



Actividades Evaluables

Odoo

Walter Martín Lopes



Índice

1. Preparando el entorno

1.1. Montando el contenedor Odoo

2. Actividad 1 (Ampliación)

2.1. Enunciado

2.2. Desarrollo

3. Actividad 2 (Ampliación)

3.1. Enunciado

3.2. Desarrollo

4. Actividad 3 (Desarrollo)

4.1. Enunciado

4.2. Desarrollo

5. Actividad 4 (Desarrollo)

5.1. Enunciado

5.2. Desarrollo

6. Bibliografía

Preparando el entorno





1. Preparando el entorno

1.1. Montando el contenedor Odoo

En primer lugar, instalamos [Docker Desktop](#) y creamos un repositorio en Git para trabajar con el contenedor en casa y en clase de manera ágil. Una vez creado el repositorio descargamos la imagen de Odoo desarrollo dada en el temario “`docker-compose.yml`” y la introducimos en la carpeta del repositorio.

Seguidamente abrimos una terminal y nos situamos en la carpeta del repositorio donde se encuentra la imagen y ejecutamos el comando “`docker-compose up -d`” para montar el contenedor.

```
cmd: C:\WINDOWS\system32\cmd.exe
      1 archivos           1.670 bytes
      2 dirs   79.205.097.472 bytes libres

C:\Users\Walter\Desktop\SSGE\Odoo>docker-compose up -d
[+] Running 25/25
  ▓ db 14 layers [████████████████████]    0B/0B      Pulled
  ▓ 2f44b7a888fa Pull complete          1.6s
  ▓ e2ab7b7507be Pull complete          0.5s
  ▓ 08eba411ca3 Pull complete          0.8s
  ▓ 100b0c654a5e4 Pull complete          1.1s
  ▓ 79f317c402ed Pull complete          1.6s
  ▓ da33959ecdff4 Pull complete          1.7s
  ▓ 2a835a615a89 Pull complete          2.1s
  ▓ 037064cd96ec Pull complete          2.1s
  ▓ 70fa2242f6e09 Pull complete          6.8s
  ▓ 4bec7ae19ef2 Pull complete          2.7s
  ▓ 2c2443a45c46 Pull complete          2.7s
  ▓ 326fd7e1c69e Pull complete          3.2s
  ▓ 2215c9c253b7 Pull complete          3.3s
  ▓ e63642645ad9 Pull complete          3.8s
  ▓ web 9 layers [██████████]    0B/0B      Pulled
  ▓ 0e0969fcfaa82 Pull complete          72.7s
  ▓ a86654a696af Pull complete          4.5s
  ▓ 1ade87532781 Pull complete          14.9s
  ▓ d092994f2bb4 Pull complete          5.4s
  ▓ 2b84941b2652 Pull complete          5.7s
  ▓ 1da77c7a47a8 Pull complete          19.7s
  ▓ 91d63cf6aa2 Pull complete          6.8s
  ▓ ccc620dfb39b6 Pull complete          7.3s
  ▓ f4aaa31640195 Pull complete          7.9s
  ▓ 0.6s
  ▓
  [+] Running 3/3
  ▓ Network odoo_default Created
  ▓ Container odoo-db-1 Started
  ▓ Container odoo-web-1 Started
  ▓ 0.1s
  ▓ 39.4s
  ▓ 0.6s

C:\Users\Walter\Desktop\SSGE\Odoo>
```

The screenshot shows a Windows Firewall alert dialog box. The title is "Alerta de seguridad de Windows". The message says: "Firewall de Windows Defender bloqueó algunas características de esta aplicación". Below the message, it lists the application: "Nombre: Docker Desktop Backend", "Editor: Docker Inc.", and "Ruta de acceso: C:\program files\docker\docker\resources\com.docker.backend.exe". There are two checkboxes at the bottom: "Permitir que Docker Desktop Backend se comunique en estas redes:" with "Redes privadas, como las domésticas o del trabajo" checked, and "Redes públicas, como las de aeropuertos y cafeterías (no se recomienda porque estas redes públicas suelen tener poca seguridad o carecer de ella)" unchecked. At the bottom right are buttons for "Permitir acceso" and "Cancelar".



1. Preparando el entorno

1.1. Montando el contenedor Odoo

Una vez hecho esto, se habrá montado el contenedor y si abrimos Docker nos aparecerá algo así. Solo tendremos que pinchar en el enlace de los puertos para abrir Odoo Web en local.

The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with options like Containers, Images, Volumes, Builds (NEW), Dev Environments (BETA), Docker Scout, Extensions, and Add Extensions. The main area is titled "Containers" and shows three items:

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
odoo	odoo:16	Running (2/2)	0.02%	8069:8069	6 minutes ago	[...]
web-1	25604f7b6	Running	0.01%	8069:8069	6 minutes ago	[...]
db-1	39aa2bd7	Running	0.01%	postgres:14	6 minutes ago	[...]

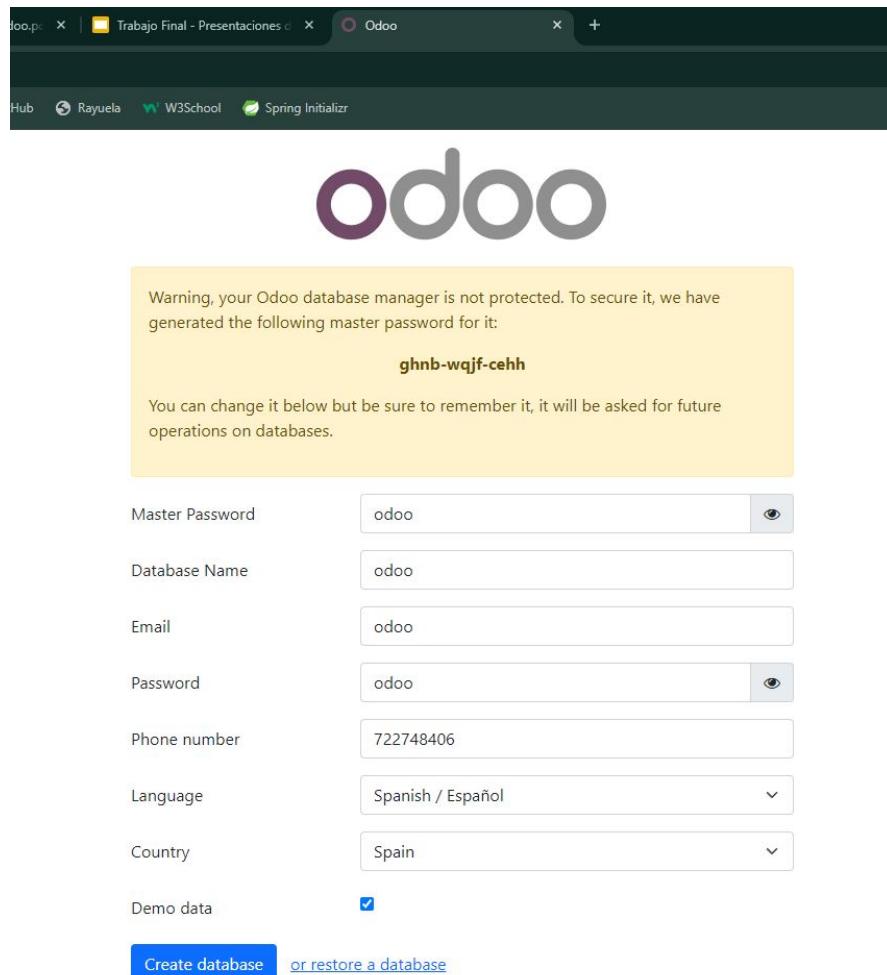
At the bottom, there are sections for "Walkthroughs" (Multi-container applications, Containerize your application) and a "View more in the Learning center" link. The status bar at the bottom shows "Engine running", system resources (RAM 2.50 GB, CPU 1.24%), and user information (Signed in). The version is v4.26.1.



1. Preparando el entorno

1.1. Montando el contenedor Odoo

Tendremos que crearnos una cuenta, para facilitar todo y que no haya problemas de contraseñas pondremos ‘odoo’ en todo.



1. Preparando el entorno

1.1. Montando el contenedor Odoo

Una vez terminado este paso solo nos queda logearnos y podremos acceder a la interfaz principal de Odoo. Solamente nos quedaría activar activar el modo desarrollador desde ajustes y tendríamos todo preparado.

Ejercicios evaluables Odoo x Actividades evaluables Odoo.x Trabajo Final - Presentaciones x Odoo - Aplicaciones x

localhost:8069/web?action=36&model=ir.module.module&view_type=kanban&cids=18&menu_id=5

Gmail Drive Trello Moodle Classroom GitHub Rayuela W3School Spring Initializr

Mitchell Admin

Aplicaciones Aplicaciones

Aplicaciones x Buscar...

Filtros Agrupar por Favoritos

1-53 / 53

CATEGORÍAS

Todos	Ventas	Servicios	Contabilidad	Inventario	Fabricación	Sitio web	Marketing	Recursos Humanos	Productividad	Administración
7	4	2	4	5	5	5	7	9	5	2

Ventas De presupuestos a facturas ACTIVAR APRENDA MÁS

Facturación Facturas y pagos ACTIVAR APRENDA MÁS

CRM Seguimiento de clientes potenciales y oportunidades próximas ACTIVAR APRENDA MÁS

MRP II Work Orders, Planning, Routing APRENDA MÁS ACTUALIZAR

Sitio web Constructor de sitio web empresarial ACTIVAR APRENDA MÁS

Inventario Gestione sus actividades de stock y logística ACTIVAR APRENDA MÁS

Contabilidad Accounting, Taxes, Budgets, Assets... APRENDA MÁS ACTUALIZAR

Conocimiento Centralice, gestione, comparta y haga crecer su librería de conocimiento APRENDA MÁS ACTUALIZAR

Compra Órdenes de compra, licitaciones y acuerdos. ACTIVAR APRENDA MÁS

Punto de venta Interfaz de Pdv amigable para usuarios para tiendas y restaurantes ACTIVAR APRENDA MÁS

Proyecto Organiza y planea tus proyectos ACTIVAR APRENDA MÁS

Comercio electrónico Venda sus productos online ACTIVAR APRENDA MÁS

Fabricación Fabricar Órdenes & Listas de Materiales ACTIVAR APRENDA MÁS

Marketing por email Diseñar, enviar y gestionar correos electrónicos ACTIVAR APRENDA MÁS

Partes de horas Track time & costs APRENDA MÁS ACTUALIZAR

Gastos Envíe, valide y refacture los gastos de los empleados. ACTIVAR APRENDA MÁS

Studio Create and Customize Applications APRENDA MÁS ACTUALIZAR

Ausencias Asigne PTO y siga las peticiones de permisos ACTIVAR APRENDA MÁS

Proceso de selección Seguimiento al flujo de reclamamiento ACTIVAR APRENDA MÁS

Servicio externo Programa y da seguimiento a operaciones en sitio, tiempo y materiales de facturación APRENDA MÁS ACTUALIZAR

Empleados Centralice la información de los empleados ACTIVAR APRENDA MÁS

Reciclaje de datos Encuentre registros antiguos y archívalos o borrélos. ACTIVAR MÁS INFORMACIÓN

Mantenimiento Seguimiento a equipo y administración de solicitudes de mantenimiento ACTIVAR APRENDA MÁS

Firmar Send documents to sign online APRENDA MÁS ACTUALIZAR

Mesa de Ayuda Track support tickets APRENDA MÁS ACTUALIZAR

Suscripciones MRR, Churn, Recurring payments APRENDA MÁS ACTUALIZAR

Calidad Quality Alerts, Control Points APRENDA MÁS ACTUALIZAR

eLearning Gestiona y publica en una plataforma eLearning APRENDA MÁS

Planificación Gestiona el horario de tus empleados APRENDA MÁS ACTUALIZAR

Eventos Pública eventos, vende entradas ACTIVAR APRENDA MÁS

Conversaciones Chat, puerta de enlace al correo y canales privados ACTIVAR APRENDA MÁS

Contactos Centralice su librería de direcciones APRENDA MÁS MÁS INFORMACIÓN

Actividad 1

(Ampliación)





2. Actividad 1 (Ampliación)

2.1. Enunciado

Modifica el ejemplo más sencillo de la lista de tareas de forma que:

- Tenga una nueva vista para visualizar las tareas en formato “Kanban”.
- Modifica las tareas para tener unos datos asignados y crea una nueva vista para visualizar en una vista “Calendar” los datos asignados.



