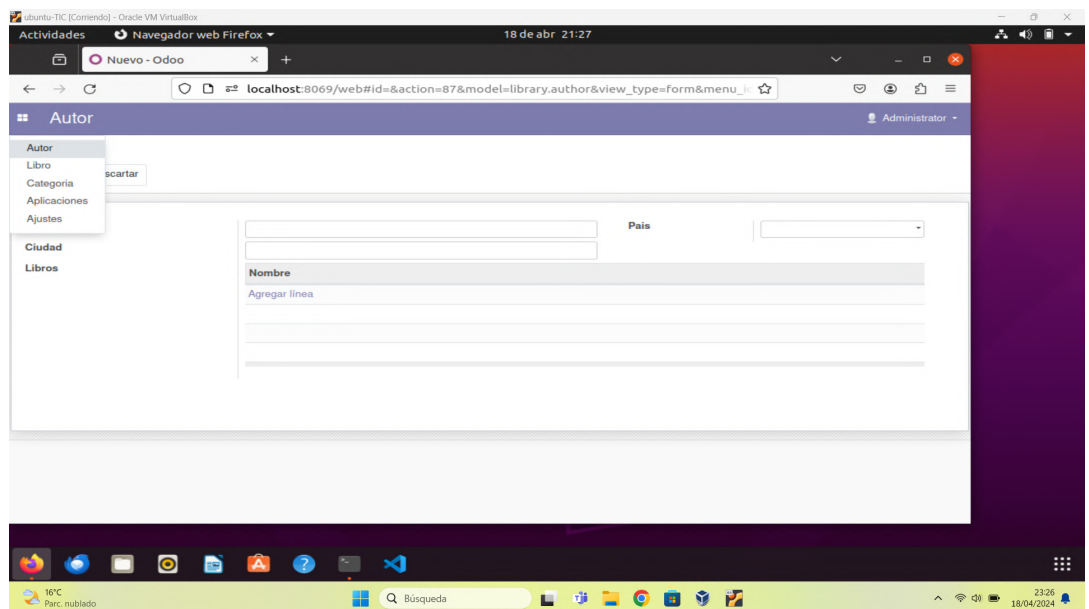
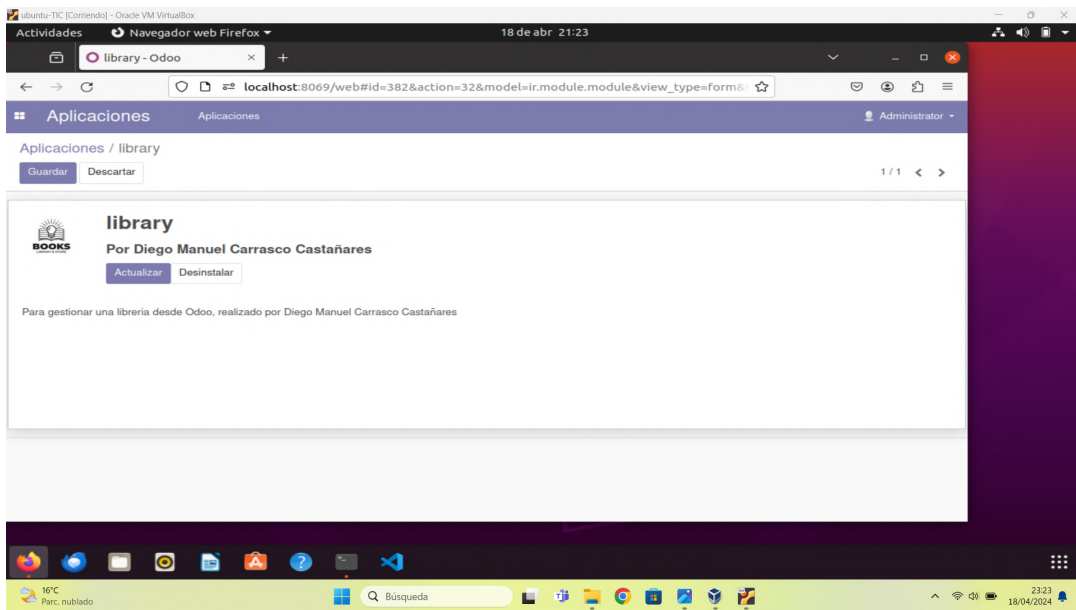


# Tarea 5 para Sistemas de Gestión Empresarial



Diego Manuel Carrasco Castañares

Módulo: Sistemas de Gestión Empresarial.

Unidad 5. Desarrollo de componentes.

Enunciado

En esta unidad hemos aprendido a crear nuevos componentes en Odoo, siguiendo el modelo MVC y apoyándonos en el lenguaje de programación Python. Se ha visto cómo crear componentes de manipulación de datos mediante módulos que crean, a su vez, tablas en la base de datos.

Para esta tarea será preciso conocer lo más básico de Python y, a continuación, se pedirá la creación de un módulo en el sistema:

1. Utilizando cualquiera de los IDE (Eclipse, Pycharm) o editor de código (Visual Studio Code), realiza los siguientes ejercicios en Python:

- a) Diseña un programa que, pidiendo por teclado la marca de un vehículo, el consumo de combustible (litros) a los 100km y los litros de combustible que le he puesto en el depósito, me calcule los kilómetros que puedo recorrer. La salida tendrá el siguiente formato

```
*****  
***** El vehículo marca que  
consume consumo litros a los 100km  
Me permite circular una distancia de kmrecorridos km con  
litros litros
```

.....

- b) Diseña un programa que lea dos números: el primero será la base y el segundo el exponente. Tendrá que mostrar el resultado de realizar dicha potencia usando un bucle iterativo.
- c) Diseña un programa que llene un vector de números enteros pedidos por teclado hasta que se introduzca el número 0; después mostrará un listado con los números pares introducidos y otro listado con los números

impares.

Para presentar los ejercicios anteriores en la tarea a entregar, genera para cada uno de ellos dos "capturas de pantalla", una del código y otra del resultado (terminal).

2. A partir de las explicaciones detalladas en los contenidos de la unidad, se pide:

- a. Crear un módulo denominado Biblioteca que permita gestionar un catálogo de libros.
- b. Desarrollar los modelos de datos en Python para las entidades Libro, Autor y Categoría, que más tarde serán mapeados en Odoo y convertidos en tablas SQL en PostgreSQL. Los atributos asociados a cada una son:

Libro:

Nombre  
Descripción  
Fecha de  
publicación Isbn  
Imagen  
Categoría a la que  
pertenece Autor

Categoria:

Nombre  
Código

Autor:

Nombre  
País  
Ciudad  
Libro\_i  
ds

- c. Crear las vistas para visualizar/manipular los registros de las entidades Libro, Categoría y Autor.

- d. Crear los menús de acceso en la interfaz de Odoo.
- e. Realizar la instalación del módulo en Odoo.
- f. Demostrar que todo funciona correctamente, dando de alta libros, asignándoles una categoría y un autor, mostrando los resultados en las vistas.

Para realizar este ejercicio se deben seguir las indicaciones que se muestran en los contenidos de la unidad. Ahí se definen los tipos de los campos y los distintos pasos a realizar para crear el módulo y permitir el alta y consulta de Categorías, Autores y Libros.

¡Importante!

Debes tener en cuenta que los contenidos del tema explican paso a paso gran parte de la creación del módulo sobre el sistema Ubuntu, pero puedes hacerlo en el sistema que prefieras.

El primer paso para crear el módulo, tanto en Windows como en Ubuntu, consiste en ejecutar el comando "scaffold" . Este comando se encarga de crear una plantilla por defecto adaptable a vuestras necesidades, y en caso de lanzarse en Windows debe ser parecido a:

```
"C:\Program Files (x86)\Odoo 12.0\python\python.exe" "C:\Program Files (x86)\Odoo 12.0\server\nodoo-bin" scaffold Biblioteca "C:\Program Files (x86)\Odoo 12.0\server\nodoo\addons".
```

Como material adicional, encontrarás un enlace a un vídeo explicativo de cómo se crea un módulo en Odoo. En él se explica cómo se realiza un módulo a partir de una "Plantilla" creada por nosotros mismos. Es un ejemplo sencillo, con un solo modelo de datos (en el ejercicio se piden 3) pero que os puede servir como guía. Podéis seguir las indicaciones para realizar vuestro módulo. Igualmente, en los contenidos de la unidad, para la edición de los ficheros Python se trabaja con Pycharm. No es requisito imprescindible que lo utilicéis. De hecho, es recomendable emplear Visual Studio Code, cuyo enlace lo tenéis en los recursos. También tenéis un documento guía sobre la creación de un módulo.

Nota: ¡mucho cuidado con la edición de los ficheros!. A Python "no le gustan nada" los caracteres extraños, espacios en blanco y saltos de línea de más que puedan incluir editores de texto como Notepad, Wordpad... Criterios de

puntuación. Total 10 puntos.

Apartado 1: 3 puntos, donde cada ejercicio se valora con 1 punto

Se valorará la correcta implementación y ejecución de cada programa.

Apartado 2: 7 puntos.

Se valorará la correcta ejecución de los distintos puntos del ejercicio:

- Creación del módulo. 1 punto.
- Configuración de los ficheros y creación de cada elemento del MVC. 2 puntos.
- Descripción de los posibles errores y sus correcciones. 1 punto.
- Obtención del resultado y comportamiento esperado, mostrando dentro del sistema Odoo el módulo creado y su correcto funcionamiento para la gestión del catálogo de libros (incluyendo categorías y autores). 3 puntos.

Consejos y recomendaciones.

Para realizar los ejercicios puedes consultar el material de apoyo de la unidad.

Indicaciones de entrega.

Una vez realizada la tarea elaborarás un único documento donde figuren las imágenes y explicaciones correspondientes. El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1\_apellido2\_nombre\_SIG05\_Tarea.pdf

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así, por ejemplo, la alumna Begoña Sánchez Mañas para la quinta unidad del módulo SGE debería nombrar esta tarea como...

sanchez\_manas\_begona\_SGE05\_Tarea.

