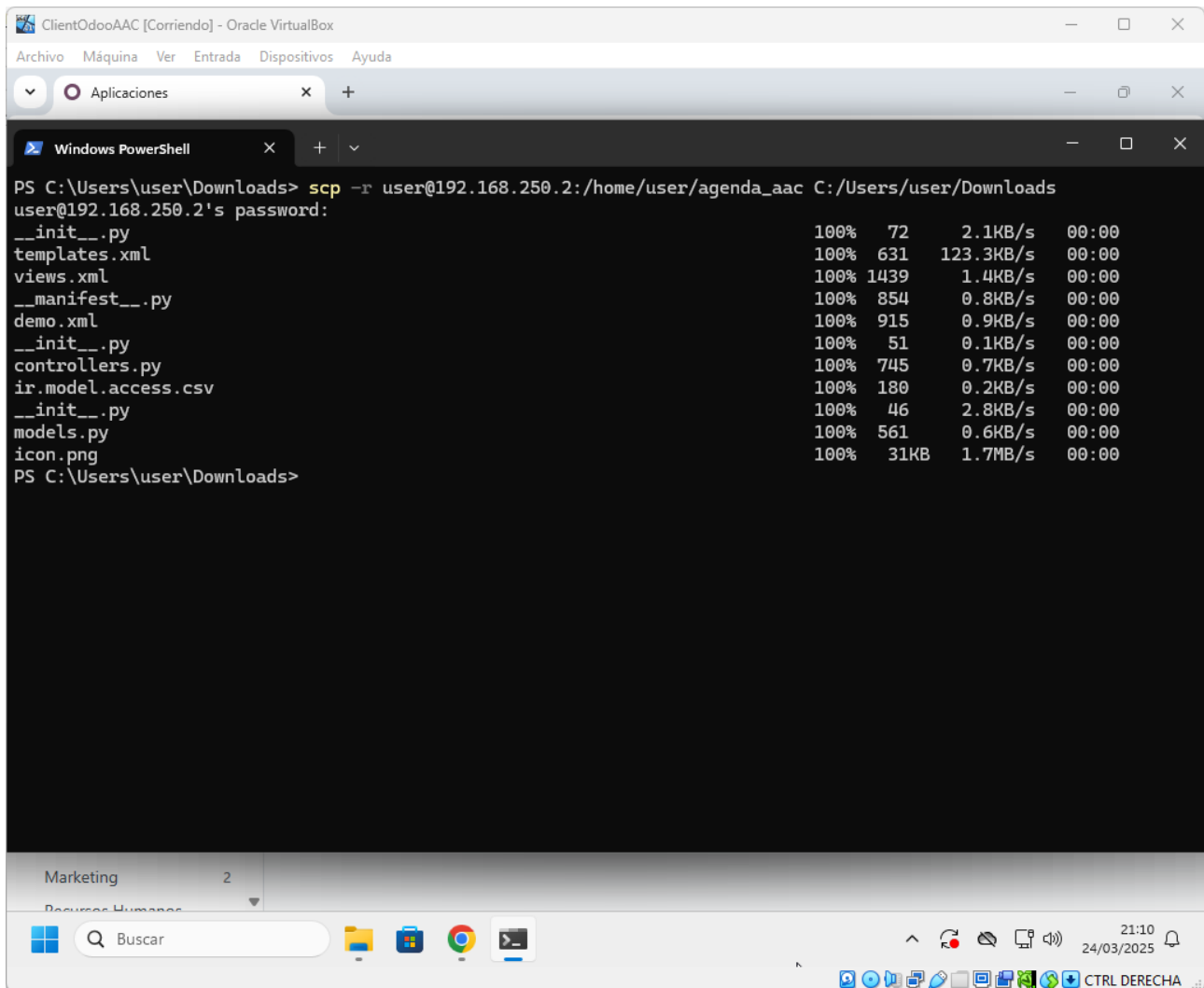


Figura 11: Archivo a exportar

Haremos el siguiente comando scp en la máquina cliente para recoger el directorio:

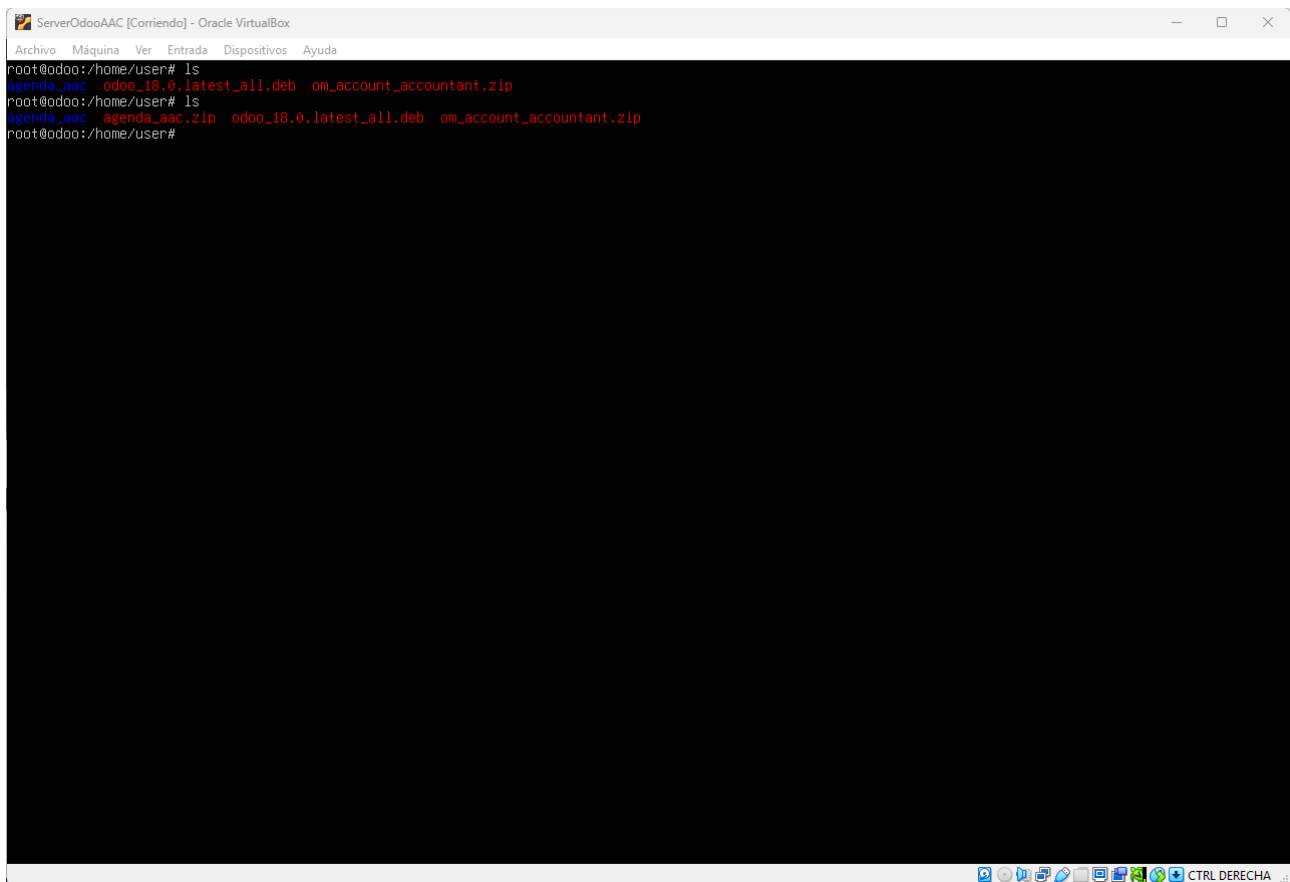


The screenshot shows a Windows PowerShell terminal window titled "ClientOdoAAC [Corriendo] - Oracle VirtualBox". The terminal displays the execution of an scp command to copy a directory from a remote server to the local machine. The command is: `PS C:\Users\user\Downloads> scp -r user@192.168.250.2:/home/user/agenda_aac C:/Users/user/Downloads`. The output shows the progress of the copy operation for various files, including `__init__.py`, `templates.xml`, `views.xml`, `__manifest__.py`, `demo.xml`, `__init__.py`, `controllers.py`, `ir.model.access.csv`, `__init__.py`, `models.py`, and `icon.png`. The progress bar indicates 100% completion for all files. The terminal also shows the prompt `PS C:\Users\user\Downloads>` after the command execution.

```
PS C:\Users\user\Downloads> scp -r user@192.168.250.2:/home/user/agenda_aac C:/Users/user/Downloads
user@192.168.250.2's password:
__init__.py                                100% 72      2.1KB/s  00:00
templates.xml                             100% 631     123.3KB/s 00:00
views.xml                                 100% 1439    1.4KB/s  00:00
__manifest__.py                           100% 854     0.8KB/s  00:00
demo.xml                                  100% 915     0.9KB/s  00:00
__init__.py                               100% 51      0.1KB/s  00:00
controllers.py                            100% 745     0.7KB/s  00:00
ir.model.access.csv                       100% 180     0.2KB/s  00:00
__init__.py                               100% 46      2.8KB/s  00:00
models.py                                 100% 561     0.6KB/s  00:00
icon.png                                  100% 31KB    1.7MB/s  00:00
PS C:\Users\user\Downloads>
```