2. Actividad 1 (Ampliación)

2.2. Desarrollo

Partimos del módulo lista de tareas simple creado en clase, a partir del cual implementaremos las modificaciones expuestas en el enunciado.

A continuación mostraré la interfaz, así como los modelos y vistas iniciales.

The screenshot shows the Odoo application interface. The top navigation bar includes links to Google Drive, Odoo - Aplicaciones, and various external services like Gmail, Drive, Trello, Moodle, Classroom, GitHub, Rayuela, W3School, and Spring Initializr. The main header bar has tabs for 'Aplicaciones', 'Actualizaciones', and 'Actualizar lista de aplicaciones'. The search bar contains 'Módulo lista de tareas'. The sidebar on the left is titled 'CATEGORÍAS' and lists categories: Todos, Ventas, Servicios, Contabilidad, Inventario, Fabricación, Sitio web, Marketing, Recursos Humanos, Productividad (with a count of 1), Técnico, and Administración. The central content area displays a module card for 'Lista de tareas simple' (Simple Task List), which includes its icon, name, status 'ACTIVAR', and a 'MÁS INFORMACIÓN' button.



2. Actividad 1 (Ampliación)

2.2. Desarrollo

The screenshot shows a web browser window with the URL `localhost:8069/web?debug=assets#action=313&model=list_tareas.lista&view_type=list&cids=1&menu_id=204`. The page title is "Odoo - Listado de tareas pendientes". The interface is a list view of tasks:

Tarea	Prioridad	Urgente	Realizada
Aprobar SSGE	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comprar jamón a Nacho	8	<input type="checkbox"/>	<input checked="" type="checkbox"/>



2. Actividad 1 (Ampliación)

2.2. Desarrollo

```
EXPLORADOR ... models.py 1 < views.xml
models > models.py > lista_tareas
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5  #Definimos el modelo de datos
6  class lista_tareas(models.Model):
7      #Nombre y descripcion del modelo de datos
8      _name = 'lista_tareas.lista'
9      _description = 'Modelo de la lista de tareas'
10     #Como no tenemos un atributo "name" en nuestro modelo, indicamos que cuando
11     #se necesite un nombre, se usara el atributo tarea
12     _rec_name="tarea"
13
14     #Elementos de cada fila del modelo de datos
15     tarea = fields.Char()
16     prioridad = fields.Integer()
17
18     #Indicamos que este valor es computado y se computara con la funcion "_value_urgente"
19     #Con store=True indicamos que pese a ser computado, cada vez que se compute se guarde en la base de datos
20     #esto se hace para que podamos utilizar el campo en busquedas, filtrados y ordenaciones
21     urgente = fields.Boolean(compute="_value_urgente", store=True)
22     realizada = fields.Boolean()
23
24
25     #Este es un ejemplo de "valor computado." Este computo depende de la variable prioridad.
26     #La dependencia se indica mediante el decorador
27     @api.depends('prioridad')
28
29     #Funcion para calcular el valor de urgente.
30     #Recibe "self" que se refiere al modelo completo (no a un registro solo)
31     def _value_urgente(self):
32         #Para cada registro... (recordamos, self es el modelo, no un registro)
33         for record in self:
34             #Si la prioridad es mayor que 10, se considera urgente, en otro caso no lo es
35             if record.prioridad>10:
36                 record.urgente = True
37             else:
38                 record.urgente = False
39
```

2. Actividad 1 (Ampliación)

2.2. Desarrollo

The screenshot shows the Odoo Studio interface with the following details:

- Toolbar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, ...
- Search Bar:** ListaTareasSimple
- Left Sidebar:** EXPLORADOR, LISTAREASIMPLE, models.py, security, views, _init_.py, _manifest_.py.
- Code Editor:** The code editor displays the content of `models.py` and `views.xml`.
- Code Content:**

```
<odoo>
    <data>
        <!-- Acciones al abrir las vistas en los modelos-->
        <record model="ir.actions.act_window" id="action_lista_tareas">
            <field name="name">Listado de tareas pendientes</field>
            <!-- Nombre del modelo que mostraremos en la vista-->
            <field name="res_model">lista_tareas.lista</field>
            <!-- Indicamos que nuestro formulario tendrá vista tree (para mostrar todos los datos y vista form (para crear o editar datos)
            La vista Form no la definimos, por lo cual tendremos la versión automática.
            La vista tree la definiremos más abajo.
        -->
            <field name="view_mode">tree,form</field>
        </record>

        <!-- Top menu item
        En este ejemplo, todo el menú de "menutem" está puesto como ejemplo, no hace una función concreta
        -->
        <menutem name="Listado de tareas" id="lista_tareas_menu_root" />

        <!-- menu categories -->
        <menutem name="Opciones Lista Tareas" id="lista_tareas_menu_1" parent="lista_tareas_menu_root" />

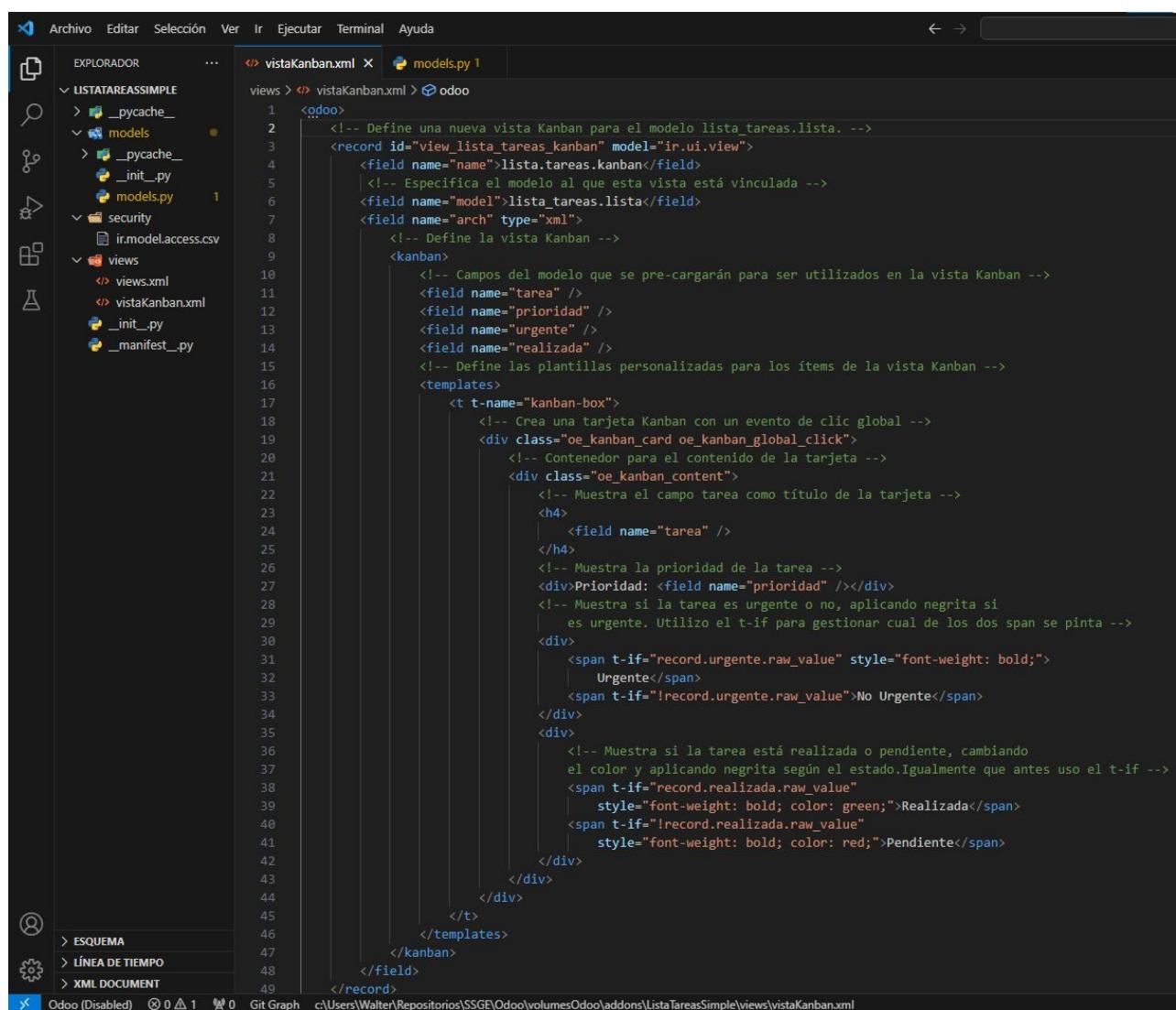
        <!-- actions -->
        <menutem name="Mostrar lista" id="lista_tareas_menu_1_list" parent="lista_tareas_menu_1" action="action_lista_tareas" />

        <!-- Odoo puede colocar vistas automáticas, aunque con un aspecto y funcionalidad mejorable
        Nosotros, para afinarlo, explicaremos de manera explícita cómo es cada vista -->
        <!-- Definimos cómo es la vista explícita de la lista-->
        <record model="ir.ui.view" id="lista_tareas">
            <field name="name">Lista de tareas</field>
            <field name="model">lista_tareas.lista</field>
            <field name="arch" type="xml">
                <tree>
                    <field name="tarea" />
                    <field name="prioridad" />
                    <field name="urgente" />
                    <field name="realizada" />
                </tree>
            </field>
        </record>
    </data>
</odoo>
```

2. Actividad 1 (Ampliación)

2.2. Desarrollo

Lo primero será crear una nueva vista para visualizar las tareas en formato Kanban y seguidamente cargar la vista en el 'data' del manifest.py.