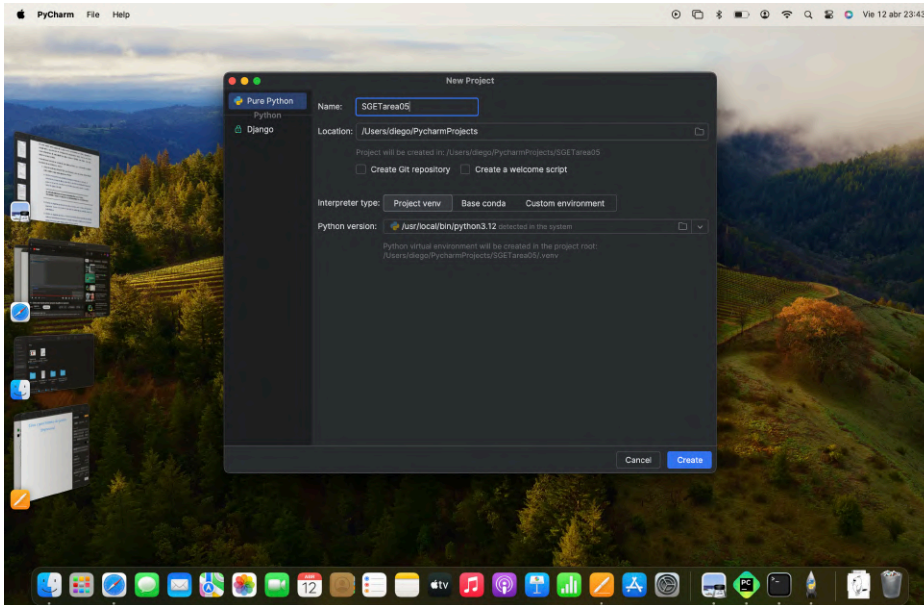
**Indice:**

Ejercicio 1.....	07
Ejercicio 1A (Distancia a recorrer).....	08
Ejercicio 1B (potencia).....	09
Ejercicio 1C (separar pares de impares).....	09
Main.....	10
Ejercicio 2.....	12
Crear carpeta para el módulo y esqueleto.....	13
Crear clases carpeta models.....	14
Modificar init.py.....	15
Modificar ir.model.access.csv.....	17
Modificar manifest.py.....	17
Crear vistas.....	18
Instalación del módulo.....	20
Vistas abiertas en odoo.....	21

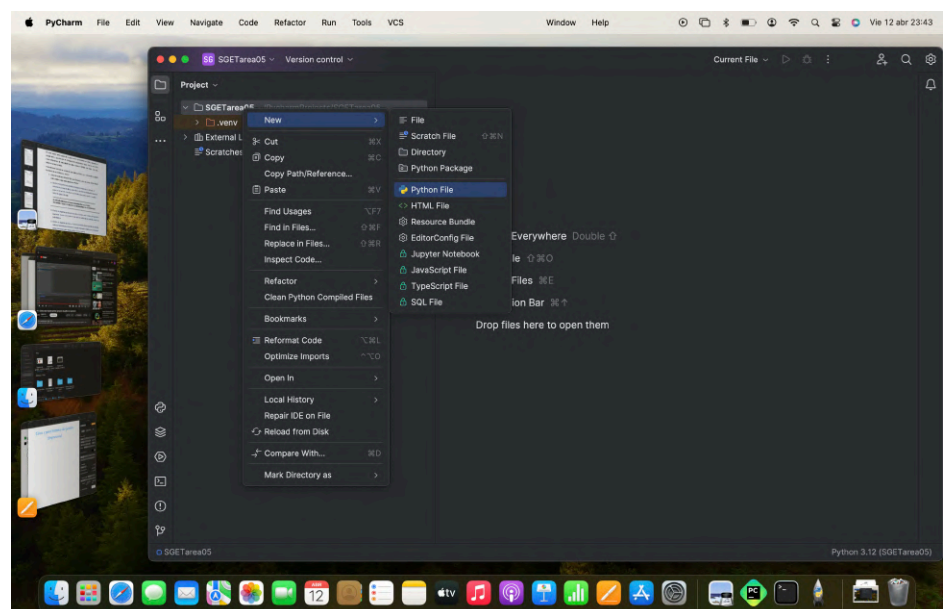
Para el primer ejercicio he escogido hacerlo con Pycharm.

Lo primero que haremos será descargarlo de la página de JetBrains e instalarlo en nuestro equipo.

Tras ello crearemos un nuevo proyecto llamado SGETarea05.

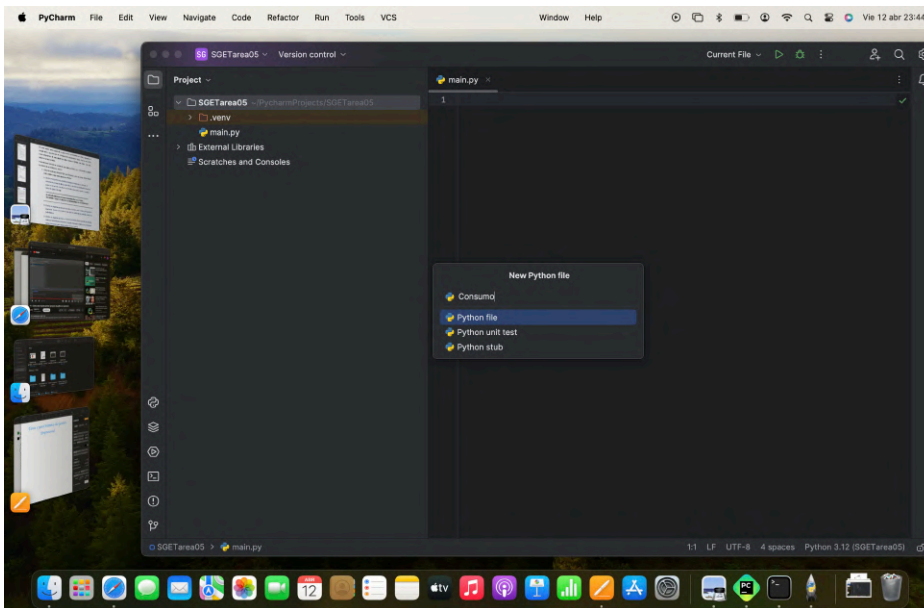


En dicho proyecto crearemos cuatro clases nueva (botón derecho → new → Python file)



Las nombraremos Main, Consumo, Potencia y Vector\_Pares\_Impares





La clase consumo alojara una función llamada `calcular_km_disponibles`, a la cual le pasaremos por parámetros el consumo medio cada 100 km los litros añadidos al depósito.

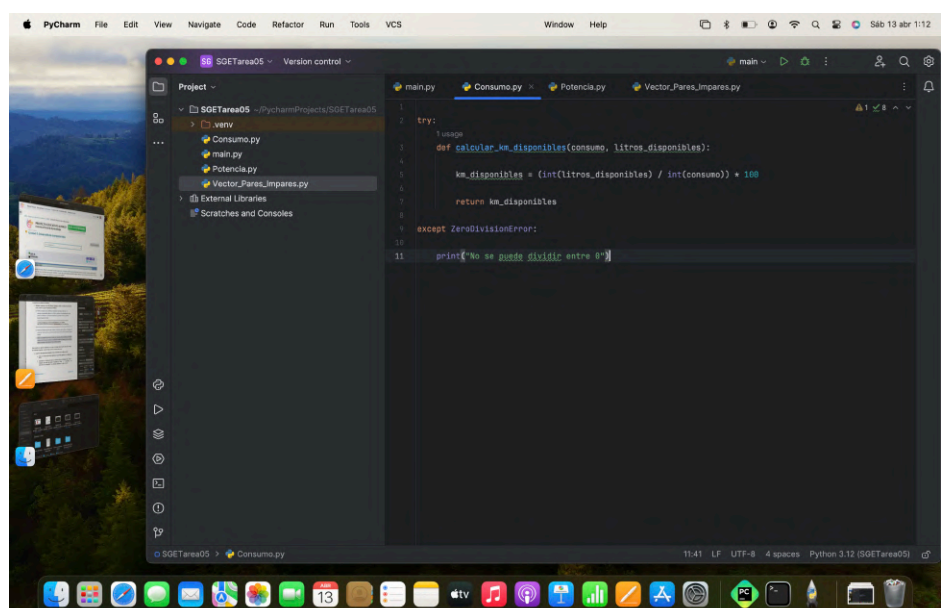
En dicho método declararemos una variable llamada `km_disponibles` al cual será igual a los litros de combustible partido del consumo y el resultado de ello multiplicaremos por 100.

Al ser los datos obtenidos por teclado de tipo string tenemos que hacer cast a int de los datos pasados por parámetro.

Esta función retornara la variable `km_disponibles`.

Todo esto lo envolveremos en try - except para controlar el `ZeroDivisionError`, en cuyo caso lanzara el mensaje correspondiente.

Podemos ver el código de esta clase en la siguiente imagen.



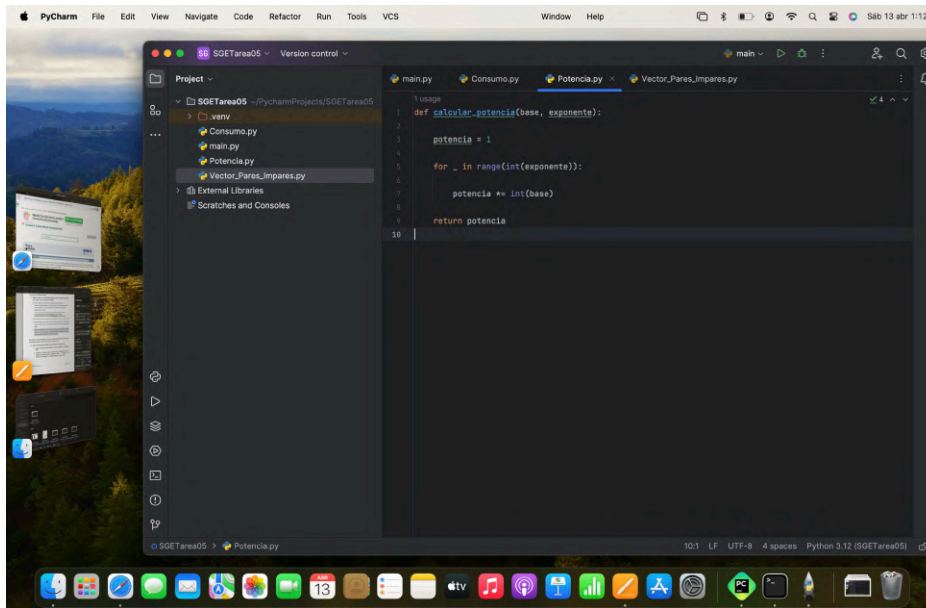


La clase Potencia contendrá un método llamado calcular potencia al cual le pasaremos por parámetros la base y el exponente.

