

Proyecto de organización, consulta y tratamiento de la información de Odoo en implantación cliente/servidor

2º Desarrollo de aplicaciones multiplataforma

Sumario

1. PgAdmin 4.....	3
1.1 Qué es PgAdmin 4.....	3
1.2 Guía de instalación.....	4
1.3 Conexión a la base de datos de Odoo.....	10
1.3 Realización de consultas SQL con PgAdmin.....	17
2. Elaboración de vistas y formularios.....	19
2.1 Elaboración de plantilla de creación de módulos.....	19
2.2 Creación de vista.....	24
2.3 Adición de datos a la vista.....	34

Índice de figuras

Figura 1: Formas de descargar PgAdmin 4.....	4
Figura 2: PgAdmin versiones.....	5
Figura 3: PgAdmin.exe.....	6
Figura 4: Modo de instalación.....	7
Figura 5: Licencia de PgAdmin.....	8
Figura 6: Inicio de instalación.....	9
Figura 7: Instalación de PgAdmin completada.....	10
Figura 8: Menú principal PgAdmin 4.....	11
Figura 9: Desactivar firewall.....	12
Figura 10: Archivo postgresql.conf.....	13
Figura 11: Archivo pg_hba.conf.....	14
Figura 12: Cambiar contraseña a usuario.....	15
Figura 13: Parámetros de conexión en PgAdmin 4.....	16
Figura 14: Acceso a la base de datos con PgAdmin.....	17
Figura 15: Query Tool.....	18
Figura 16: Consulta sobre la base de datos.....	19
Figura 17: Creación de directorio modules.....	20
Figura 18: Añadir directorio al archivo odoo.conf.....	21
Figura 19: Comando odoo scaffold.....	22
Figura 20: Plantilla creada.....	23
Figura 21: Reinicio de Odoo.....	24
Figura 22: Módulo creado.....	25
Figura 23: Models.py.....	26
Figura 24: Modificación archivo models.py.....	27
Figura 25: views.xml.....	28
Figura 26: Modificación del archivo views.xml.....	29
Figura 27: Error ids Many2many.....	31
Figura 28: Módulo creado correctamente.....	32
Figura 29: Añadir campos nuevos a la vista.....	33
Figura 30: Vista actualizada.....	34
Figura 31: Nueva compañía.....	35
Figura 32: Nueva compañía creada correctamente.....	36

1. PgAdmin 4

1.1 Qué es PgAdmin 4

PgAdmin 4 es una herramienta gráfica para gestionar bases de datos en PostgreSQL.

Enlace de descarga: <https://www.pgadmin.org/download/>

El programa lo podremos descargar para diferentes formatos como se muestra en la figura 1.

En nuestro caso, elegiremos la opción de Windows.



Figura 1: Formas de descargar PgAdmin 4

1.2 Guía de instalación

Seleccionamos la última versión (8.13).

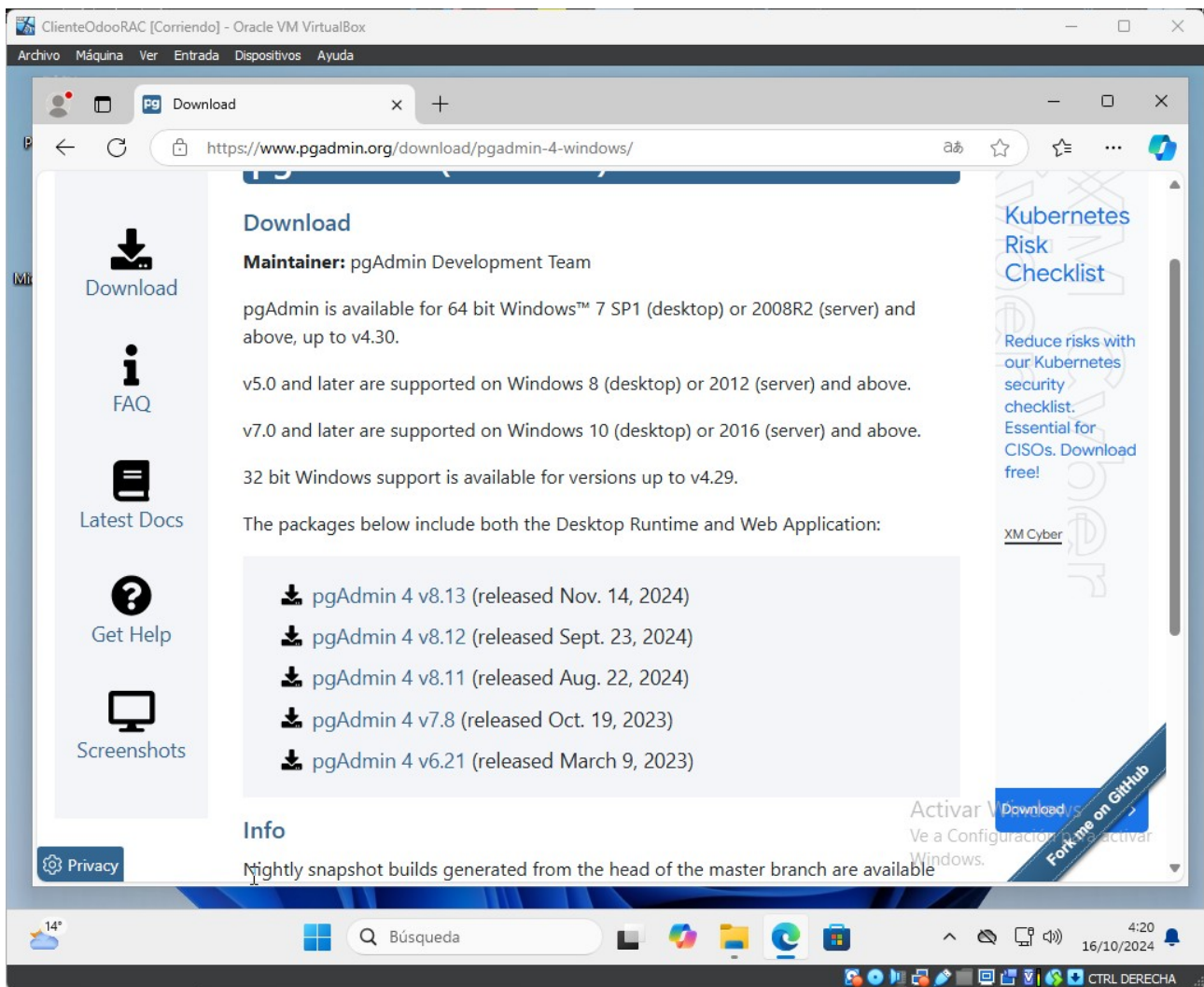


Figura 2: PgAdmin versiones

Descargaremos el archivo.exe que será el instalador.

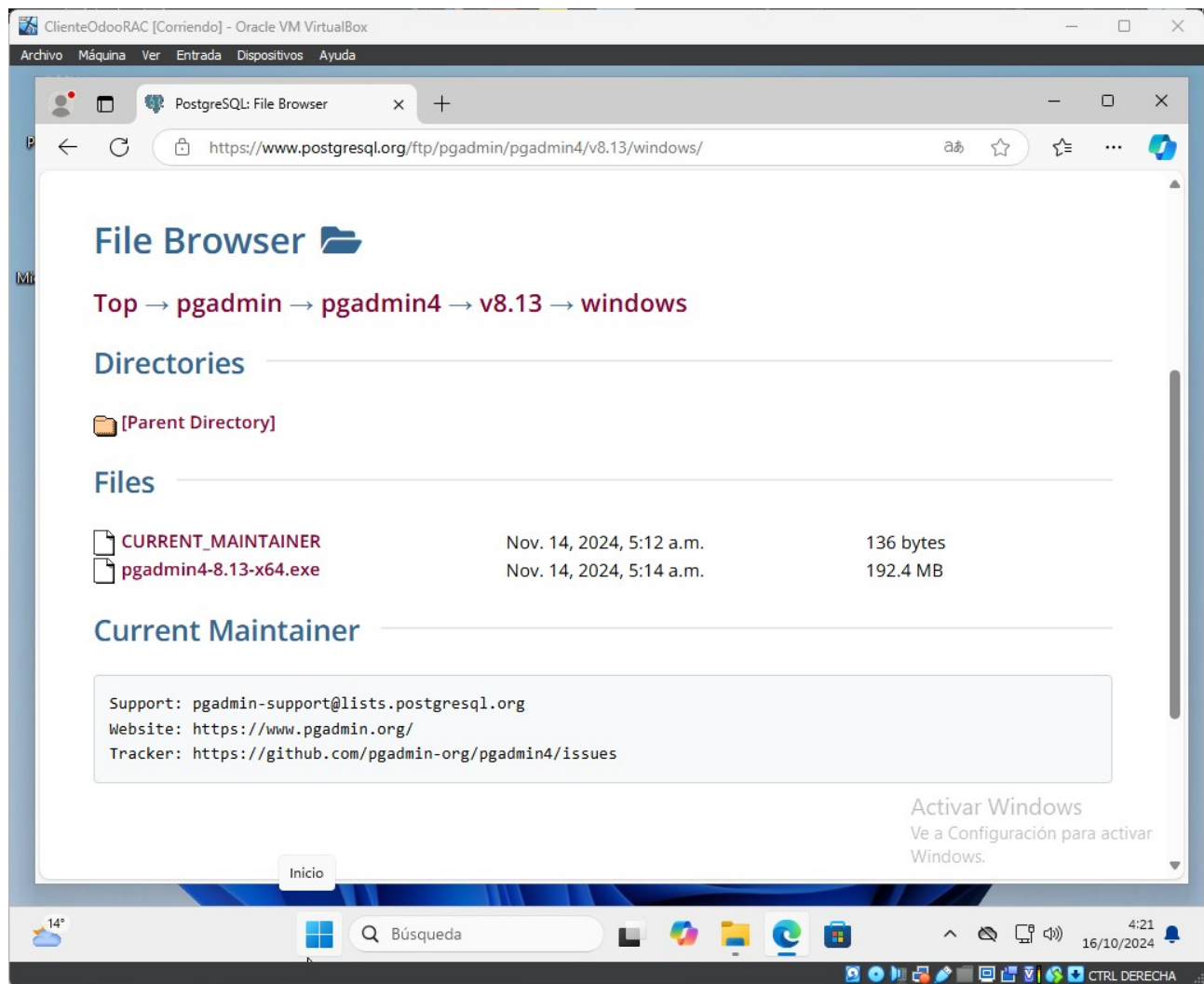


Figura 3: PgAdmin.exe

Una vez descargado, lo ejecutamos y seleccionamos la opción de instalar solo para mí.

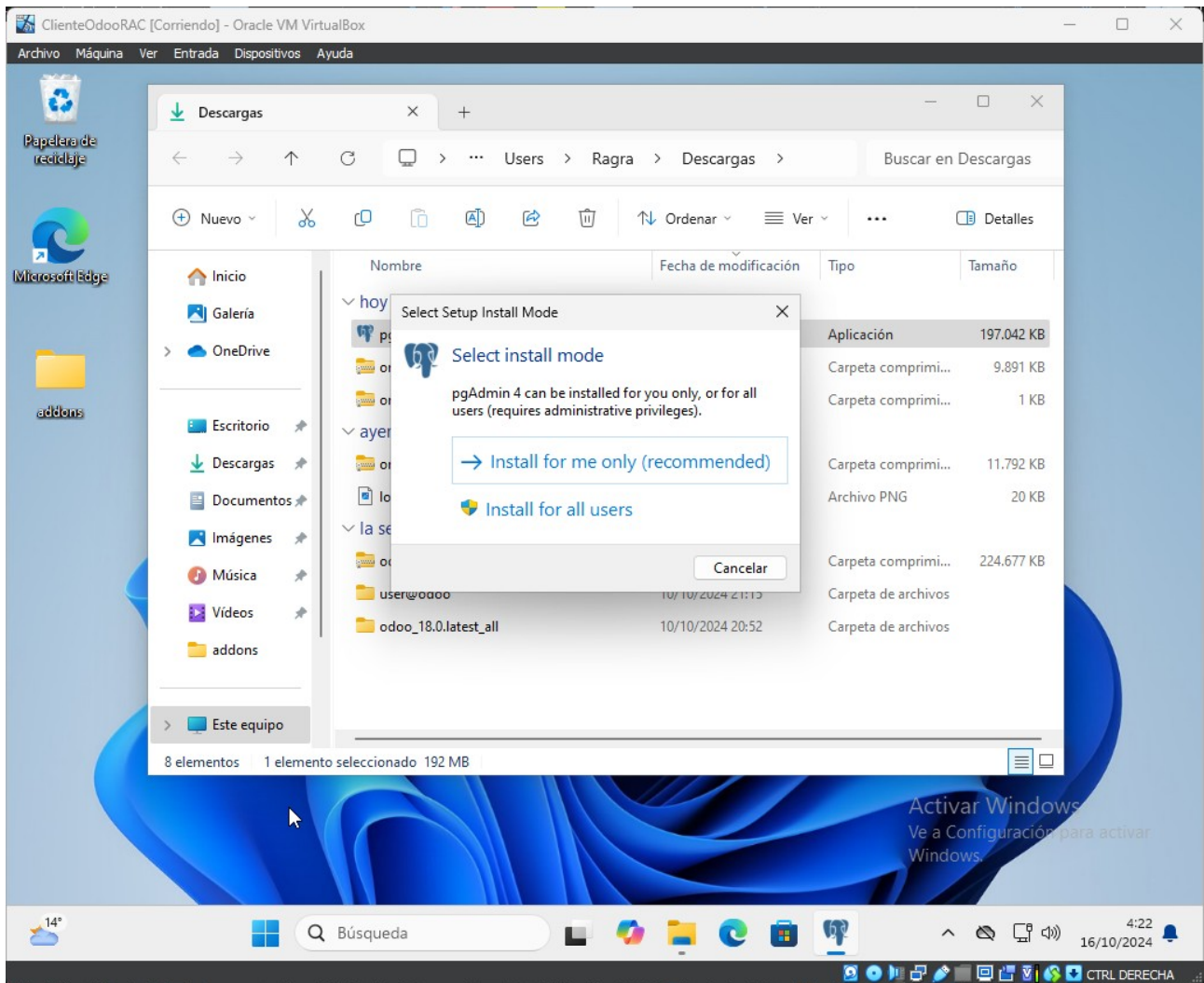


Figura 4: Modo de instalación

Aceptamos las condiciones de la licencia.