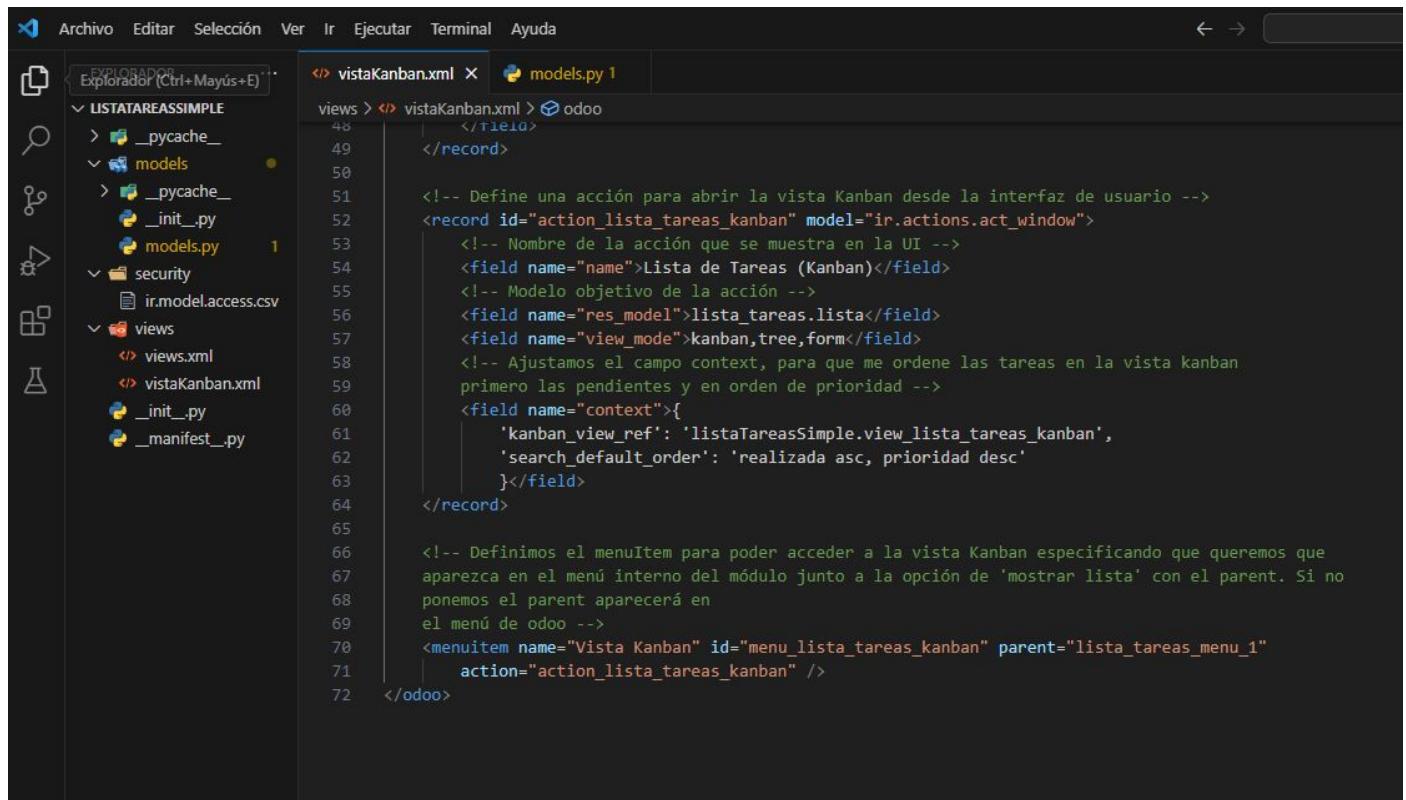


The screenshot shows a code editor interface with several tabs and panels. The main tab is 'vistaKanban.xml'. Below it, there's a 'models.py 1' tab. On the left, there's a file tree for 'LISTATAREASSIMPLE' containing 'views', 'security', and 'views.xml'. The 'views.xml' file is also visible in the tree. At the bottom, there are status bars for 'Ondo (Disabled)', 'Git Graph', and a file path 'c:\Users\Walter\Repositorios\SSGE\Odoo\addons\ListaTareasSimple\views\vistaKanban.xml'.

```
<odoo>
    <!-- Define una nueva vista Kanban para el modelo lista_tareas.lista. -->
    <record id="view_lista_tareas_kanban" model="ir.ui.view">
        <field name="name">lista.tareas.kanban</field>
        <!-- Especifica el modelo al que esta vista está vinculada -->
        <field name="model">lista_tareas.lista</field>
        <field name="arch" type="xml">
            <!-- Define la vista Kanban -->
            <kanban>
                <!-- Campos del modelo que se pre-cargarán para ser utilizados en la vista Kanban -->
                <field name="tarea" />
                <field name="prioridad" />
                <field name="urgente" />
                <field name="realizada" />
                <!-- Define las plantillas personalizadas para los ítems de la vista Kanban -->
                <templates>
                    <t t-name="kanban-box">
                        <!-- Crea una tarjeta Kanban con un evento de clic global -->
                        <div class="oe_kanban_card oe_kanban_global_click">
                            <!-- Contenedor para el contenido de la tarjeta -->
                            <div class="oe_kanban_content">
                                <!-- Muestra el campo tarea como título de la tarjeta -->
                                <h4>
                                    <field name="tarea" />
                                </h4>
                                <!-- Muestra la prioridad de la tarea -->
                                <div>Prioridad: <field name="prioridad" /></div>
                                <!-- Muestra si la tarea es urgente o no, aplicando negrita si es urgente. Utilizo el t-if para gestionar cual de los dos span se pinta -->
                                <div>
                                    <span t-if="record.urgente.raw_value" style="font-weight: bold;">
                                        Urgente</span>
                                    <span t-if="!record.urgente.raw_value">No Urgente</span>
                                </div>
                                <div>
                                    <!-- Muestra si la tarea está realizada o pendiente, cambiando el color y aplicando negrita según el estado. Igualmente que antes uso el t-if -->
                                    <span t-if="record.realizada.raw_value"
                                          style="font-weight: bold; color: green;">Realizada</span>
                                    <span t-if="!record.realizada.raw_value"
                                          style="font-weight: bold; color: red;">Pendiente</span>
                                </div>
                            </div>
                        </div>
                    </t>
                </templates>
            </kanban>
        </field>
    </record>
```

2. Actividad 1 (Ampliación)

2.2. Desarrollo



The screenshot shows a code editor interface with the following details:

- Menu Bar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda.
- Left Sidebar (Explorador):** Shows the file structure of the module "LISTATAREASSIMPLE".
 - __pycache__
 - models (selected)
 - __pycache__
 - __init__.py
 - models.py
 - security (with ir.model.access.csv)
 - views (with views.xml, vistaKanban.xml)
 - __init__.py
 - __manifest__.py
- Central Area:** The "vistaKanban.xml" file is open in the editor.

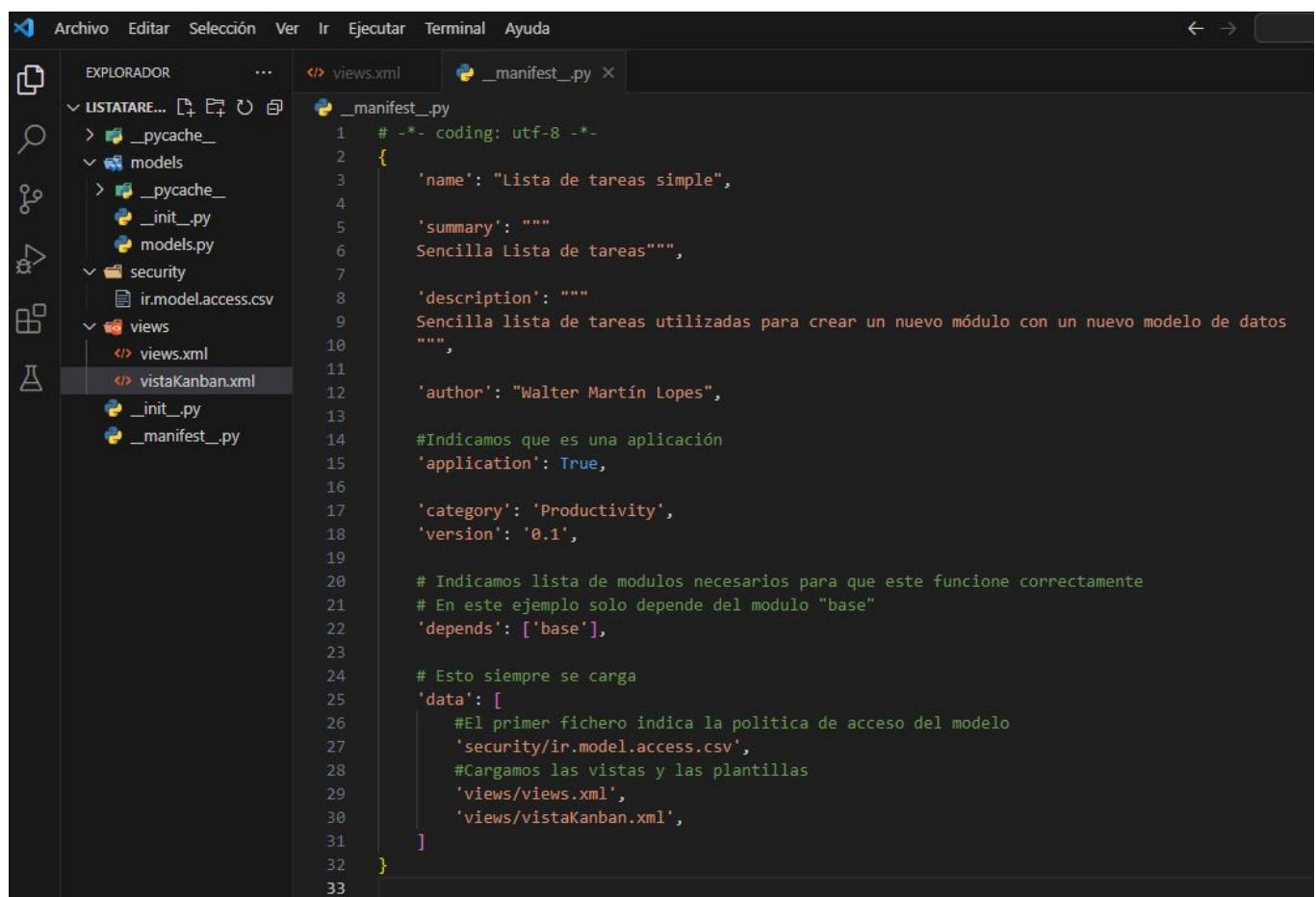
```
<odoo>
    </record>
    <!-- Define una acción para abrir la vista Kanban desde la interfaz de usuario -->
    <record id="action_lista_tareas_kanban" model="ir.actions.act_window">
        <!-- Nombre de la acción que se muestra en la UI -->
        <field name="name">Lista de Tareas (Kanban)</field>
        <!-- Modelo objetivo de la acción -->
        <field name="res_model">lista_tareas.lista</field>
        <field name="view_mode">kanban,tree,form</field>
        <!-- Ajustamos el campo context, para que me ordene las tareas en la vista kanban
        primero las pendientes y en orden de prioridad -->
        <field name="context">
            'kanban_view_ref': 'listaTareasSimple.view_lista_tareas_kanban',
            'search_default_order': 'realizada asc, prioridad desc'
        </field>
    </record>
    <!-- Definimos el menuItem para poder acceder a la vista Kanban especificando que queremos que
    aparezca en el menú interno del módulo junto a la opción de 'mostrar lista' con el parent. Si no
    ponemos el parent aparecerá en
    el menú de odoo -->
    <menuitem name="Vista Kanban" id="menu_lista_tareas_kanban" parent="lista_tareas_menu_1"
              action="action_lista_tareas_kanban" />
</odoo>
```



2. Actividad 1 (Ampliación)

2.2. Desarrollo

Cargamos la vista en el 'data'.



```
# -*- coding: utf-8 -*-
{
    'name': "Lista de tareas simple",
    'summary': '',
    'description': '',
    'author': "Walter Martín Lopes",
    'application': True,
    'category': 'Productivity',
    'version': '0.1',
    # Indicamos lista de modulos necesarios para que este funcione correctamente
    # En este ejemplo solo depende del modulo "base"
    'depends': ['base'],
    # Esto siempre se carga
    'data': [
        #El primer fichero indica la politica de acceso del modelo
        'security/ir.model.access.csv',
        #Cargamos las vistas y las plantillas
        'views/views.xml',
        'views/vistaKanban.xml',
    ]
}
```



2. Actividad 1 (Ampliación)

2.2. Desarrollo

Para hacer que funcione el 'order' no basta con implementarlo en la vista, también hay que implementarlo en el modelo añadiendo el siguiente código.

```
# Establecemos el orden por defecto de los registros, primero las no realizadas y en orden de prioridad
_order = 'realizada asc, prioridad desc'
```

A continuación muestro el resultado de la implementación de la nueva vista Kanban.