Declaramos e instanciamos la potencia a 1.

Mediante un bucle for iteramos sobre la variable exponente.

En cada interacción multiplicamos dicha variable por la variable base y lo guardamos en la variable potencia, retornando posteriormente la variable potencia.

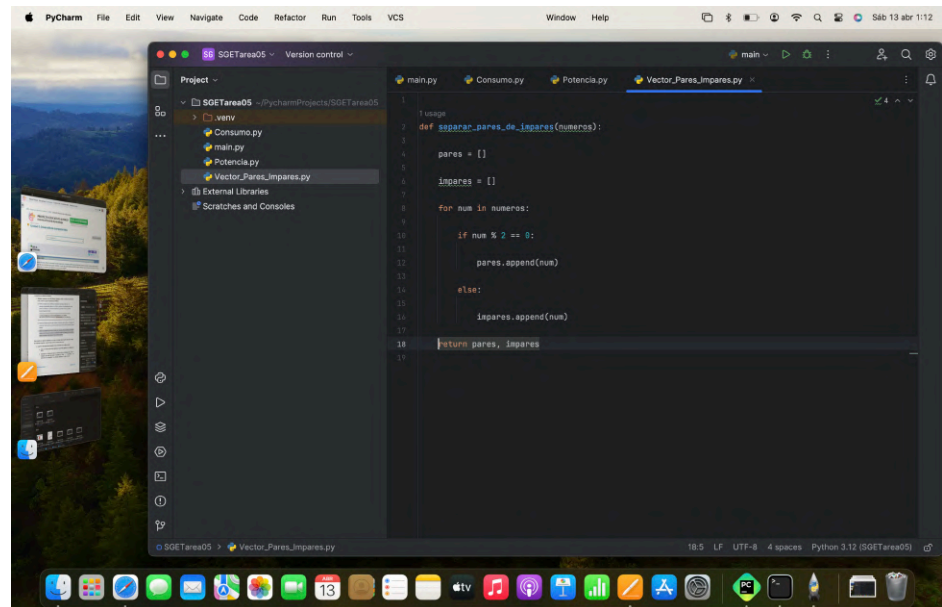


La clase separar\_pares\_de\_impares le pasaremos por parámetro un array de números llamado números.

Declararemos dos array dentro, uno llamado pares y otro impares.

Mediante un bucle for recorreremos el array números y evaluamos si el numero que estamos iterando es divisible entre 2 sin que quede resto, de ser así, el número es par y lo guardamos en el array pares, de lo contrario lo guardamos en el array impares.

Posteriormente retornamos los array pares e impares.



En la clase Main vamos a interactuar con el usuario.

Para el ejercicio de calcular la distancia a recorrer con el carburante repostado le pediremos al usuario la marca del vehículo, el consumo y los litros, que guardaremos en variables para tal fin.

Tras ello llamaremos al método `calcular_km_disponibles` y le pasaremos por parámetro las variables `consumo` y `litros_disponibles` y guardaremos en la variable `km_disponibles` el valor retornado por dicho método.

Por último, mostraremos, mediante un array concatenado con las variables el mensaje indicado en la tarea.

Para el ejercicio calcular potencia pediremos al usuario la base y el exponente y lo guardaremos en variable declaradas para tal fin.

Tras ello llamaremos al método `calcular_potencia`, al cual le pasaremos las variables `base` y `exponente` y guardaremos en la variable `potencia` el valor retornado por dicho método.

Y para terminar mostraremos mediante un string concatenado con las variables pertinentes el mensaje correspondiente.

Podemos ver el código de ambos ejercicios en la siguiente imagen.

```

1 import Consumo
2 import Potencia
3 import Vector_Pares_Impares
4
5 print("Calcular distancia a recorrer.")
6
7 marca = input("Que marca de vehiculo tienes: ")
8
9 consumo = input("Cuantos litros consume cada 100 km: ")
10
11 litros_disponibles = input("Cuantos litros de combustible le has puesto: ")
12
13 km_disponibles = Consumo.calcular_km_disponibles(consumo, litros_disponibles)
14
15 print("El vehiculo de la marca " + marca + " tiene un consumo de " + str(consumo) +
16       " litros cada 100 km y me permite circular una distancia de " +
17       str(km_disponibles) + " kilometros con " + str(litros_disponibles) + " litros")
18
19 print("-----")
20
21 print("Calcular potencia de dos numeros")
22
23 base = input("Introduce la base: ")
24
25 exponente = input("Introduce el exponente: ")
26
27 potencia = Potencia.calcular_potencia(base, exponente)
28
29 print(base + " elevado a " + exponente + " da un resultado de " + str(potencia))
30
31 print("-----")
32
33 print("Separar pares de impares")

```

Para el ejercicio de separar pares de impares declararemos un array llamado números.

Mediante un while con la condición true iremos pidiendo numero al usuario y los iremos almacenando en dicho array, mediante el método append, hasta que introduzca el número 0.

Comprobaremos cada número que introduce el usuario mediante un if, que, en caso de ser 0 hará un break y saldremos del while.

Posteriormente llamaremos al método vector\_pares\_impares, al cual le pasaremos el array en el que hemos almacenado los números introducidos por el usuario y guardamos los arrays devueltos en arrays creados para tal fin.