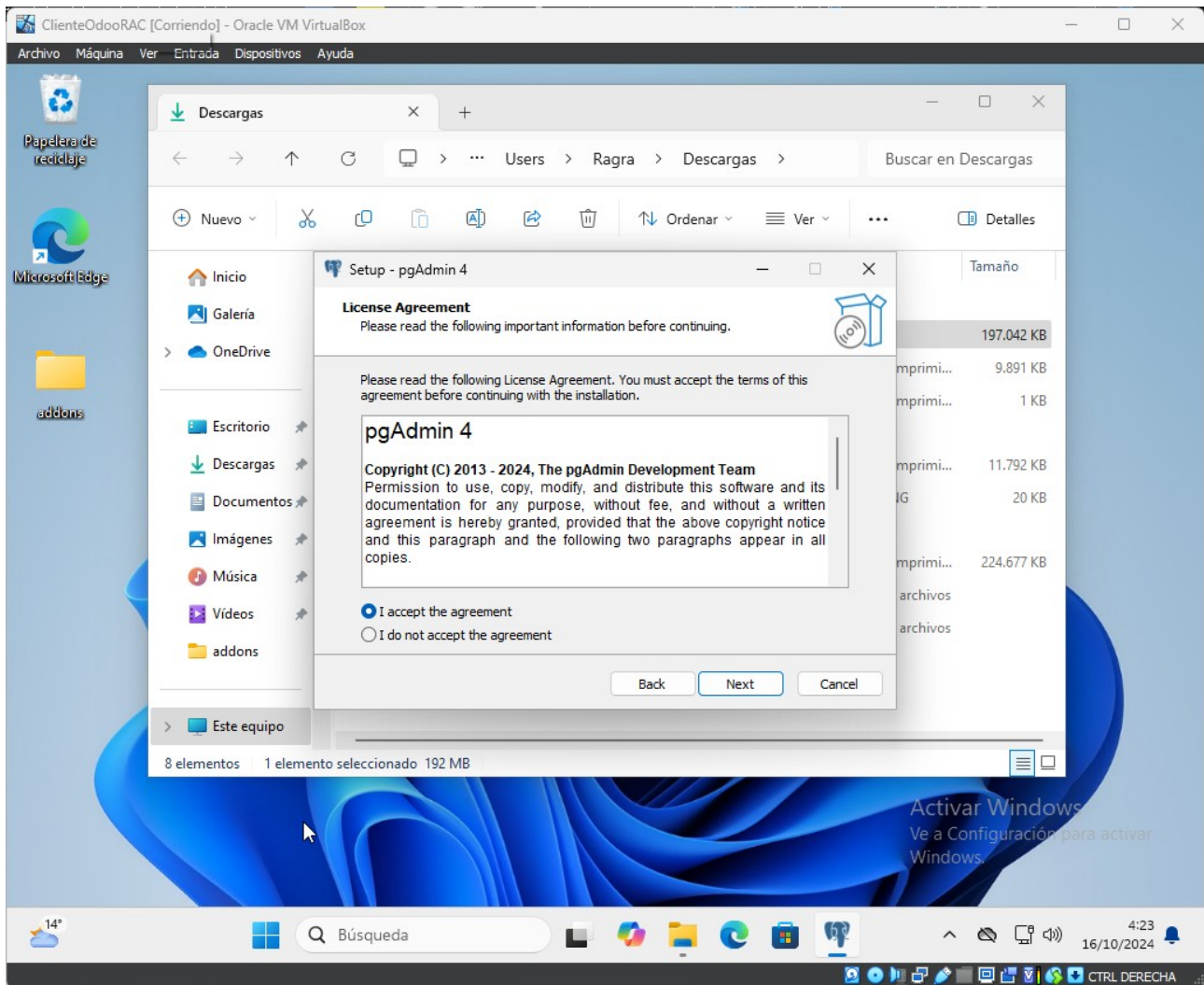


Figura 5: Licencia de PgAdmin

Esperamos a que el programa complete la instalación

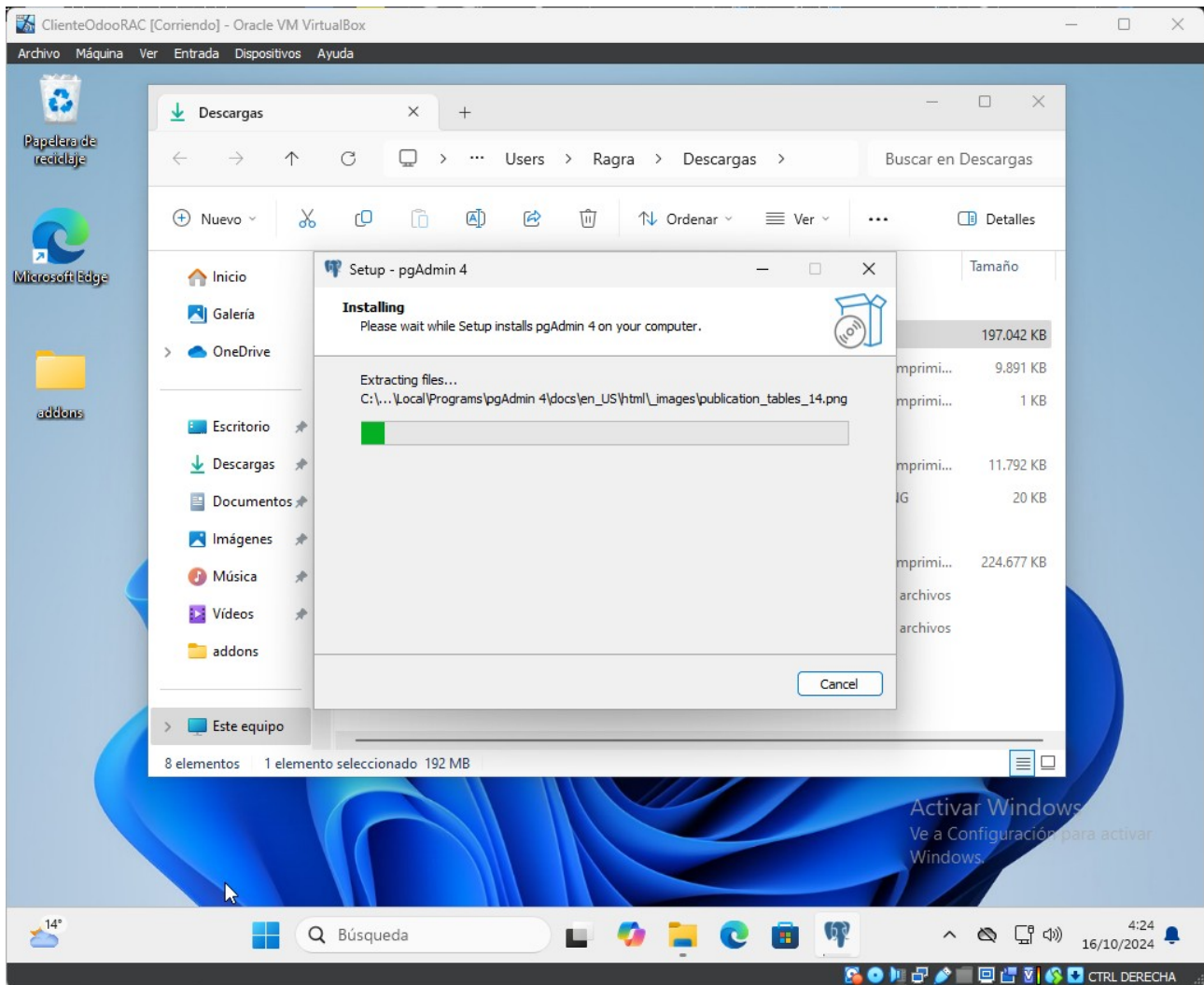


Figura 6: Inicio de instalación

Una vez instalado ya lo podremos abrir.

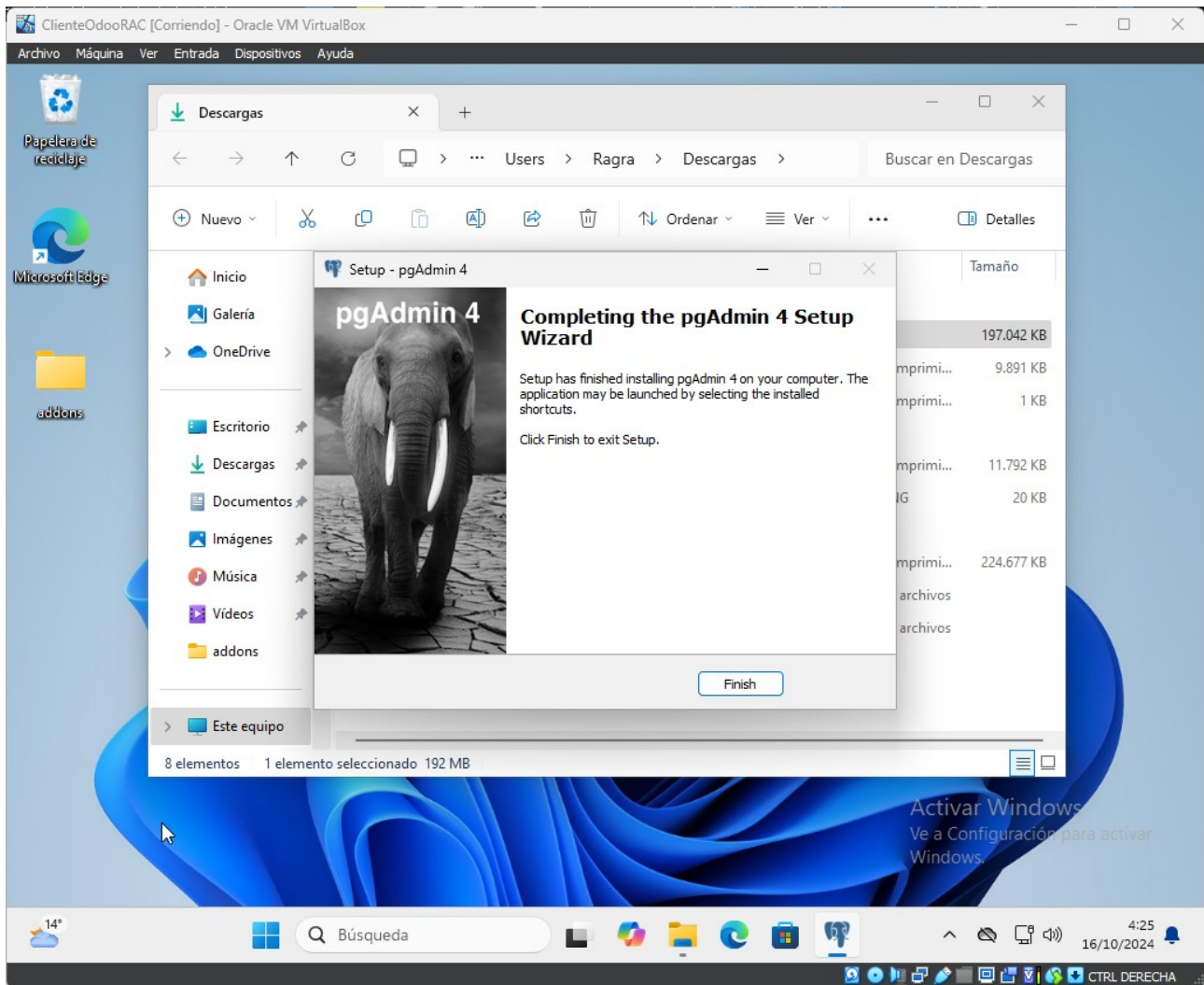


Figura 7: Instalación de PgAdmin completada

1.3 Conexión a la base de datos de Odoo

El siguiente paso será conectarnos a la base de datos de Odoo, en el cual debemos hacer diferentes pasos tanto en la máquina cliente como en la máquina servidor.

Lo primero será abrir PgAdmin 4 y pulsar en “Add new server”.

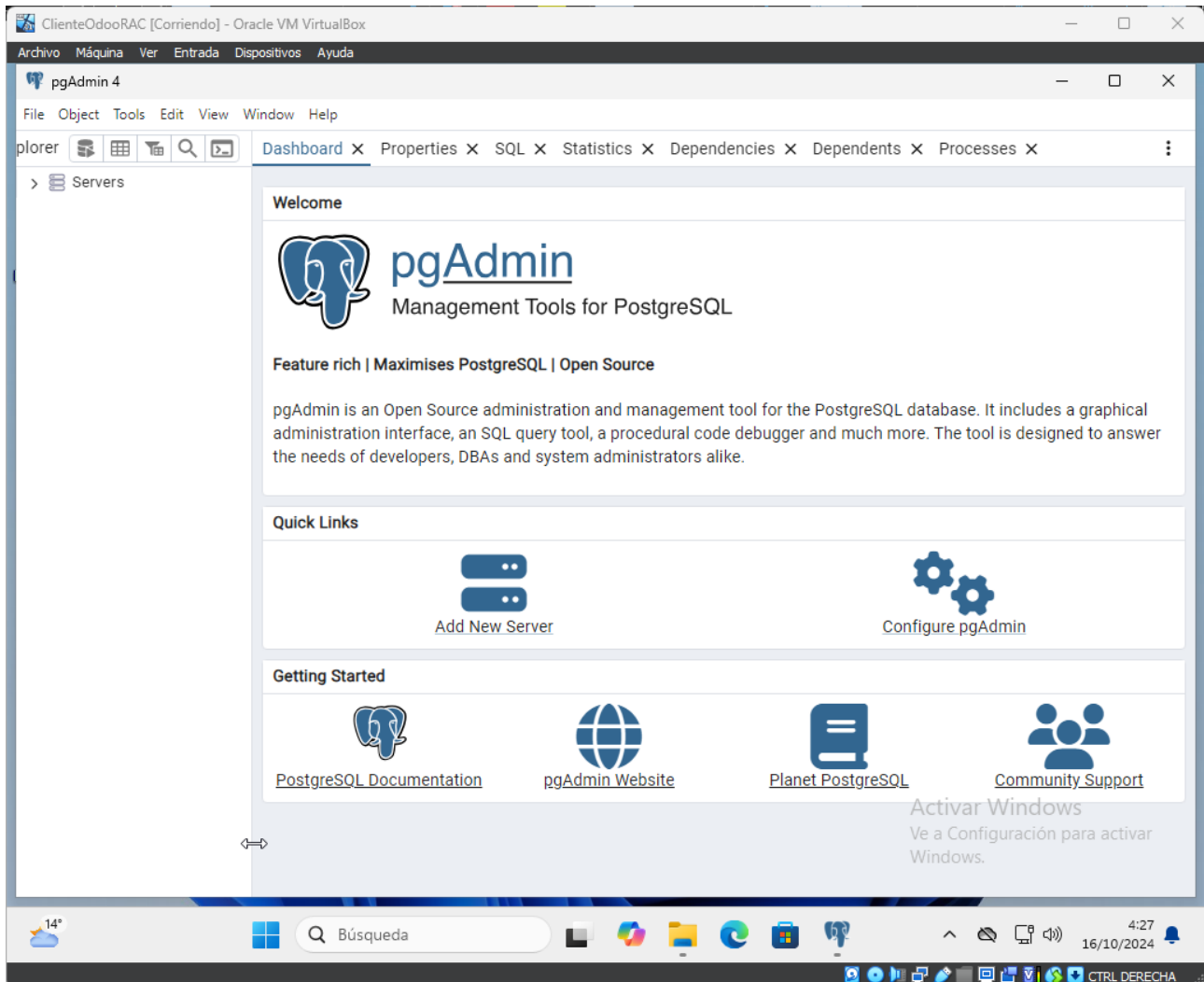
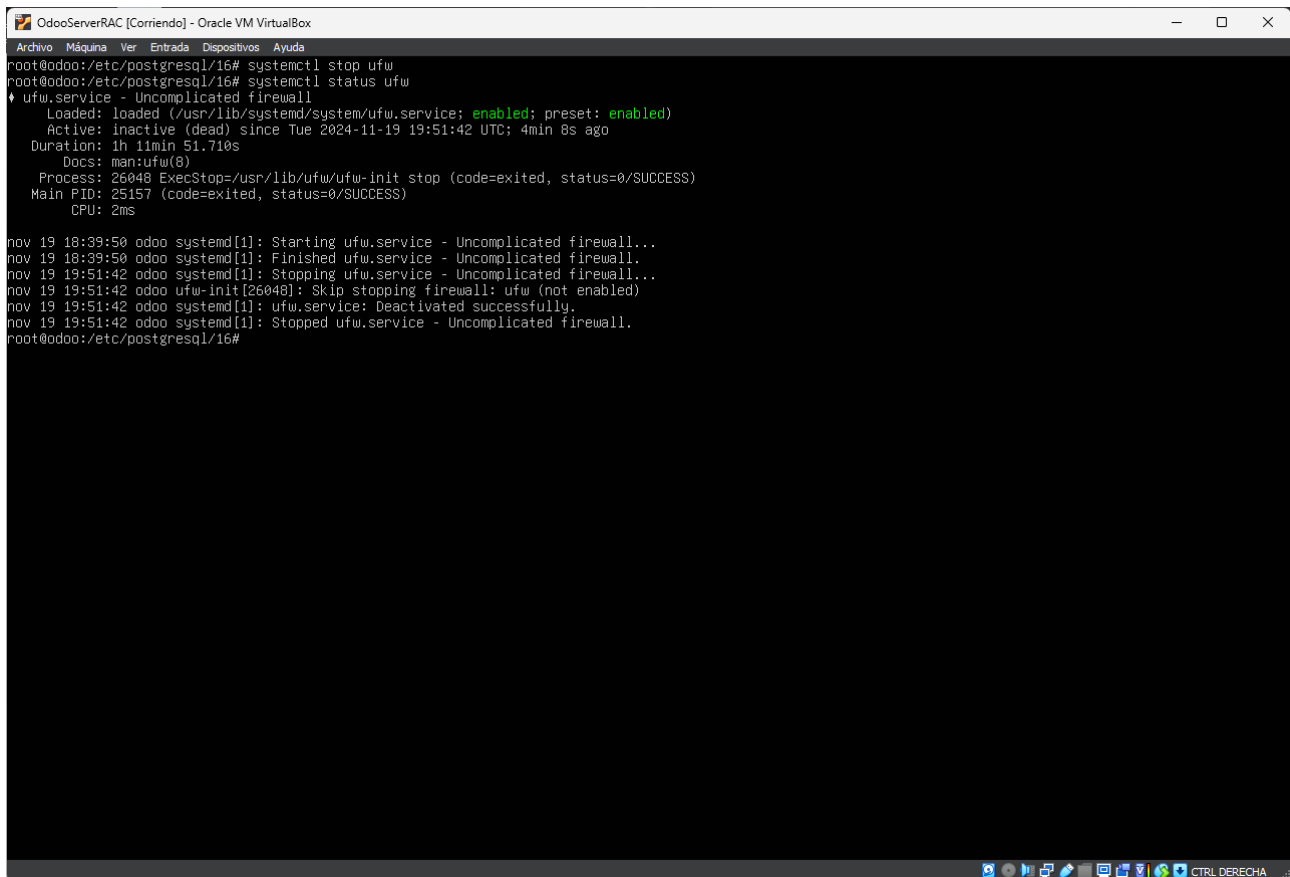


Figura 8: Menú principal PgAdmin 4

En la máquina servidor, debemos desactivar el firewall que impide la conexión a la base de datos. Para ello usaremos el comando “`systemctl stop ufw`”. Luego comprobaremos su estado haciendo “`systemctl status ufw`”.

