Desarrollo de Aplicaciones Multiplataforma

(e-learning)

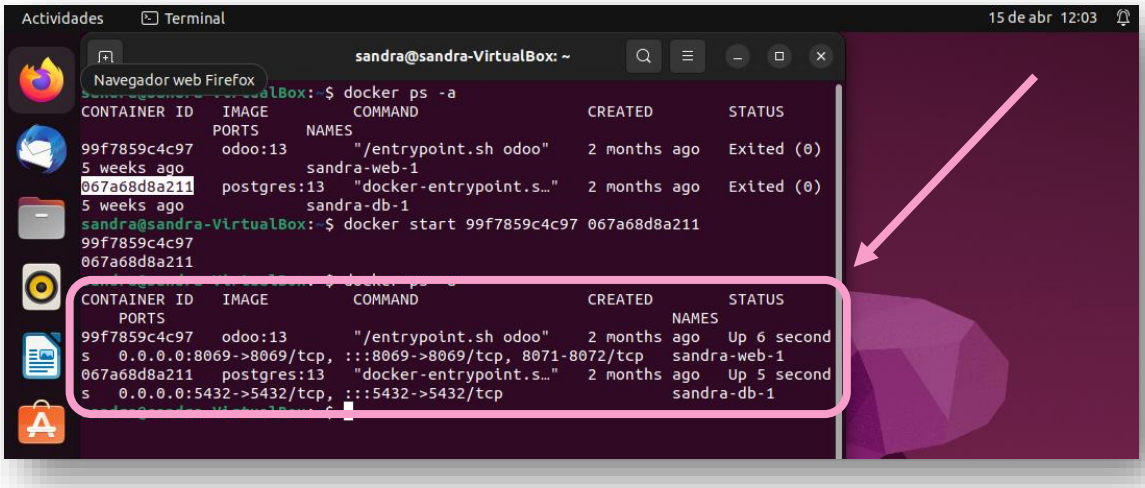
ASIGNATURA: Sistemas de Gestión Empresarial. Tema 5.

ALUMNO: Sandra Sanabrias Viller

CURSO: 2022 - 2023

- **Instalación y funcionamiento de la aplicación.**

En primer lugar, arranco la máquina virtual e inicializo el contenedor de Docker con *Odoo* y *Postgres.*

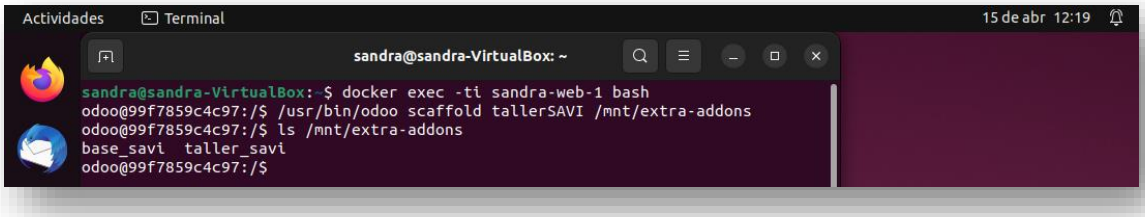


Para realizar la estructura de la aplicación necesito acceder al contenedor con el comando:

docker exec –ti sandra-web-1 bash

y ejecutar el siguiente comando para instalar el módulo:

/usr/bin/odoo scaffold tallerSAVI /mnt/extra-addons

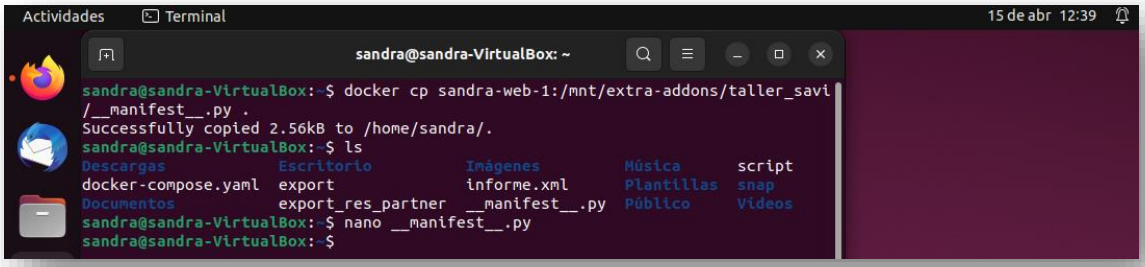


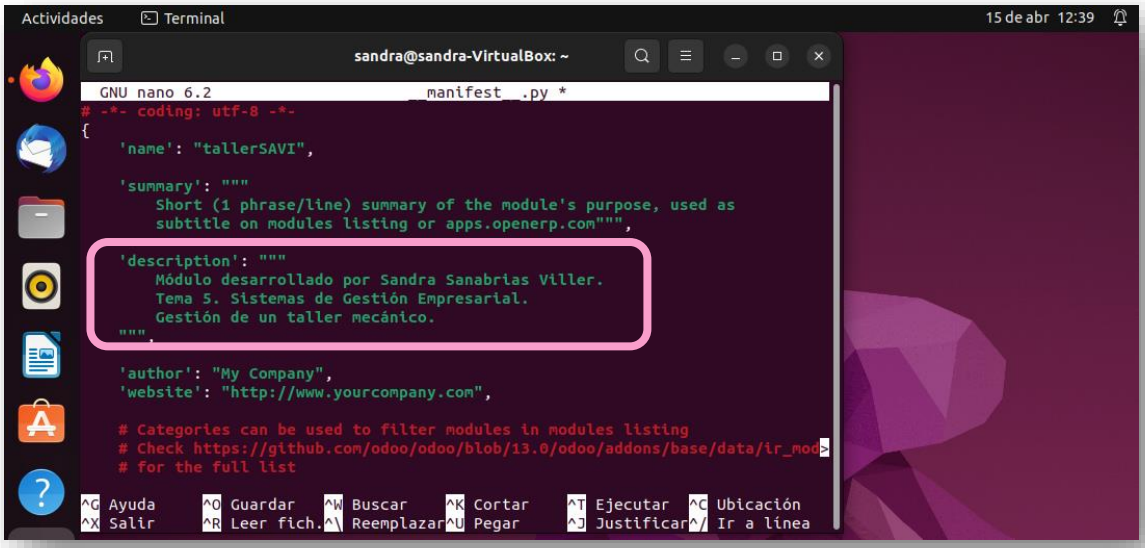
Al listar compruebo que están instaladas la aplicación del tema anterior (*baseSAVI*) y la aplicación para este nuevo tema (*tallerSAVI, a la carpeta le asigna por defecto el nombre taller_SAVI*).

A continuación, he de modificar el archivo *__manifest__.py* para cambiar la descripción del módulo. Para poder editarlo, debo extraerlo del contenedor a la máquina virtual con el siguiente comando:

docker cp sandra-web-1:/mnt/extra-addons/taller_savi/__manifest__.py .

Una vez hecho esto listo para comprobar que tengo el archivo en mi carpeta personal y lo edito: ***nano __manifest__.py***

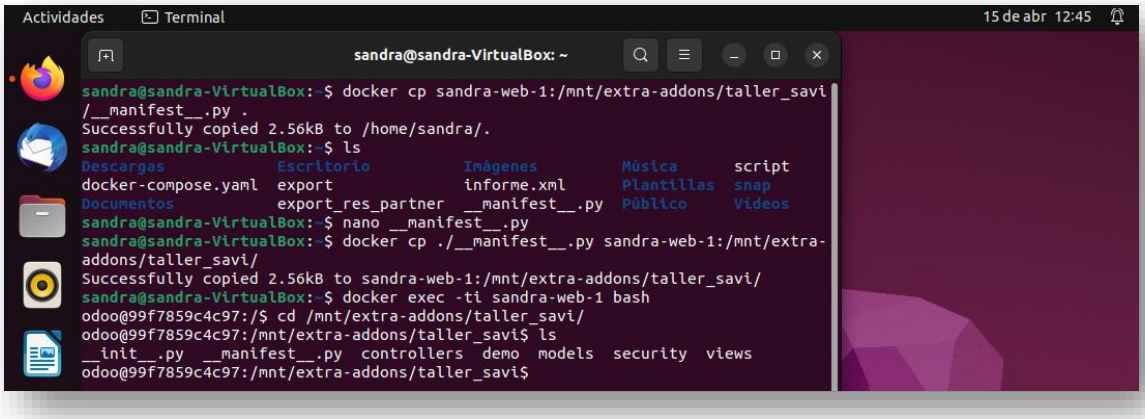




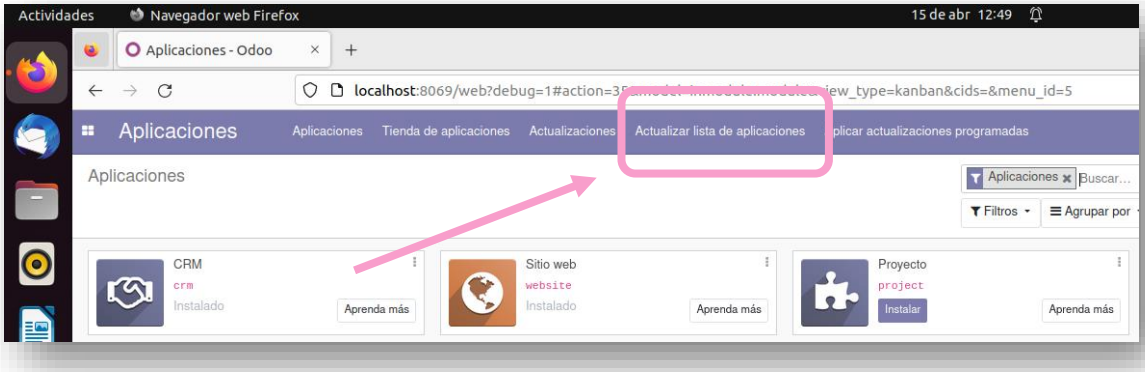
Ahora necesito devolver el archivo a su origen, es decir, a la carpeta del contenedor:

docker cp ./__manifest__.py sandra-web-1:/mnt/extra-addons/taller_savi/

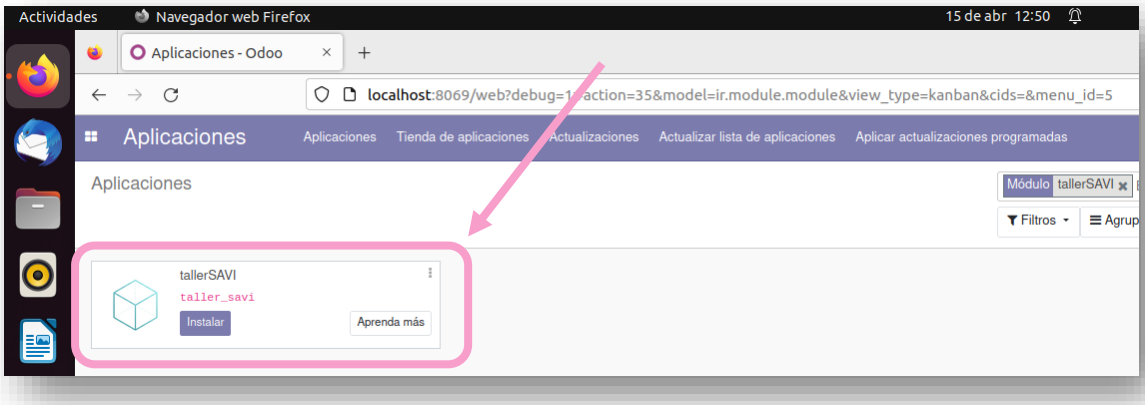
Vuelvo a acceder al contenedor con el comando ***docker exec –ti sandra-web-1 bash***, me sitúo en la carpeta donde se ha debido copiar el archivo *__manifest__.py:* ***cd /mnt/extra-addons/taller_savi/*** y listo (***ls***) para comprobar que el archivo se ha copiado correctamente.