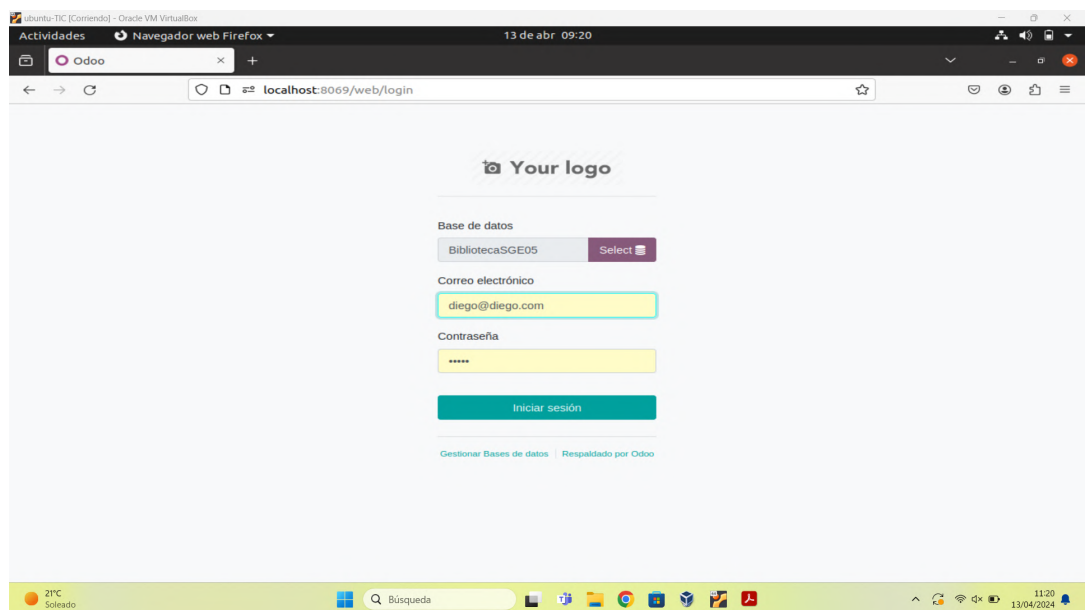
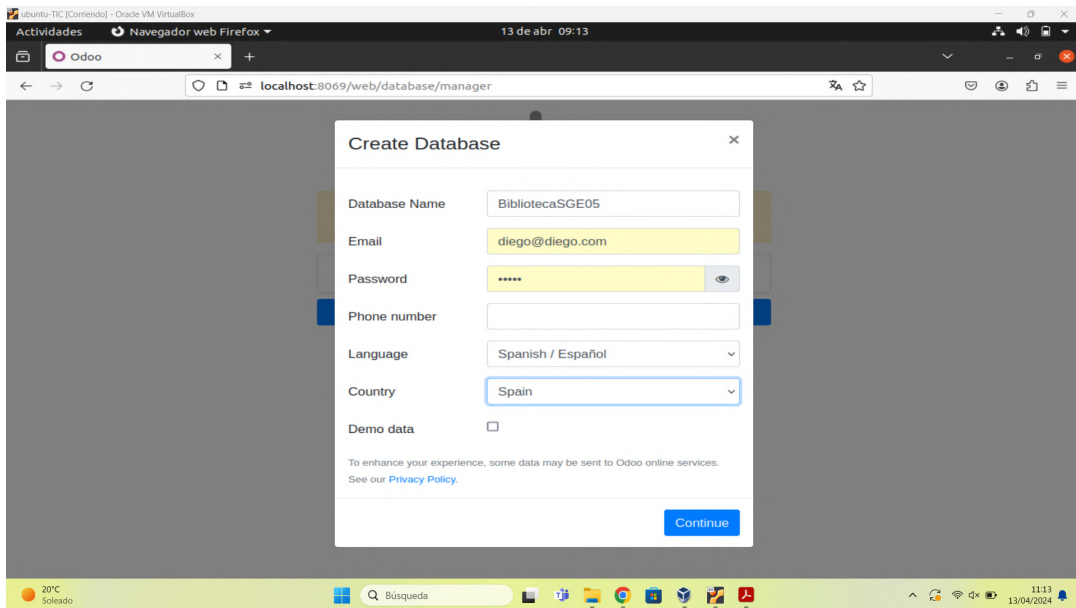
Por último, mediante dos bucles for, recorreremos los arrays y mostraremos el contenido de los mismos.

```

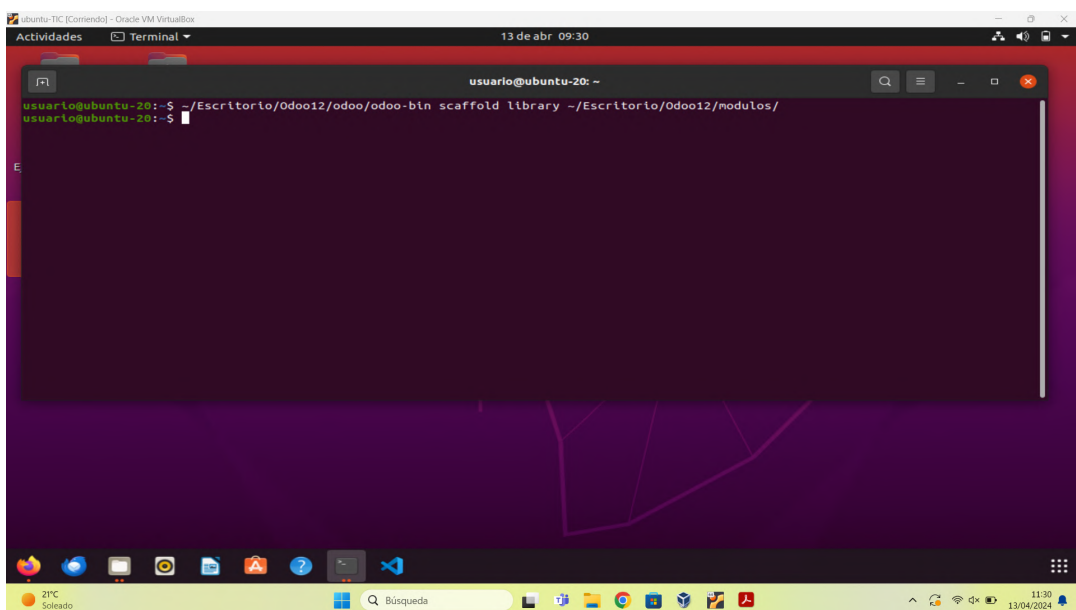
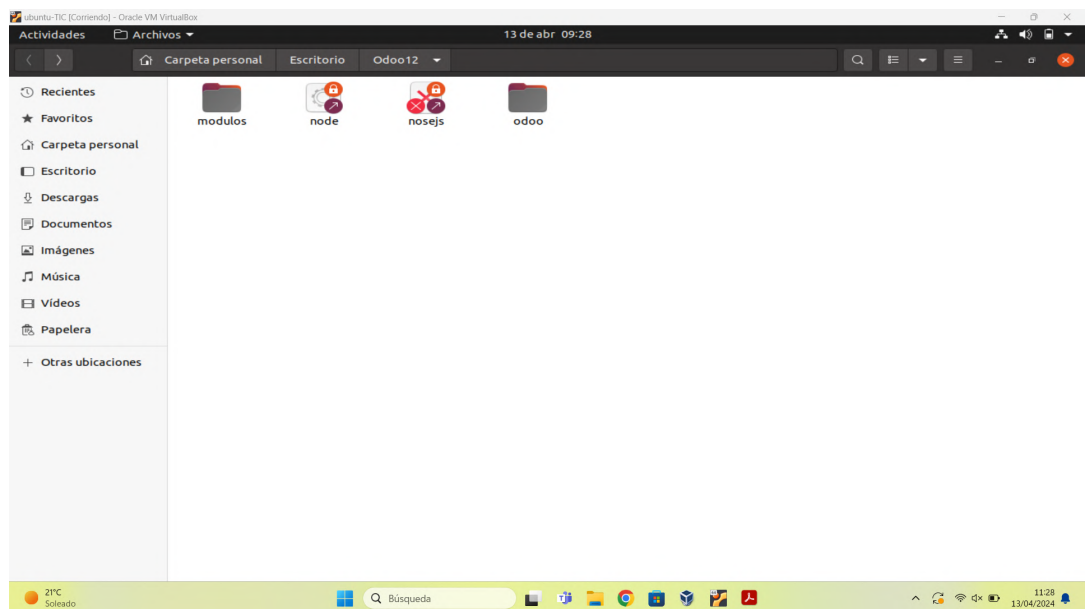
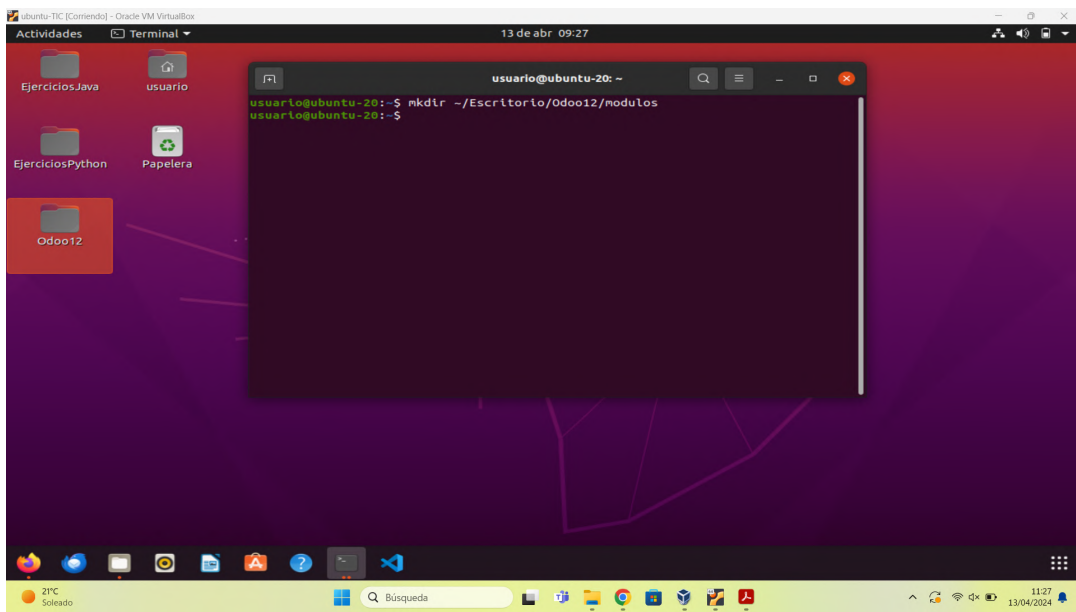
30
31 print("-----")
32
33 print("Separar pares de impares")
34
35 numeros = []
36
37 while True:
38
39     num = int(input("Introduce un numero entero. El 0 termina el programa "))
40
41     if num == 0:
42         break
43
44     numeros.append(num)
45
46 pares, impares = Vector_Pares_Impares.separar_pares_de_impares(numeros)
47
48 print("Los numeros pares son: ")
49
50 for par in pares:
51     print(par)
52
53 print("Los numeros impares son: ")
54
55 for imp in impares:
56     print(imp)

```

Para la segunda parte de la tarea crearé una base de datos llamada BibliotecaSGE05 desde la pagina principal de odoo y nos conectaremos a ella.



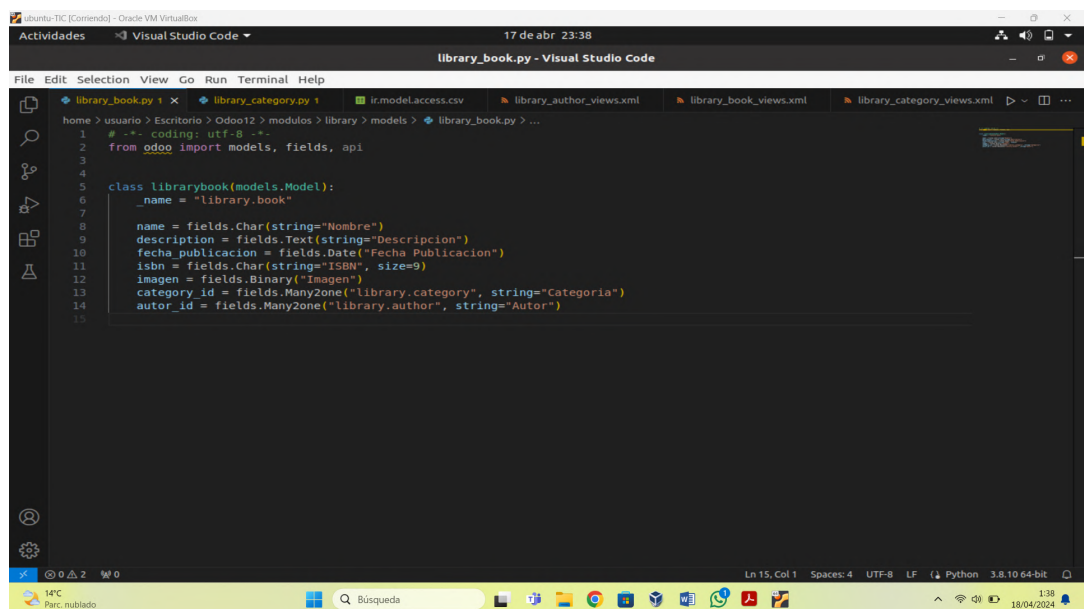
A continuación usaremos dos comando del temario, uno (mkdir) para crear el directorio módulos dentro de la carpeta de Odoo12 y otro (scaffold library) para crear la estructura vacía o esqueleto del módulo.



El siguiente paso será crear tres clases Python, dentro de la carpeta models, lo cual haremos con Visual Studio Code, programa con el cual también editaremos los archivos que necesitemos de dicho modulo.

Tras ello vamos a componer la clase librarybook, en la cual importaremos models, fields y api.

A dicha clase la llamaremos library.book y ella contendrá el nombre del libro, que será de tipo Char, descripción, de tipo text, fecha\_publicacion, de tipo Date, isbn, de tipo Char, imagen de tipo Binary, category\_id que será del tipo many2one, que hará referencia al campo de igual nombre de la clase library.category y autor\_id, del tipo many2one, que hará referencia al campo de igual nombre de la clase library.author.