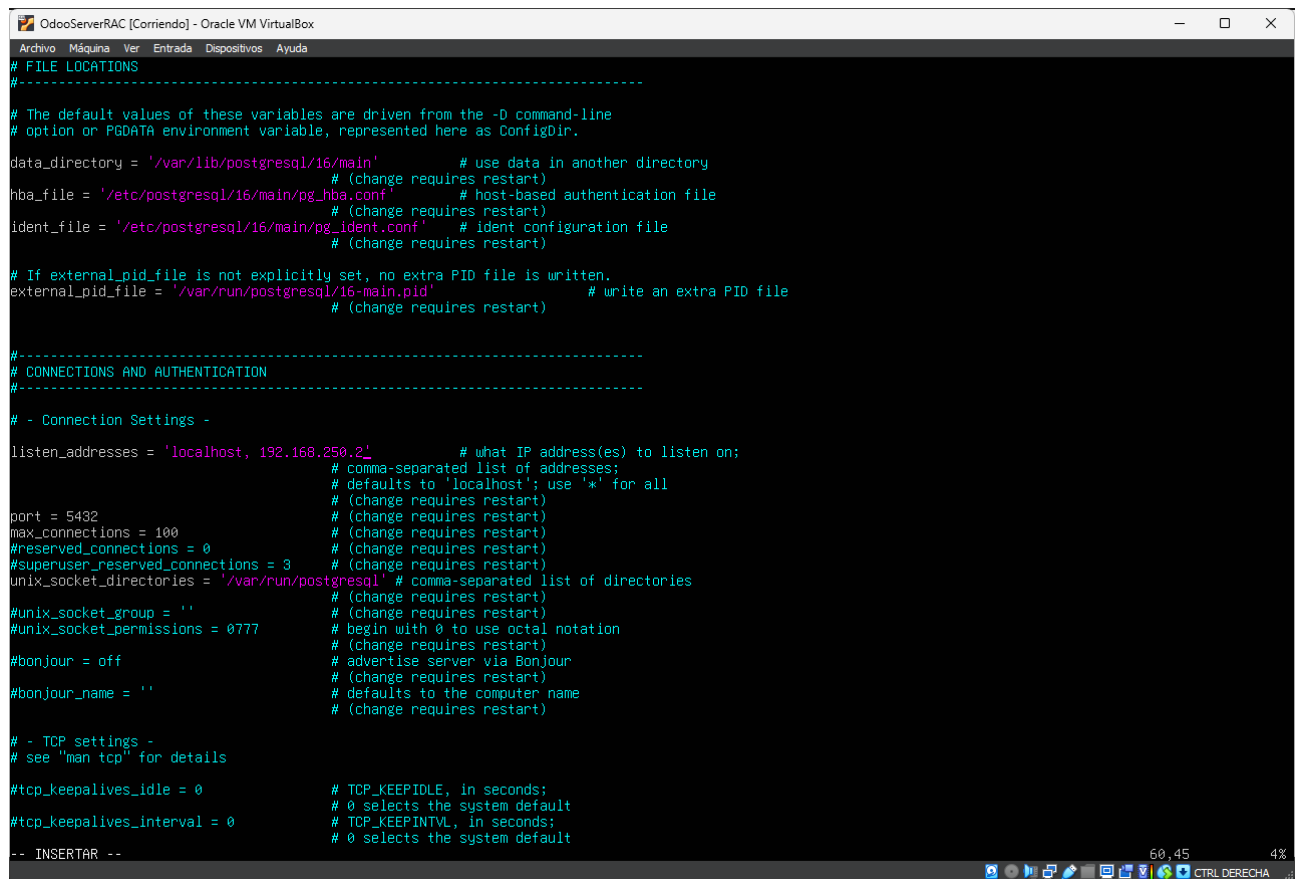


```
OdooServerRAC [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/etc/postgresql/16# systemctl stop ufw
root@odoo:/etc/postgresql/16# systemctl status ufw
* ufw.service - Uncomplicated firewall
   Loaded: loaded (/usr/lib/systemd/system/ufw.service; enabled; preset: enabled)
   Active: inactive (dead) since Tue 2024-11-19 19:51:42 UTC; 4min 8s ago
     Duration: 1h 11min 51.710s
       Docs: man:ufw(8)
   Process: 26048 ExecStop=/usr/lib/ufw/ufw-init stop (code=exited, status=0/SUCCESS)
    Main PID: 25157 (code=exited, status=0/SUCCESS)
       CPU: 2ms

nov 19 18:39:50 odoo systemd[1]: Starting ufw.service - Uncomplicated firewall...
nov 19 18:39:50 odoo systemd[1]: Finished ufw.service - Uncomplicated firewall.
nov 19 19:51:42 odoo systemd[1]: Stopping ufw.service - Uncomplicated firewall...
nov 19 19:51:42 odoo ufw-init[26048]: Skip stopping firewall: ufw (not enabled)
nov 19 19:51:42 odoo systemd[1]: ufw.service: Deactivated successfully.
nov 19 19:51:42 odoo systemd[1]: Stopped ufw.service - Uncomplicated firewall.
root@odoo:/etc/postgresql/16#
```

Figura 9: Desactivar firewall

A continuación deberemos ir a `/etc/postgresql/16/main` y abrir el archivo `postgresql.conf`. Buscaremos el apartado de conexiones y autenticaciones y en `listen_addresses =` pondremos `localhost, 192.168.250.2` (que es la dirección IP de la máquina del servidor) o simplemente podremos poner `*` para que escuche cualquier dirección.

A screenshot of a terminal window titled "OdooServerRAC [Corriendo] - Oracle VM VirtualBox". The terminal displays the content of the `postgresql.conf` file. The file is divided into sections: "FILE LOCATIONS", "CONNECTIONS AND AUTHENTICATION", and "TCP Settings". The "listen_addresses" parameter is set to `'localhost, 192.168.250.2'`. Other parameters like `port`, `max_connections`, and `unix_socket_directories` are also visible. The terminal has a dark background with light-colored text. The window's title bar and menu bar are visible at the top. The bottom of the window shows a taskbar with various icons and system information like "60.45" and "4%".

```
# FILE LOCATIONS
#-----
# The default values of these variables are driven from the -D command-line
# option or PGDATA environment variable, represented here as ConfigDir.

data_directory = '/var/lib/postgresql/16/main'      # use data in another directory
                                                    # (change requires restart)
hba_file = '/etc/postgresql/16/main/pg_hba.conf'    # host-based authentication file
                                                    # (change requires restart)
ident_file = '/etc/postgresql/16/main/pg_ident.conf' # ident configuration file
                                                    # (change requires restart)

# If external_pid_file is not explicitly set, no extra PID file is written.
external_pid_file = '/var/run/postgresql/16-main.pid' # write an extra PID file
                                                    # (change requires restart)

#-----
# CONNECTIONS AND AUTHENTICATION
#-----

# - Connection Settings -

listen_addresses = 'localhost, 192.168.250.2'      # what IP address(es) to listen on;
                                                    # comma-separated list of addresses;
                                                    # defaults to 'localhost'; use '*' for all
                                                    # (change requires restart)
port = 5432                                         # (change requires restart)
max_connections = 100                             # (change requires restart)
#reserved_connections = 0                         # (change requires restart)
#superuser_reserved_connections = 3               # (change requires restart)
unix_socket_directories = '/var/run/postgresql'    # comma-separated list of directories
                                                    # (change requires restart)
#unix_socket_group = ''                           # (change requires restart)
#unix_socket_permissions = 0777                   # begin with 0 to use octal notation
                                                    # (change requires restart)
#bonjour = off                                     # advertise server via Bonjour
                                                    # (change requires restart)
#bonjour_name = ''                                # defaults to the computer name
                                                    # (change requires restart)

# - TCP Settings -
# see "man tcp" for details

#tcp_keepalives_idle = 0                           # TCP_KEEPIRL, in seconds;
                                                    # 0 selects the system default
#tcp_keepalives_interval = 0                       # TCP_KEEPIRL, in seconds;
                                                    # 0 selects the system default

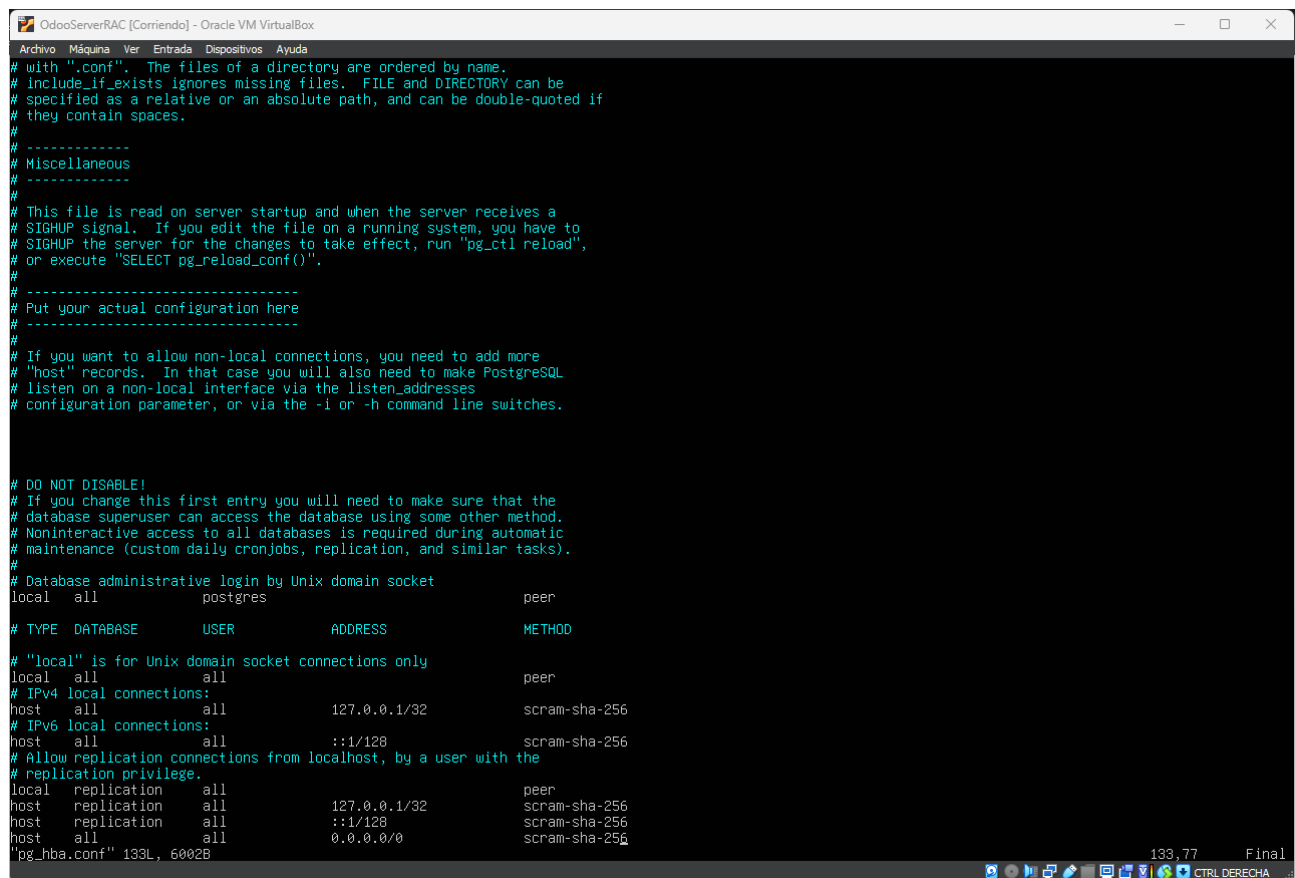
-- INSERTAR --
```

Figura 10: Archivo `postgresql.conf`

Una vez modificado el archivo, guardaremos los cambios.

Luego, en el mismo directorio, modificaremos el archivo `pg_hba.conf` añadiremos una línea al final del documento:

```
host all all 0.0.0.0/0 scram-sha-256
```

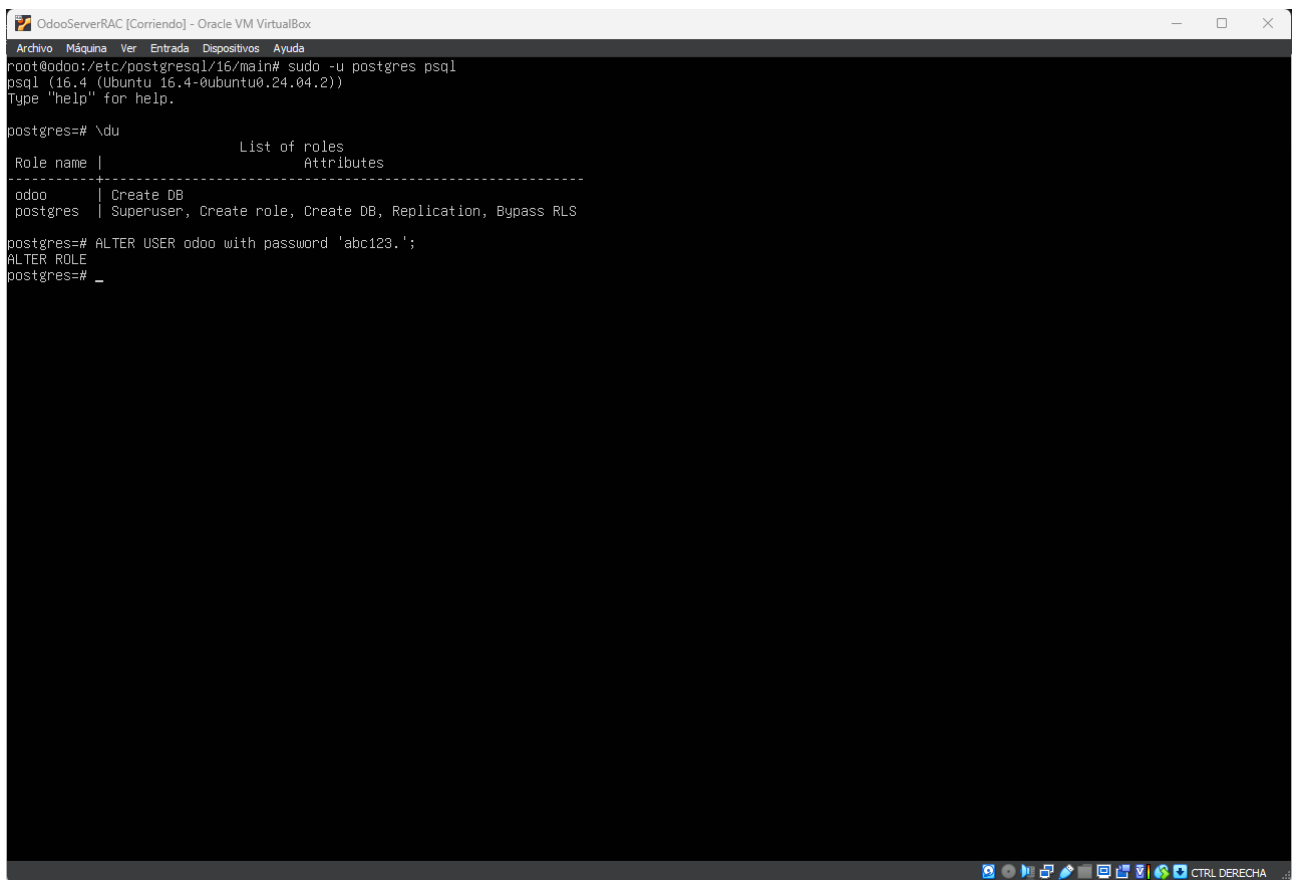
A screenshot of a terminal window titled "OdooServerRAC [Corriendo] - Oracle VM VirtualBox". The terminal displays the content of the `pg_hba.conf` file. The file contains various configuration comments and a table of database connections. The table has columns: TYPE, DATABASE, USER, ADDRESS, and METHOD. The last line of the file is the configuration entry added in the previous step: `host all all 0.0.0.0/0 scram-sha-256`. The terminal window shows a standard Linux-style command prompt and file editor interface.

```
# with ".conf". The files of a directory are ordered by name,
# include_if_exists ignores missing files. FILE and DIRECTORY can be
# specified as a relative or an absolute path, and can be double-quoted if
# they contain spaces.
#
# -----
# Miscellaneous
# -----
#
# This file is read on server startup and when the server receives a
# SIGHUP signal. If you edit the file on a running system, you have to
# SIGHUP the server for the changes to take effect, run "pg_ctl reload",
# or execute "SELECT pg_reload_conf()".
#
# -----
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -l or -h command line switches.
#
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local all postgres peer
# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256
host all all 0.0.0.0/0 scram-sha-256
"pg_hba.conf" 133L, 6002B
```

Figura 11: Archivo `pg_hba.conf`

Una vez modificado el archivo, guardaremos los cambios.