Figura 12: Importación de carpeta agenda_rac

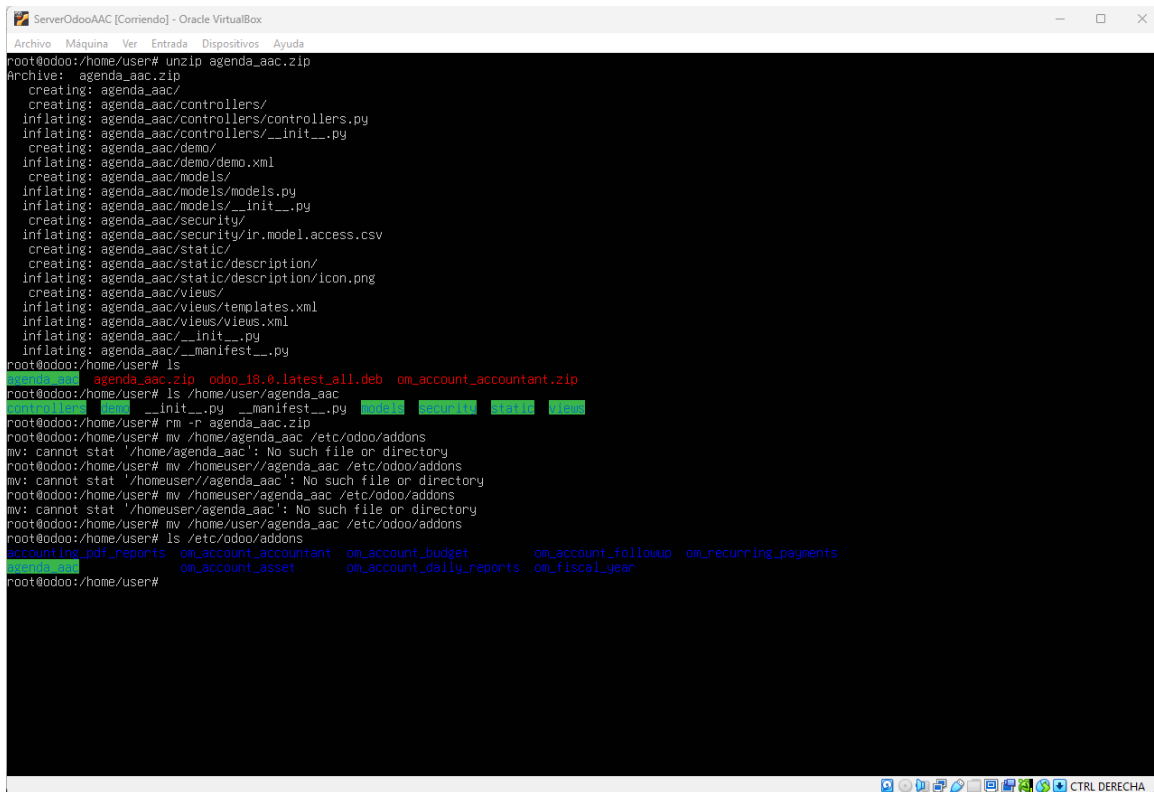
Con la carpeta que hemos importado, haremos un comprimido .zip y lo pasaremos otra vez a la máquina del servidor simulando que es un módulo externo.



```
ServerOdooAAC [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/home/user# ls
agenda_aac  odoo_18.0.latest_all.deb  om_account_accountant.zip
root@odoo:/home/user# ls
agenda_aac  agenda_aac.zip  odoo_18.0.latest_all.deb  om_account_accountant.zip
root@odoo:/home/user#
```

Figura 13: Exportación de archivo .zip

Una vez tengamos el archivo en la máquina del servidor, lo moveremos a la carpeta donde se almacenan los demás módulos externos `/etc/odoo/addons` y descomprimos el archivo con el comando `unzip`.



```
ServerOdooAAC [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/home/user# unzip agenda_aac.zip
Archive:  agenda_aac.zip
  creating: agenda_aac/
  creating: agenda_aac/controllers/
  inflating: agenda_aac/controllers/controllers.py
  inflating: agenda_aac/controllers/__init__.py
  creating: agenda_aac/demo/
  inflating: agenda_aac/demo/demo.xml
  creating: agenda_aac/models/
  inflating: agenda_aac/models/models.py
  inflating: agenda_aac/models/__init__.py
  creating: agenda_aac/security/
  inflating: agenda_aac/security/ir.model.access.csv
  creating: agenda_aac/static/
  inflating: agenda_aac/static/description/
  inflating: agenda_aac/static/description/icon.png
  creating: agenda_aac/views/
  inflating: agenda_aac/views/templates.xml
  inflating: agenda_aac/views/views.xml
  inflating: agenda_aac/__init__.py
  inflating: agenda_aac/__manifest__.py
root@odoo:/home/user# ls
agenda_aac.zip  odoo-18.0.latest.all.deb  om_account_accountant.zip
root@odoo:/home/user# ls /home/user/agenda_aac
__init__.py  __manifest__.py
root@odoo:/home/user# rm -r agenda_aac.zip
root@odoo:/home/user# mv /home/agenda_aac /etc/odoo/addons
mv: cannot stat '/home/agenda_aac': No such file or directory
root@odoo:/home/user# mv /homeuser//agenda_aac /etc/odoo/addons
mv: cannot stat '/homeuser//agenda_aac': No such file or directory
root@odoo:/home/user# mv /homeuser/agenda_aac /etc/odoo/addons
mv: cannot stat '/homeuser/agenda_aac': No such file or directory
root@odoo:/home/user# mv /home/user/agenda_aac /etc/odoo/addons
root@odoo:/home/user# ls /etc/odoo/addons
accounting_pdf_reports  om_account_accountant  om_account_budget  om_account_followup  om_recurring_payments
om_account_asset        om_account_daily_reports  om_fiscal_year
```

Figura 14: Colocación de archivo en ruta `/etc/odoo/addons`

Reiniciaremos Odoo con `systemctl restart odoo` y actualizamos la lista de aplicaciones.