2. Actividad 1 (Ampliación)

2.2. Desarrollo

The screenshot shows a web browser window with the Odoo application running on localhost:8069. The title bar reads "Odoo - Listado de tareas pendiente". The main content area is titled "Listado de tareas simple" and "Opciones Lista Tareas". A purple header bar contains buttons for "Nuevo" and "Mostrar lista" (selected), and a "Vista Kanban" button. Below this is a table with three columns: "Tarea" (checkboxes), "Prioridad/Urgente" (with values 15, 11, and 8), and "Realizada" (checkboxes). The tasks listed are:

Tarea	Prioridad/Urgente	Realizada
Terminar Actividad 1 Odoo	15 <input checked="" type="checkbox"/>	<input type="checkbox"/>
Aprobar SSGE	11 <input checked="" type="checkbox"/>	<input type="checkbox"/>
Comprar jamón a Nacho	8 <input type="checkbox"/>	<input checked="" type="checkbox"/>

2. Actividad 1 (Ampliación)

2.2. Desarrollo

The screenshot shows a web browser displaying the Odoo Kanban view at localhost:8069/web?action=354&model=list_tareas.lista&view_type=kanban&cids=1&menu_id=204. The interface includes a header with 'Listado de tareas simple' and 'Opciones Lista Tareas', a search bar, and a toolbar with filters, grouping, and favorites. Three tasks are listed in columns:

- Terminar Actividad 1 Odoo**
Prioridad: 15
Urgente
Pendiente
- Aprobar SSGE**
Prioridad: 11
Urgente
Pendiente
- Comprar jamón a Nacho**
Prioridad: 8
No Urgente
Realizada



2. Actividad 1 (Ampliación)

2.2. Desarrollo

Para implementar una nueva vista Calendar necesitamos actualizar el modelo añadiendo los campos de 'fecha_inicio' y 'fecha_fin'.

```
# Nuevos campos para fechas de asignación
fecha_inicio = fields.Date(string='Fecha de Inicio')
fecha_fin = fields.Date(string='Fecha de Fin')
```

Seguidamente creamos e implementamos la vista calendar.

```
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
```

```
Explorador
```

```
LISTATAREASSIMPLE
```

```
vistaKanban.xml models.py vistaCalendar.xml
```

```
views
```

```
vistaCalendar.xml
```

```
models.py
```

```
security ir.model.access.csv
```

```
vistaKanban.xml
```

```
views.xml
```

```
_init_.py
```

```
_manifest_.py
```

```
<odoo>
    <!-- Define una vista de Calendario para el modelo lista_tareas.lista -->
    <record id="view_lista_tareas_calendar" model="ir.ui.view">
        <!-- Asigna un nombre descriptivo a la vista de Calendario -->
        <field name="name">lista.tareas.calendar</field>
        <!-- Especifica el modelo al que esta vista está vinculada -->
        <field name="model">lista_tareas.lista</field>
        <field name="arch" type="xml">
            <!-- Configura la vista de Calendario, especificando las fechas de inicio y fin de las tareas -->
            <calendar string="Calendario de Tareas" date_start="fecha_inicio" date_stop="fecha_fin">
                <!-- Incluye el campo tarea para mostrar en los eventos del calendario -->
                <field name="tarea"/>
                <!-- Incluye los campos urgente y realizada para poder filtrar en el calendario -->
                <field name="urgente"/>
                <field name="realizada"/>
            </calendar>
        </field>
    </record>
    <!-- Define una acción para abrir la vista de Calendario desde la interfaz de usuario -->
    <record id="action_lista_tareas_calendar" model="ir.actions.act_window">
        <!-- Nombre de la acción que se muestra en la UI -->
        <field name="name">Tareas en Calendar</field>
        <!-- Modelo objetivo de la acción -->
        <field name="res_model">lista_tareas.lista</field>
        <field name="view_mode">calendar,tree,form</field>
        <field name="view_id" ref="view_lista_tareas_calendar"/>
    </record>
    <!-- Como en la vista Kanban, definimos el menuItem para poder acceder a la vista Calendar especificando que queremos que aparezca en el menú interno del módulo. -->
    <menuitem id="menu_lista_tareas_calendar" name="Vista Calendario" parent="lista_tareas_menu_1" action="action_lista_tareas_calendar" />
</odoo>
```



2. Actividad 1 (Ampliación)

2.2. Desarrollo

No hay que olvidar actualizar el 'data' en el manifest cargando la vista nueva.

```
# -*- coding: utf-8 -*-
{
    'name': "Lista de tareas simple",
    'summary': """
        Sencilla Lista de tareas""",
    'description': """
        Sencilla lista de tareas utilizadas para crear un nuevo módulo con un nuevo modelo de datos""",
    'author': "Walter Martín Lopes",
    '#Indicamos que es una aplicación
    'application': True,
    'category': 'Productivity',
    'version': '0.1',
    '# Indicamos lista de modulos necesarios para que este funcione correctamente
    # En este ejemplo solo depende del modulo "base"
    'depends': ['base'],
    '# Esto siempre se carga
    'data': [
        #El primer fichero indica la política de acceso del modelo
        'security/ir.model.access.csv',
        #Cargamos las vistas y las plantillas
        'views/views.xml',
        'views/vistaKanban.xml',
        'views/vistaCalendar.xml',
    ]
}
```



2. Actividad 1 (Ampliación)

2.2. Desarrollo

A continuación muestro el resultado final de la implementación de la nueva vista calendar.

The image contains two screenshots of the Odoo application interface, specifically the 'Listado de tareas simple' (Simple Task List) module.

Screenshot 1: This screenshot shows a single task entry in a form view. The task details are as follows:

Tarea	Terminar Actividad 1 Odoo	Prioridad	15
Urgente	<input checked="" type="checkbox"/>	Realizada	<input type="checkbox"/>
Fecha de Inicio	05/02/2024	Fecha de Fin	06/02/2024

Screenshot 2: This screenshot shows a list of tasks in a Kanban view. The tasks listed are:

Tarea	Prioridad	Urgente	Realizada
Terminar Actividad 1 Odoo	15	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Aprobar SSGE	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comprar jamón a Nacho	8	<input type="checkbox"/>	<input checked="" type="checkbox"/>



2. Actividad 1 (Ampliación)

2.2. Desarrollo

The screenshot shows the Odoo calendar interface for February 2024. The main view displays a grid of days from Monday to Sunday. Several tasks are listed as events:

- On February 5, there is a task titled "Terminar Actividad 1 Odoo".
- On February 19, there is a task titled "Aprobar SSGE".
- On February 29, there is a task titled "Comprar jamón a N...".

A sidebar on the right shows a detailed monthly calendar for February 2024, with the 5th highlighted in red. The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and Mitchell Admin.

Actividad 2

(Ampliación)





3. Actividad 2 (Ampliación)

3.1. Enunciado

Amplía el módulo del ejemplo de biblioteca de cómics de forma que:

- Incluya la posibilidad de gestionar socios, almacenar nombres, apellidos e identificador.
- Introduzca la posibilidad de que haya ejemplares de comics para prestar.
 - El modelo de cómic actual ha de servir como referencia de la información del cómic. A parte de este modelo, tendrá que plantearse un nuevo modelo para indicar ejemplares de préstamo de este cómic.
 - De aquellos ejemplares de préstamo deberemos controlar a quién están prestados, y la fecha de inicio y fin del préstamo. No es necesario tener un histórico de préstamos, solo quién tiene el cómic en cada momento, cuando se le ha prestado y la fecha prevista de entrega.
 - La fecha de préstamo no puede ser posterior a la fecha de hoy.
 - La fecha prevista de retorno no puede ser anterior al día de hoy.



3. Actividad 2 (Ampliación)

3.2. Desarrollo

Partimos del módulo biblioteca de cómic simple creado en clase, a partir del cual implementaremos las modificaciones expuestas en el enunciado.

A continuación mostraré la interfaz, así como los modelos y vistas iniciales.

The screenshot shows the Odoo application manager interface. The top navigation bar includes links to Google Drive, Odoo - Aplicaciones, and other external services like Gmail, Drive, Trello, Moodle, Classroom, GitHub, Rayuela, W3School, and Spring Initializr. The main menu has tabs for 'Aplicaciones', 'Aplicaciones', 'Tienda de aplicaciones', 'Actualizaciones', 'Actualizar lista de aplicaciones', and 'Aplicar actualizaciones programadas'. A search bar at the top right contains filters for 'Aplicaciones', 'Módulo comic', and a search term 'Buscar...'. On the left, a sidebar titled 'CATEGORÍAS' lists various application categories: Todos, Ventas, Servicios, Contabilidad, Inventario, Fabricación, Sitio web, Marketing, Recursos Humanos, Productividad, Técnico, and Administración. The main content area displays a card for the 'Biblioteca Comics Simple' module, which is listed under the 'Todos' category. The card shows a thumbnail image of a comic book store, the name 'Biblioteca Comics Simple', the module identifier 'Comicssimple', its status as 'Instalado' (Installed), and a 'APRENDA MÁS' (Learn More) button.