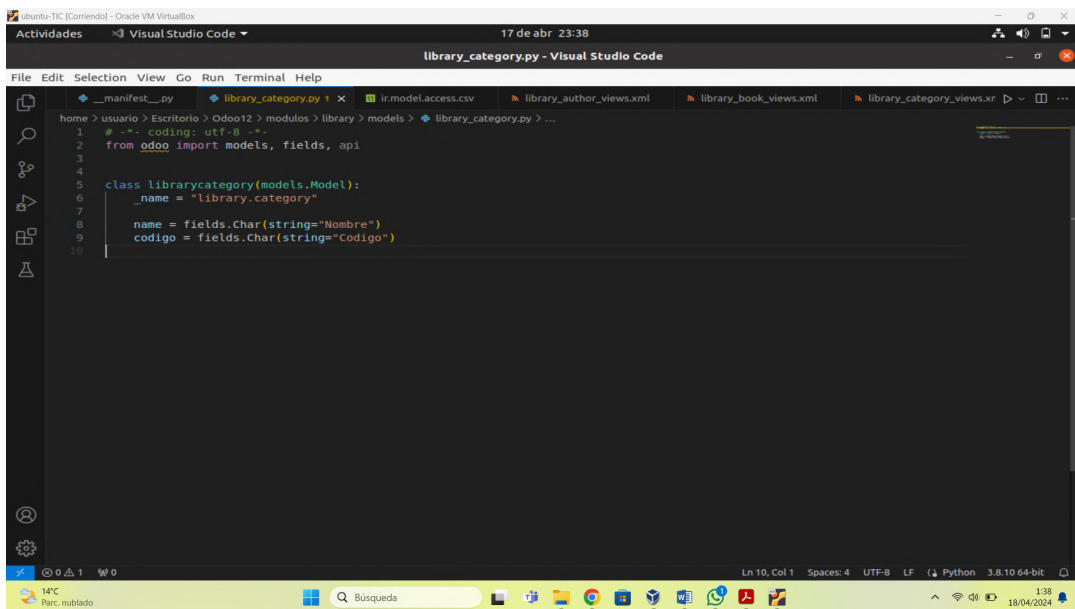


```
1  # -*- coding: utf-8 -*-
2  from odoo import models, fields, api
3
4
5  class librarybook(models.Model):
6      _name = "library.book"
7
8      name = fields.Char(string="Nombre")
9      description = fields.Text(string="Descripcion")
10     fecha_publicacion = fields.Date(string="Fecha Publicacion")
11     isbn = fields.Char(string="ISBN", size=9)
12     imagen = fields.Binary(string="Imagen")
13     category_id = fields.Many2one("library.category", string="Categoria")
14     autor_id = fields.Many2one("library.author", string="Autor")
15
```

En la clase librarycategory importaremos, al igual que en la clase anterior, importaremos models, fields y api.

Esta clase la llamaremos library.category y contendrá los atributos nombre, de tipo char y código de tipo char.





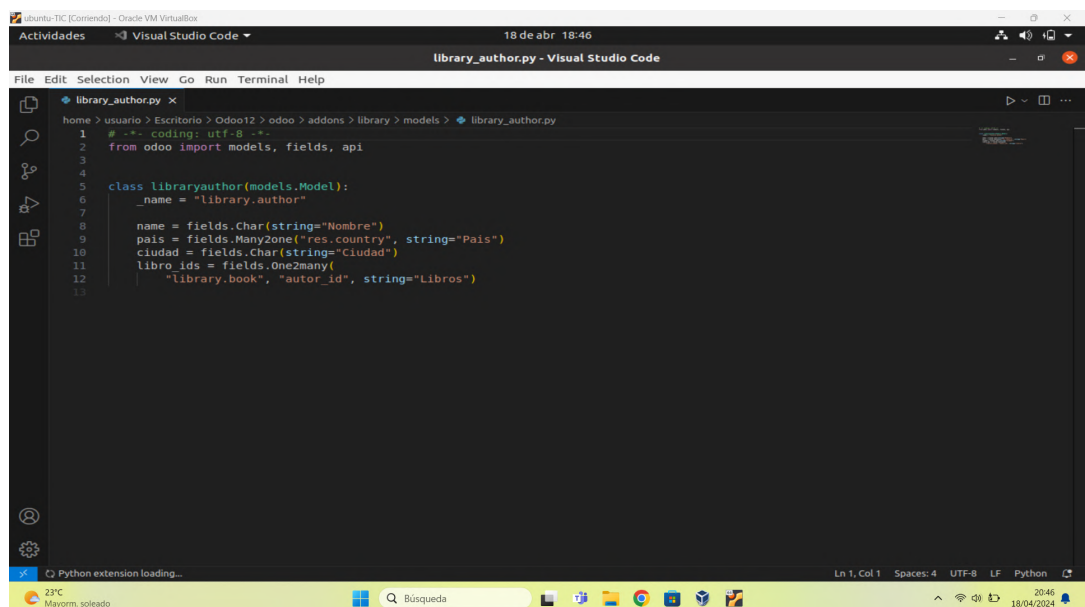
```

1  # -*- coding: utf-8 -*-
2  from odoo import models, fields, api
3
4
5  class librarycategory(models.Model):
6      _name = "library.category"
7
8      name = fields.Char(string="Nombre")
9      codigo = fields.Char(string="Codigo")
10

```

En la clase libraryauthor importamos models, fields y api, al igual que en las clases anteriores.

Esta clase le daremos de nombre library.author y contendrá un atributo name, de tipo char, uno de tipo país de tipo many2one que hará referencia a la tabla res\_country, otro llamado ciudad de tipo char y otro libro\_ids, del tipo one2many que hará referencia a la clase library.autor.



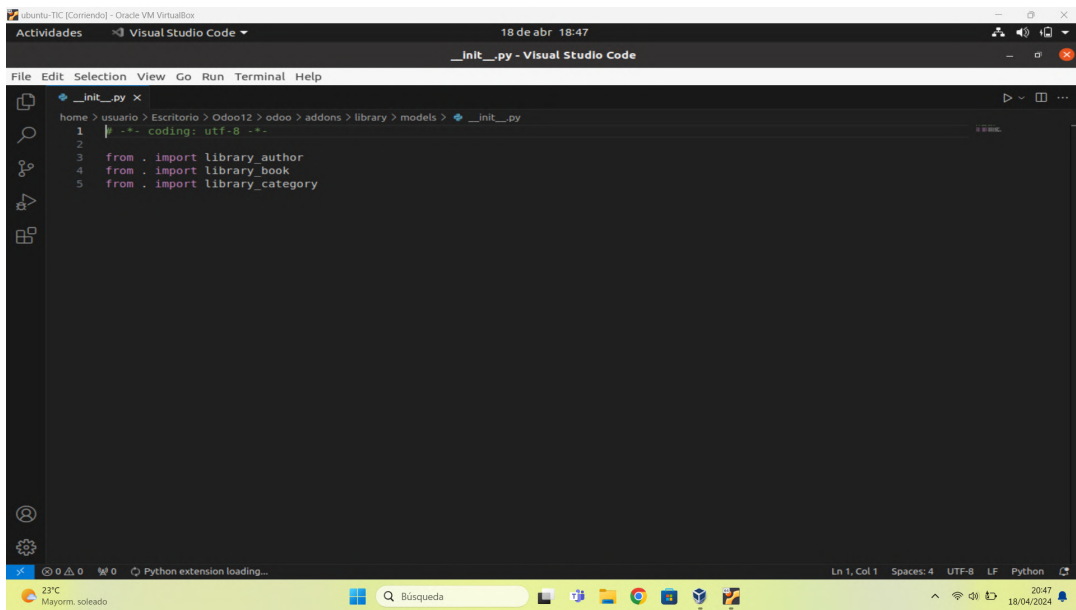
```

1  # -*- coding: utf-8 -*-
2  from odoo import models, fields, api
3
4
5  class libraryauthor(models.Model):
6      _name = "library.author"
7
8      name = fields.Char(string="Nombre")
9      pais = fields.Many2one("res.country", string="País")
10     ciudad = fields.Char(string="Ciudad")
11     libro_ids = fields.One2many(
12         "library.book", "autor_id", string="Libros")
13

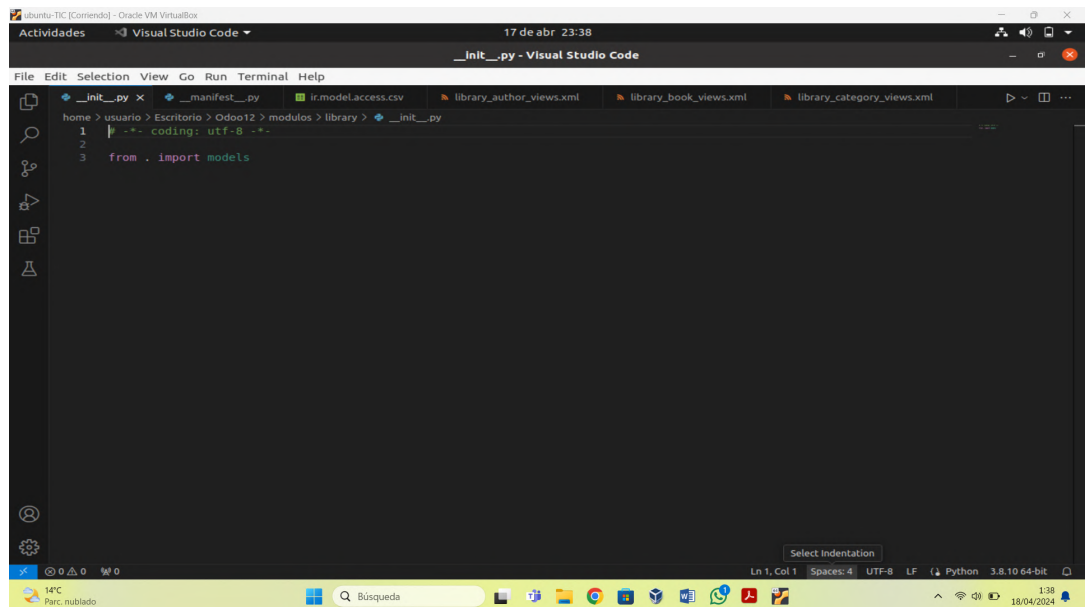
```

Posteriormente abriremos el archivo init.py alojado en models e importaremos las tres clases creadas en el.