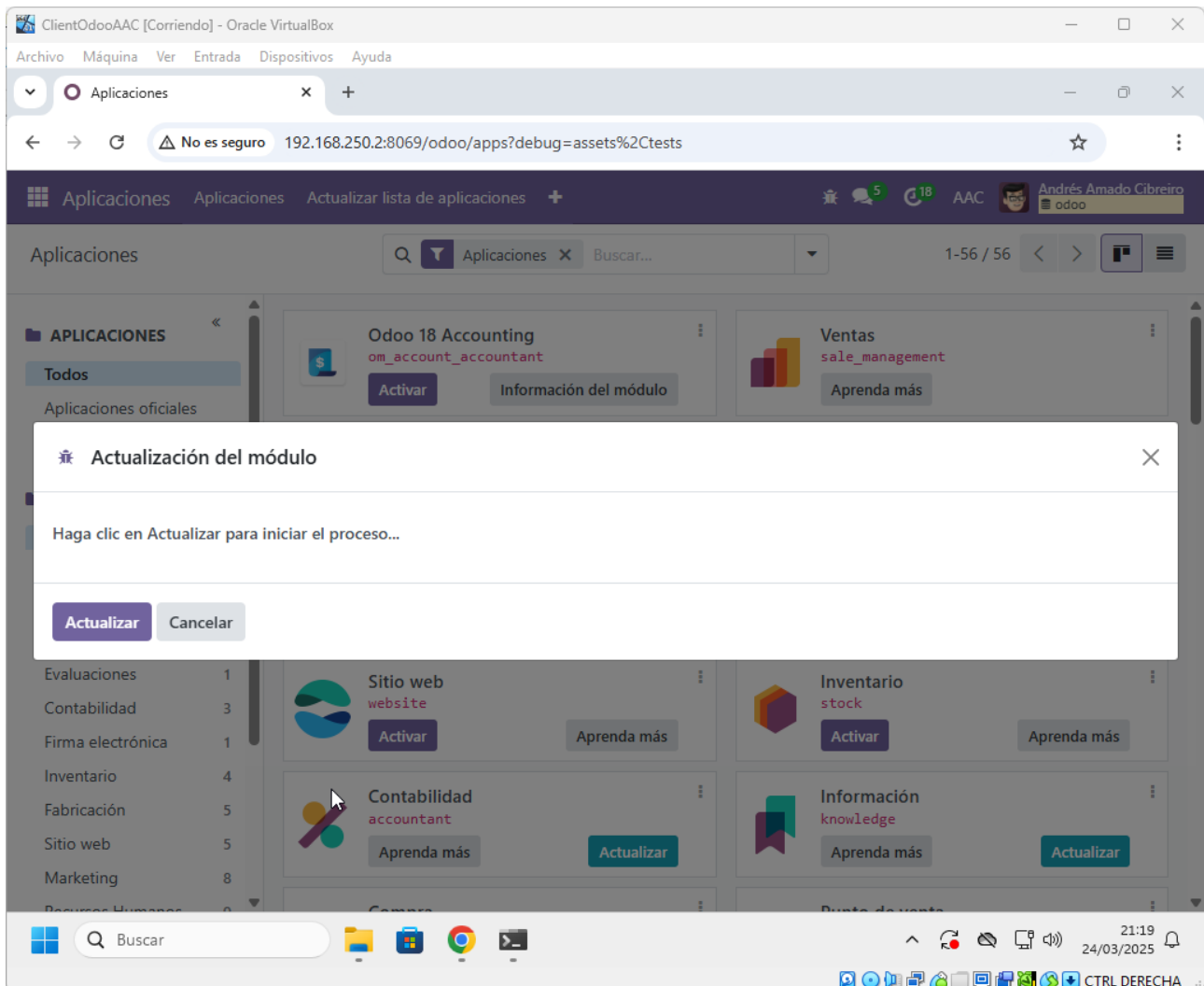


Figura 15: Actualización del nuevo módulo

Y nos aparecerá como al principio para activarlo.