Para poder ver la aplicación en Odoo, debo activar el ***modo desarrollador*** en Ajustes y seguidamente, actualizar la lista de aplicaciones para que aparezca el módulo:

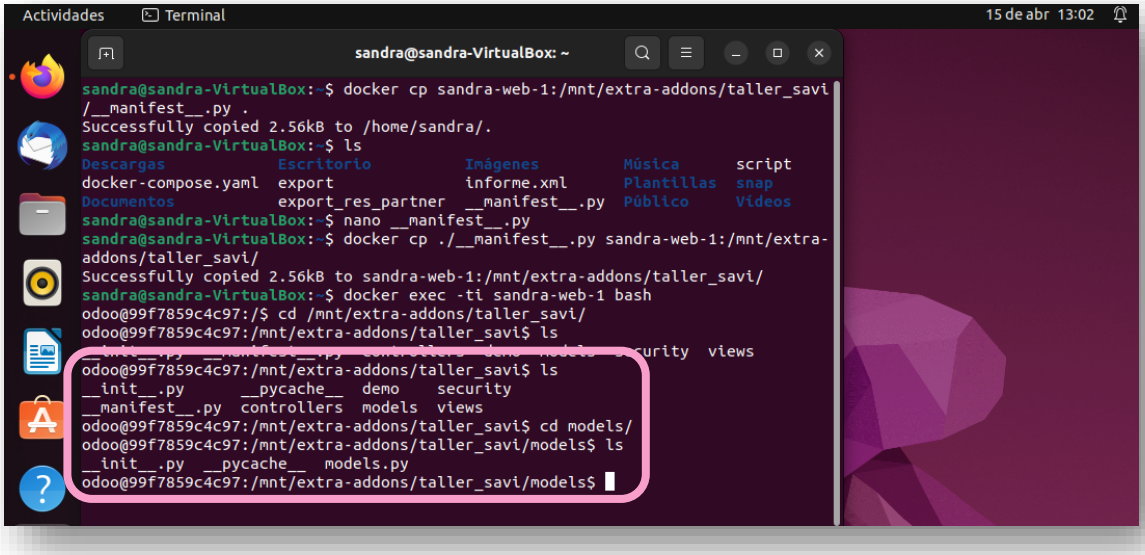


Una vez que he actualizado, procedo a instalar el módulo:



- ## Creación de modelos.

Para crear los modelos necesito volver a hacer lo mismo que hice antes con el archivo *__manifest__.py*, es decir, necesito copiar el archivo *models.py* (que se encontrará en la carpeta *taller_savi/models,* por defecto), en mi carpeta personal de la máquina virtual para poder editarlo, con los comandos:



Compruebo que el archivo está en la carpeta *models* y procedo a copiarlo fuera del contenedor:

docker cp sandra-web-1:/mnt/extra-addons/taller_savi/models/models.py .

Para modificarlo he instalado Visual Studio Code en la máquina virtual, y lo realizaré desde ahí.

Para cada modelo es necesario realizar una clase, por tanto, obtendré tres clases diferentes: ***propietario, vehículo*** y ***reparación***.

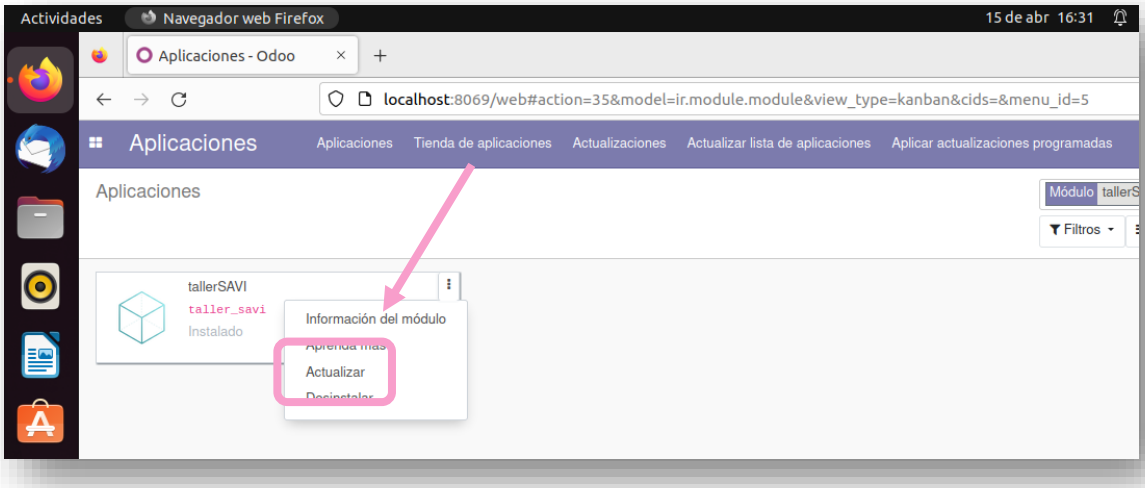
```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api #IMPORTANTE: descomentar esta línea

class Propietario(models.Model):
    _name = 'taller.propietario'

    nombre = fields.Char(string = "Nombre", required = True)
    dni = fields.Char(string = "DNI", required = True)
    telefono = fields.Integer(string = "Telefono contacto", required = True)
    email = fields.Char(string = "Email", required = True)

class Vehiculo(models.Model):
    _name = 'taller.vehiculo'

    matricula = fields.Char(string = "Matricula", required = True)
    _rec_name="matricula"
    tipo = fields.Selection(selection = [('coche','Coche'),('moto','Moto')], required = True)
    marca = fields.Char(string = "Marca", required = True)
    modelo = fields.Char(string = "Modelo", required = True)
    propietario = fields.Many2one('taller.propietario', string = "Propietario", required = True)

class Reparacion(models.Model):
    _name = 'taller.reparacion'

    numero = fields.Integer(string = "Numero reparacion", required = True)
    fecha = fields.Datetime(string = "Fecha y hora")
    cliente = fields.Many2one('taller.propietario', string = "Cliente")
    vehiculo = fields.Many2one('taller.vehiculo', string = "Vehiculo",
                               domain = "[('propietario', '=', cliente)]", rec_name="matricula")
    reparacion = fields.Char(string = "Reparacion")
    total = fields.Float(string = "Total")
```

Una vez hecho esto, con el siguiente comando devuelvo el archivo al contenedor y lo reinicio, junto a la base de datos, para que se apliquen los cambios:

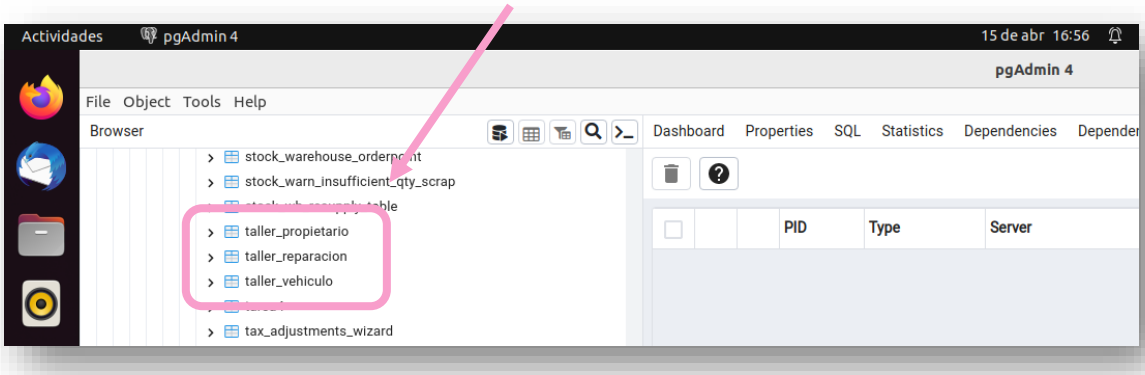
docker cp ./models.py sandra-web-1:/mnt/extra-addons/taller_savi/models && docker restart sandra-web-1 sandra-db-1

```
sandra@sandra-VirtualBox:~$ docker cp ./models.py sandra-web-1:/mnt/extra-addons
/taller_savi/models/ && docker restart sandra-web-1 sandra-db-1
Successfully copied 4.096kB to sandra-web-1:/mnt/extra-addons/taller_savi/models
/
sandra-web-1
sandra-db-1
sandra@sandra-VirtualBox:~$
```

Además, en Odoo voy a proceder a actualizar de nuevo la lista de aplicaciones y, voy a actualizar también mi aplicación. El proceso se ha realizado satisfactoriamente sin ninguna interrupción.

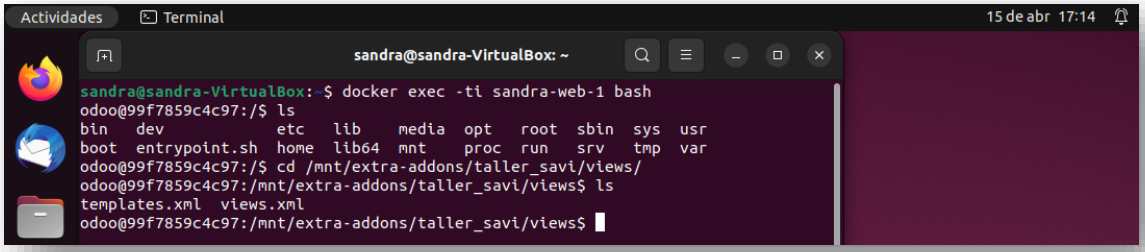


Si compruebo en pgAdmin en la base de datos en la que estoy trabajando (*baseSAVI* en mi caso), observo que se han creado los modelos correctamente:



- ## Creación de vistas.