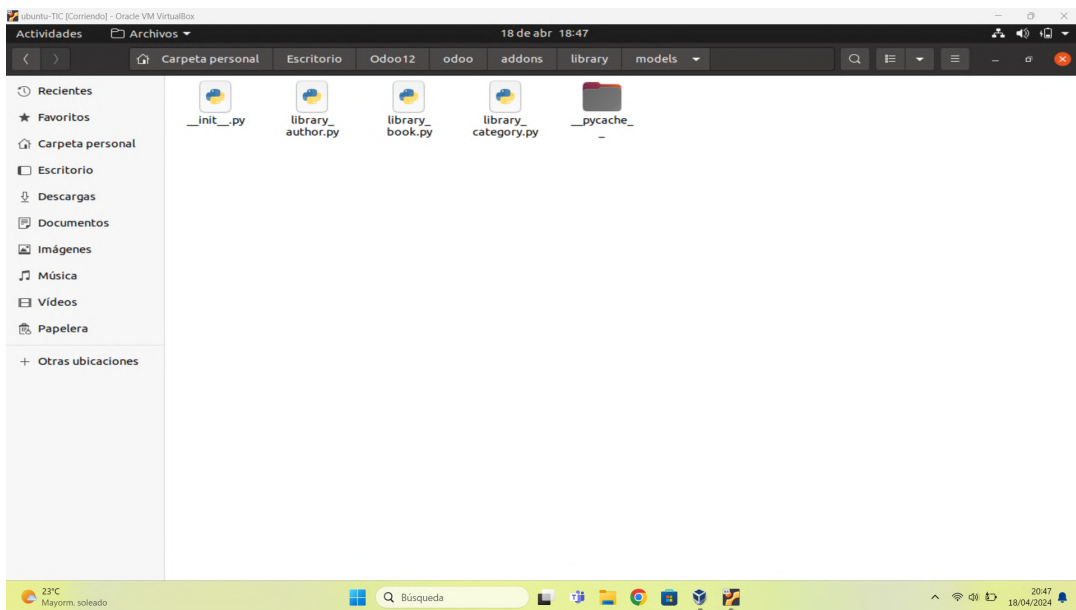




A continuación abriremos el archivo `init.py` alojado en `library`, en el cual solo vamos a importar `models`.



La carpeta `models` quedaría como se muestra en la siguiente imagen.

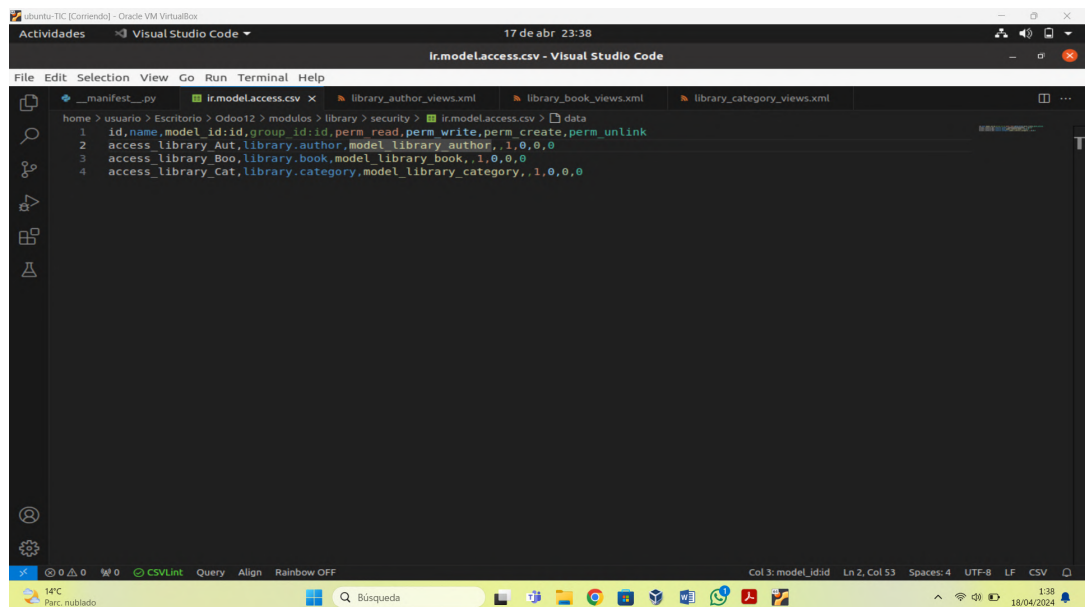


Tras ello abriremos el archivo `ir.model.access.csv`.

En dicho archivo vamos a dar acceso a las tres clases que tenemos creadas.

Para ello le daremos un id, haremos referencia al nombre del modelo que le pusimos al construirlas, le daremos un id, dejaremos en blanco la parte de si pertenecen a un grupo y le daremos todos los permisos.

Puede verse el código en la imagen siguiente.



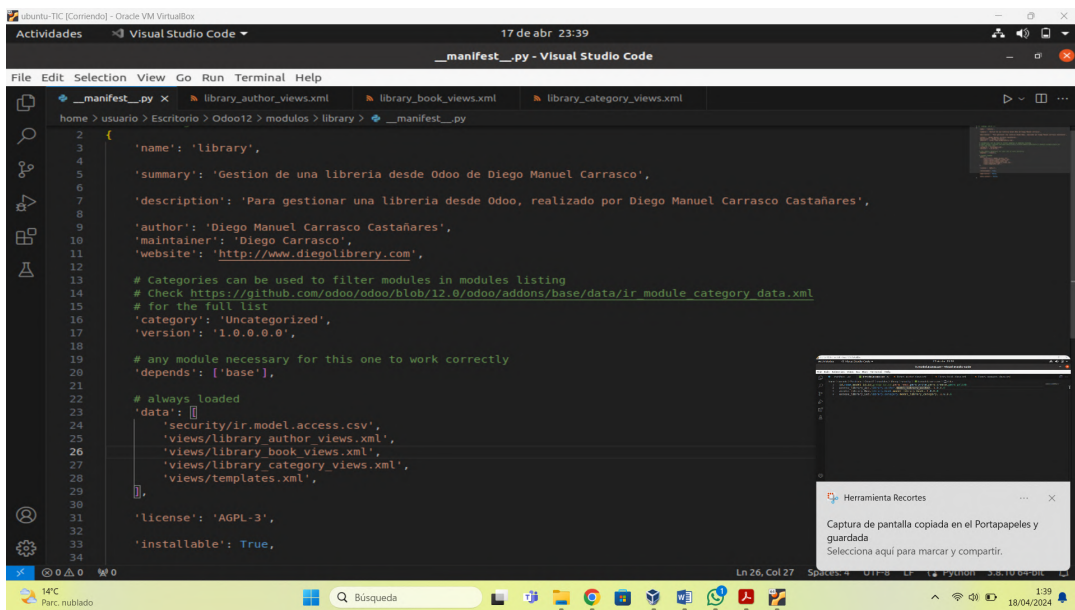
En el archivo `manifest.py` le daremos el nombre que le hemos puesto al modulo, `library` en mi caso, pondremos un resumen de la descripción, una descripción completa, el autor, quien mantiene el

módulo, al web del creador, la categoría, la versión y las dependencias, en este caso dependerá del modulo base.

En data de dicho archivo manifest haremos referencia al archivo ir.model.access.csv y a las tres vistas que posteriormente crearemos.

En license le diremos que es del tipo AGPL-3 y en instalable le daremos true para que nos permita instalar el módulo en odoo.

Podemos ver el contenido de dicho archivo en la siguiente imagen.



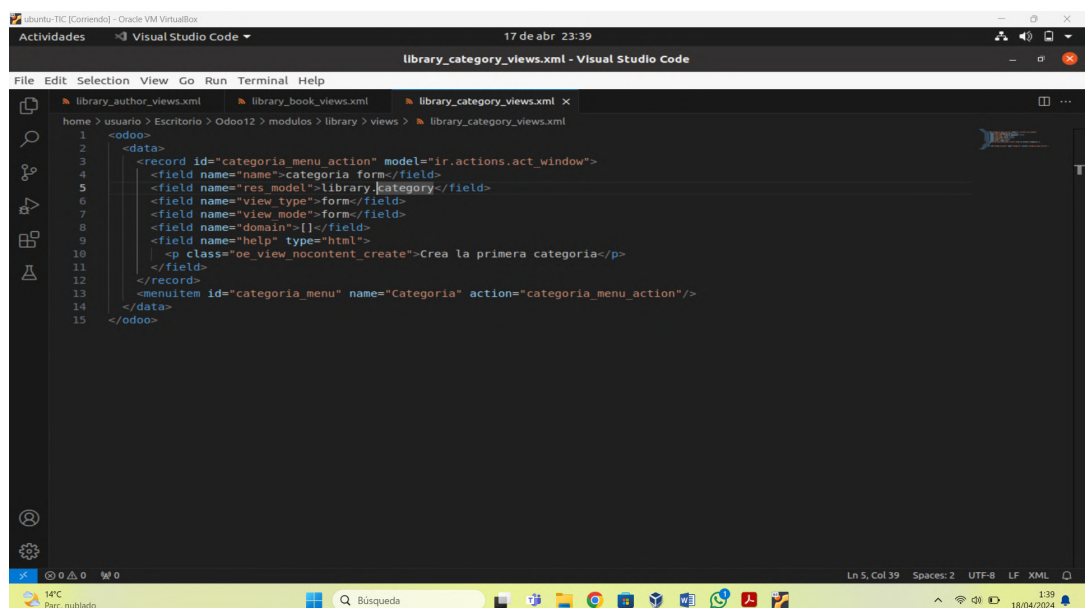
```
1 {
2     'name': 'library',
3
4     'summary': 'Gestion de una libreria desde Odoo de Diego Manuel Carrasco',
5
6     'description': 'Para gestionar una libreria desde Odoo, realizado por Diego Manuel Carrasco Castañares',
7
8     'author': 'Diego Manuel Carrasco Castañares',
9     'maintainer': 'Diego Carrasco',
10    'website': 'http://www.diegolibrery.com',
11
12    # Categories can be used to filter modules in modules listing
13    # Check https://github.com/odoo/odoo/blob/12.0/odoo/addons/base/data/ir_module_category_data.xml
14    # for the full list
15    'category': 'Uncategorized',
16    'version': '1.0.0.0.0',
17
18    # any module necessary for this one to work correctly
19    'depends': ['base'],
20
21    # always loaded
22    'data': [
23        'security/ir.model.access.csv',
24        'views/library_author_views.xml',
25        'views/library_book_views.xml',
26        'views/library_category_views.xml',
27        'views/templates.xml',
28    ],
29
30    'license': 'AGPL-3',
31
32    'installable': True,
33
34 }
```

Para crear las vistas también lo haremos con Visual Studio Code.

Dichas vistas las alojaremos en la carpeta views.

Cada vista hará referencia su clase homónima creada en models.