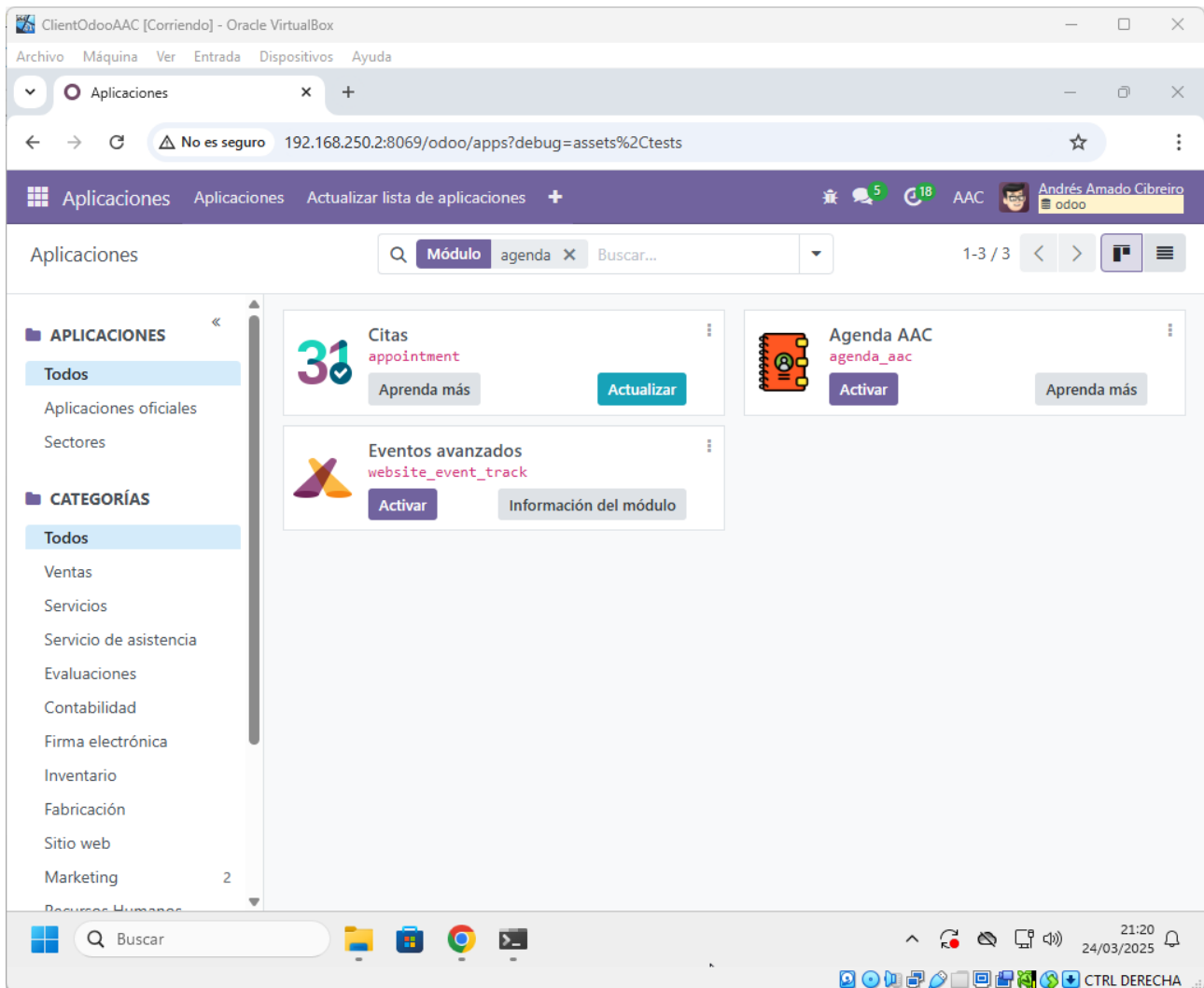


Figura 16: Nuevo módulo externo instalado correctamente

3 Diseño y documentación de pruebas para verificar el funcionamiento correcto del módulo

Para nuestra nueva aplicación, podemos aplicar algunas normas de seguridad como por ejemplo que el usuario deba rellenar todos los campos de forma obligatoria. Para esto, utilizaremos el atributo `required="true"`.

También podemos añadir unos “tooltips” para que el usuario tenga más información sobre el campo si coloca el ratón encima de el. Para aplicarlo utilizaremos el atributo `help`.