El siguiente paso será cambiar la contraseña al usuario odoo en la base de datos de PostgreSQL con el que, posteriormente, utilizaremos para acceder. Para ello usaremos primero el comando: `sudo -u postgres psql` para poder entrar. Luego, haremos `\du` para poder ver la lista de usuarios que hay en la base de datos.

A screenshot of a terminal window titled 'OdooServerRAC [Corriendo] - Oracle VM VirtualBox'. The terminal shows the following commands and output:

```
root@odoo:/etc/postgresql/16/main# sudo -u postgres psql
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
Type "help" for help.

postgres=# \du
               List of roles
Role name | Attributes
-----+-----
odoo      | Create DB
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS

postgres=# ALTER USER odoo with password 'abc123.';
ALTER ROLE
postgres=#
```

Figura 12: Cambiar contraseña a usuario

Como se puede ver en la figura, tenemos tanto el usuario odoo como el usuario postgres. Lo que haremos será cambiar la contraseña al usuario odoo.

Para ello escribiremos la sentencia: `ALTER USER odoo with password 'abc123.';`

A continuación saldremos y reiniciaremos PostgreSQL, ufw y Odoo con los siguientes comandos:

```
systemctl restart postgresql
systemctl restart ufw
systemctl restart odoo
```

Luego, volveremos a la máquina cliente y al añadir el servidor pondremos los siguientes parámetros (la contraseña será 'abc123.')

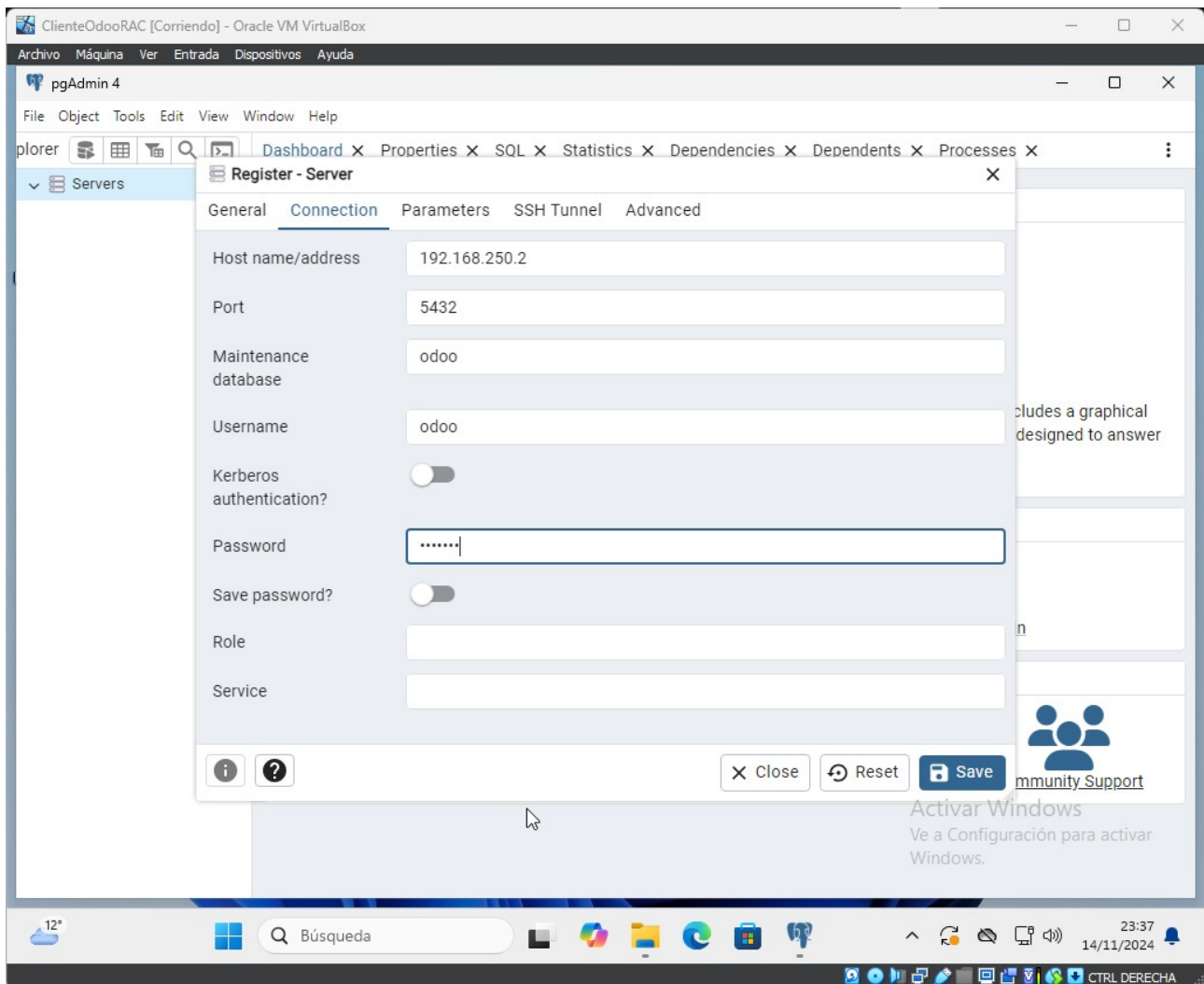


Figura 13: Parámetros de conexión en PgAdmin 4

Una vez puesto todo, pulsaremos en "Save".

De esta forma ya entraríamos en la base de datos de Odoo con PgAdmin.

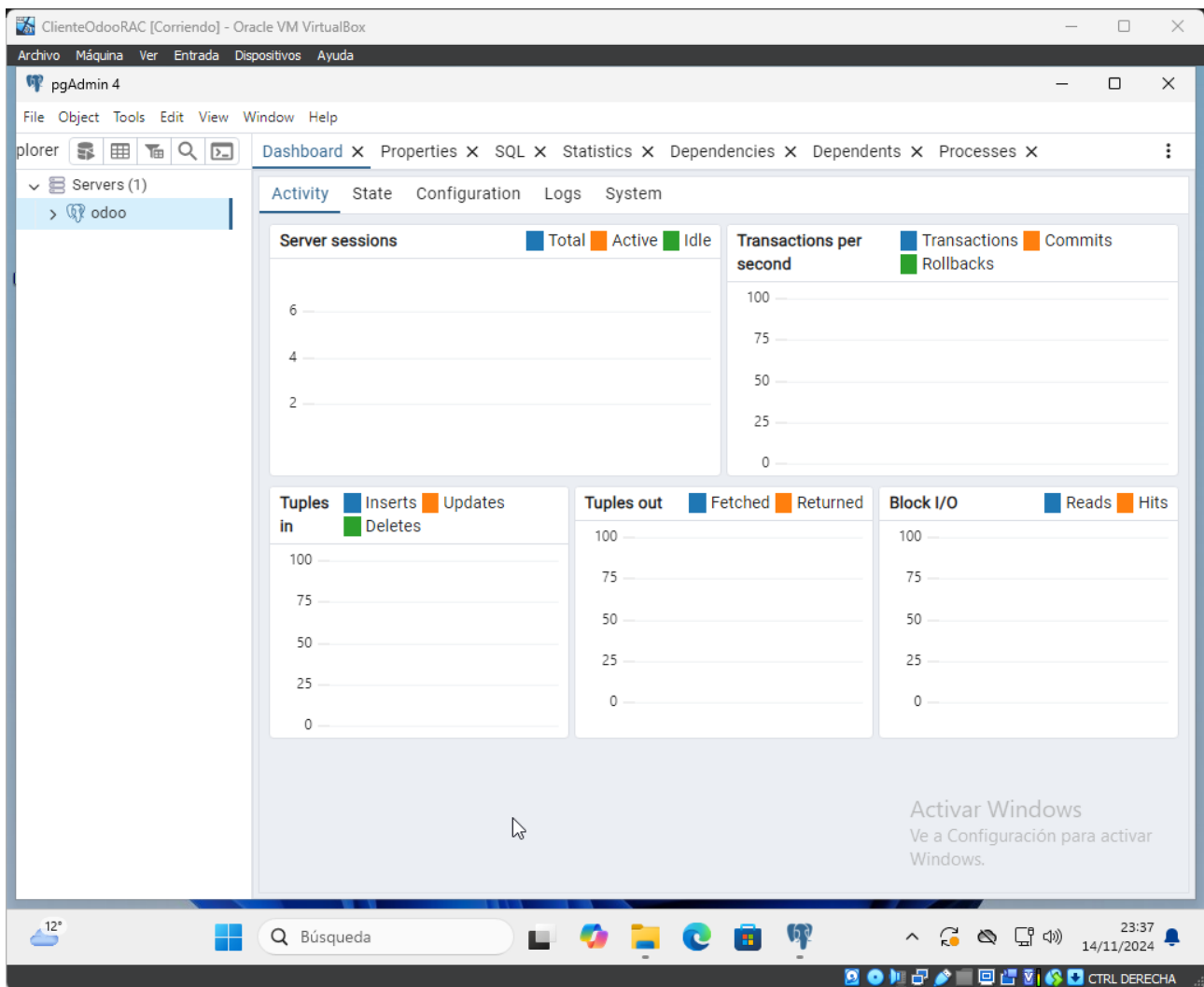


Figura 14: Acceso a la base de datos con PgAdmin

1.3 Realización de consultas SQL con PgAdmin

El siguiente paso que debemos realizar es hacer una consulta de la tabla res.partner. Para ello, lo primero que debemos hacer es pulsar botón derecho sobre la base de datos de Odoo y pulsar en “Query Tool”

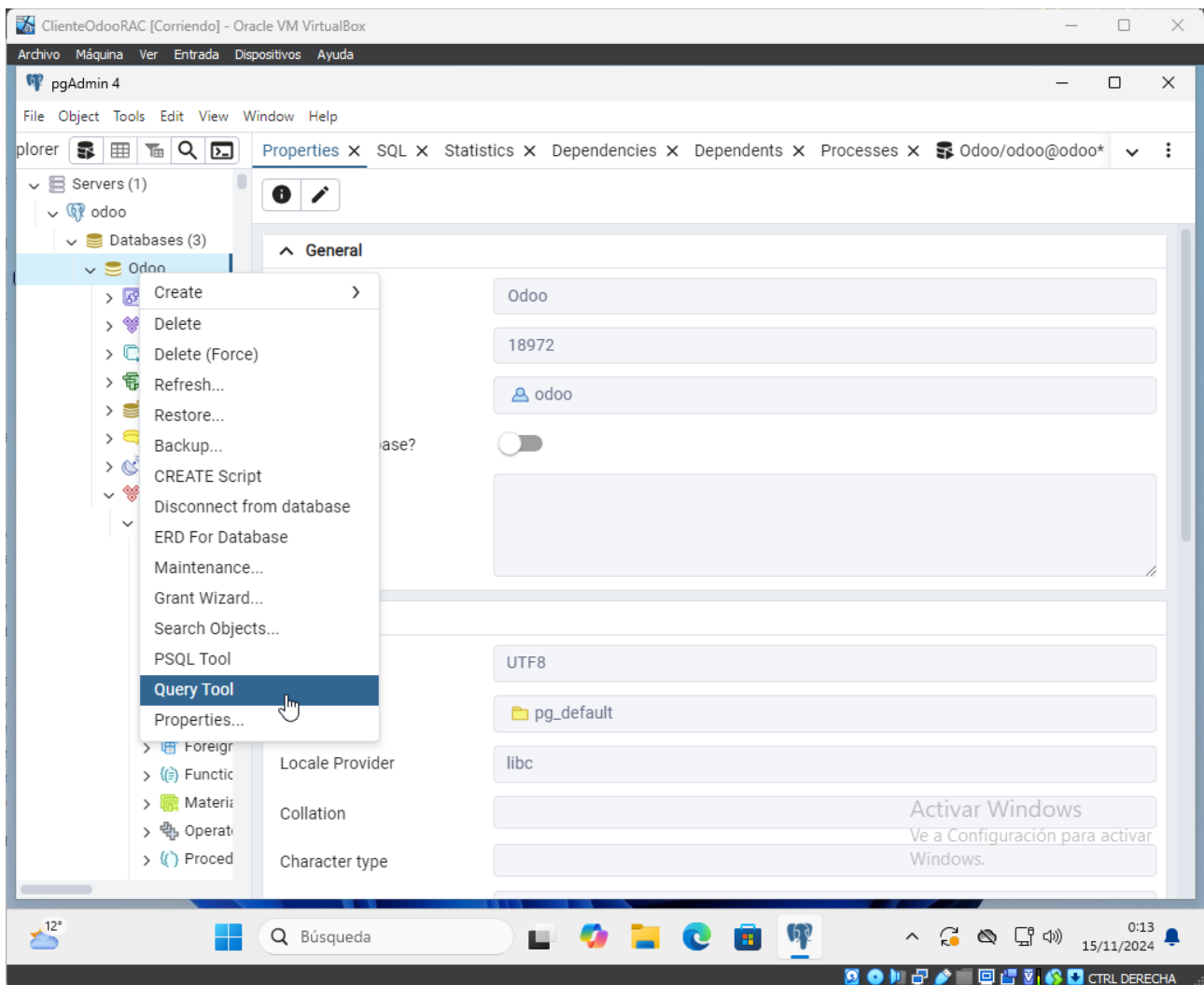


Figura 15: Query Tool

Se nos abrirá un cuadro de texto en el que podremos escribir consultas. Aquí, escribiremos la consulta:

```
SELECT id, name, title, lang, debit_limit, street, zip, city,
phone FROM public.res_partner WHERE is_company = TRUE
```