3. Actividad 2 (Ampliación)

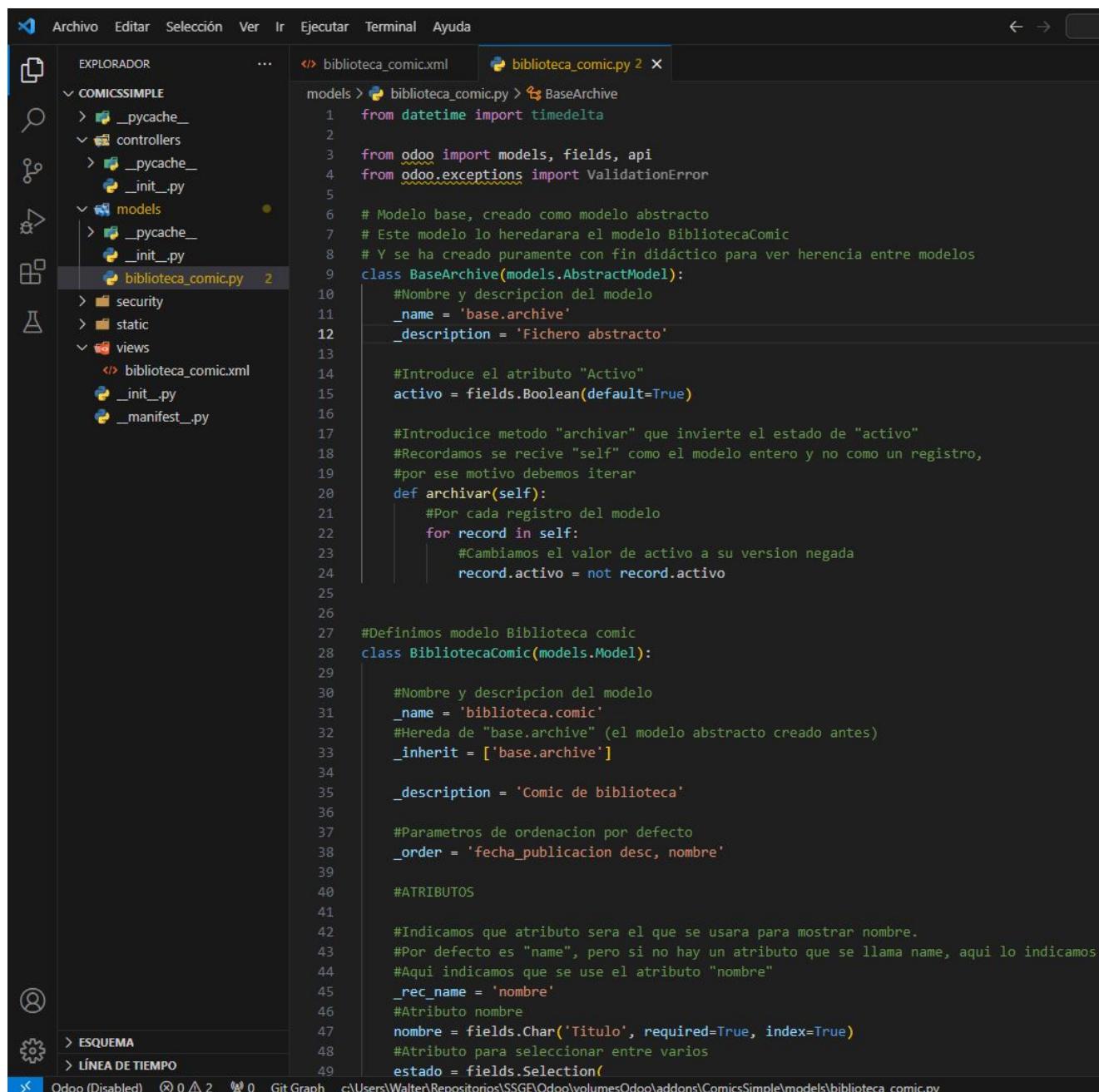
3.2. Desarrollo

A screenshot of a web browser window. The address bar shows the URL: localhost:8069/web?action=356&model=biblioteca.comic&view_type=list&cids=1&menu_id=228. The browser has several tabs open, including "Mi unidad - Google Drive", "Odoo - Biblioteca de Comics", "superman comic - Buscar con Google", and "The Beano - Búsqueda de Google". The main content area displays a list of comic books in a table format. The table has three columns: "Titulo", "Fecha publicación", and "Estado". There are three rows of data:

Titulo	Fecha publicación	Estado
Micky Maus	05/02/2024	Disponible
The Beano	02/02/2024	Disponible
SuperMan	09/11/2023	Perdido

3. Actividad 2 (Ampliación)

3.2. Desarrollo

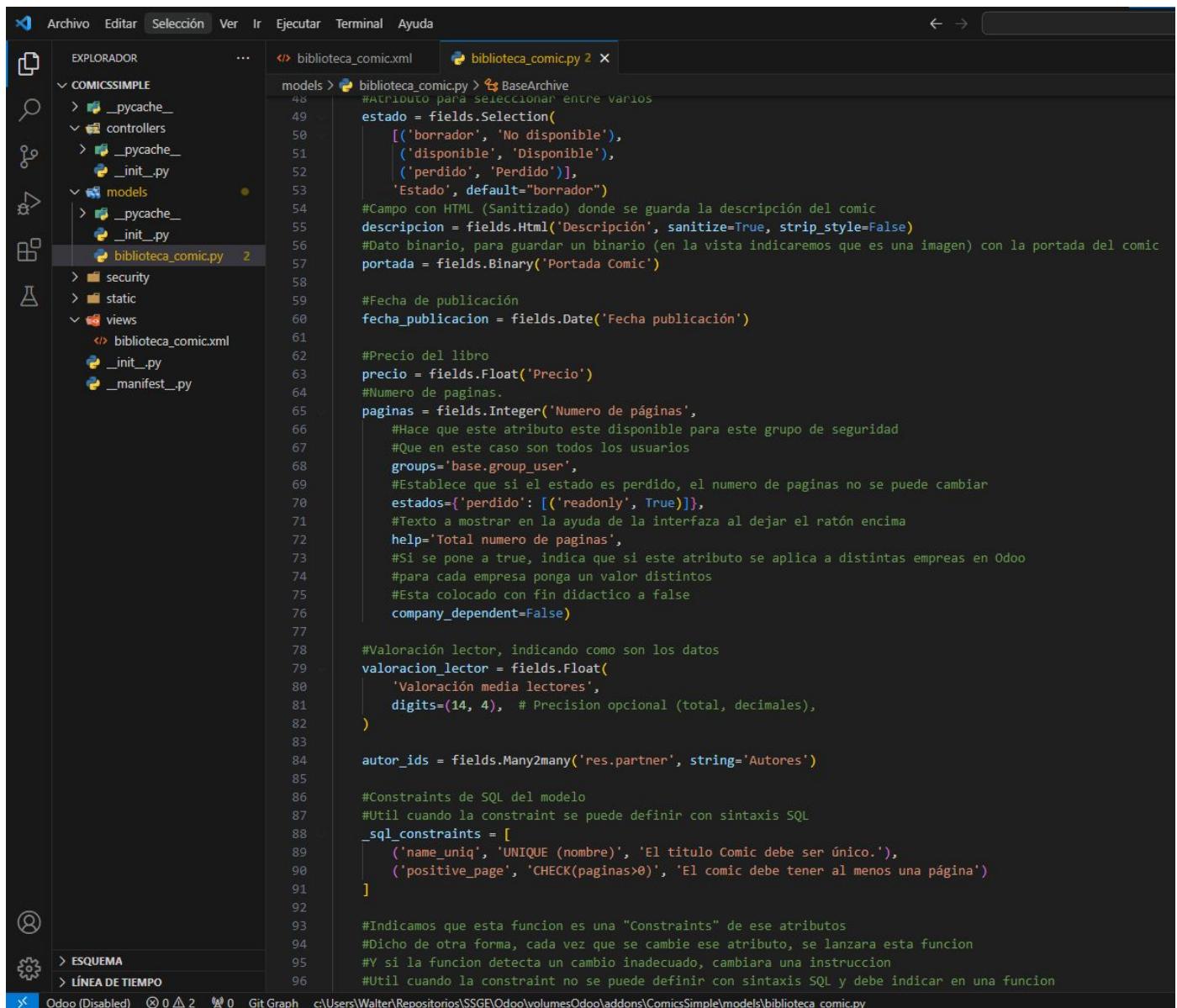


The screenshot shows the Odoo development interface. On the left is a sidebar with icons for Explorer, Search, Models, Security, Static, and Views. The Explorer section shows a folder structure for 'COMICSSIMPLE' containing '_pycache_', 'controllers', 'models', 'security', 'static', and 'views'. The 'views' folder contains 'biblioteca_comic.xml', '_init_.py', and '_manifest_.py'. The main area is a code editor titled 'biblioteca_comic.py' with two changes (version 2). The code is as follows:

```
models > biblioteca_comic.py > BaseArchive
1   from datetime import timedelta
2
3   from odoo import models, fields, api
4   from odoo.exceptions import ValidationError
5
6   # Modelo base, creado como modelo abstracto
7   # Este modelo lo heredara el modelo BibliotecaComic
8   # Y se ha creado puramente con fin didáctico para ver herencia entre modelos
9   class BaseArchive(models.AbstractModel):
10      #Nombre y descripción del modelo
11      _name = 'base.archive'
12      _description = 'Fichero abstracto'
13
14      #Introduce el atributo "Activo"
15      activo = fields.Boolean(default=True)
16
17      #Introduce método "archivar" que invierte el estado de "activo"
18      #Recordamos se recibe "self" como el modelo entero y no como un registro,
19      #por ese motivo debemos iterar
20      def archivar(self):
21          #Por cada registro del modelo
22          for record in self:
23              #Cambiemos el valor de activo a su versión negada
24              record.activo = not record.activo
25
26
27      #Definimos modelo Biblioteca comic
28      class BibliotecaComic(models.Model):
29
30          #Nombre y descripción del modelo
31          _name = 'biblioteca.comic'
32          #Hereda de "base.archive" (el modelo abstracto creado antes)
33          _inherit = ['base.archive']
34
35          _description = 'Comic de biblioteca'
36
37          #Parámetros de ordenación por defecto
38          _order = 'fecha_publicacion desc, nombre'
39
40          #ATRIBUTOS
41
42          #Indicamos que atributo será el que se usará para mostrar nombre.
43          #Por defecto es "name", pero si no hay un atributo que se llama name, aquí lo indicamos
44          #Aquí indicamos que se use el atributo "nombre"
45          _rec_name = 'nombre'
46          #Atributo nombre
47          nombre = fields.Char('Titul', required=True, index=True)
48          #Atributo para seleccionar entre varios
49          estado = fields.Selection(
```