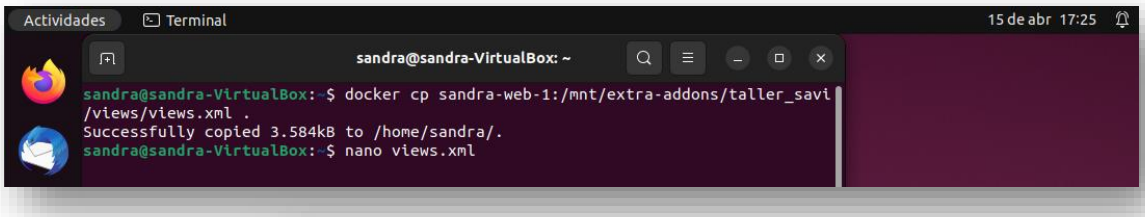
Como se puede observar, dentro de la carpeta *taller_savi* también se encuentra la carpeta *views* y, a su vez, dentro de ella, el archivo *views.xml*.



A continuación, para crear las vistas, nuevamente, debo copiar el archivo *views.xml* en mi carpeta personal de la máquina virtual, para poder modificarlo, con el comando:

docker cp sandra-web-1:/mnt/extra-addons/taller_savi/views/views.xml .

Y lo modifico con: ***nano views.xml***



Por defecto, la estructura es la siguiente (vistas, menús, submenús y acciones):

```xml
<odoo>
  <data>
    <!-- explicit list view definition -->
<!--
    <record model="ir.ui.view" id="taller_savi.list">
      <field name="name">tallerSAVI list</field>
      <field name="model">taller_savi.taller_savi</field>
      <field name="arch" type="xml">
        <tree>
          <field name="name"/>
          <field name="value"/>
          <field name="value2"/>
        </tree>
      </field>
    </record>
-->

    <!-- actions opening views on models -->
<!--
    <record model="ir.actions.act_window" id="taller_savi.action_window">
      <field name="name">tallerSAVI window</field>
      <field name="res_model">taller_savi.taller_savi</field>
      <field name="view_mode">tree,form</field>
    </record>
-->

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

    <!-- Top menu item -->
<!--
    <menuitem name="tallerSAVI" id="taller_savi.menu_root"/>
-->
    <!-- menu categories -->
<!--
    <menuitem name="Menu 1" id="taller_savi.menu_1" parent="taller_savi.menu_root"/>
    <menuitem name="Menu 2" id="taller_savi.menu_2" parent="taller_savi.menu_root"/>
-->
    <!-- actions -->
<!--
    <menuitem name="List" id="taller_savi.menu_1_list" parent="taller_savi.menu_1"
              action="taller_savi.action_window"/>
    <menuitem name="Server to list" id="taller_savi" parent="taller_savi.menu_2"
              action="taller_savi.action_server"/>
-->
  </data>
</odoo>
```

El código para cada vista quedaría de la siguiente forma:

__manifest__.py | views.xml | models.py 1

home > sandra > views.xml

```xml
<odoo>
  <data>
    <!--PROPIETARIOS-->
    <!--Vista tipo tree-->
    <record model="ir.ui.view" id="taller_savi.propietarios_tree">
      <field name="name">Lista de propietarios</field>
      <field name="model">taller.propietario</field>
      <field name="arch" type="xml">
        <tree>
          <field name="nombre"/>
          <field name="dni"/>
          <field name="telefono"/>
          <field name="email"/>
        </tree>
      </field>
    </record>
    <!--Vista tipo form-->
    <record model="ir.ui.view" id="taller_savi.propietarios_form">
      <field name="name">Formulario de propietarios</field>
      <field name="model">taller.propietario</field>
      <field name="arch" type="xml">
        <form>
          <group>
            <field name="nombre"/>
            <field name="dni"/>
            <field name="telefono"/>
            <field name="email"/>
          </group>
        </form>
      </field>
    </record>
```

__manifest__.py | views.xml | models.py 1

home > sandra > views.xml

```xml
<!--VEHICULOS-->
<!--Vista tipo tree-->
<record model="ir.ui.view" id="taller_savi.vehiculos_tree">
  <field name="name">Lista de vehiculos</field>
  <field name="model">taller.vehiculo</field>
  <field name="arch" type="xml">
    <tree>
      <field name="matricula"/>
      <field name="tipo"/>
      <field name="marca"/>
      <field name="modelo"/>
      <field name="propietario"/>
    </tree>
  </field>
</record>
<!--Vista tipo form-->
<record model="ir.ui.view" id="taller_savi.vehiculos_form">
  <field name="name">Formulario de vehiculos</field>
  <field name="model">taller.vehiculo</field>
  <field name="arch" type="xml">
    <form>
      <group>
        <field name="matricula"/>
        <field name="tipo"/>
        <field name="marca"/>
        <field name="modelo"/>
        <field name="propietario"/>
      </group>
    </form>
  </field>
</record>
```

__manifest__.py × views.xml × models.py 1

home > sandra > views.xml

```xml
<!--REPARACIONES-->
<!--Vista tipo tree-->
<record model="ir.ui.view" id="taller_savi.reparaciones_tree">
  <field name="name">Lista de reparaciones</field>
  <field name="model">taller.reparacion</field>
  <field name="arch" type="xml">
    <tree>
      <field name="numero"/>
      <field name="fecha"/>
      <field name="cliente"/>
      <field name="vehiculo"/>
      <field name="reparacion"/>
      <field name="total"/>
    </tree>
  </field>
</record>
<!--Vista tipo form-->
<record model="ir.ui.view" id="taller_savi.reparaciones_form">
  <field name="name">Formulario de reparaciones</field>
  <field name="model">taller.reparacion</field>
  <field name="arch" type="xml">
    <form>
      <group>
        <field name="numero"/>
        <field name="fecha"/>
        <field name="cliente"/>
        <field name="vehiculo"/>
        <field name="reparacion"/>
        <field name="total"/>
      </group>
    </form>
  </field>
</record>
```

Antes de volver a copiar el archivo en el contenedor y reiniciarlo con el siguiente comando, voy a añadir en el siguiente punto también los menús y submenús:

- **Creación de menús adecuados.**

Para crear los menús y submenús, he de modificar el mismo archivo de las vistas anterior también y añadirlos ahí. (*views.xml*)

__manifest__.py | views.xml | models.py 1

home > sandra > views.xml

```xml
<!--ACCIONES QUE ABREN VISTAS EN LOS MODELOS-->
<record model="ir.actions.act_window" id="taller_savi.propietarios_action_window">
  <field name="name">Propietarios</field>
  <field name="res_model">taller.propietario</field>
  <field name="view_mode">tree,form</field>
</record>
<record model="ir.actions.act_window" id="taller_savi.vehiculos_action_window">
  <field name="name">Vehiculos</field>
  <field name="res_model">taller.vehiculo</field>
  <field name="view_mode">tree,form</field>
</record>
<record model="ir.actions.act_window" id="taller_savi.reparaciones_action_window">
  <field name="name">Reparaciones</field>
  <field name="res_model">taller.reparacion</field>
  <field name="view_mode">tree,form</field>
</record>
<!--<menuitem name="tallerSAVI" id="taller_savi.menu_root"/>-->
<!-- menu categories -->
<!--
<menuitem name="Menu 1" id="taller_savi.menu_1" parent="taller_savi.menu_root"/>
<menuitem name="Menu 2" id="taller_savi.menu_2" parent="taller_savi.menu_root"/>
-->
<!-- actions -->
<!--
<menuitem name="List" id="taller_savi.menu_1_list" parent="taller_savi.menu_1"
          action="taller_savi.action_window"/>
<menuitem name="Server to list" id="taller_savi" parent="taller_savi.menu_2"
          action="taller_savi.action_server"/>
-->
```

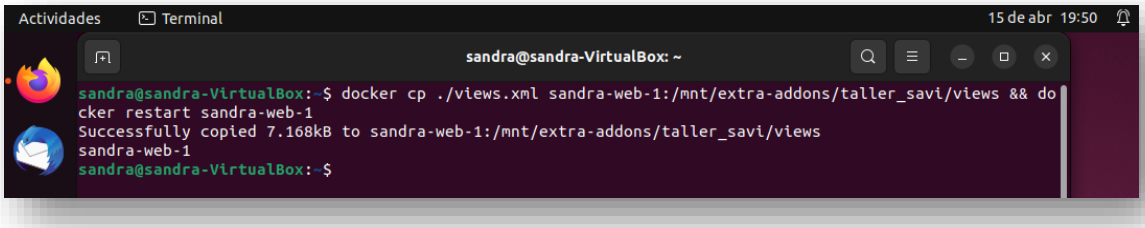
__manifest__.py | views.xml | models.py 1

home > sandra > views.xml

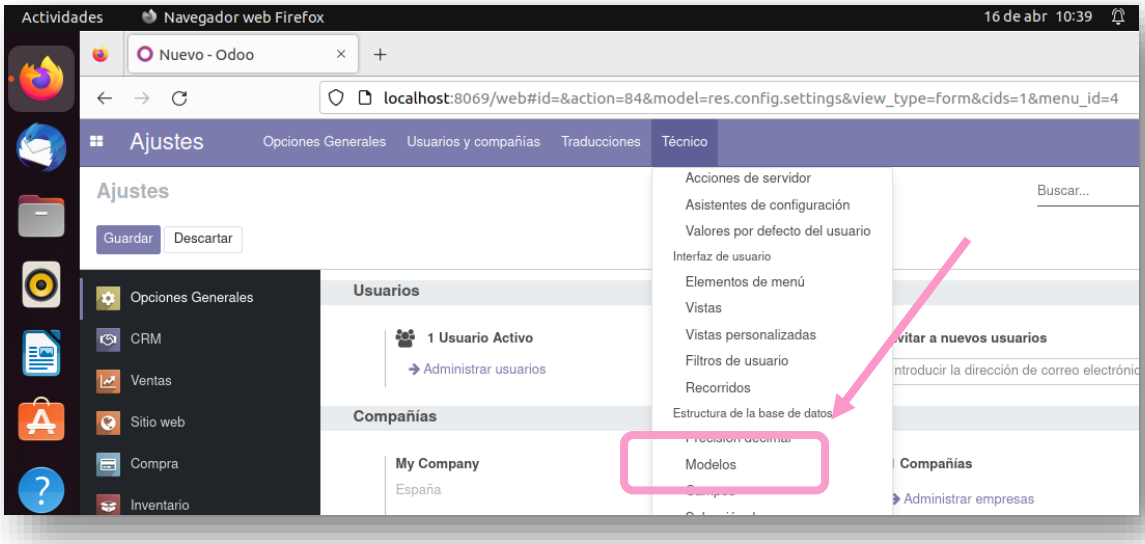
```xml
<!--MENÚ-->
<!--MENÚ PRINCIPAL-->
<menuitem name="Taller SAVI" id="taller_savi.menu_root"/>

<!--SUBMENÚS - CATEGORÍAS-->
<menuitem name="Propietarios" id="taller_savi.propietarios" parent="taller_savi.menu_root"/>
<menuitem name="Vehiculos" id="taller_savi.vehiculos" parent="taller_savi.menu_root"/>
<menuitem name="Reparaciones" id="taller_savi.reparaciones" parent="taller_savi.menu_root"/>

<!--ACCIONES-->
<menuitem name="Propietarios" id="taller_savi.propietarios_lista" parent="taller_savi.propietarios"
          action="taller_savi.propietarios_action_window"/>
<menuitem name="Vehiculos" id="taller_savi.vehiculos_lista" parent="taller_savi.vehiculos"
          action="taller_savi.vehiculos_action_window"/>
<menuitem name="Reparaciones" id="taller_savi.reparaciones_lista" parent="taller_savi.reparaciones"
          action="taller_savi.reparaciones_action_window"/>
</data>
</odoo>
```