Podemos ver el código de dichas clases en las siguientes imágenes.



```
1 <odoo>
2   <data>
3     <record id="categoria_menu_action" model="ir.actions.act_window">
4       <field name="name">categoria form</field>
5       <field name="res_model">library.category</field>
6       <field name="view_type">form</field>
7       <field name="view_mode">form</field>
8       <field name="domain">[]</field>
9       <field name="help" type="html">
10         <p class="oe_view_nocontent_create">Crea la primera categoria</p>
11       </field>
12     </record>
13     <menuitem id="categoria_menu" name="Categoria" action="categoria_menu_action">
14   </data>
15 </odoo>
```

```

1 <odoo>
2 <data>
3   <record id="book_menu_action" model="ir.actions.act_window">
4     <field name="name">book form</field>
5     <field name="res_model">library.book</field>
6     <field name="view_type">form</field>
7     <field name="view_mode">form</field>
8     <field name="domain">[]</field>
9     <field name="help" type="html">
10      <p class="oe_view_nocontent_create">Crea el primer libro</p>
11    </field>
12  </record>
13  <menuitem id="book_menu" name="Libro" action="book_menu_action"/>
14 </data>
15 </odoo>

```

```

1 <odoo>
2 <data>
3   <record id="author_menu_action" model="ir.actions.act_window">
4     <field name="name">author form</field>
5     <field name="res_model">library.author</field>
6     <field name="view_type">form</field>
7     <field name="view_mode">form</field>
8     <field name="domain">[]</field>
9     <field name="help" type="html">
10      <p class="oe_view_nocontent_create">Crea el primer autor</p>
11    </field>
12  </record>
13  <menuitem id="author_menu" name="Autor" action="author_menu_action"/>
14 </data>
15 </odoo>

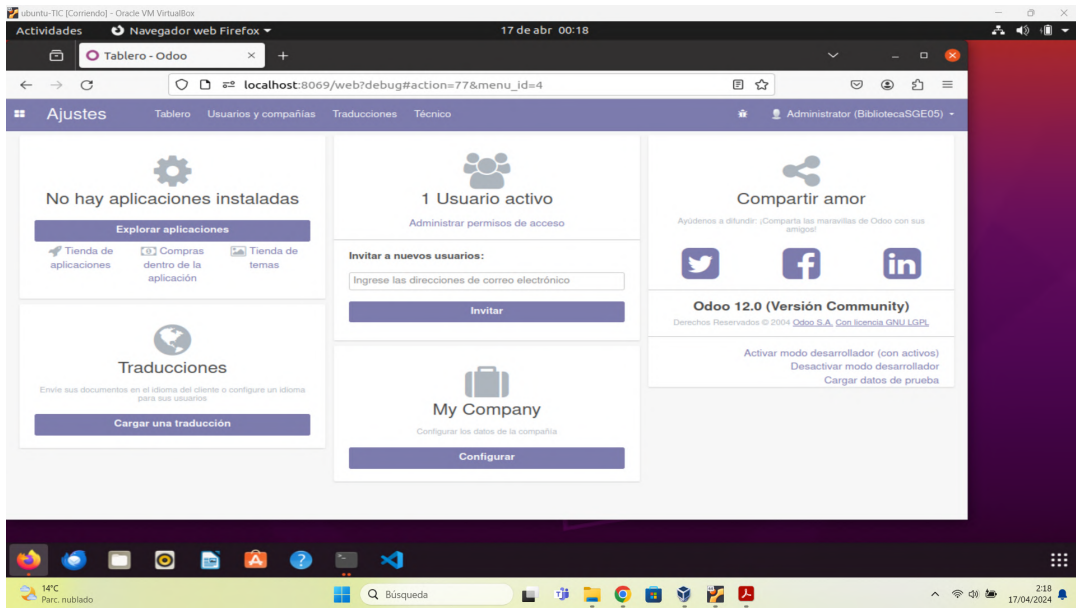
```

Como puede apreciarse, en resumen, le diremos que el modelo de cada una de ellas es `ir.actions.act_window` y el nombre será `category form`, `book form` y `author form` respectivamente.

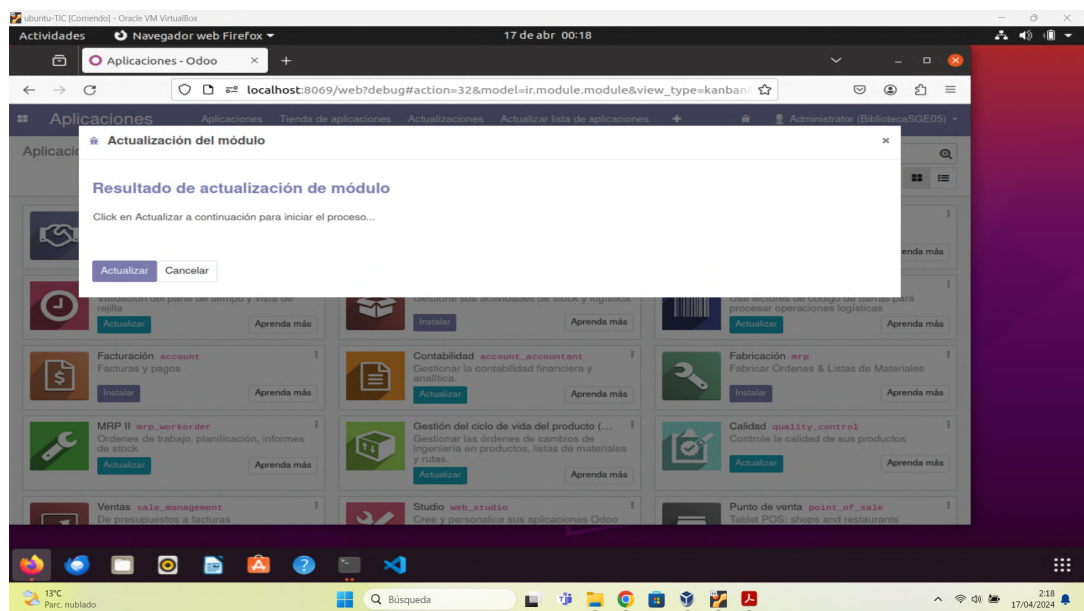
En `res_model` le diremos a que modelo pertenece cada una, `library.category`, `library.book` y `Library.author` respectivamente.

Tras ello copiaremos la carpeta `library` dentro de la carpeta `addons` de `odoo`.

Lo siguiente que haremos será dirigirnos a `odoo` y activaremos el modo desarrollador.

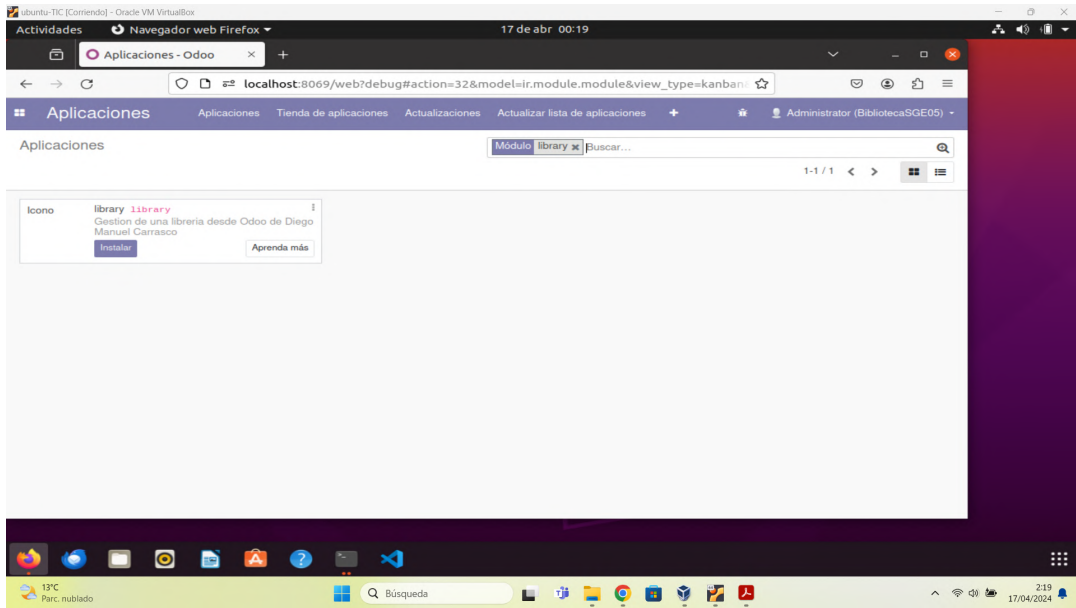


A continuación nos dirigimos a actualizar lista de aplicaciones y clicamos sobre actualizar.

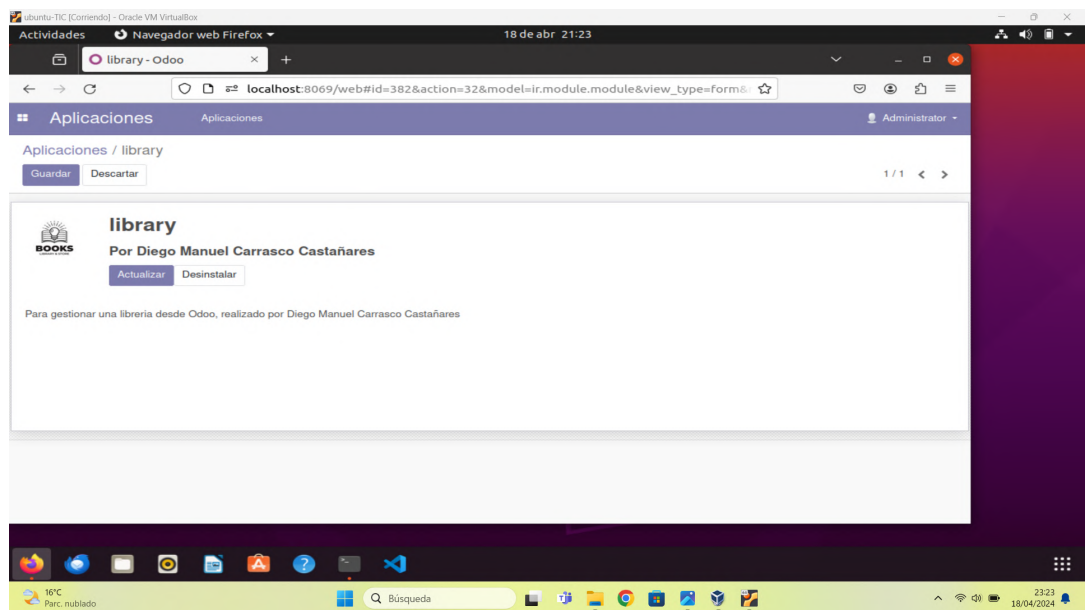


A continuación se nos quedará activado el filtro "aplicaciones", lo quitamos y buscamos el modulo creado, que en mi caso la he llamado library y clicamos en instalar.