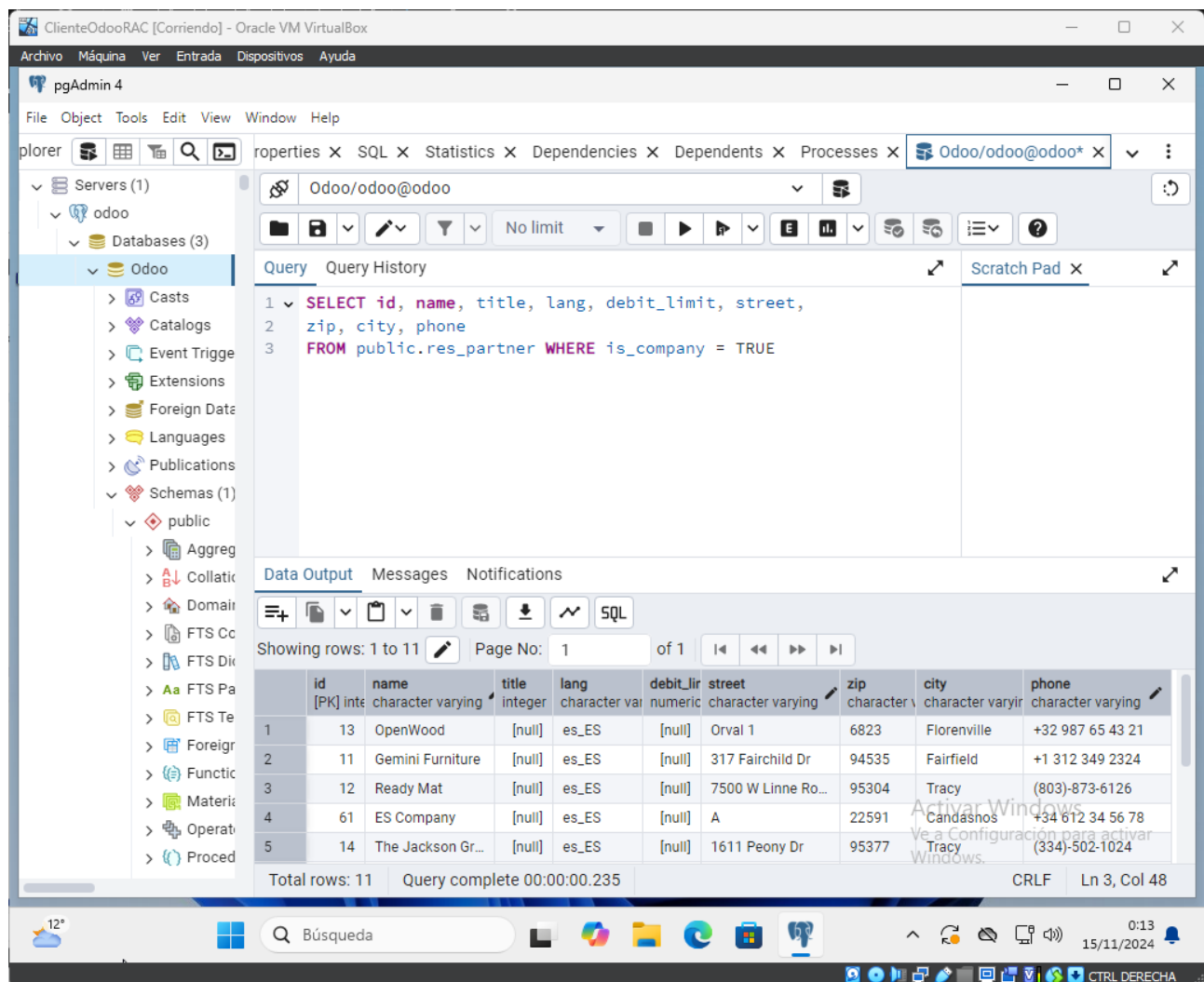


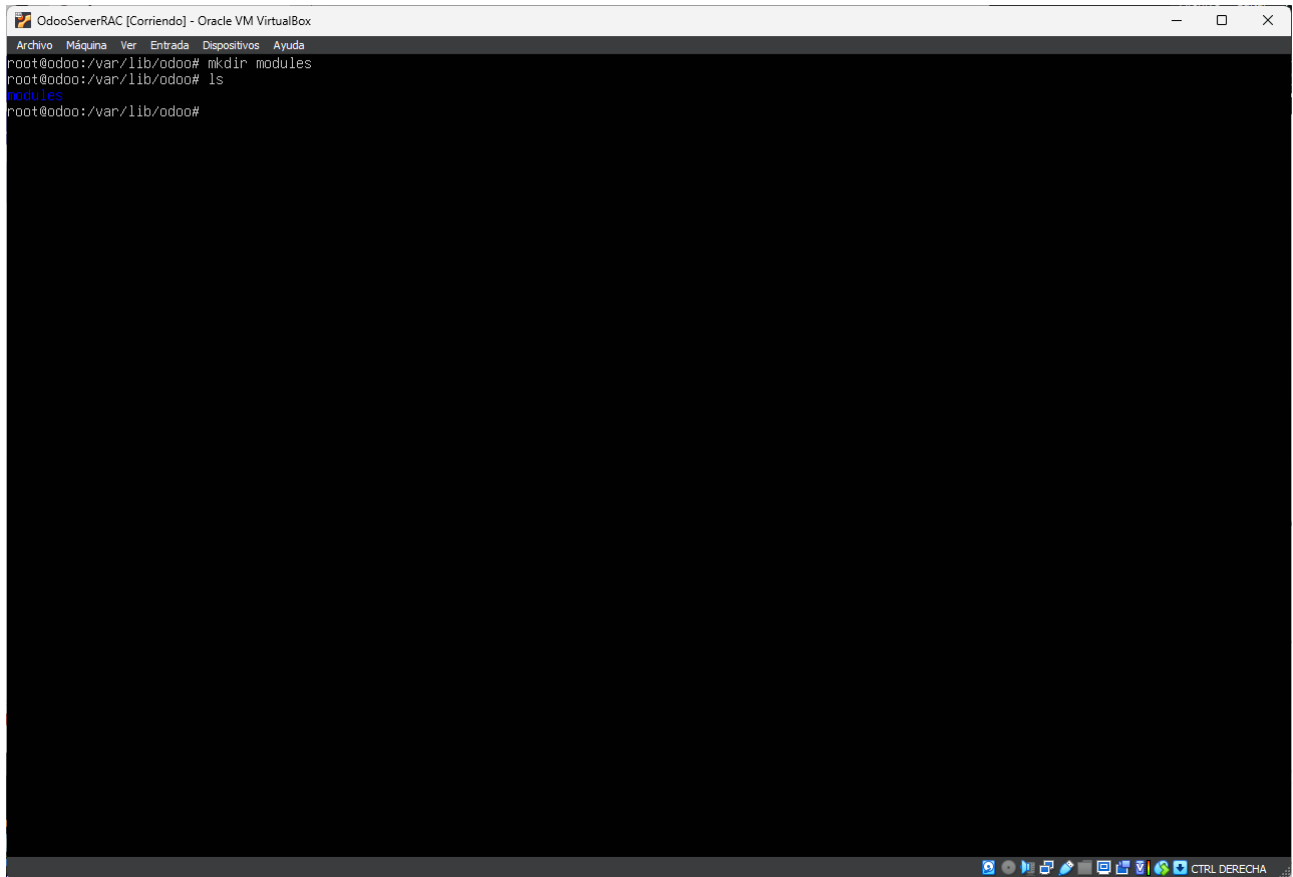
Figura 16: Consulta sobre la base de datos

De esta forma se nos mostrarán los datos en la parte inferior de la ventana.

2. Elaboración de vistas y formularios

2.1 Elaboración de plantilla de creación de módulos

Lo primero que debemos hacer será crear el directorio donde se almacenará el módulo que crearemos. En este caso será en `/var/lib/odoo` y haremos un `mkdir modules`

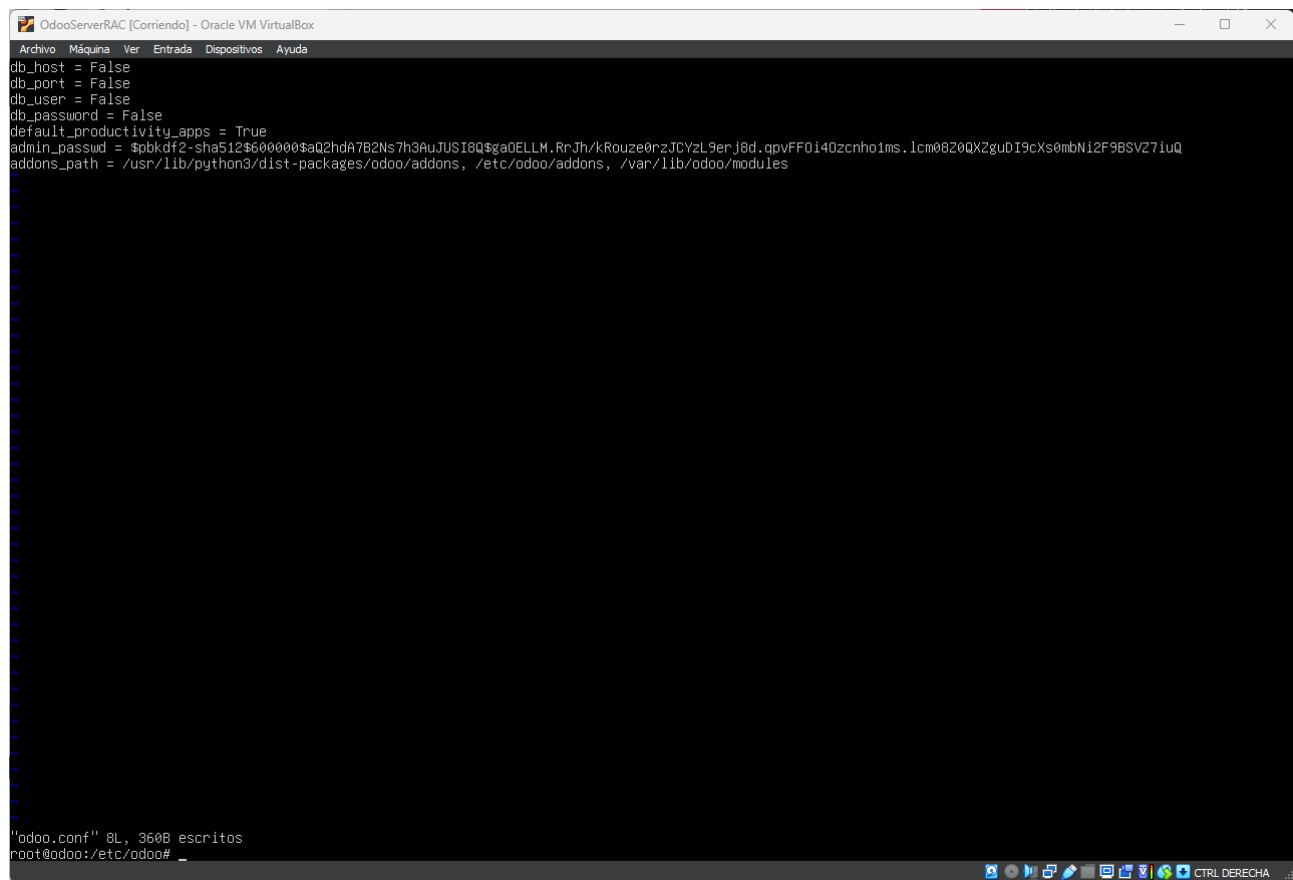


```
OdooServerRAC [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/var/lib/odoo# mkdir modules
root@odoo:/var/lib/odoo# ls
modules
root@odoo:/var/lib/odoo#
```

Figura 17: Creación de directorio modules

Lo siguiente que haremos será añadir el directorio de la carpeta que hemos creado en el archivo de configuración de odoo que hemos utilizado en proyectos anteriores.

`/etc/odoo/odoo.conf`



The screenshot shows a terminal window titled "OdooServerRAC [Corriendo] - Oracle VM VirtualBox". The terminal displays the configuration of the `/etc/odoo/odoo.conf` file. The configuration includes the following settings:

```
db_host = False
db_port = False
db_user = False
db_password = False
default_productivity_apps = True
admin_passwd = $pbkdf2-sha512$600000$aQ2hdA7B2Ns7h3AuJUSI8Q$ga0ELLM.RnJh/kRouze0rzJCYzL9erJ8d.qpvFF0i40zcnho1ms.lcm08Z0QXZguDI9cXs0mbN12F9BSVZ7iuQ
addons_path = /usr/lib/python3/dist-packages/odoo/addons, /etc/odoo/addons, /var/lib/odoo/modules
```

At the bottom of the terminal, a message indicates that the file was successfully updated: `"odoo.conf" 8L, 360B escritos`. The prompt shows the user is root at the odoo machine, located in the `/etc/odoo` directory.

Figura 18: Añadir directorio al archivo odoo.conf

Lo siguiente será crear la plantilla del módulo, Odoo nos proporciona un comando para realizarlo llamado scaffold. Para poder crearla, escribiremos el siguiente comando:

```
odoo scaffold (nombre) /var/lib/odoo/modules
```

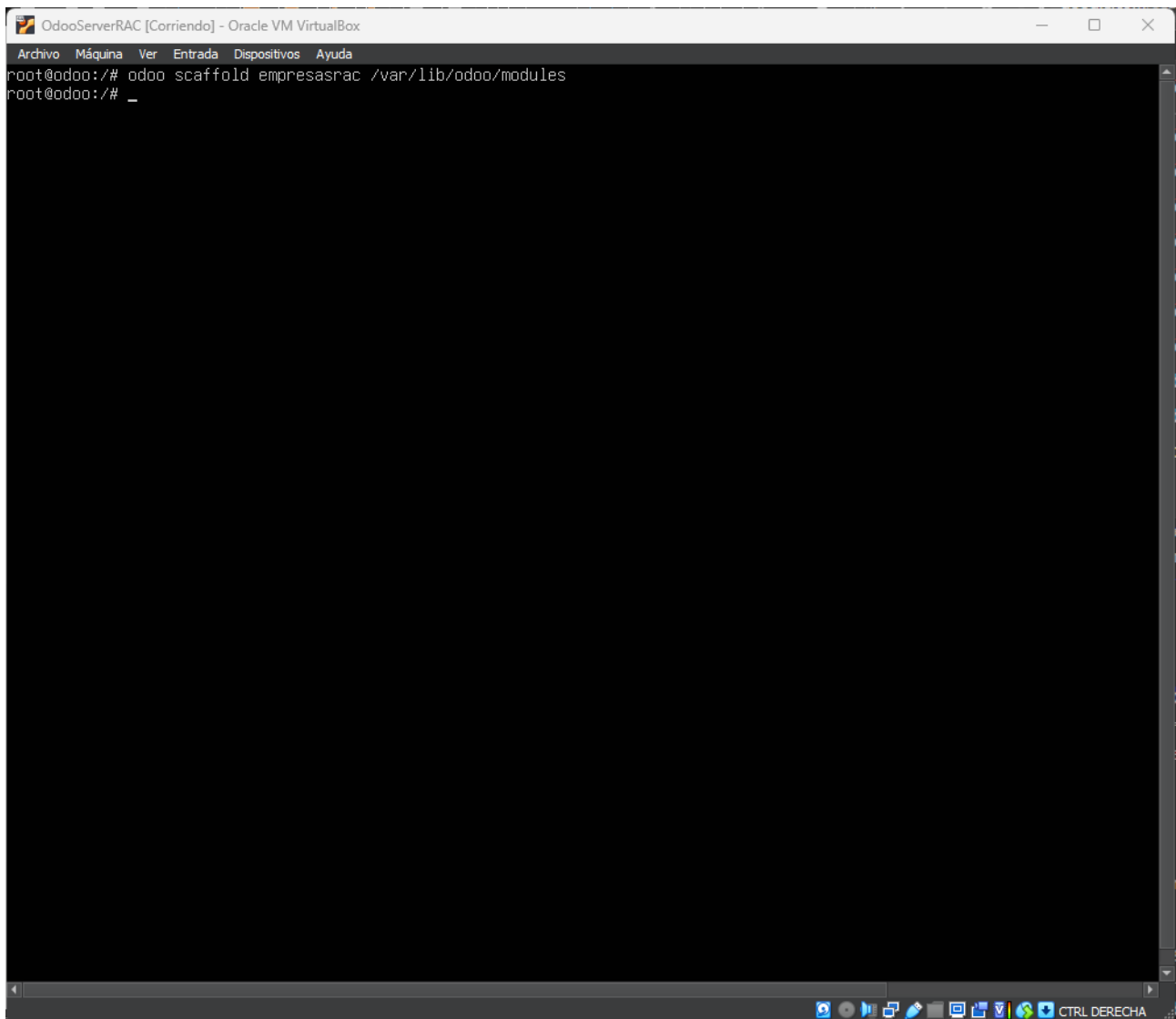
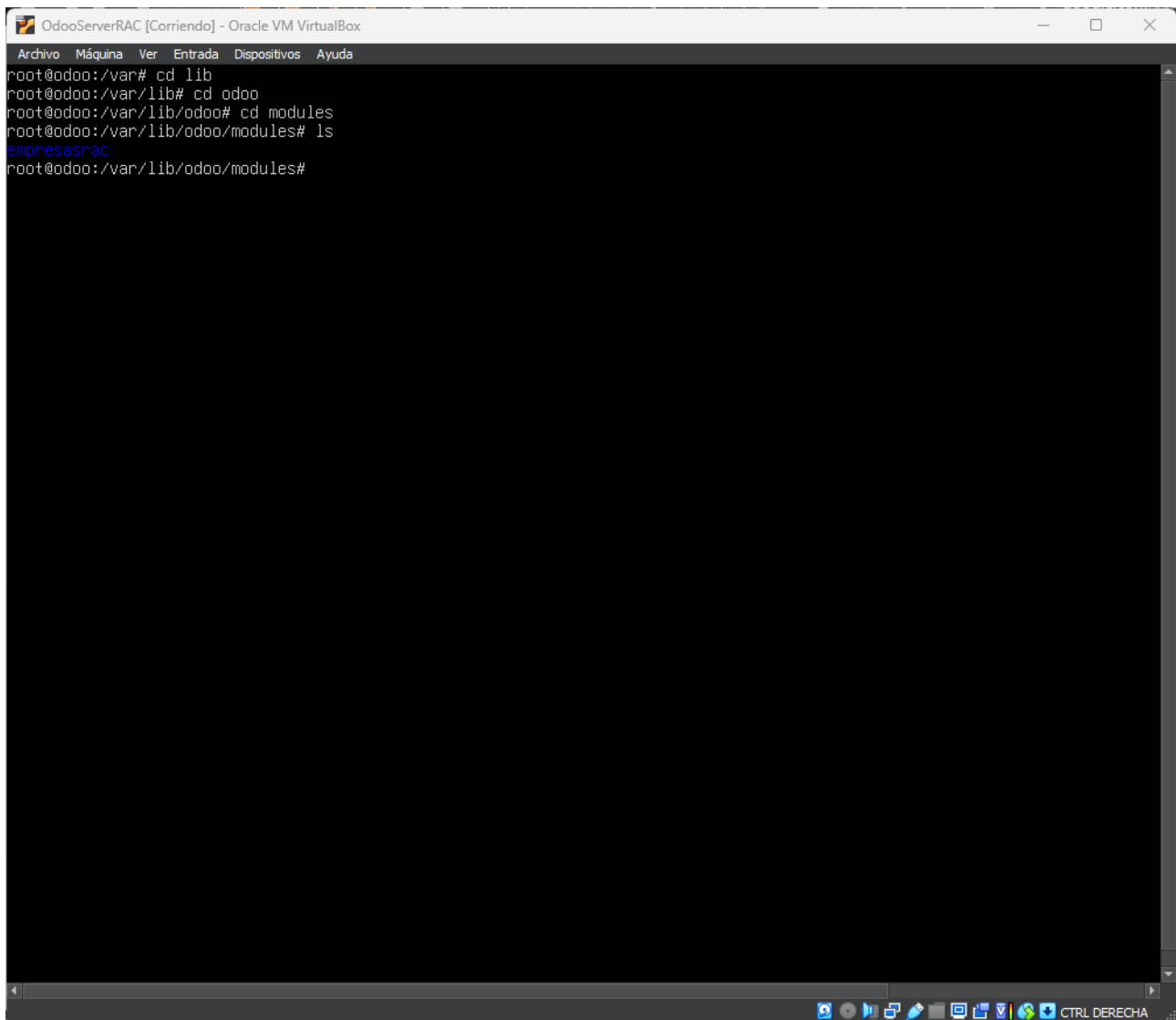