3. Actividad 2 (Ampliación)

3.2. Desarrollo



The screenshot shows the Odoo Python code editor interface. On the left is a sidebar with icons for Explorer, Schema, and Timeline. The main area has tabs for 'EXPLORADOR' (showing the project structure), 'ESQUEMA' (schema browser), and 'LÍNEA DE TIEMPO' (time travel). The current tab is 'biblioteca_comic.py'. The code editor displays the following Python code:

```
models > biblioteca_comic.py > BaseArchive
48     #Atributo para seleccionar entre varios
49     estado = fields.Selection(
50         [('borrador', 'No disponible'),
51          ('disponible', 'Disponible'),
52          ('perdido', 'Perdido')],
53          'Estado', default="borrador")
54
55     #Campo con HTML (Sanitizado) donde se guarda la descripción del comic
56     descripcion = fields.Html('Descripción', sanitize=True, strip_style=False)
57
58     #Dato binario, para guardar un binario (en la vista indicaremos que es una imagen) con la portada del comic
59     portada = fields.Binary('Portada Comic')
60
61     #Fecha de publicación
62     fecha_publicacion = fields.Date('Fecha publicación')
63
64     #Precio del libro
65     precio = fields.Float('Precio')
66
67     #Número de páginas.
68     paginas = fields.Integer('Número de páginas',
69         #Hace que este atributo esté disponible para este grupo de seguridad
70         #Que en este caso son todos los usuarios
71         groups='base.group_user',
72         #Establece que si el estado es perdido, el número de páginas no se puede cambiar
73         estados={'perdido': [('readonly', True)]},
74         #Texto a mostrar en la ayuda de la interfaz al dejar el ratón encima
75         help='Total número de páginas',
76         #Si se pone a true, indica que si este atributo se aplica a distintas empresas en Odoo
77         #para cada empresa ponga un valor distinto
78         #Esta colocado con fin didactico a false
79         company_dependent=False)
80
81         #Valoración lector, indicando como son los datos
82         valoracion_lector = fields.Float(
83             'Valoración media lectores',
84             digits=(14, 4), # Precision opcional (total, decimales),
85         )
86
87         autor_ids = fields.Many2many('res.partner', string='Autores')
88
89         #Constraints de SQL del modelo
90         #Util cuando la constraint se puede definir con sintaxis SQL
91         _sql_constraints = [
92             ('name_uniq', 'UNIQUE (nombre)', 'El título Comic debe ser único.'),
93             ('positive_page', 'CHECK(paginas>0)', 'El comic debe tener al menos una página')
94         ]
95
96         #Indicamos que esta función es una "Constraints" de ese atributos
97         #Dicho de otra forma, cada vez que se cambie ese atributo, se lanzara esta función
98         #Y si la función detecta un cambio inadecuado, cambiara una instrucción
99         #Util cuando la constraint no se puede definir con sintaxis SQL y debe indicar en una función
```

At the bottom, there are status indicators: Odoo (Disabled), Git Graph, and a path bar showing 'c:\Users\Walter\Repositorios\SSGE\Odoo\addons\ComicsSimple\models\biblioteca_comic.py'.



3. Actividad 2 (Ampliación)

3.2. Desarrollo

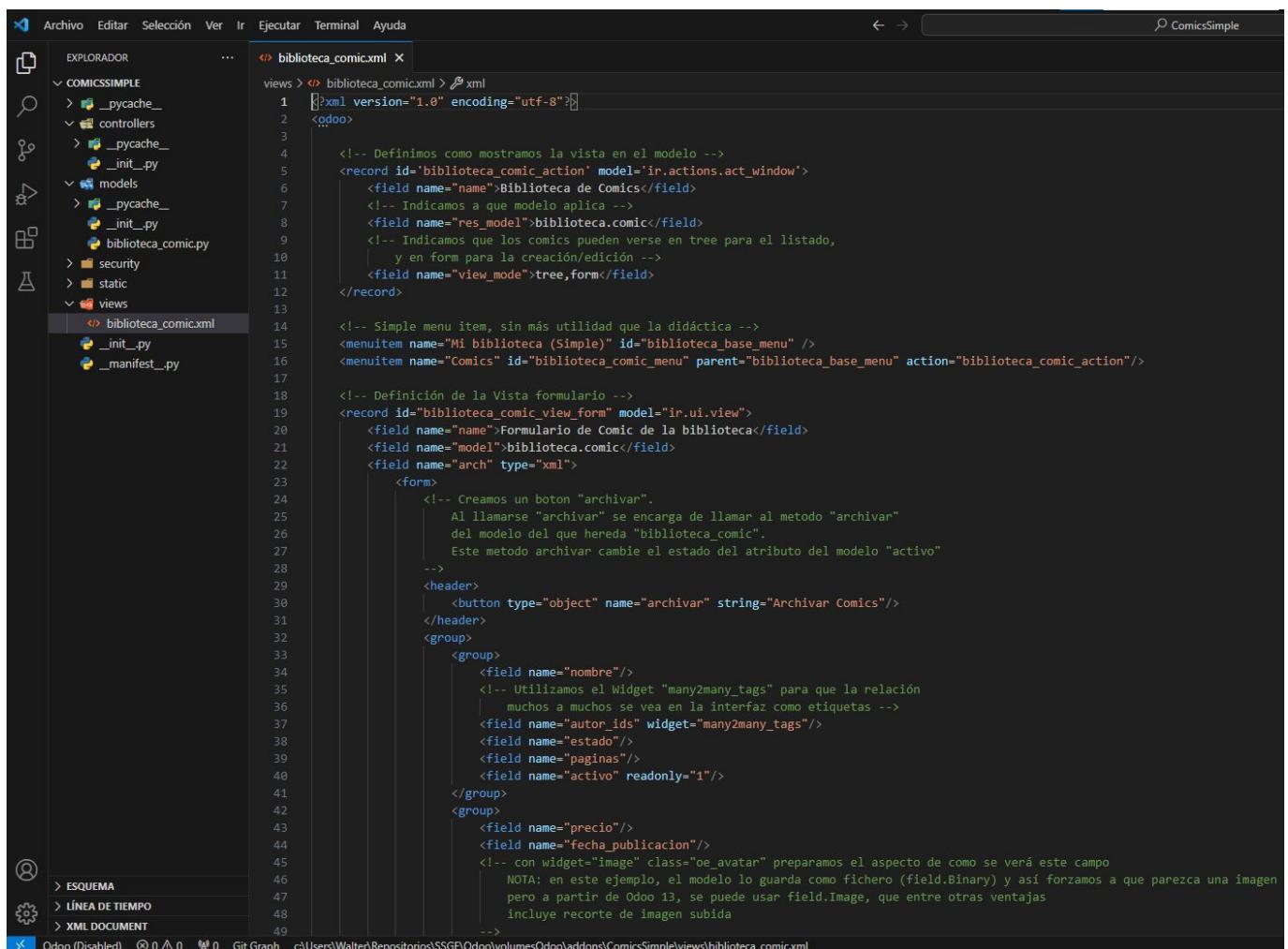
The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it shows the project structure for "COMICSSIMPLE". The "models" folder is selected, containing files like "__init__.py", "biblioteca_comic.py", and "views/biblioteca_comic.xml".
- Code Editor:** The main window displays the content of "biblioteca_comic.py".
- Code Content:**

```
models > biblioteca_comic.py > BaseArchive
75     #Esta colocado con fin didactico a false
76     company_dependent=False)
77
78     #Valoración lector, indicando como son los datos
79     valoracion_lector = fields.Float(
80         'Valoración media lectores',
81         digits=(14, 4), # Precision opcional (total, decimales),
82     )
83
84     autor_ids = fields.Many2many('res.partner', string='Autores')
85
86     #Constraints de SQL del modelo
87     #Util cuando la constraint se puede definir con sintaxis SQL
88     _sql_constraints = [
89         ('name_uniq', 'UNIQUE (nombre)', 'El titulo Comic debe ser único.'),
90         ('positive_page', 'CHECK(paginas>0)', 'El comic debe tener al menos una página')
91     ]
92
93     #Indicamos que esta funcion es una "Constraints" de ese atributos
94     #Dicho de otra forma, cada vez que se cambie ese atributo, se lanzara esta funcion
95     #Y si la funcion detecta un cambio inadecuado, cambiara una instruccion
96     #Util cuando la constraint no se puede definir con sintaxis SQL y debe indicar en una funcion
97     @api.constrains('fecha_publicacion')
98     def _check_release_date(self):
99         # Recorremos el modelo
100        for record in self:
101            #Comprobamos de cada registro, primero que haya una fecha_publicacion
102            #y tras ello, que la fecha no sea posterior a la actual.
103            if record.fecha_publicacion and record.fecha_publicacion > fields.Date.today():
104                #Si procede, lanzamos una excepcion
105                raise models.ValidationError('La fecha de lanzamiento debe ser anterior a la actual')
```

3. Actividad 2 (Ampliación)

3.2. Desarrollo



The screenshot shows the Odoo XML editor interface. The left sidebar displays the project structure for 'COMICSIMPLE'. The main area shows the XML code for the 'biblioteca_comic.xml' file.

```
<?xml version="1.0" encoding="utf-8"?>
<record id="biblioteca_comic_action" model="ir.actions.act_window">
    <field name="name">Biblioteca de Comics</field>
    <!-- Indicamos a que modelo aplica -->
    <field name="res_model">biblioteca.comic</field>
    <!-- Indicamos que los comics pueden verse en tree para el listado,
        y en form para la creación/edición -->
    <field name="view_mode">tree,form</field>
</record>

<!-- Simple menu item, sin más utilidad que la didáctica -->
<menuitem name="Mi biblioteca (Simple)" id="biblioteca_base_menu" />
<menuitem name="Comics" id="biblioteca_comic_menu" parent="biblioteca_base_menu" action="biblioteca_comic_action"/>

<!-- Definición de la Vista formulario -->
<record id="biblioteca_comic_view_form" model="ir.ui.view">
    <field name="name">Formulario de Comic de la biblioteca</field>
    <field name="model">biblioteca.comic</field>
    <field name="arch" type="xml">
        <form>
            <!-- Creamos un botón "archivar".
                Al llamarse "archivar" se encarga de llamar al método "archivar"
                del modelo del que hereda "biblioteca_comic".
                Este método archivar cambia el estado del atributo del modelo "activo"
            -->
            <header>
                <button type="object" name="archivar" string="Archivar Comics"/>
            </header>
            <group>
                <group>
                    <field name="nombre"/>
                    <!-- Utilizamos el Widget "many2many_tags" para que la relación
                        a muchos a muchos se vea en la interfaz como etiquetas -->
                    <field name="autor_ids" widget="many2many_tags"/>
                    <field name="estado"/>
                    <field name="paginas"/>
                    <field name="activo" readonly="1"/>
                </group>
                <group>
                    <field name="precio"/>
                    <field name="fecha_publicacion"/>
                    <!-- con widget="image" class="oe_avatar" preparamos el aspecto de como se verá este campo
                        NOTA: en este ejemplo, el modelo lo guarda como fichero (field.Binary) y así forzamos a que parezca una imagen
                        pero a partir de Odoo 13, se puede usar field.Image, que entre otras ventajas
                        incluye recorte de imagen subida
                </group>
            </group>
        </form>
    </field>
</record>
```