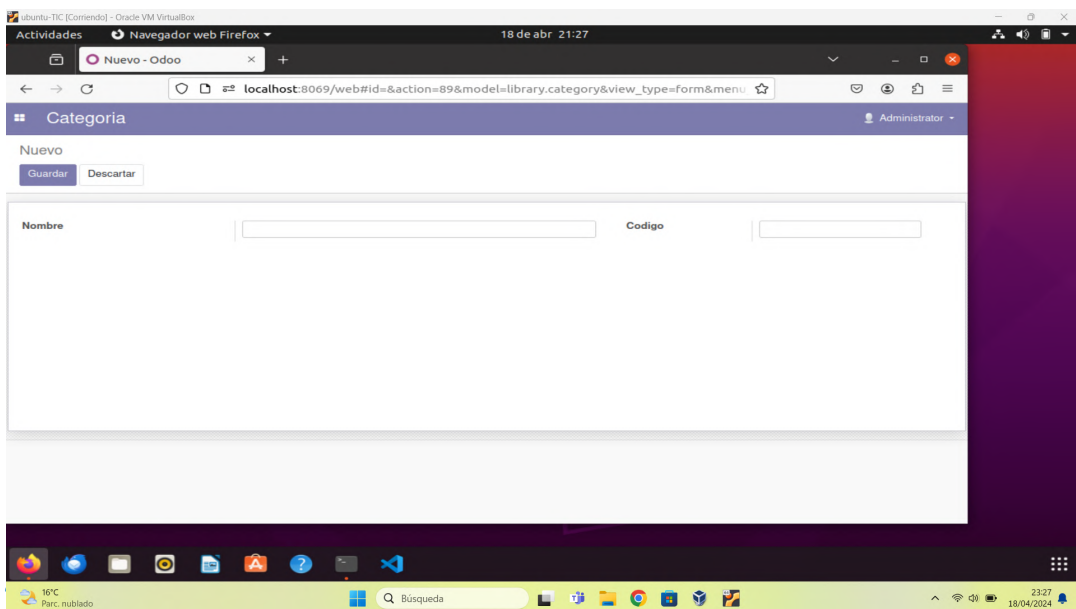
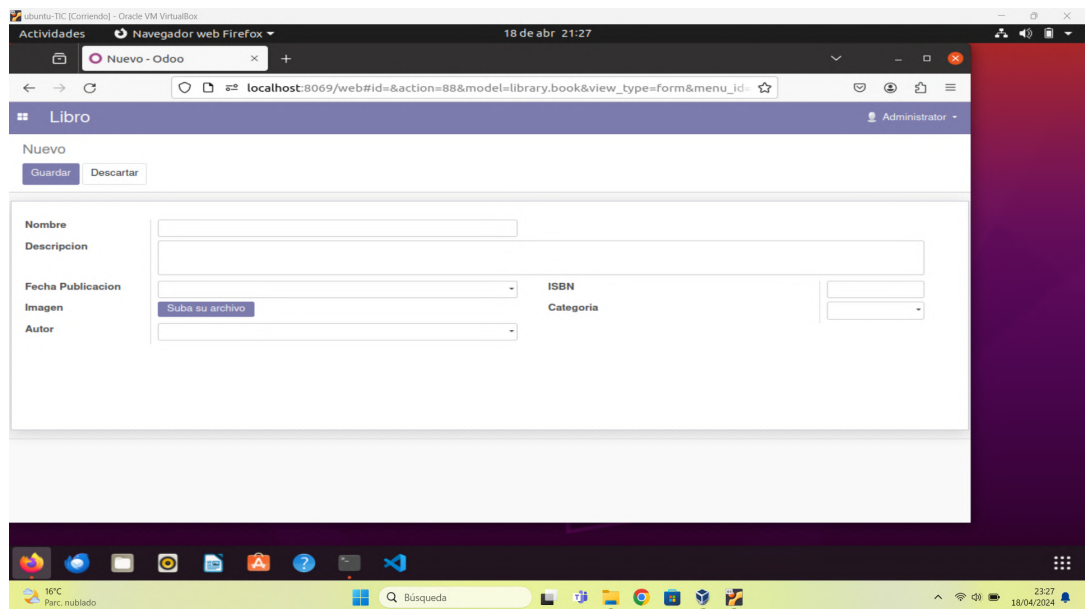
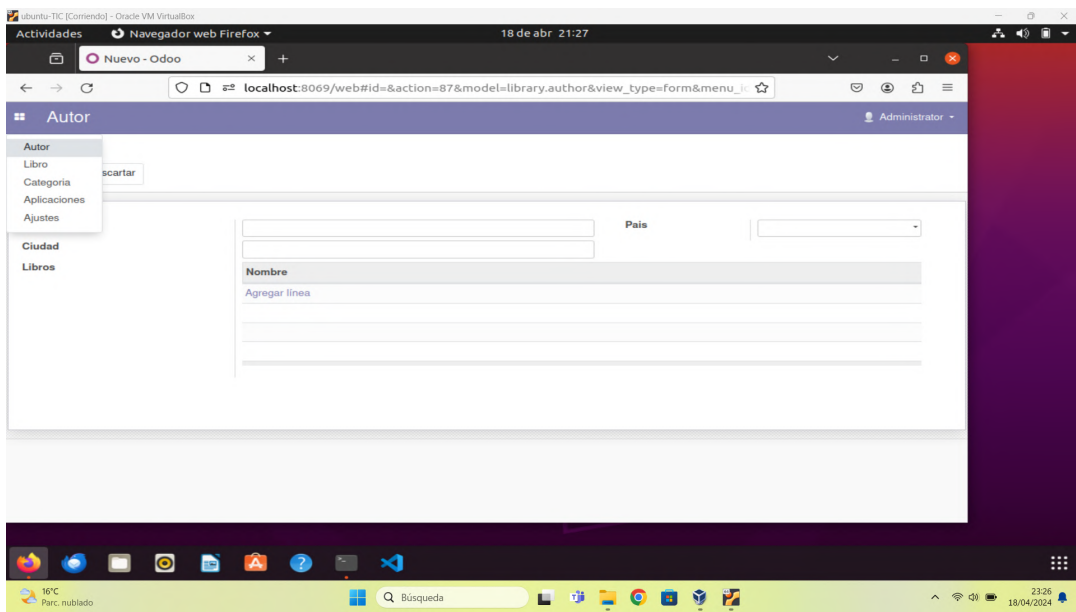


En la siguiente imagen podemos ver la información del módulo, ya el icono en la foto anterior no se ve.



En las siguientes imágenes podemos ver cada una de las vistas que creamos y como nos sale la opción en el menú.

Tras esas imágenes adjunto las imágenes donde puede verse como voy creando un libro, un autor y una categoría.





Ubuntu-TIC [Corriendo] - Oracle VM VirtualBox

Actividades Navegador web Firefox 18 de abr 21:38

Nuevo - Odoo

localhost:8069/web#id=&action=88&model=library.book&view\_type=form&menu\_id=

Libro Administrator

Nuevo

Guardar Descartar

Nombre La fragilidad de un corazón bajo la lluvia

Descripción Trata sobre dos personajes que tienen que superar muchos problemas, de los cuales comparten el mayor o mas importante de ellos, se ayudan con sus problemas y al final acabarán enamorando se.

Fecha Publicacion 01/06/2022

ISBN 840824715

Imagen la-fragilidad-de-un-corazon-bajo-la-lluvia.jpg

Categoria Romantica

Autor Maria Martinez

16°C Parc, nublado 23:38 18/04/2024

Ubuntu-TIC [Corriendo] - Oracle VM VirtualBox

Actividades Navegador web Firefox 18 de abr 21:40

Nuevo - Odoo

localhost:8069/web#id=&action=87&model=library.author&view\_type=form&menu\_id=

Autor Administrator

Nuevo

Guardar Descartar

Nombre Maria Martinez Pais España

Ciudad Elche

Libros

Nombre

Agregar línea

16°C Parc, nublado 23:40 18/04/2024

Ubuntu-TIC [Corriendo] - Oracle VM VirtualBox

Actividades Navegador web Firefox 18 de abr 21:42

Nuevo - Odoo

localhost:8069/web#id=3&action=87&model=library.author&view\_type=form&menu\_id=

Autor Administrator

Maria Martinez

Guardar Descartar

1 / 1

Nombre Maria Martinez Pais España

Ciudad Elche

Libros

Nombre

La fragilidad de un corazón bajo la lluvia

Agregar línea

16°C Parc, nublado 23:42 18/04/2024

Y con esto damos por finalizada la tarea.

Durante la ejecución de la tarea he tenido una serie de problemas que he ido solucionando hasta terminarla.

Vi que había un video cuando leí el temario, pero a la hora de ponerme con la tarea ni me acordé de él y simplemente fui siguiendo los pasos del temario.

El primer problema me lo encontré a la hora de instalar el modulo, que no se instalaba sin generar ningún error, problema por el cual te escribí.

Tras tu respuesta visualice el video, me aclaró muchas cosas y llegue hasta el momento de instalar el módulo en el cual me di de frente con un error, el cual me llevo hasta el fichero csv, en el que había cometido errores de sintaxis a la hora de dar acceso a los modelos.

Tras solucionar este problema me encontré otro, el cual me llevo hasta las vistas creadas (library\_book\_views, library\_author\_views y library\_category\_views) que también tenían errores de sintaxis los cuales estaban en la forma de llamar a al modelo al que hacen referencia, que solo nombre `<field name = "model">library.author</field>` cuando la sintaxis correcta es `<field name = "res_model">library.author</field>`. En las tres vistas cometí el mismo error.

Además de los errores anteriores, la primera que cree las visitas (siguiendo los pasos del temario) solo cree una vista en la cual metí los tres récord para que me generara las tres vistas, cosa que tiene que hacerse por separado ya que cada vista necesita su propio archivo xml.