Figura 19: Comando odoo scaffold

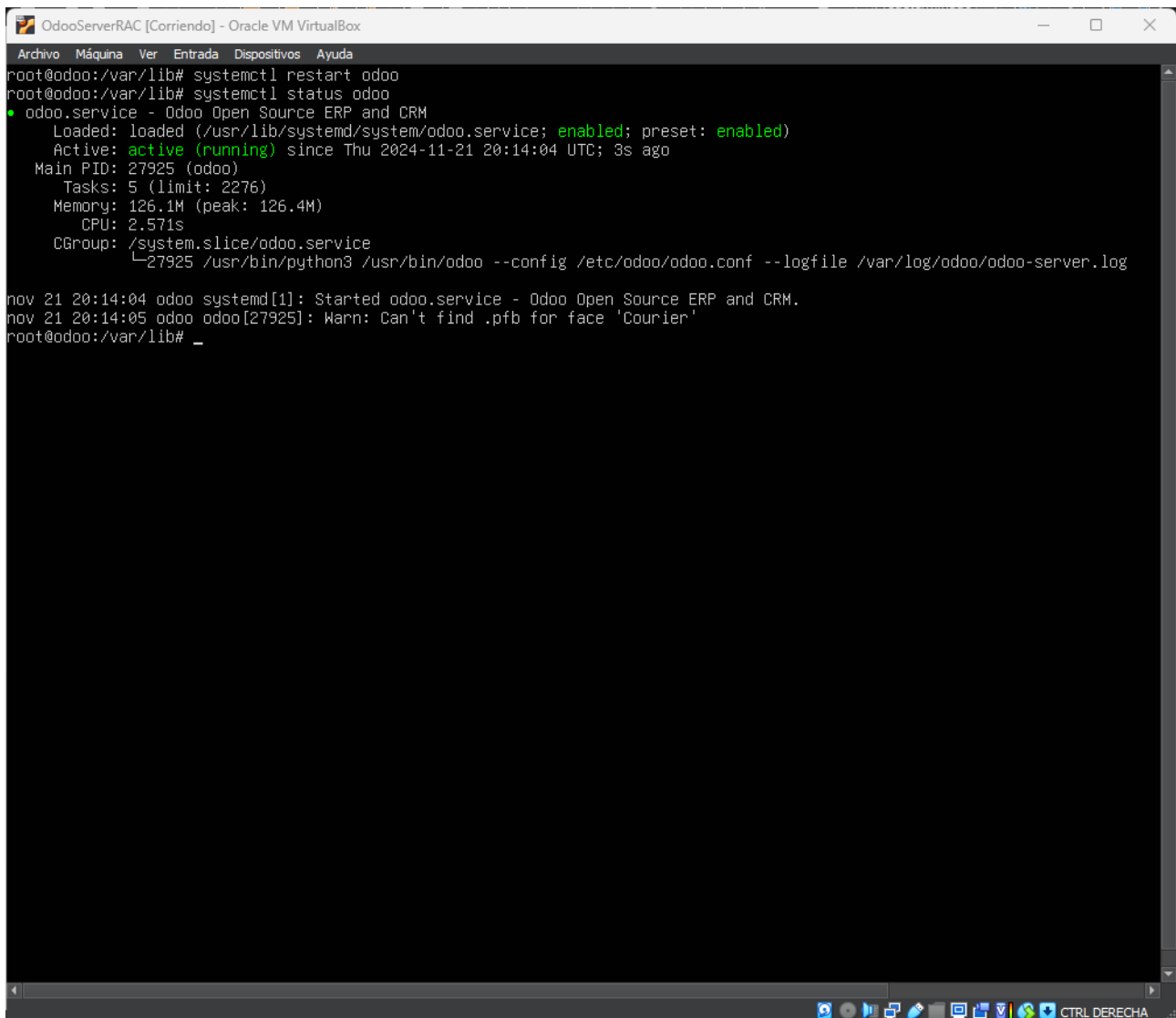
Podremos ver que después de haber efectuado el comando, se ha creado un nuevo directorio con el nombre que le hemos indicado anteriormente con varios archivos dentro.



```
OdooServerRAC [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/var# cd lib
root@odoo:/var/lib# cd odoo
root@odoo:/var/lib/odoo# cd modules
root@odoo:/var/lib/odoo/modules# ls
empresasrac
root@odoo:/var/lib/odoo/modules#
```

Figura 20: Plantilla creada

Reiniciaremos Odoo con `systemctl restart odoo` y comprobaremos su estado con `systemctl status odoo`



```
OdooServerRAC [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/var/lib# systemctl restart odoo
root@odoo:/var/lib# systemctl status odoo
● odoo.service - Odoo Open Source ERP and CRM
   Loaded: loaded (/usr/lib/systemd/system/odoo.service; enabled; preset: enabled)
   Active: active (running) since Thu 2024-11-21 20:14:04 UTC; 3s ago
     Main PID: 27925 (odoo)
        Tasks: 5 (limit: 2276)
      Memory: 126.1M (peak: 126.4M)
         CPU: 2.571s
       CGroup: /system.slice/odoo.service
               └─27925 /usr/bin/python3 /usr/bin/odoo --config /etc/odoo/odoo.conf --logfile /var/log/odoo/odoo-server.log

nov 21 20:14:04 odoo systemd[1]: Started odoo.service - Odoo Open Source ERP and CRM.
nov 21 20:14:05 odoo odoo[27925]: Warn: Can't find .pfb for face 'Courier'
root@odoo:/var/lib# _
```

Figura 21: Reinicio de Odoo

2.2 Creación de vista

Si nos dirigimos a Odoo y buscamos por el módulo nos aparecerá pero debemos configurarlo para crear la vista.

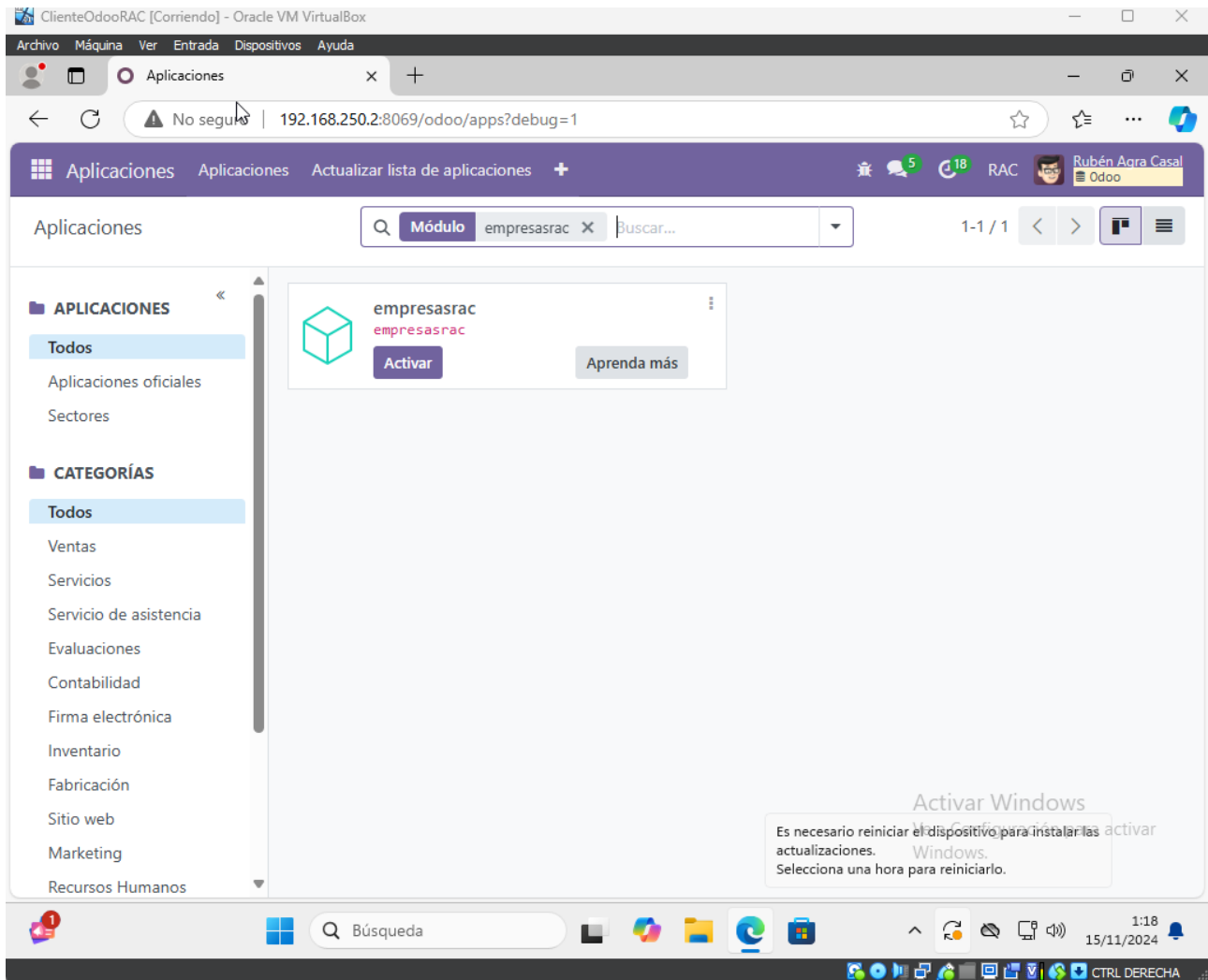
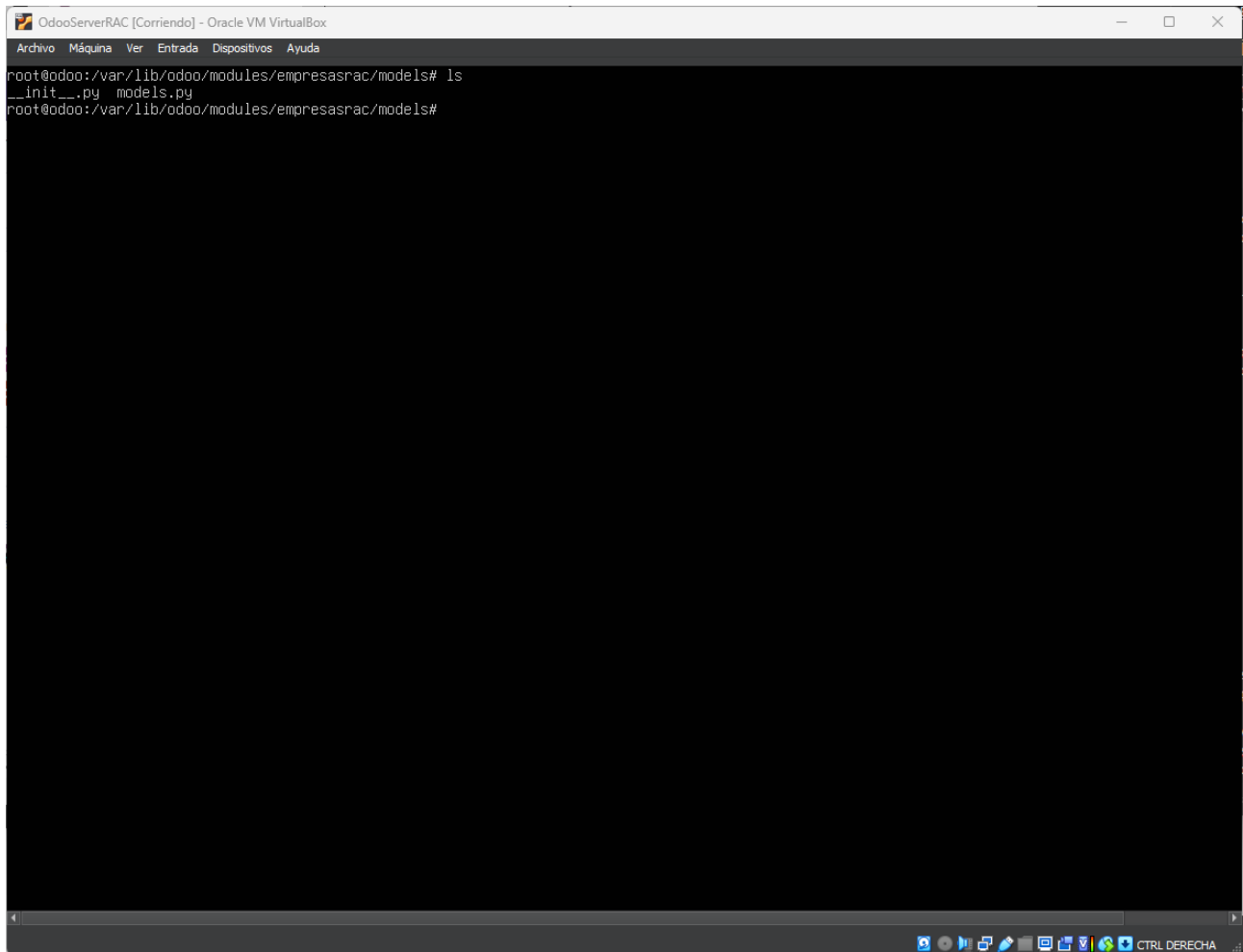


Figura 22: Módulo creado

Si nos dirigimos a la carpeta que se ha creado antes con la plantilla podremos ver varios directorios, uno de ellos es models y dentro habrá un archivo .py que deberemos modificar. Antes de modificarlo es recomendable hacer una copia con el comando cp



```
OdooServerRAC [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/var/lib/odoo/modules/empresasrac/models# ls
__init__.py  models.py
root@odoo:/var/lib/odoo/modules/empresasrac/models#
```