3. Actividad 2 (Ampliación)

3.2. Desarrollo

```
<!-- Definición de la vista Tree -->
<record id="biblioteca_comic_view_tree" model="ir.ui.view">
    <field name="name">Lista de Comics de la biblioteca</field>
    <field name="model">biblioteca.comic</field>
    <field name="arch" type="xml">
        <tree>
            <field name="nombre"/>
            <field name="fecha_publicacion"/>
            <field name="estado"/>
        </tree>
    </field>
</record>

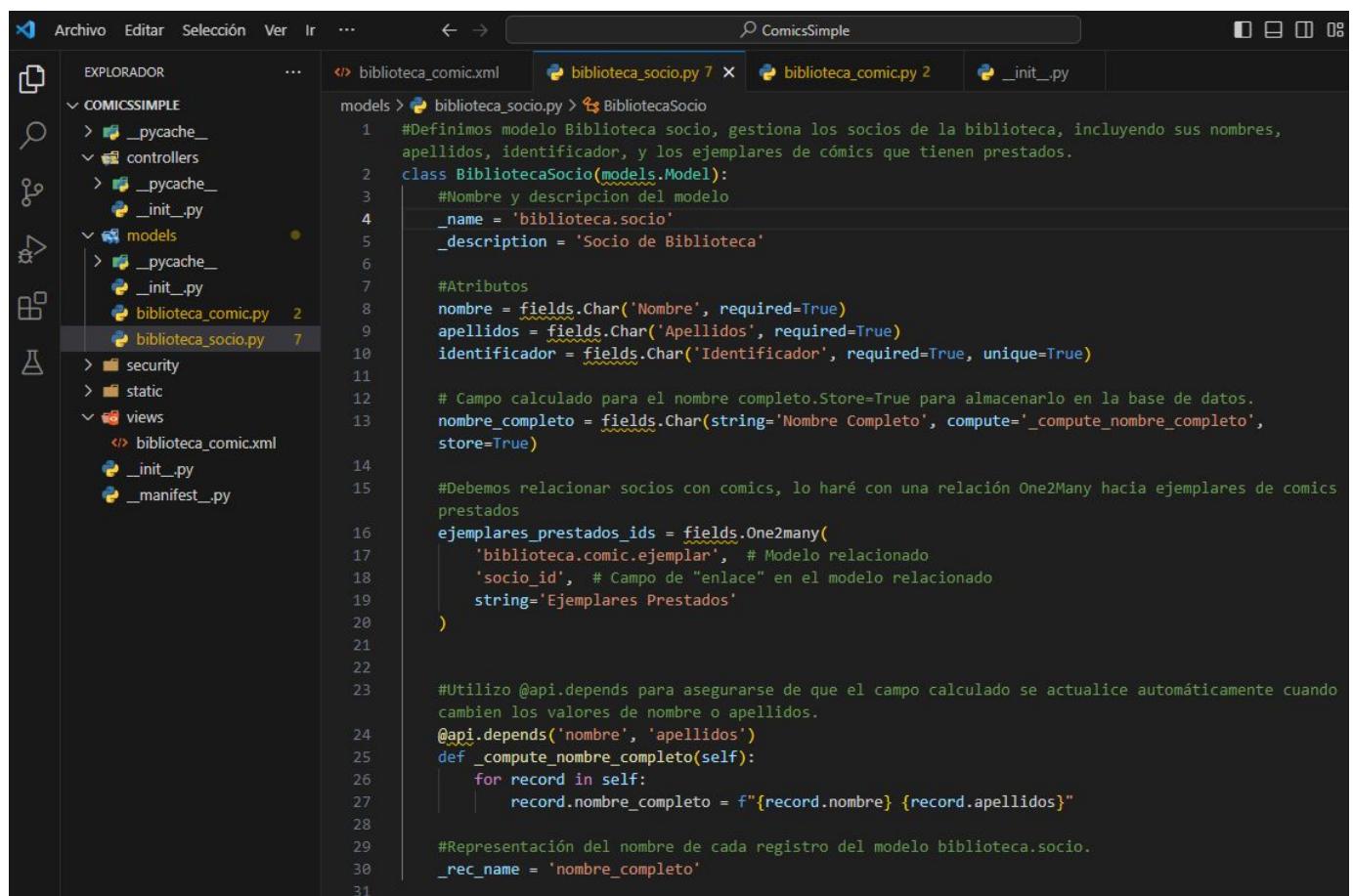
<!-- Definición de la vista busqueda-->
<record id="biblioteca_comic_view_search" model="ir.ui.view">
    <field name="name">Búsqueda de Comics en la biblioteca</field>
    <field name="model">biblioteca.comic</field>
    <field name="arch" type="xml">
        <search>
            <field name="nombre"/>
            <field name="autor_ids"/>
            <!-- Indicamos que para si filtramos por los del dominio "autor_ids=False" se muestre el texto "Sin autor"-->
            <filter string="Sin autor" name="sin_autor" domain="['autor_ids','=',False]"/>
        </search>
    </field>
</record>
```



3. Actividad 2 (Ampliación)

3.2. Desarrollo

Lo primero que haremos será crear un nuevo modelo para gestionar los socios.



The screenshot shows a code editor with a dark theme. On the left is a file explorer window titled 'EXPLORADOR' showing a project structure for 'COMICSSIMPLE'. The structure includes 'controllers', 'models' (which contains 'biblioteca_comic.py' and 'biblioteca_socio.py'), 'views', and XML files. The main editor window has tabs for 'biblioteca_comic.xml', 'biblioteca_socio.py 7', 'biblioteca_comic.py 2', and '_init_.py'. The 'biblioteca_socio.py' tab is active, displaying the following Python code:

```
#Definimos modelo Biblioteca socio, gestiona los socios de la biblioteca, incluyendo sus nombres, apellidos, identificador, y los ejemplares de cómics que tienen prestados.
class BibliotecaSocio(models.Model):
    #Nombre y descripción del modelo
    _name = 'biblioteca.socio'
    _description = 'Socio de Biblioteca'

    #Atributos
    nombre = fields.Char('Nombre', required=True)
    apellidos = fields.Char('Apellidos', required=True)
    identificador = fields.Char('Identificador', required=True, unique=True)

    # Campo calculado para el nombre completo. Store=True para almacenarlo en la base de datos.
    nombre_completo = fields.Char(string='Nombre Completo', compute='_compute_nombre_completo', store=True)

    #Debemos relacionar socios con comics, lo haré con una relación One2Many hacia ejemplares de comics prestados
    ejemplares_prestados_ids = fields.One2many(
        'biblioteca.comic.ejemplar', # Modelo relacionado
        'socio_id', # Campo de "enlace" en el modelo relacionado
        string='Ejemplares Prestados'
    )

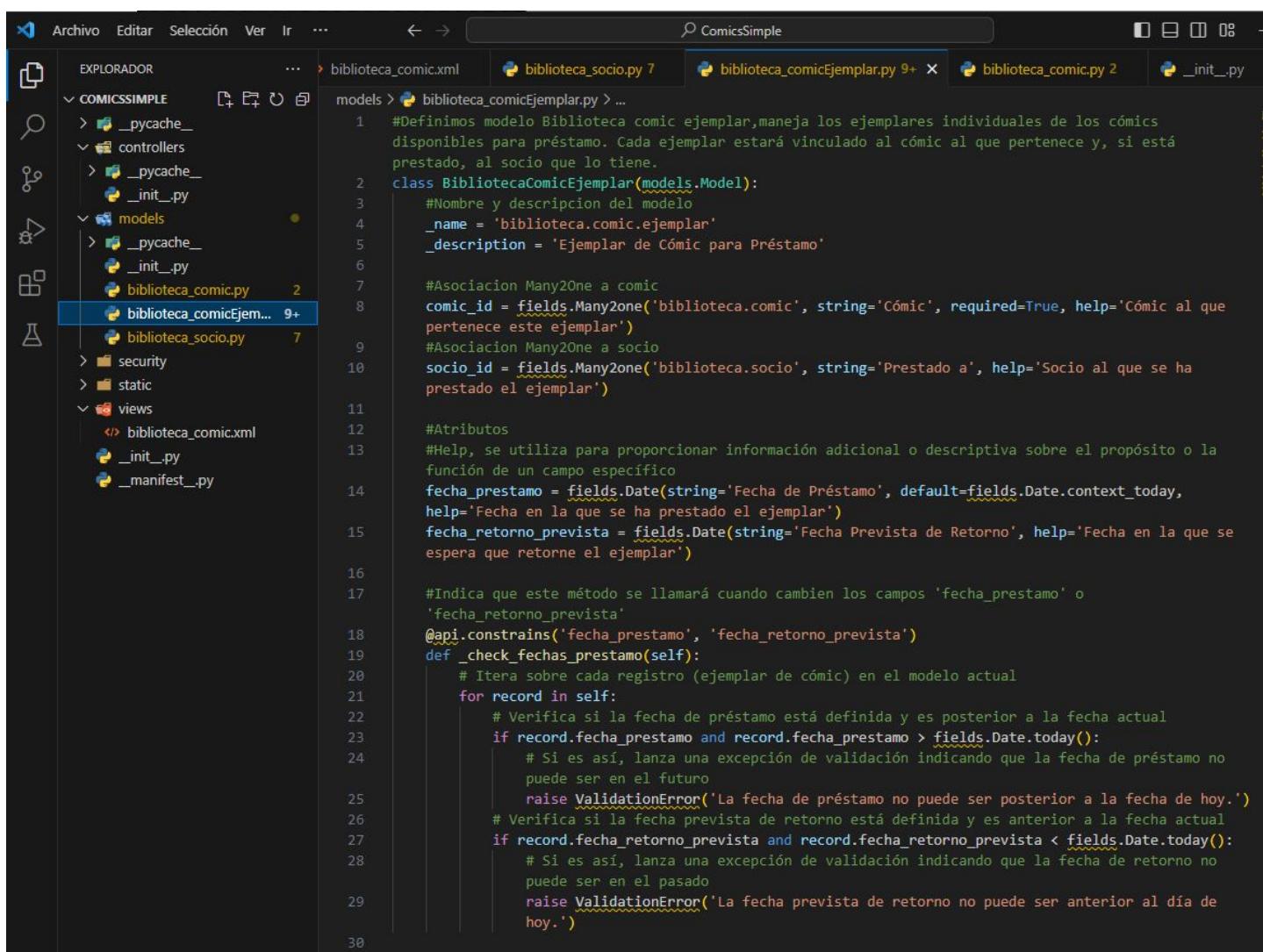
    #Utilizo @api.depends para asegurarse de que el campo calculado se actualice automáticamente cuando cambien los valores de nombre o apellidos.
    @api.depends('nombre', 'apellidos')
    def _compute_nombre_completo(self):
        for record in self:
            record.nombre_completo = f'{record.nombre} {record.apellidos}'

    #Representación del nombre de cada registro del modelo biblioteca.socio.
    _rec_name = 'nombre_completo'
```