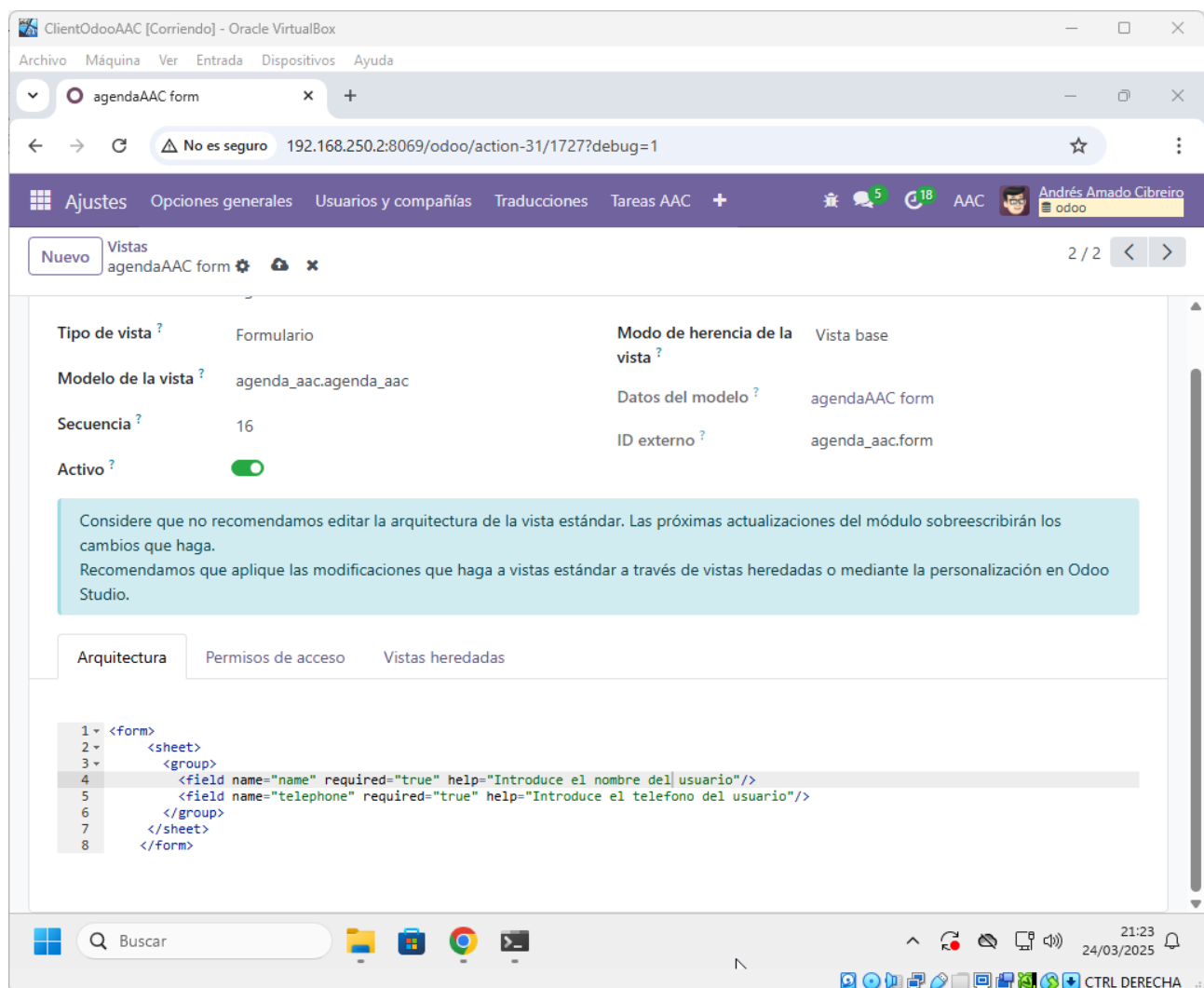


Figura 17: Mejora de diseño y seguridad del módulo

Ahora probaremos nuestra aplicación.