Y ahora sí, he de volver a copiar el archivo *views.xml* en el contenedor y reiniciar el servicio, para que los cambios surtan efecto:

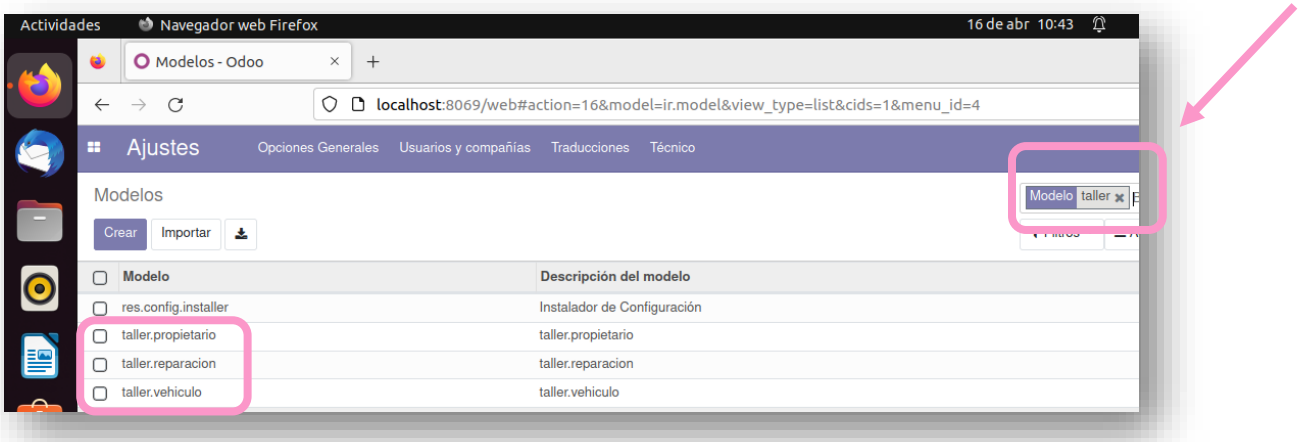
docker cp ./views.xml sandra-web-1:/mnt/extra-addons/taller_savi/views && docker restart sandra-web-1



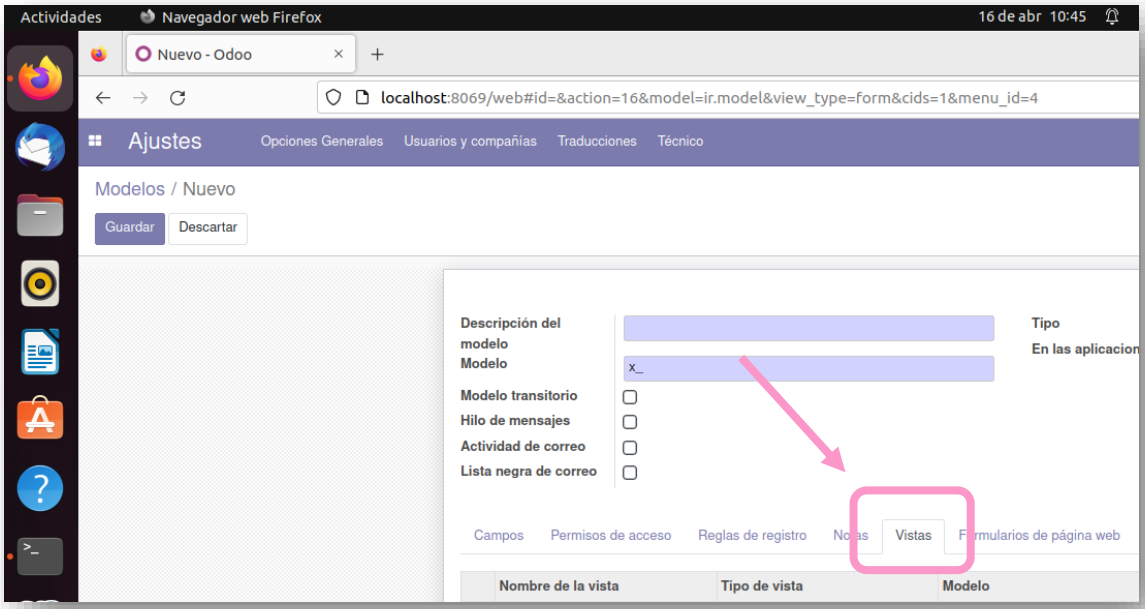
Compruebo que se han creado las vistas correctamente. Selecciono ***Ajustes – Técnico – Estructura de la base de datos – Modelos:***



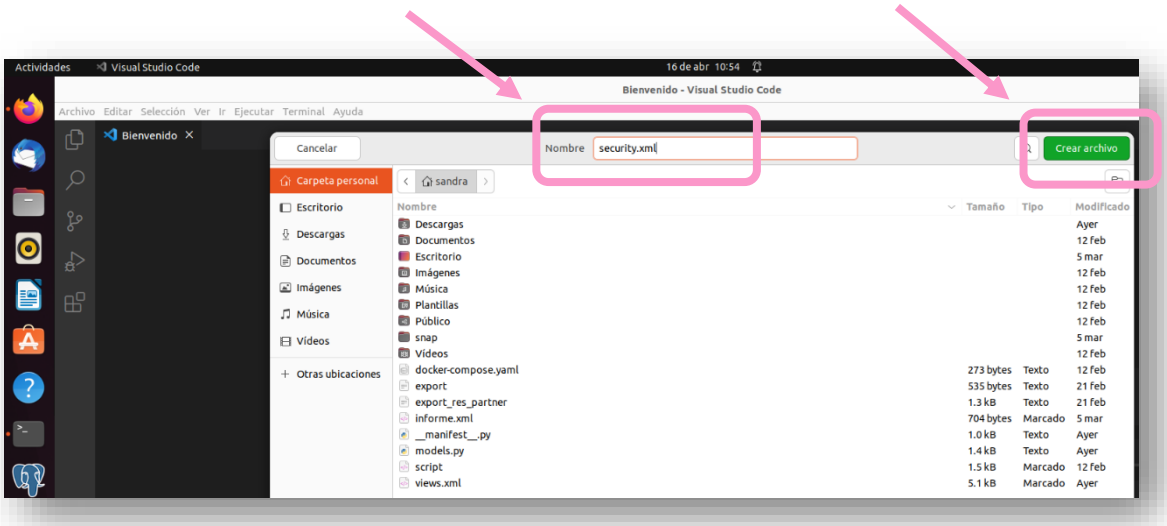
Filtro por ***taller:***



Y selecciono algún modelo, por ejemplo ***taller.propietario -> Crear*** y observo que se ha creado correctamente el apartado ***"Vistas"***:



Para poder ver el módulo en la lista de aplicaciones en Odoo, debo crear un fichero denominado *security.xml* para conceder los permisos de acceso. En este caso lo creo en Visual Studio Code, fuera del contenedor y, una vez hecho, lo copiaré dentro.



security.xml - Visual Studio Code

home > sandra > security.xml

```xml
<odoo>
    <record id="model_access_taller1" model="ir.model.access">
        <field name="name">model.access.taller1</field>
        <field name="model_id" ref="taller_savi.model_taller_propietario"></field>
        <field name="group_id" ref="base.group_system"></field>
        <field name="perm_read" eval="1"></field>
        <field name="perm_write" eval="1"></field>
        <field name="perm_create" eval="1"></field>
        <field name="perm_unlink" eval="1"></field>
    </record>
    <record id="model_access_taller2" model="ir.model.access">
        <field name="name">model.access.taller2</field>
        <field name="model_id" ref="taller_savi.model_taller_vehiculo"></field>
        <field name="group_id" ref="base.group_system"></field>
        <field name="perm_read" eval="1"></field>
        <field name="perm_write" eval="1"></field>
        <field name="perm_create" eval="1"></field>
        <field name="perm_unlink" eval="1"></field>
    </record>
    <record id="model_access_taller3" model="ir.model.access">
        <field name="name">model.access.taller3</field>
        <field name="model_id" ref="taller_savi.model_taller_reparacion"></field>
        <field name="group_id" ref="base.group_system"></field>
        <field name="perm_read" eval="1"></field>
        <field name="perm_write" eval="1"></field>
        <field name="perm_create" eval="1"></field>
        <field name="perm_unlink" eval="1"></field>
    </record>
</odoo>
```

Ahora procedo a copiar dicho archivo en la carpeta ***"taller_savi"*** con el siguiente comando:

docker cp ./security.xml sandra-web-1:/mnt/extra-addons/taller_savi/

Para comprobar que se ha copiado correctamente, entro dentro del contenedor:

docker exec –ti sandra-web-1 bash

Me sitúo en la carpeta de mi módulo (***taller_savi***):

cd mnt/extra-addons/taller_savi/

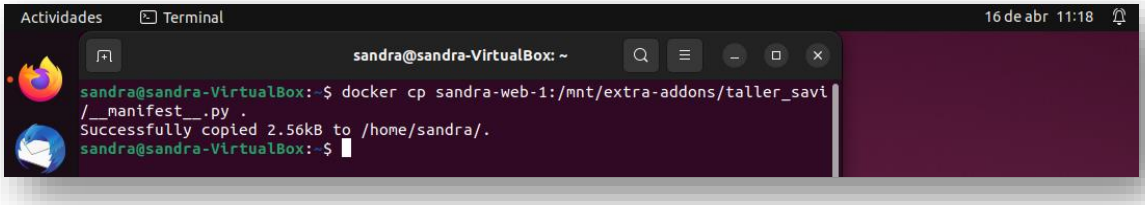
Y listo (***ls***), para comprobar que, efectivamente, el archivo se ha copiado correctamente:

```
sandra@sandra-VirtualBox:~$ docker cp ./security.xml sandra-web-1:/mnt/extra-add
ons/taller_savi/
Successfully copied 3.072kB to sandra-web-1:/mnt/extra-addons/taller_savi/
sandra@sandra-VirtualBox:~$ docker exec -ti sandra-web-1 bash
odoo@99f7859c4c97:/$ cd mnt/extra-addons/taller_savi/
odoo@99f7859c4c97:/mnt/extra-addons/taller_savi$ ls
__init__.py      __pycache__  demo   security   views
__manifest__.py  controllers  models security.xml
odoo@99f7859c4c97:/mnt/extra-addons/taller_savi$
```