Figura 23: Models.py

Una vez dentro del archivo, deberemos descomentar las líneas y deberemos modificarlo como se muestra en la siguiente figura.

```
OdooServerRAC [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

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

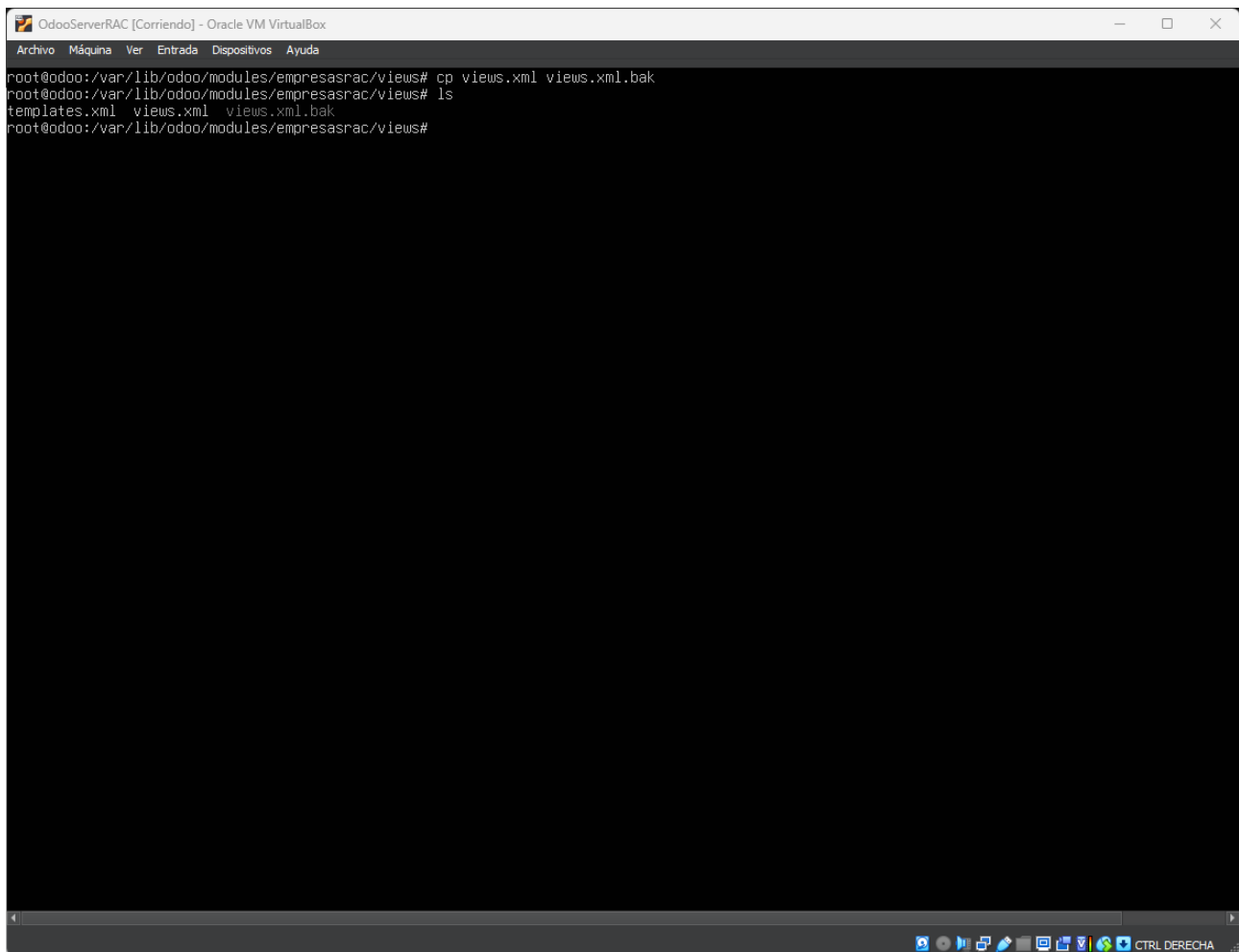
from odoo import models, fields, api

class empresasrac(models.Model):
    _name = 'empresasrac.empresasrac'
    _description = 'empresasrac.empresasrac'
    _inherit = 'res.partner'
    # 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 24: Modificación archivo `models.py`

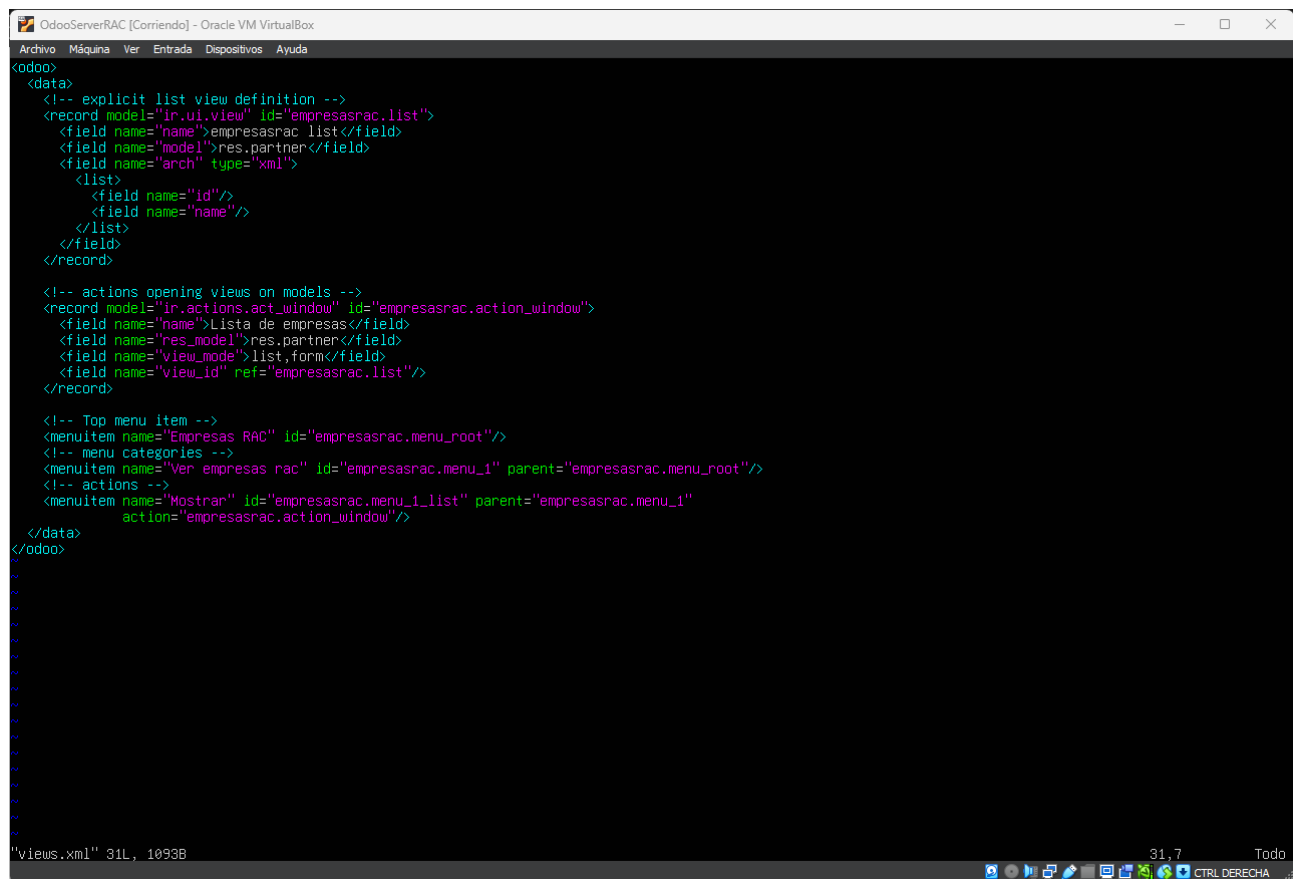
A continuación saldremos de la anterior carpeta e iremos a views. Dentro tendremos un archivo xml que también deberemos modificar.



```
OdooServerRAC [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@odoo:/var/lib/odoo/modules/empresasrac/views# cp views.xml views.xml.bak
root@odoo:/var/lib/odoo/modules/empresasrac/views# ls
templates.xml  views.xml  views.xml.bak
root@odoo:/var/lib/odoo/modules/empresasrac/views#
```

Figura 25: views.xml

Modificaremos el xml de tal manera que se vea como en la siguiente figura:



```
<?xml version="1.0"?>
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="empresasrac.list">
      <field name="name">empresasrac list</field>
      <field name="model">res.partner</field>
      <field name="arch" type="xml">
        <list>
          <field name="id"/>
          <field name="name"/>
        </list>
      </field>
    </record>

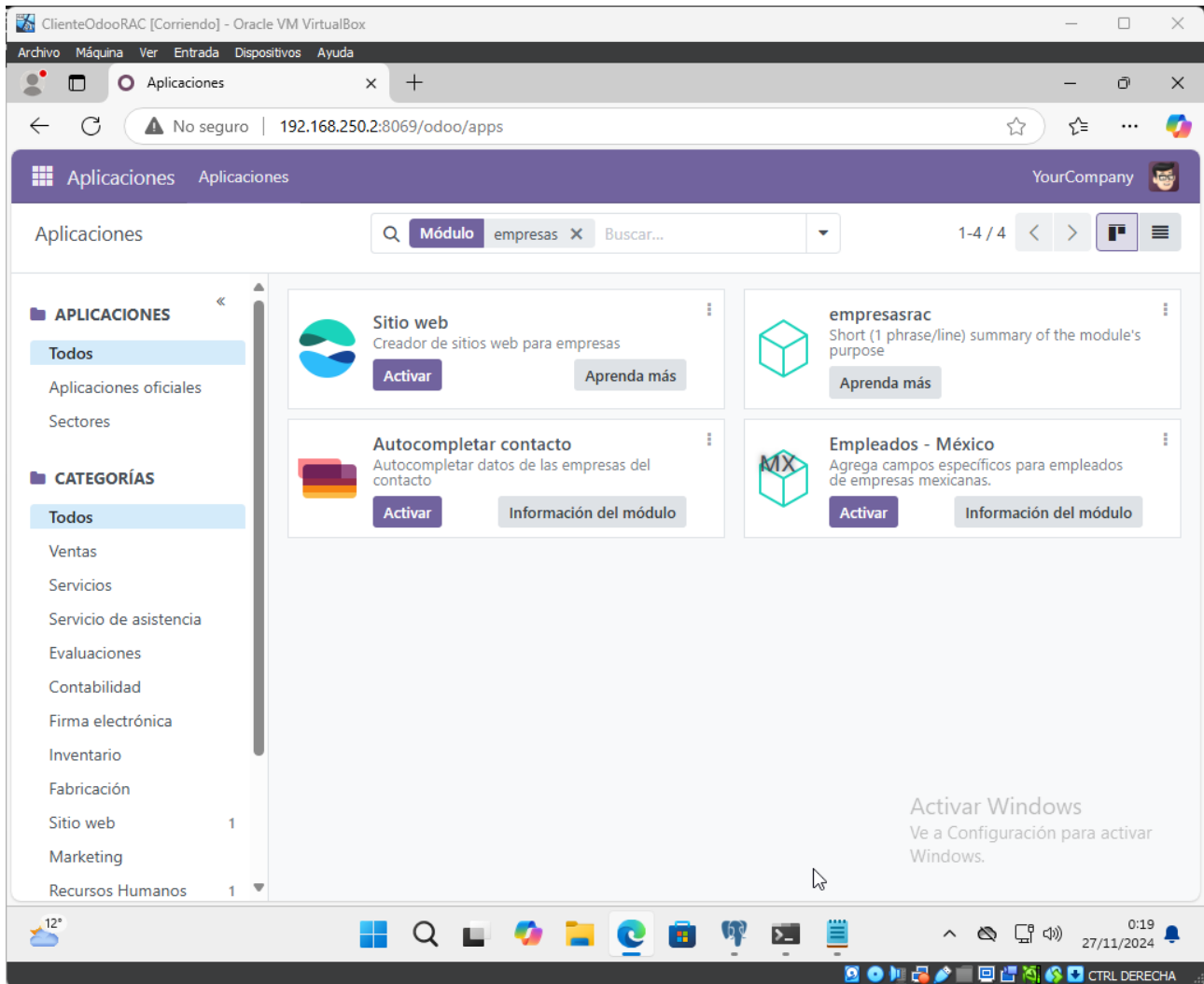
    <!-- actions opening views on models -->
    <record model="ir.actions.act_window" id="empresasrac.action_window">
      <field name="name">Lista de empresas</field>
      <field name="res_model">res.partner</field>
      <field name="view_mode">list,form</field>
      <field name="view_id" ref="empresasrac.list"/>
    </record>

    <!-- Top menu item -->
    <menuitem name="Empresas RAC" id="empresasrac.menu_root"/>
    <!-- menu categories -->
    <menuitem name="Ver empresas rac" id="empresasrac.menu_1" parent="empresasrac.menu_root"/>
    <!-- actions -->
    <menuitem name="Mostrar" id="empresasrac.menu_1_list" parent="empresasrac.menu_1"
      action="empresasrac.action_window"/>
  </data>
</odoo>
```

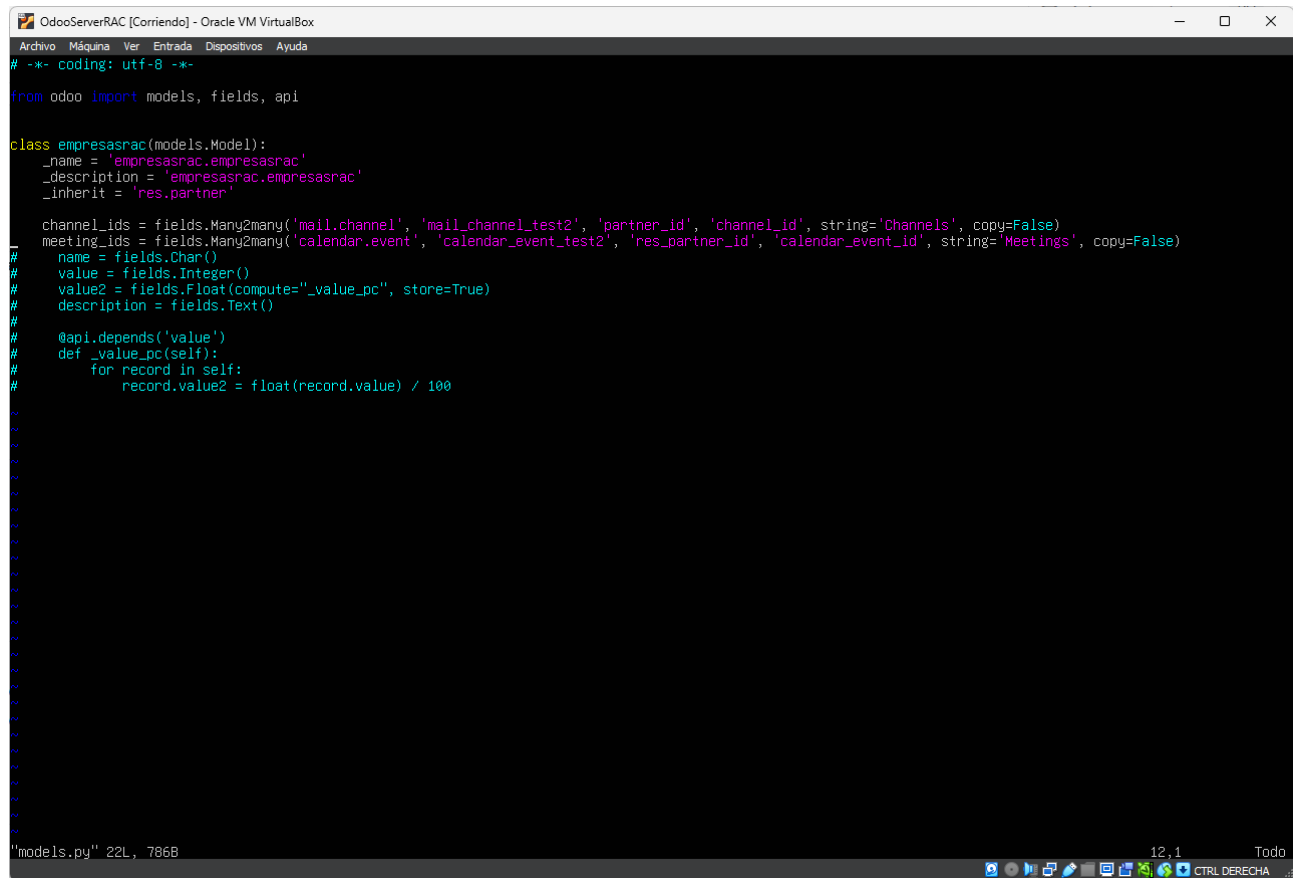
"views.xml" 31L, 1093B 31, 7 Todo CTRL DERECHA

Figura 26: Modificación del archivo views.xml

El siguiente paso será comprobar si funciona en la máquina cliente.



* Puede que nos de un error 500 y no nos deje acceder al servidor. Para solucionar esto, lo más probable es que tengamos que añadir las siguientes líneas adicionales en el archivo models.py



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

from odoo import models, fields, api

class empresasrac(models.Model):
    _name = 'empresasrac.empresasrac'
    _description = 'empresasrac.empresasrac'
    _inherit = 'res.partner'

    channel_ids = fields.Many2many('mail.channel', 'mail_channel_test2', 'partner_id', 'channel_id', string='Channels', copy=False)
    meeting_ids = fields.Many2many('calendar.event', 'calendar_event_test2', 'res_partner_id', 'calendar_event_id', string='Meetings', copy=False)
    # 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

"models.py" 22L, 786B
```

Figura 27: Error ids Many2many

Si la aplicación abre sin problemas, se mostrará de la siguiente manera:

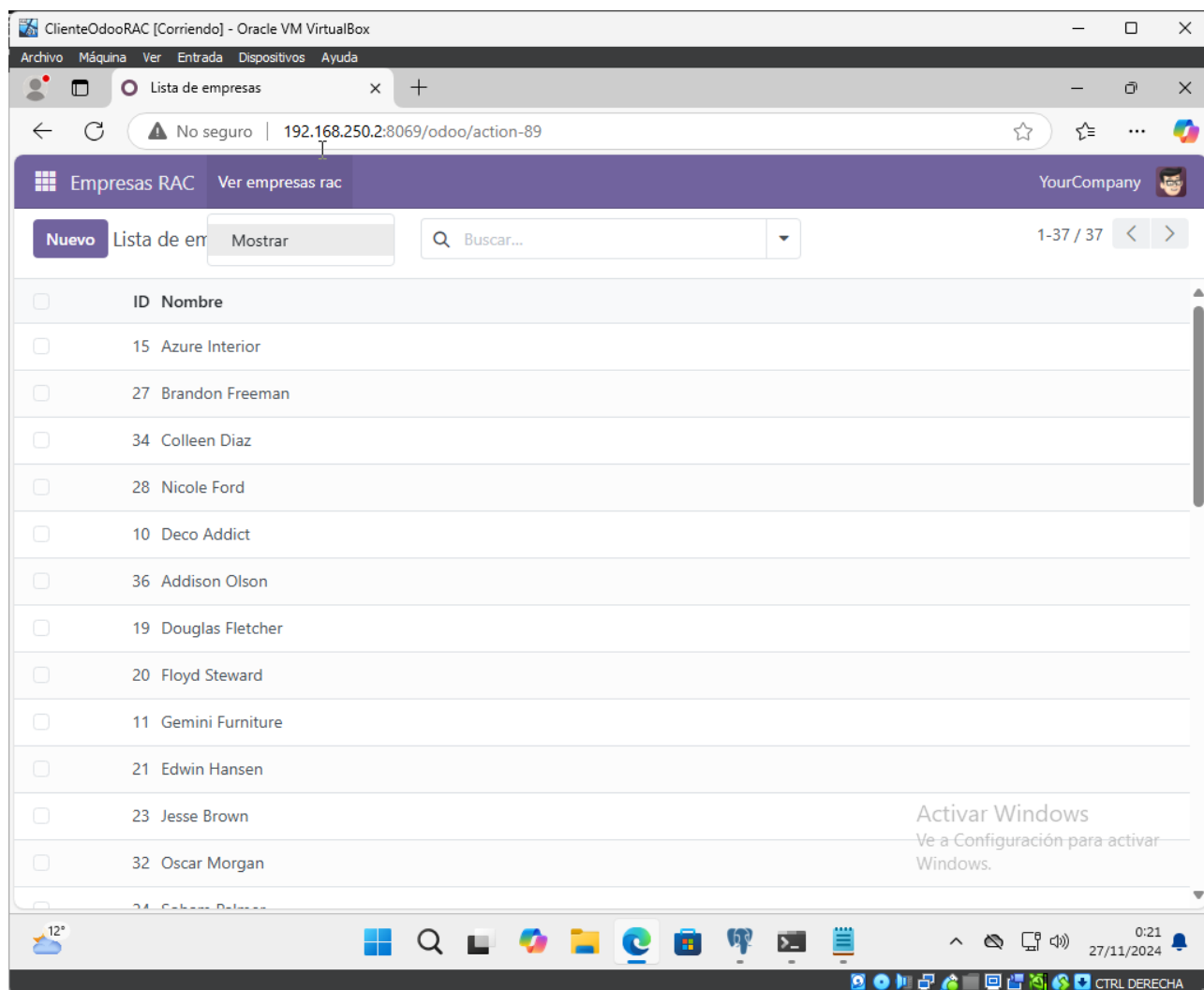


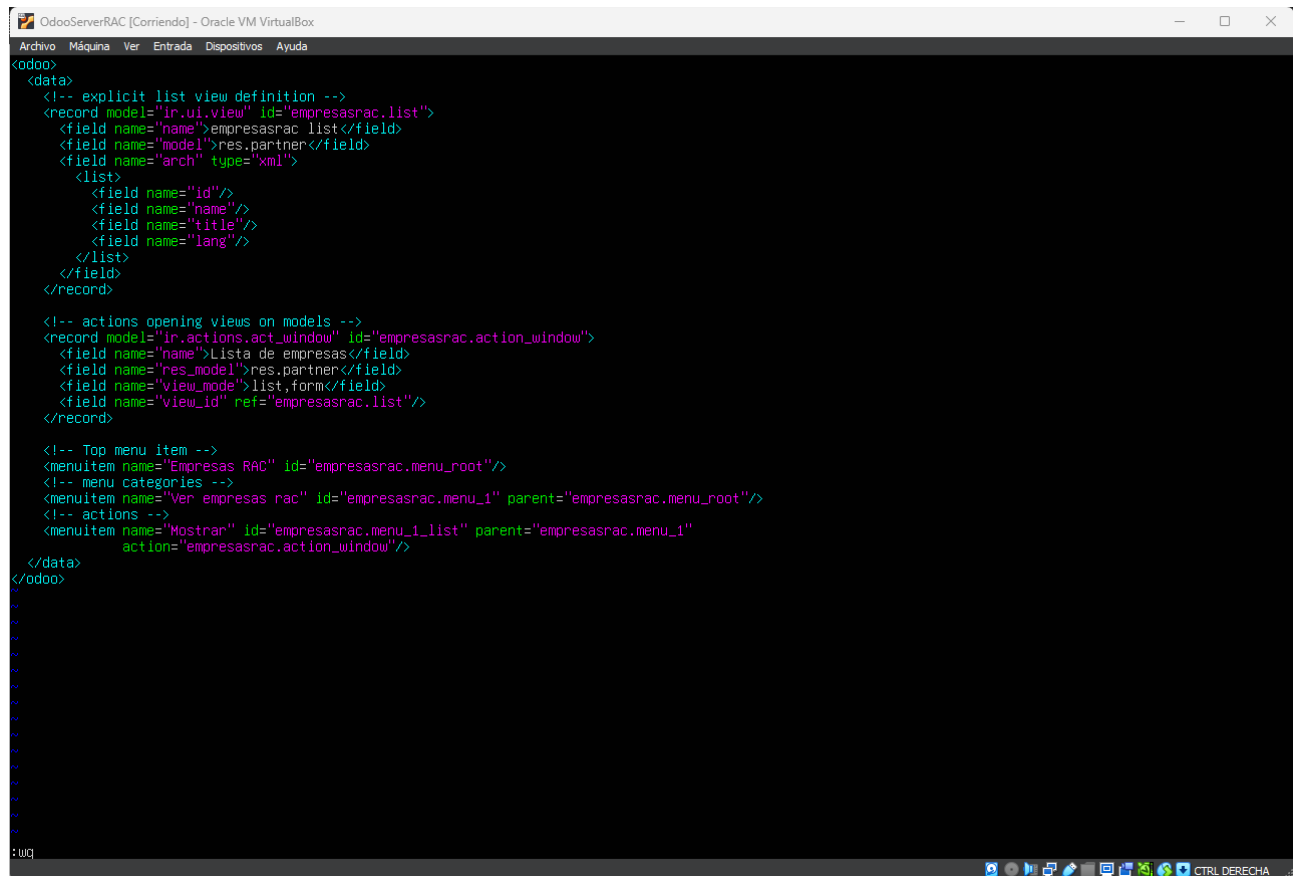
Figura 28: Módulo creado correctamente

Si no nos deja abrir la aplicación, probaremos a crear una nueva base de datos e intentarlo desde ahí.

Se nos pide que la vista muestre unos campos más, por lo que debemos volver al archivo views.xml y añadirle los siguientes campos:

```
<field name="title"/>
```

```
<field name="lang"/>
```



```
<odoo>
<data>
  <!-- explicit list view definition -->
  <record model="ir.ui.view" id="empresasrac.list">
    <field name="name">empresasrac.list</field>
    <field name="model">res.partner</field>
    <field name="arch" type="xml">
      <list>
        <field name="id"/>
        <field name="name"/>
        <field name="title"/>
        <field name="lang"/>
      </list>
    </field>
  </record>

  <!-- actions opening views on models -->
  <record model="ir.actions.act_window" id="empresasrac.action_window">
    <field name="name">Lista de empresas</field>
    <field name="res_model">res.partner</field>
    <field name="view_model">list.form</field>
    <field name="view_id" ref="empresasrac.list"/>
  </record>

  <!-- Top menu item -->
  <menuitem name="Empresas RAC" id="empresasrac.menu_root"/>
  <!-- menu categories -->
  <menuitem name="Ver empresas rac" id="empresasrac.menu_1" parent="empresasrac.menu_root"/>
  <!-- actions -->
  <menuitem name="Mostrar" id="empresasrac.menu_1_list" parent="empresasrac.menu_1"
    action="empresasrac.action_window"/>
</data>
</odoo>
```

Figura 29: Añadir campos nuevos a la vista

Ahora reiniciaremos Odoo.

Ahora si volvemos a abrir la aplicación, se nos mostrarán los nuevos campos

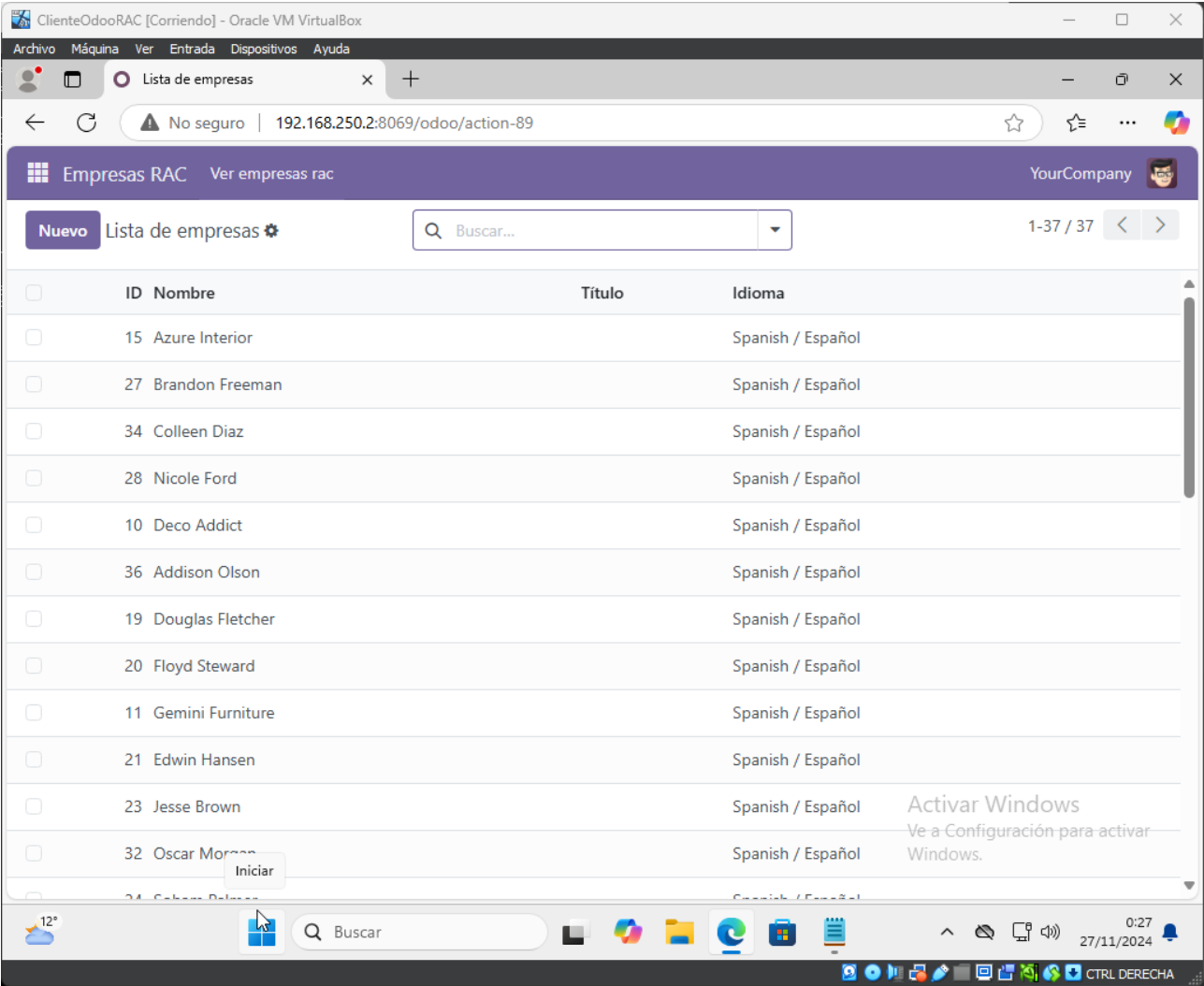


Figura 30: Vista actualizada

2.3 Adición de datos a la vista

Si pulsamos en el botón “Nuevo” de arriba a la izquierda, podremos añadir un nuevo usuario o una nueva compañía en la base de datos.

ClienteOdooRAC [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

RAC

No seguro | 192.168.250.2:8069/odoo/action-89/new

Empresas RAC Ver empresas rac YourCompany

Nuevo Lista de empresas RAC

☐ Individuo ☒ Compañía

RAC

Dirección Calle falsa 13 2D
Calle 2...
15202 Noia A Coruña (La Coruña)
España

NIF ? p. ej., ESA00000000

Teléfono 981828282

Móvil

Correo electrónico a24rubenac@iesantonlosada.gal

Sitio web www.rac.com

Etiquetas p. ej., "B2B", "VIP", "consultoría", ...

Contactos y direcciones Ventas y compras Notas internas

Agregar

Buscar

Activar Windows
Ve a Configuración para activar Windows.

12° Buscar 0:29 27/11/2024

Figura 31: Nueva compañía

Luego comprobaremos que existe utilizando la barra de búsqueda.

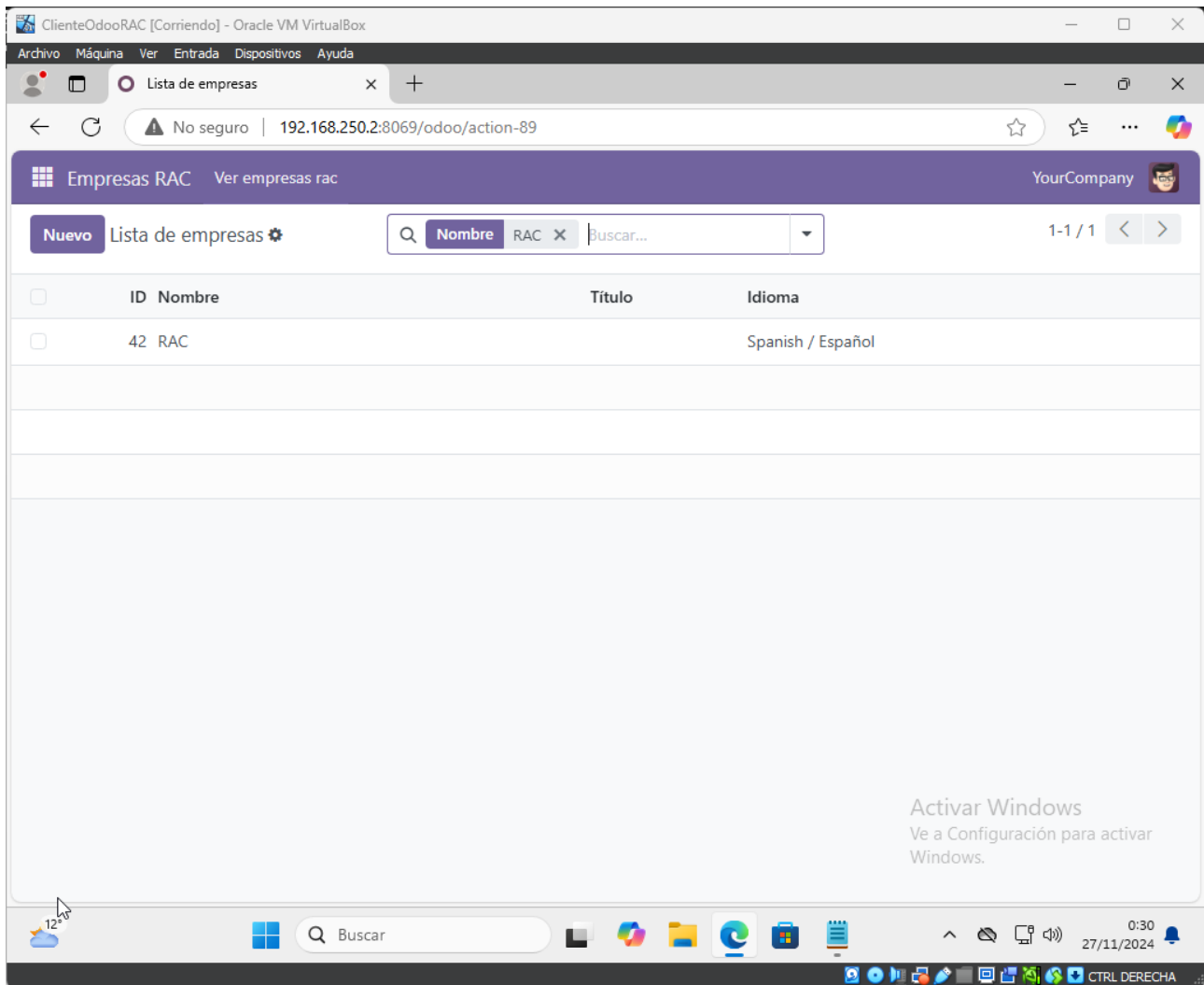


Figura 32: Nueva compañía creada correctamente