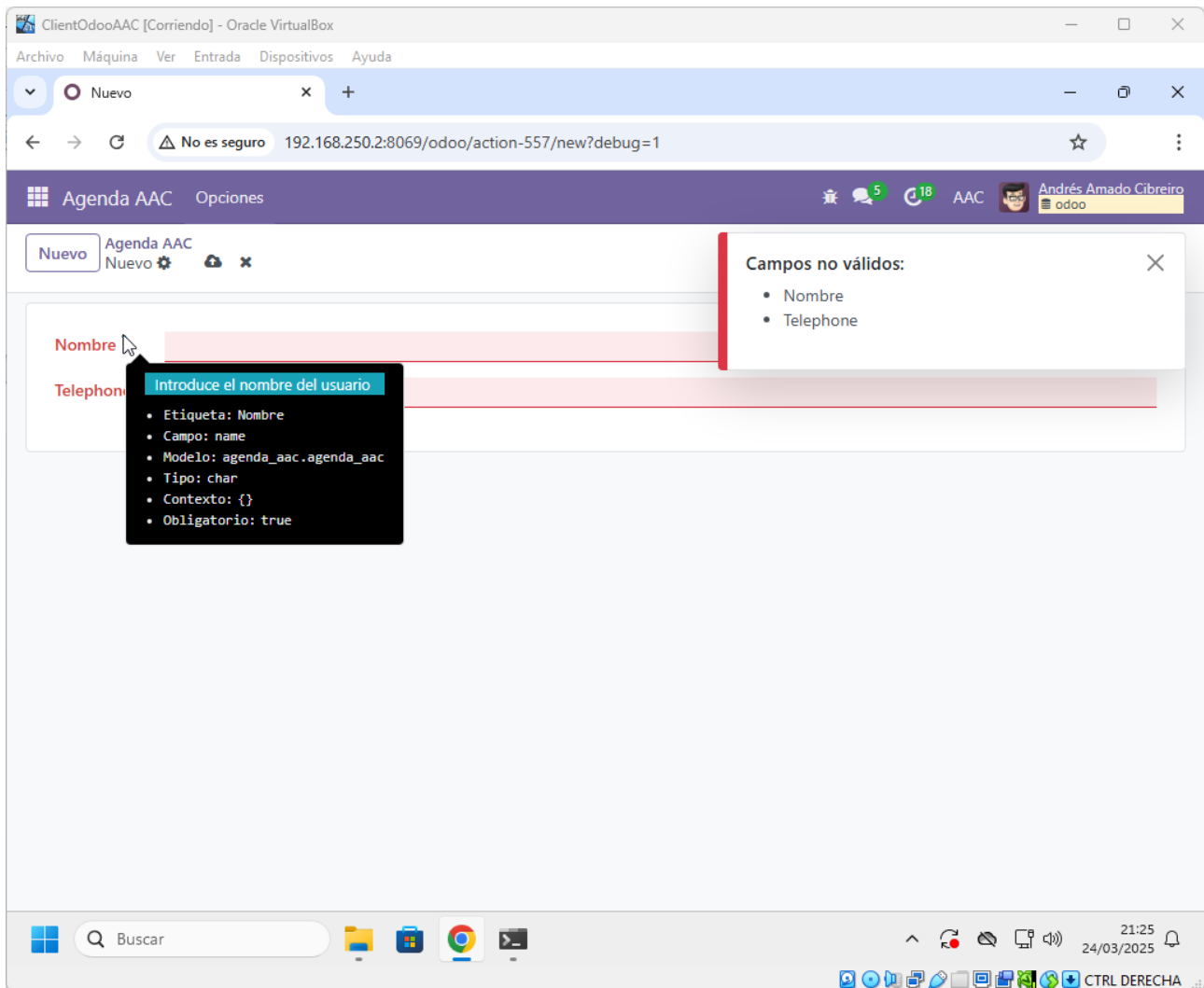
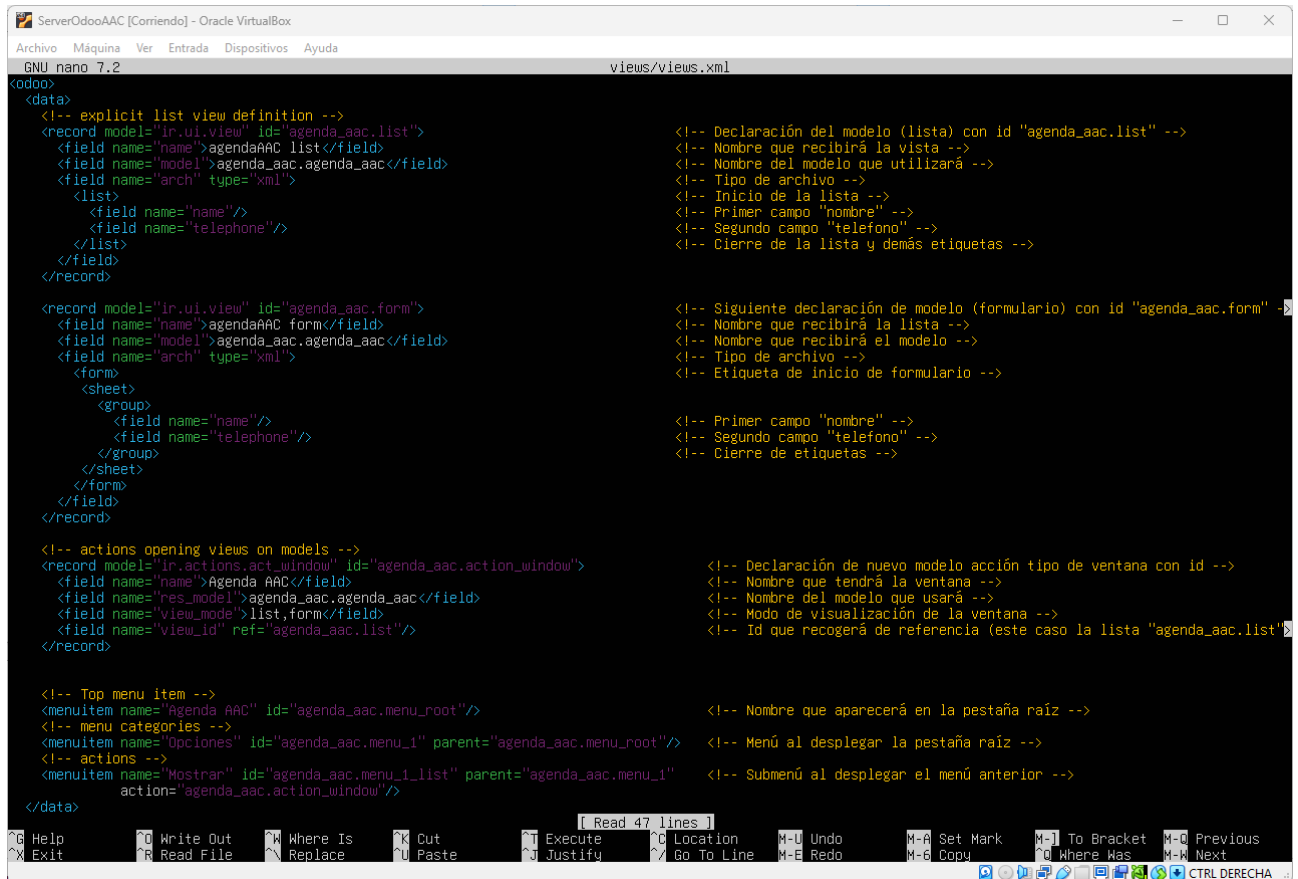


Figura 18: Prueba de la aplicación

4 4. Documentación del código fuente

El código de los siguientes archivos funcionan de la siguiente manera:

view.xml:



```
<?xml version="1.0" encoding="UTF-8" ?>
<data>
  <!-- explicit list view definition -->
  <record model="ir.ui.view" id="agenda_aac.list">
    <field name="name">agendaAAC list</field>
    <field name="model">agenda_aac.agenda_aac</field>
    <field name="arch" type="xml">
      <list>
        <field name="name"/>
        <field name="telephone"/>
      </list>
    </field>
  </record>

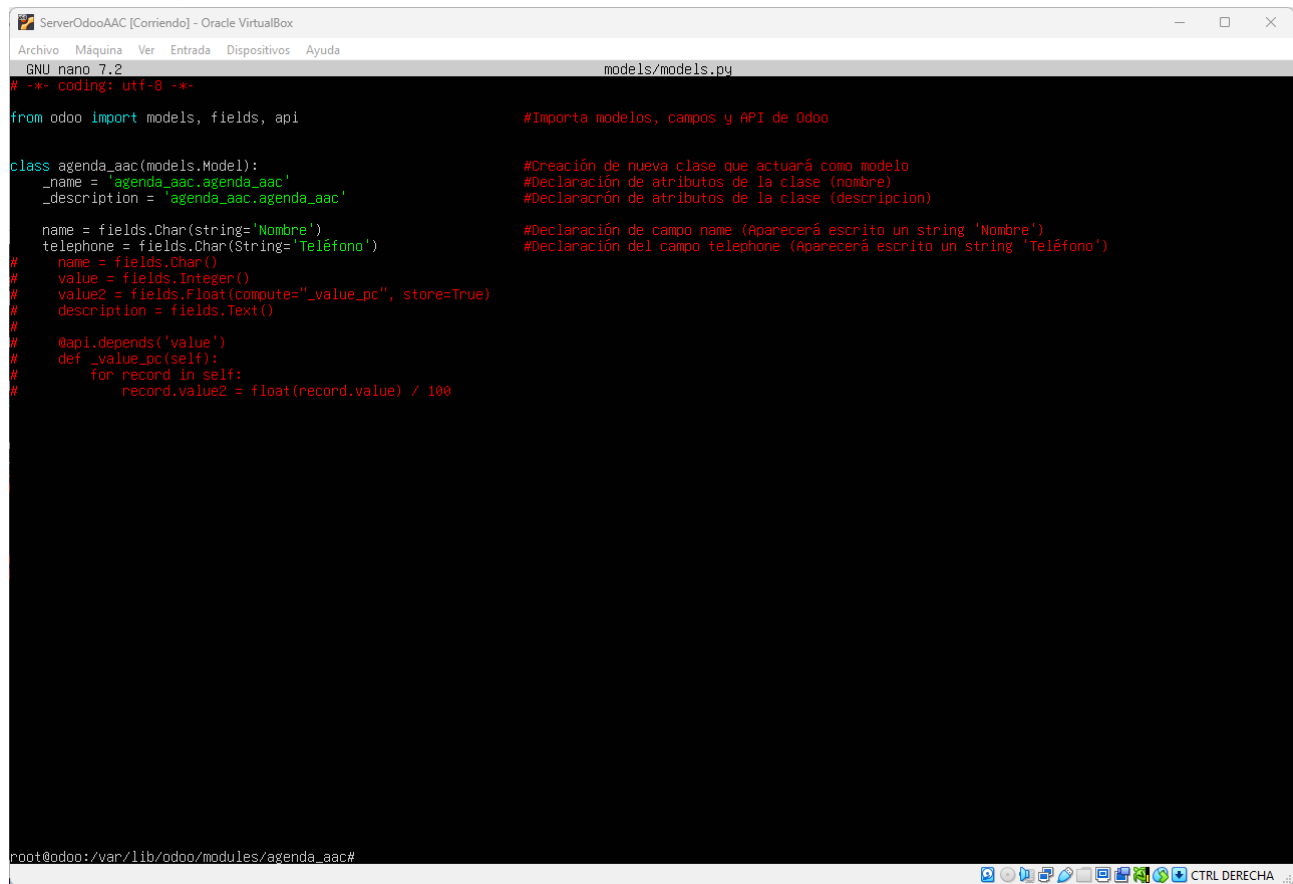
  <record model="ir.ui.view" id="agenda_aac.form">
    <field name="name">agendaAAC form</field>
    <field name="model">agenda_aac.agenda_aac</field>
    <field name="arch" type="xml">
      <form>
        <sheet>
          <group>
            <field name="name"/>
            <field name="telephone"/>
          </group>
        </sheet>
      </form>
    </field>
  </record>

  <!-- actions opening views on models -->
  <record model="ir.actions.act_window" id="agenda_aac.action_window">
    <field name="name">Agenda AAC</field>
    <field name="res_model">agenda_aac.agenda_aac</field>
    <field name="view_mode">list,form</field>
    <field name="view_id" ref="agenda_aac.list"/>
  </record>

  <!-- Top menu item -->
  <menuitem name="Agenda AAC" id="agenda_aac.menu_root"/>
  <!-- menu categories -->
  <menuitem name="Opciones" id="agenda_aac.menu_1" parent="agenda_aac.menu_root"/>
  <!-- actions -->
  <menuitem name="Mostrar" id="agenda_aac.menu_1_list" parent="agenda_aac.menu_1"
    action="agenda_aac.action_window"/>
</data>
```

Figura 19: view.xml

model.py:



```
ServerOdooAAC [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 7.2 models/models.py
# -*- coding: utf-8 -*-

from odoo import models, fields, api

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

    name = fields.Char(string='Nombre')
    telephone = fields.Char(string='Teléfono')
    value = fields.Integer()
    value2 = fields.Float(compute='_value_pc', store=True)
    description = fields.Text()

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

root@odoo:/var/lib/odoo/modules/agenda_aac#
```

Figura 20: model.py