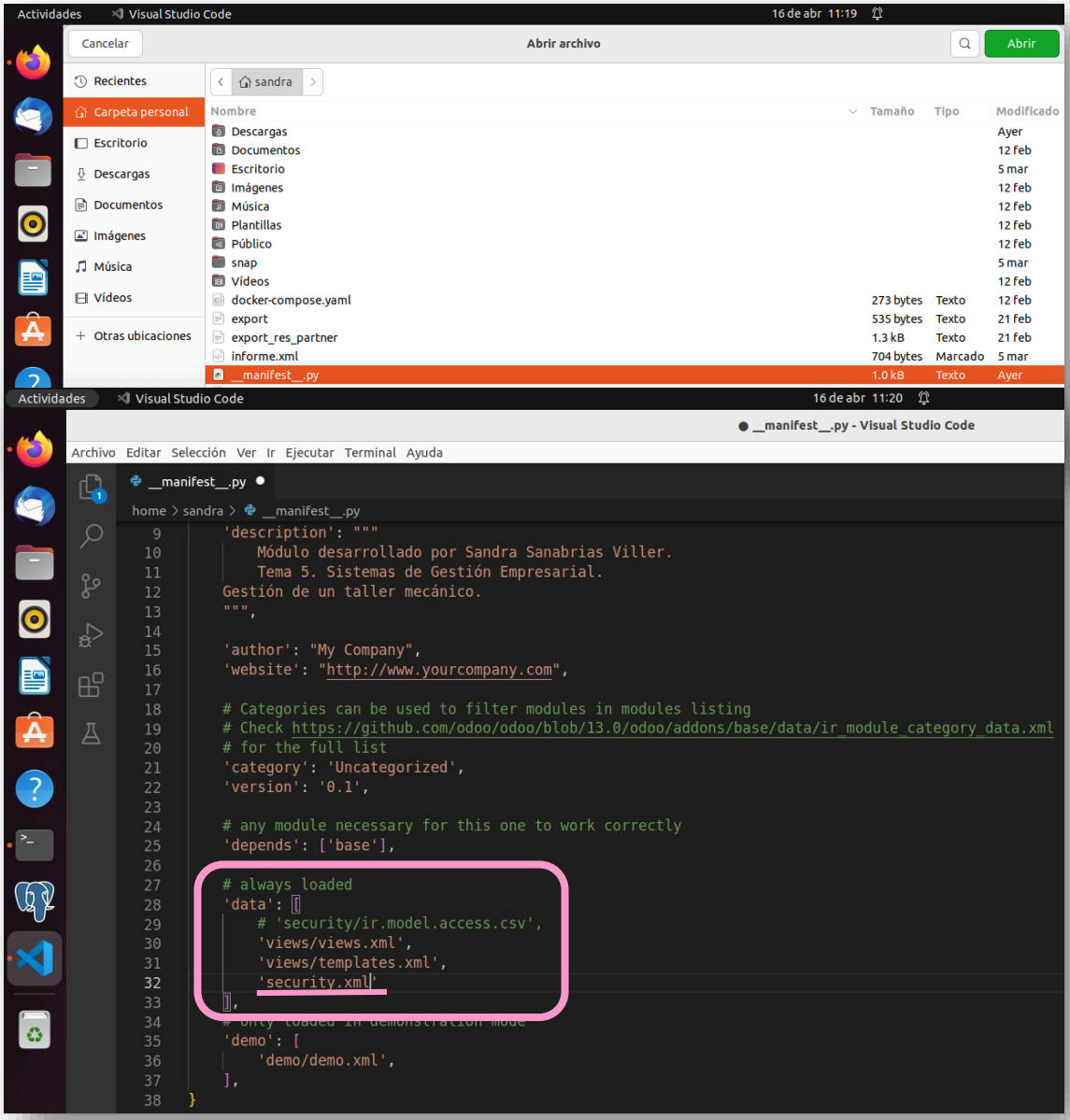
Una vez creado este archivo, debo introducirlo en el archivo *__manifest__.py*, así que nuevamente, he de copiar el archivo *__manifest__.py* en mi carpeta personal de Ubuntu, para poder modificarlo y, después, volverlo a copiar dentro del contenedor:

Copiarlo a mi carpeta personal:

docker cp sandra-web-1:/mnt/extra-addons/taller_savi/__manifest__.py .



Modificarlo en Visual Studio Code:

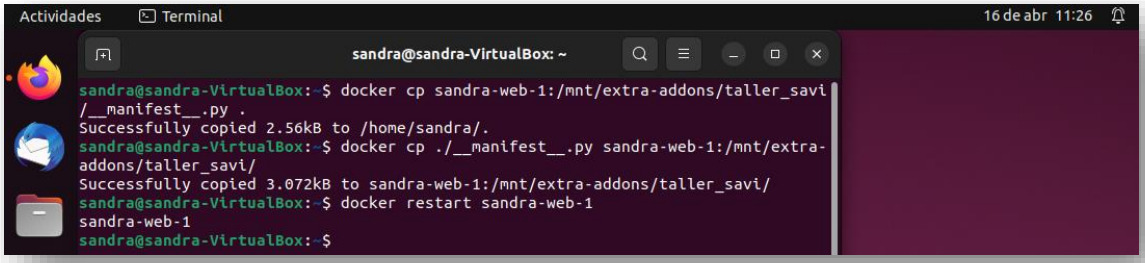


Copiarlo de nuevo en el contenedor:

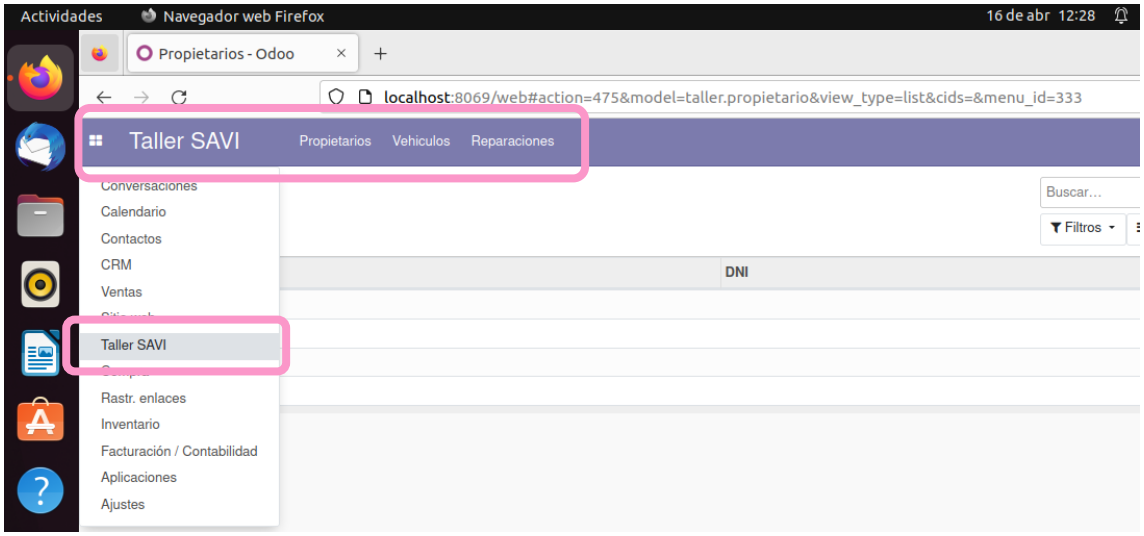
docker cp ./__manifest__.py sandra-web-1:/mnt/extra-addons/taller_savi/

Además, voy a reiniciar el servicio para que se apliquen los cambios:

docker restart sandra-web-1



Voy a comprobar que puedo ver la aplicación correctamente desde Odoo (a parte de reiniciar el servicio, he vuelto a actualizar la aplicación también en el panel):



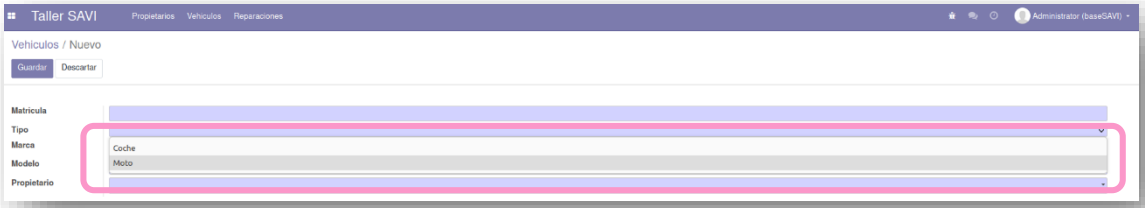
- **Creación del informe.**

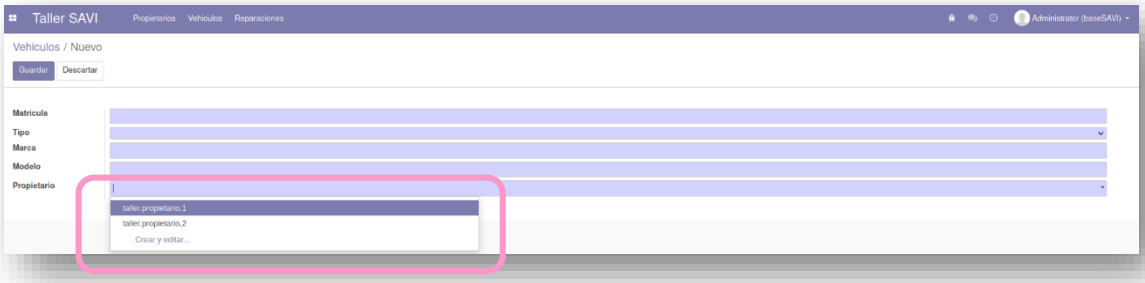
  Para crear el informe voy a introducir datos en las vistas que hemos creado.

Propietarios/Clientes:

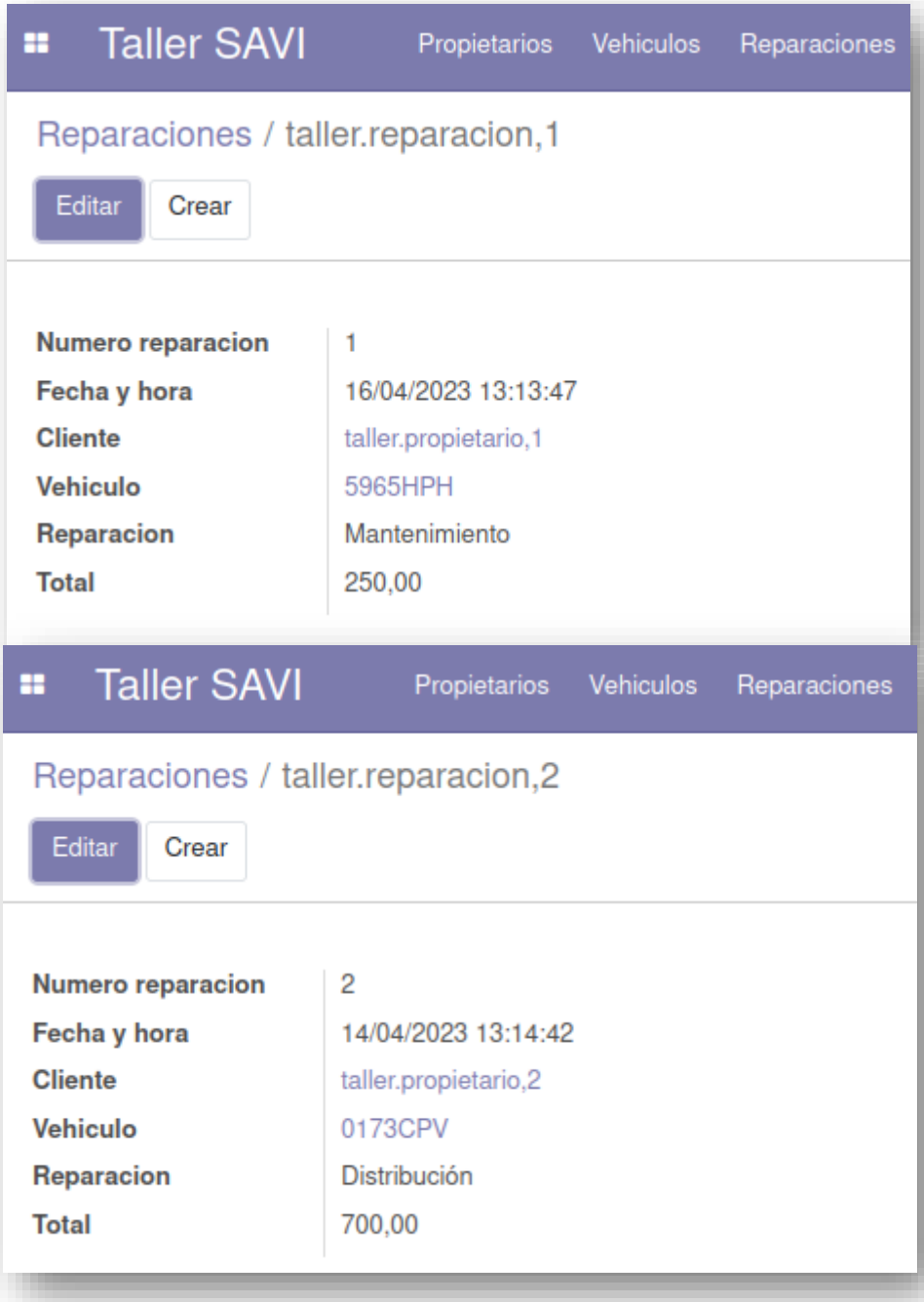
Vehículos:

Se observa que al crear un vehículo permite seleccionar el tipo y el propietario:

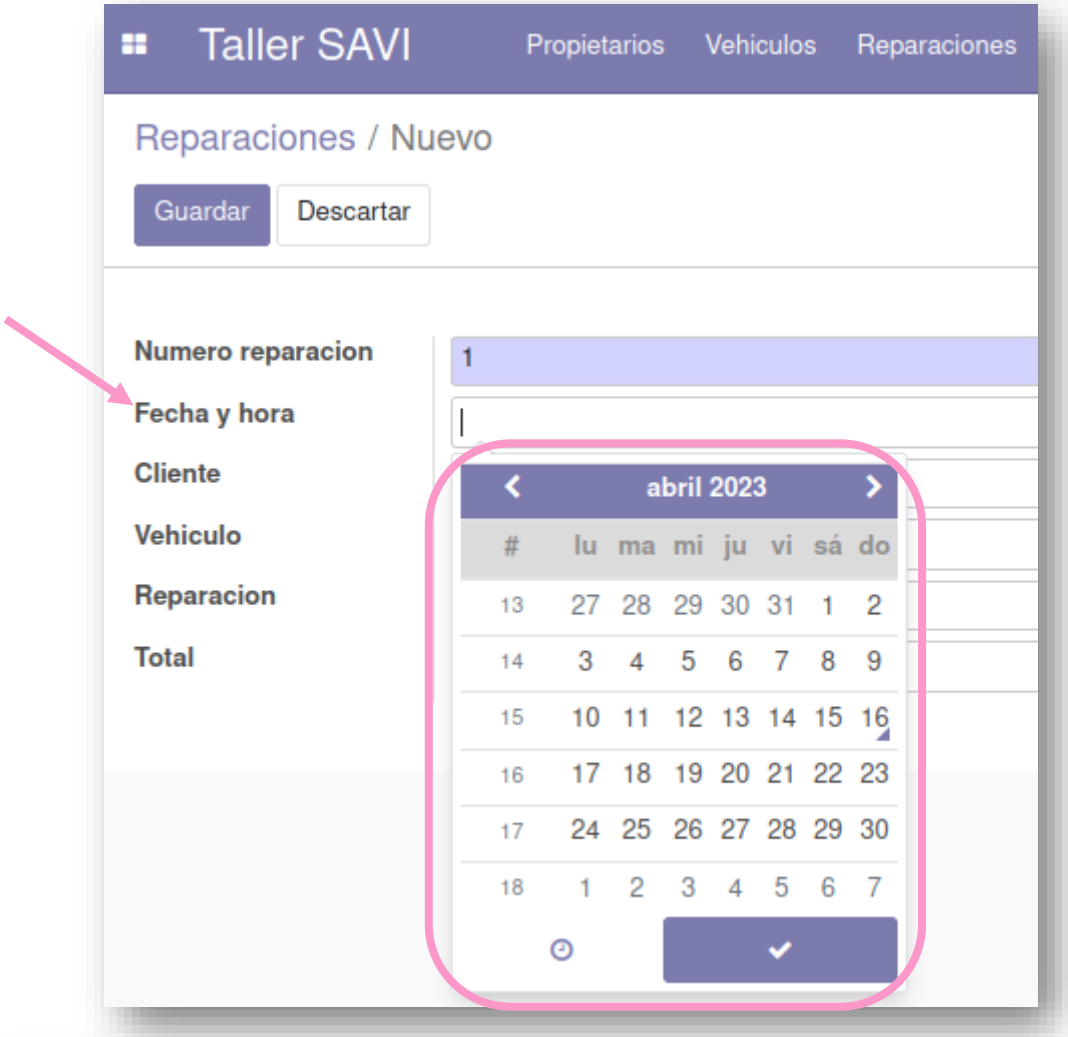


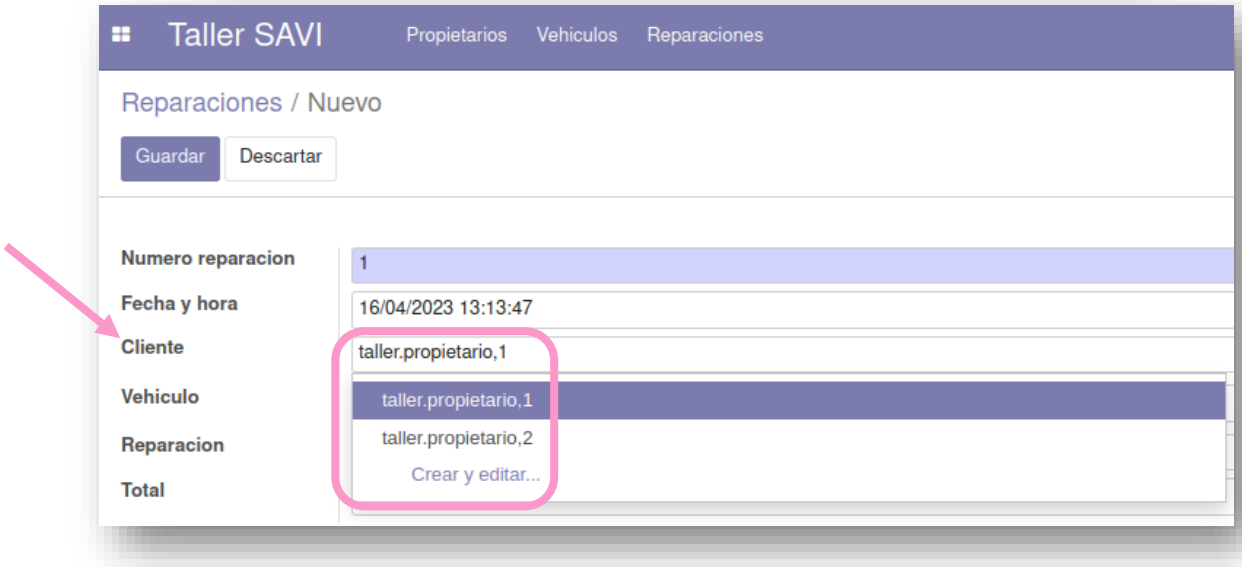


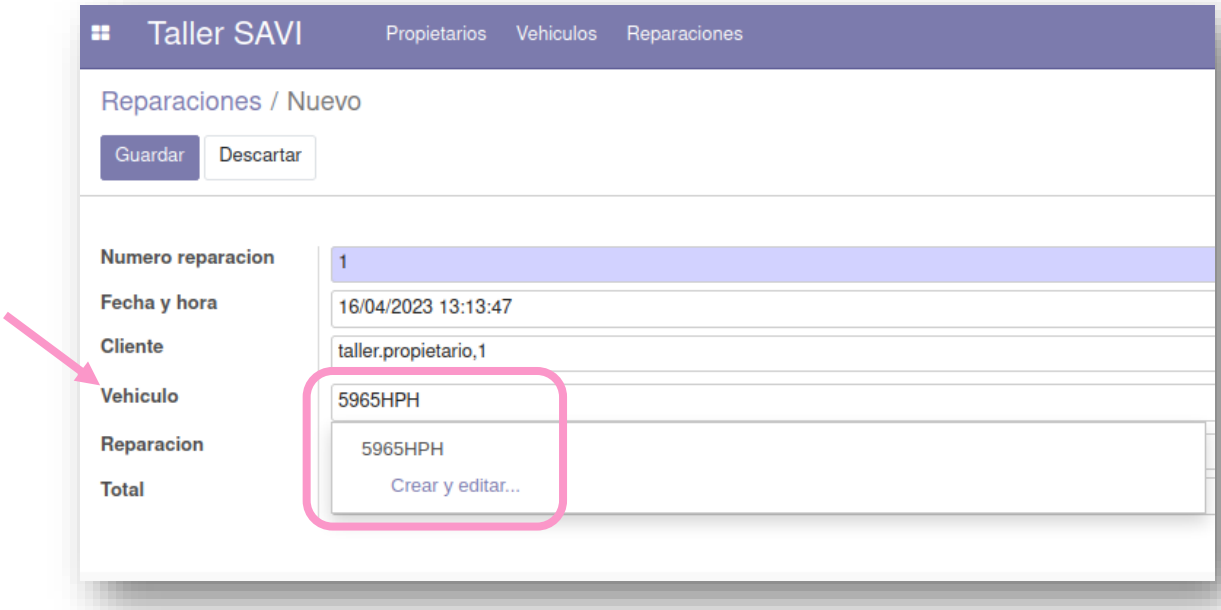
Reparaciones:



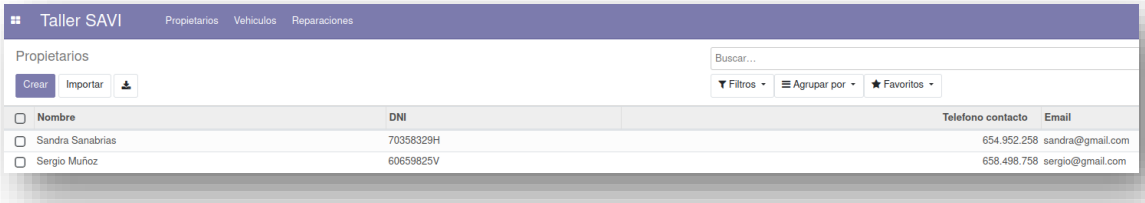
Se observa también cómo permite seleccionar la fecha y hora y se abren dos desplegables con el cliente y la matrícula asociada:

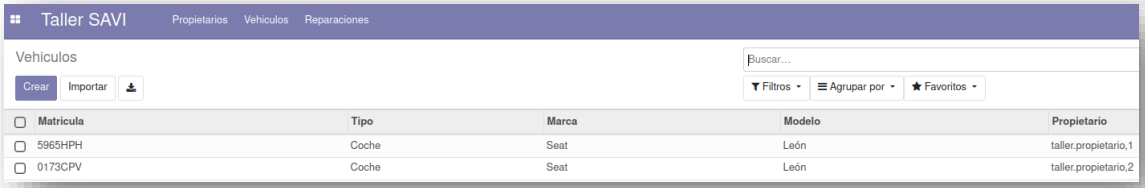


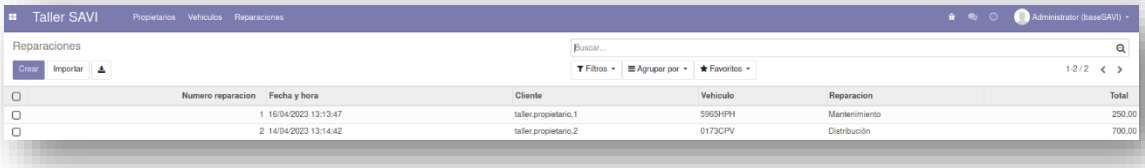




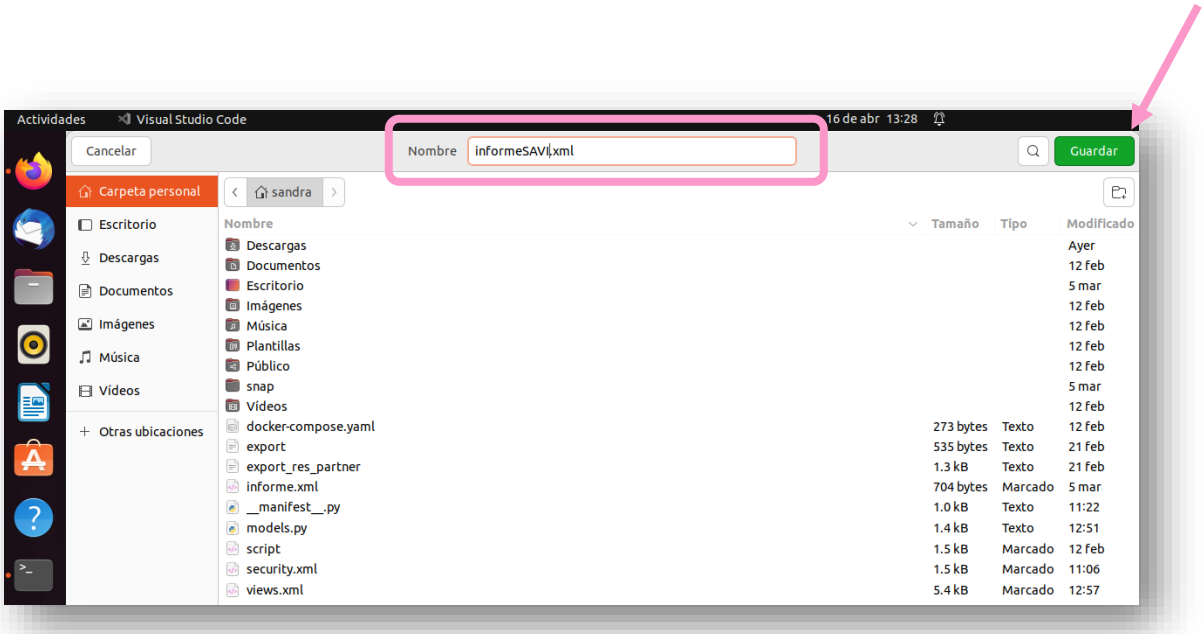
Resultado:

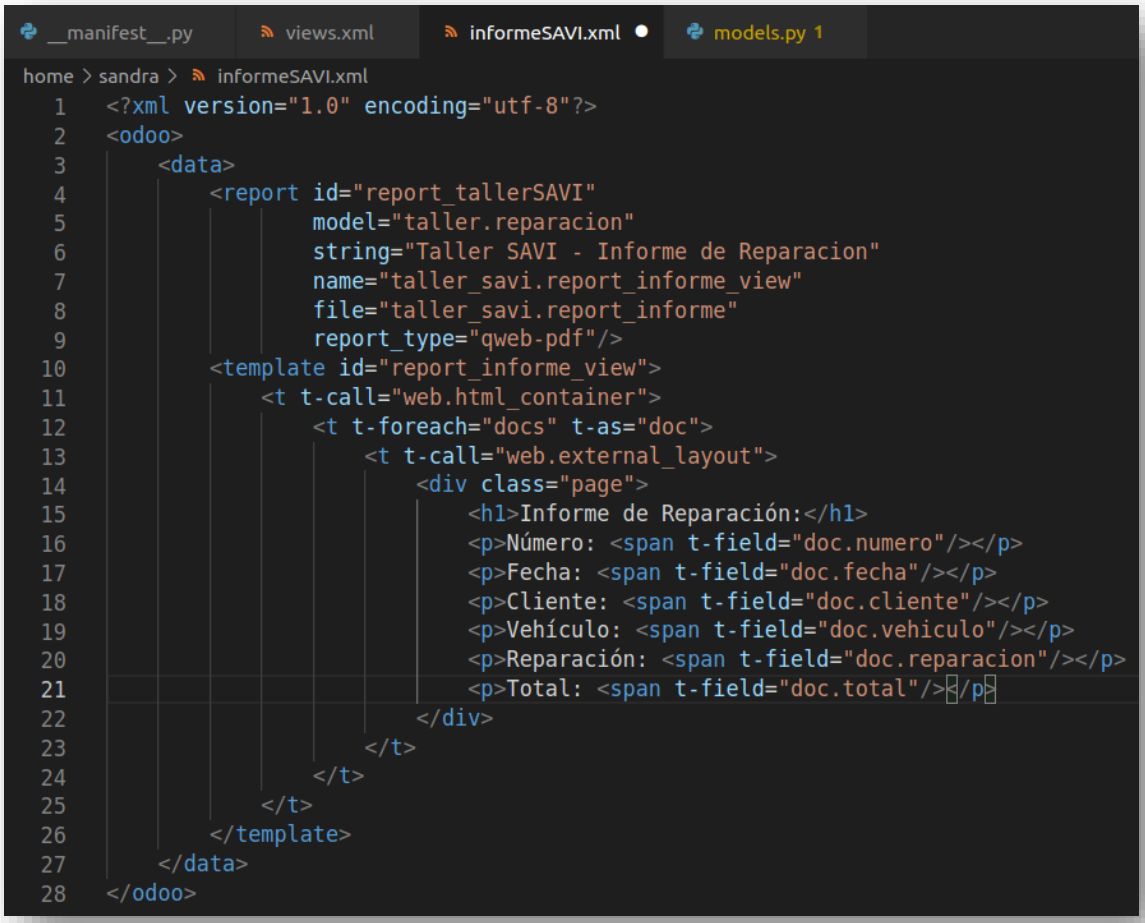






Una vez introducidos los datos, para realizar el informe, al igual que en la tarea 4 y a lo largo de toda esta tarea, debo crear un archivo ***"informeSAVI.xml"*** dentro de la carpeta ***"views"***, es decir, lo crearé en Visual Studio Code y lo añadiré al contenedor.





Procedo a copiar el archivo en la carpeta ***"views"*** y reinicializo de nuevo el servicio:

docker cp ./informeSAVI.xml sandra-web-1:/mnt/extra-addons/taller_savi/views && docker restart sandra-web-1

Compruebo que el archivo se ha copiado correctamente en el contenedor:

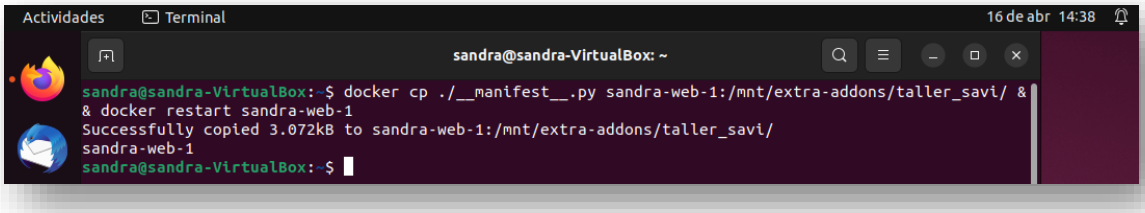
```
Actividades   Terminal                                   16 de abr 14:28
                          sandra@sandra-VirtualBox: ~
sandra@sandra-VirtualBox:~$ docker cp ./informeSAVI.xml sandra-web-1:/mnt/extra-addons/taller_savi/vi
ews && docker restart sandra-web-1
Successfully copied 3.072kB to sandra-web-1:/mnt/extra-addons/taller_savi/views
sandra-web-1
sandra@sandra-VirtualBox:~$ docker exec -ti sandra-web-1 bash
odoo@99f7859c4c97:/$ cd mnt/extra-addons/taller_savi/views
odoo@99f7859c4c97:/mnt/extra-addons/taller_savi/views$ ls
informeSAVI.xml  templates.xml  views.xml
odoo@99f7859c4c97:/mnt/extra-addons/taller_savi/views$
```

A continuación, debo añadir el fichero ***"informeSAVI.xml"*** en el archivo *__manifest__.py*, por tanto, seguiré todo el proceso nuevamente. Como ya copié anteriormente dicho archivo en mi carpeta personal de Ubuntu, tan solo necesito modificarlo en Visual Studio Code:

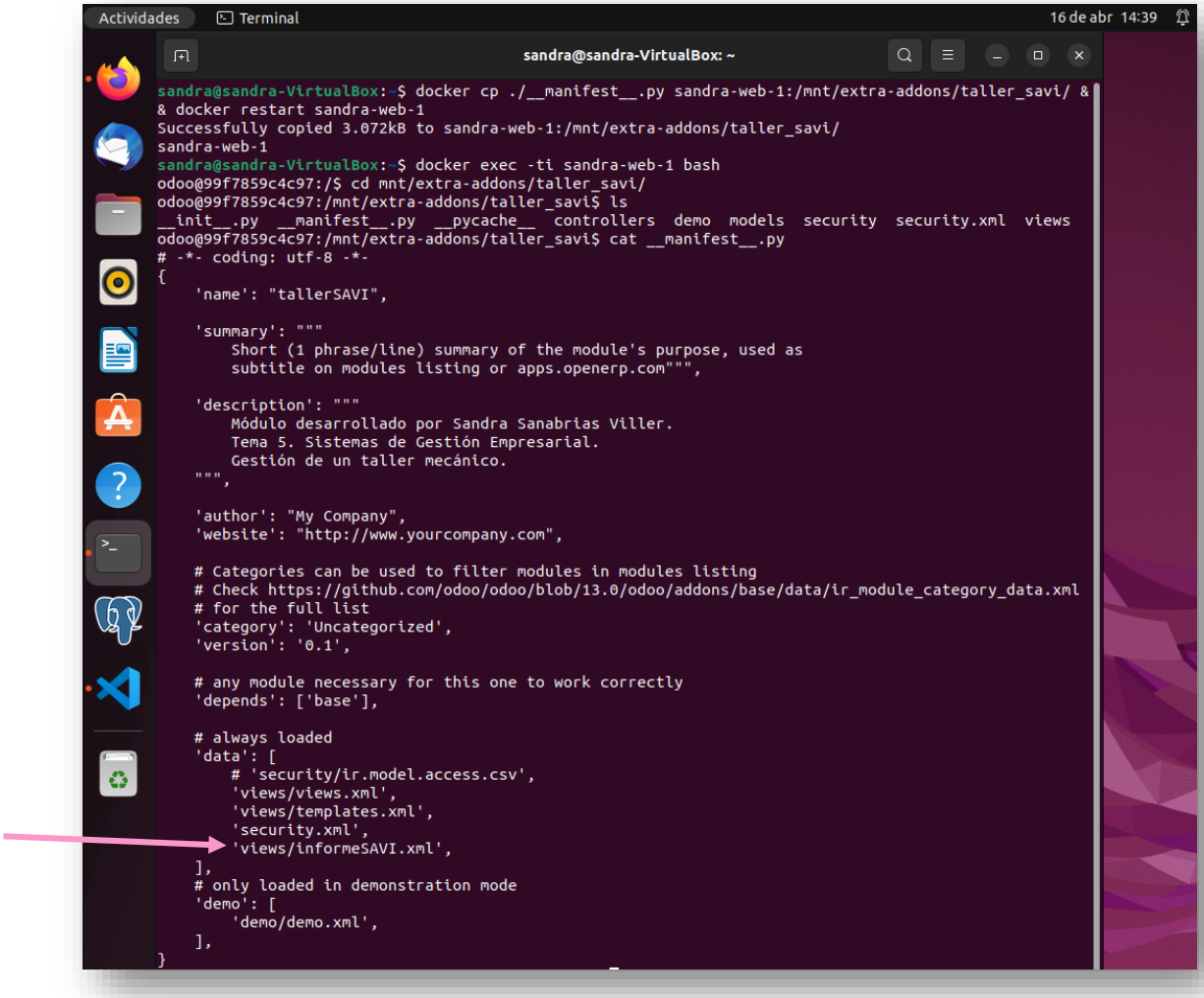
```
__manifest__.py ●   views.xml   informeSAVI.xml   models.py 1
home > sandra > __manifest__.py
6           short (1 phrase/line) summary of the module's purpose, used as
7           subtitle on modules listing or apps.openerp.com""",
8
9       'description': """
10          Módulo desarrollado por Sandra Sanabrias Viller.
11          Tema 5. Sistemas de Gestión Empresarial.
12      Gestión de un taller mecánico.
13      """,
14
15      'author': "My Company",
16      'website': "http://www.yourcompany.com",
17
18      # Categories can be used to filter modules in modules listing
19      # Check https://github.com/odoo/odoo/blob/13.0/odoo/addons/base/data/ir_module_category_data.xml
20      # for the full list
21      'category': 'Uncategorized',
22      'version': '0.1',
23
24      # any module necessary for this one to work correctly
25      'depends': ['base'],
26
27      # always loaded
28      'data': [
29          # 'security/ir.model.access.csv',
30          'views/views.xml',
31          'views/templates.xml',
32          'security.xml',
33          'views/informeSAVI.xml',
34      ],
35      # only loaded in demonstration mode
36      'demo': [
37          'demo/demo.xml',
38      ],
39  }
```

Guardo, copio el archivo en el contenedor y reinicializo nuevamente:

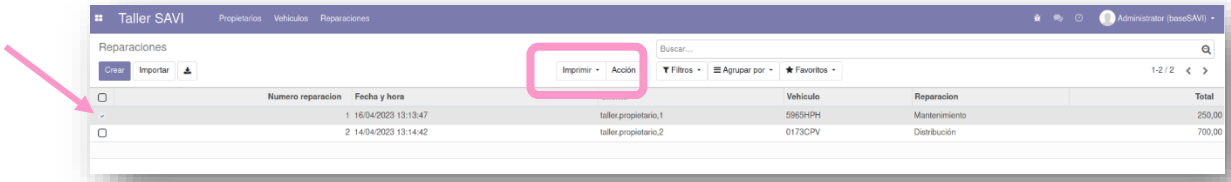
docker cp ./__manifest__.py sandra-web-1:/mnt/extra-addons/taller_savi/ && docker restart sandra-web-1



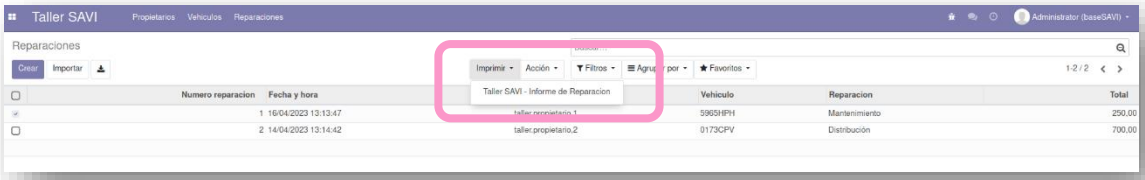
Para comprobar que el archivo se ha modificado correctamente:



Seguidamente vuelvo a actualizar tanto la página como la aplicación de Odoo y, si compruebo en el apartado Reparaciones, se puede observar que ahora muestra también la opción ***"Imprimir"*** al seleccionar algún informe de reparación:

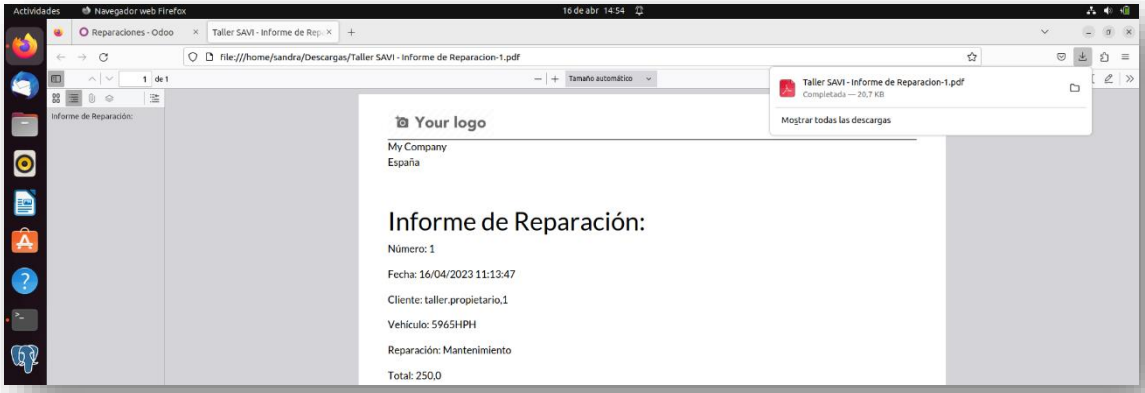
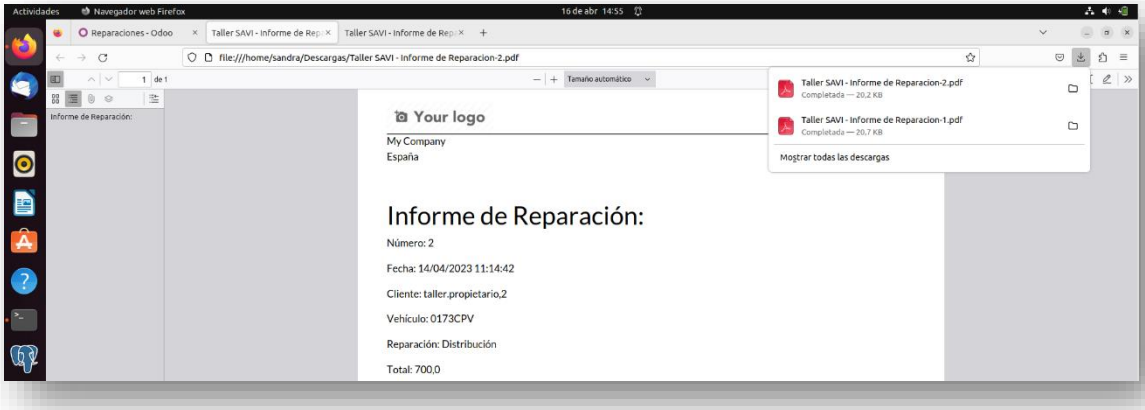


Si pinchamos sobre dicha opción, nos permite descargar el informe que se ha creado con los datos:



Descargando así el informe en PDF:

Informe 1: Sandra Sanabrias (propietario1)



Informe 2: Sergio Muñoz (propietario2)