3. Actividad 2 (Ampliación)

3.2. Desarrollo

Seguidamente crearemos otro modelo para gestionar los ejemplares de los cómics. Desarrollamos el modelo comicEjemplar, teniendo en cuenta las restricciones que tenemos que añadir para las fechas de los préstamos expuestas en el enunciado.



```
EXPLORADOR          ... > biblioteca_comic.xml  biblioteca_socio.py 7  biblioteca_comicEjemplar.py 9+  biblioteca_comic.py 2  _init_.py
COMICSSIMPLE
> __pycache__ 
controllers
> __pycache__ 
__init__.py
models
> __pycache__ 
__init__.py
biblioteca_comic.py 2
biblioteca_comicEjem... 9+
biblioteca_socio.py 7
security
static
views
< biblioteca_comic.xml
__init__.py
__manifest__.py

models > biblioteca_comicEjemplar.py > ...
1  #Definimos modelo Biblioteca comic ejemplar, maneja los ejemplares individuales de los cómics
2  disponibles para préstamo. Cada ejemplar estará vinculado al cómic al que pertenece y, si está
3  prestado, al socio que lo tiene.
4  class BibliotecaComicEjemplar(models.Model):
5      #Nombre y descripción del modelo
6      _name = 'biblioteca.comic.ejemplar'
7      _description = 'Ejemplar de Cómic para Préstamo'
8
9      #Asociación Many2One a comic
10     comic_id = fields.Many2one('biblioteca.comic', string='Cómic', required=True, help='Cómic al que
11     pertenece este ejemplar')
12     #Asociación Many2One a socio
13     socio_id = fields.Many2one('biblioteca.socio', string='Prestado a', help='Socio al que se ha
14     prestado el ejemplar')
15
16     #Atributos
17     #Help, se utiliza para proporcionar información adicional o descriptiva sobre el propósito o la
18     #función de un campo específico
19     fecha_prestamo = fields.Date(string='Fecha de Préstamo', default=fields.Date.context_today,
20                                   help='Fecha en la que se ha prestado el ejemplar')
21     fecha_retorno_prevista = fields.Date(string='Fecha Prevista de Retorno', help='Fecha en la que se
22     espera que retorne el ejemplar')
23
24     #Indica que este método se llamará cuando cambien los campos 'fecha_prestamo' o
25     'fecha_retorno_prevista'
26     @api.constrains('fecha_prestamo', 'fecha_retorno_prevista')
27     def _check_fechas_prestamo(self):
28         # Itera sobre cada registro (ejemplar de cómic) en el modelo actual
29         for record in self:
30             # Verifica si la fecha de préstamo está definida y es posterior a la fecha actual
            if record.fecha_prestamo and record.fecha_prestamo > fields.Date.today():
                # Si es así, lanza una excepción de validación indicando que la fecha de préstamo no
                # puede ser en el futuro
                raise ValidationError('La fecha de préstamo no puede ser posterior a la fecha de hoy.')
            # Verifica si la fecha prevista de retorno está definida y es anterior a la fecha actual
            if record.fecha_retorno_prevista and record.fecha_retorno_prevista < fields.Date.today():
                # Si es así, lanza una excepción de validación indicando que la fecha de retorno no
                # puede ser en el pasado
                raise ValidationError('La fecha prevista de retorno no puede ser anterior al día de
                hoy.')
```



3. Actividad 2 (Ampliación)

3.2. Desarrollo

A continuación debemos importar los modelos creados en el init.py

The screenshot shows a code editor window with the title "ComicsSimple". The left sidebar displays a file tree for a project named "COMICSIMPLE". The "models" directory contains an "__init__.py" file. The code in this file is:

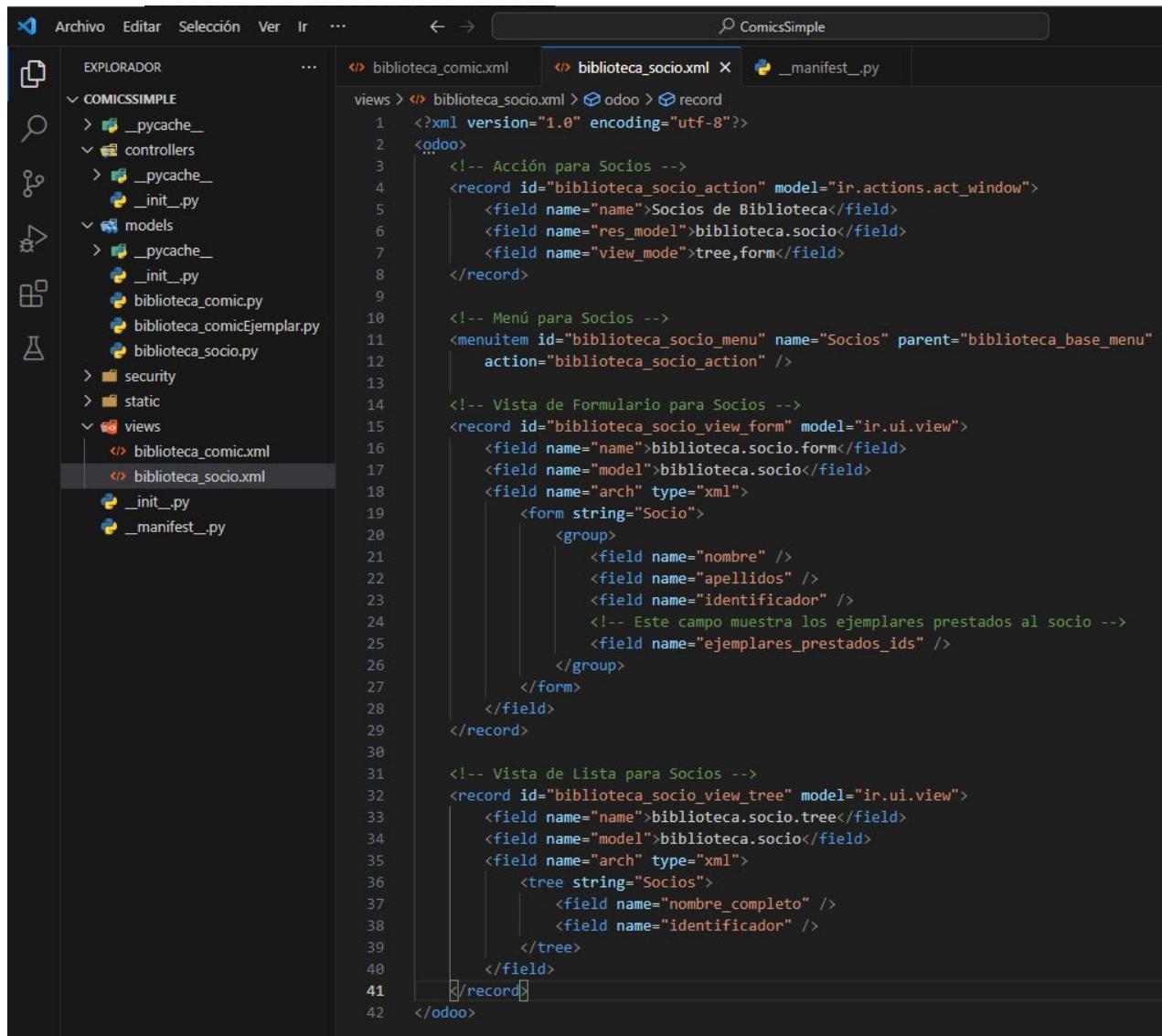
```
# -*- coding: utf-8 -*-
# Aquí indicamos que se cargara el fichero "biblioteca_comic.py"
# Si creamos mas modelos, deben importarse en este fichero
from . import biblioteca_comic
from . import biblioteca_socio
from . import biblioteca_comicEjemplar
```

3. Actividad 2 (Ampliación)

3.2. Desarrollo

Ahora necesitamos crear una vista para el modelo socio, y otra para el modelo comicEjemplar. Comenzaremos con la vista para socio.

Desarrollamos la vista de manera sencilla, creando una lista (árbol) y una vista específica para el formulario, también añadimos la acción para abrir esta vista y un menú item.



The screenshot shows a code editor with several tabs open. The main tab is 'biblioteca_socio.xml'. The left sidebar shows the project structure:

- EXPLORADOR**
 - COMICSSIMPLE
 - __pycache__
 - controllers
 - models
 - __pycache__
 - __init__.py
 - biblioteca_comic.py
 - biblioteca_comicEjemplar.py
 - biblioteca_socio.py
 - security
 - static
 - views
 - biblioteca_comic.xml
 - biblioteca_socio.xml**
 - __init__.py
 - __manifest__.py

The 'biblioteca_socio.xml' file contains the following XML code:

```

<?xml version="1.0" encoding="utf-8"?>
<odoor>
    <!-- Acción para Socios -->
    <record id="biblioteca_socio_action" model="ir.actions.act_window">
        <field name="name">Socios de Biblioteca</field>
        <field name="res_model">biblioteca.socio</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- Menú para Socios -->
    <menuitem id="biblioteca_socio_menu" name="Socios" parent="biblioteca_base_menu"
              action="biblioteca_socio_action" />

    <!-- Vista de Formulario para Socios -->
    <record id="biblioteca_socio_view_form" model="ir.ui.view">
        <field name="name">biblioteca.socio.form</field>
        <field name="model">biblioteca.socio</field>
        <field name="arch" type="xml">
            <form string="Socio">
                <group>
                    <field name="nombre" />
                    <field name="apellidos" />
                    <field name="identificador" />
                    <!-- Este campo muestra los ejemplares prestados al socio -->
                    <field name="ejemplares_prestados_ids" />
                </group>
            </form>
        </field>
    </record>

    <!-- Vista de Lista para Socios -->
    <record id="biblioteca_socio_view_tree" model="ir.ui.view">
        <field name="name">biblioteca.socio.tree</field>
        <field name="model">biblioteca.socio</field>
        <field name="arch" type="xml">
            <tree string="Socios">
                <field name="nombre_completo" />
                <field name="identificador" />
            </tree>
        </field>
    </record>
</odoor>

```

3. Actividad 2 (Ampliación)

3.2. Desarrollo

Ahora crearemos una vista para el modelo comicEjemplar, desarrollamos una vista similar a la anterior.

```

<?xml version="1.0" encoding="utf-8"?>
<odoor>
    <!-- Acción para Ejemplares de Cómics -->
    <record id="biblioteca_comic_ejemplar_action" model="ir.actions.act_window">
        <field name="name">Ejemplares de Cómics</field>
        <field name="res_model">biblioteca.comic.ejemplar</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- Menú para Ejemplares de Cómics -->
    <menuitem id="biblioteca_comic_ejemplar_menu" name="Ejemplares de Cómics"
              parent="biblioteca_base_menu" action="biblioteca_comic_ejemplar_action" />

    <!-- Vista de Formulario para Ejemplares de Cómics -->
    <record id="biblioteca_comic_ejemplar_view_form" model="ir.ui.view">
        <field name="name">biblioteca.comic.ejemplar.form</field>
        <field name="model">biblioteca.comic.ejemplar</field>
        <field name="arch" type="xml">
            <form string="Ejemplar de Cómic">
                <group>
                    <field name="comic_id" />
                    <field name="socio_id" />
                    <field name="fecha_prestamo" />
                    <field name="fecha_retorno_prevista" />
                </group>
            </form>
        </field>
    </record>

    <!-- Vista de Lista para Ejemplares de Cómics -->
    <record id="biblioteca_comic_ejemplar_view_tree" model="ir.ui.view">
        <field name="name">biblioteca.comic.ejemplar.tree</field>
        <field name="model">biblioteca.comic.ejemplar</field>
        <field name="arch" type="xml">
            <tree string="Ejemplares de Cómics">
                <field name="comic_id" />
                <field name="socio_id" />
                <field name="fecha_prestamo" />
                <field name="fecha_retorno_prevista" />
            </tree>
        </field>
    </record>
</odoor>

```



3. Actividad 2 (Ampliación)

3.2. Desarrollo

Cargamos las vistas en el 'data' del manifest.py.

```
EXPLORADOR          Archivo Editar Selección Ver Ir ...
COMICSSIMPLE
  > __pycache__      ...      <- > ComicsSimple
  controllers
    > __pycache__
    __init__.py
  models
    > __pycache__
    __init__.py
    biblioteca_comic.py
    biblioteca_comicEjemplar.py
    biblioteca_socio.py
  security
  static
  views
    biblioteca_comic.xml
    biblioteca_comicEjemplar.xml 1
    biblioteca_socio.xml
    __init__.py
    __manifest__.py

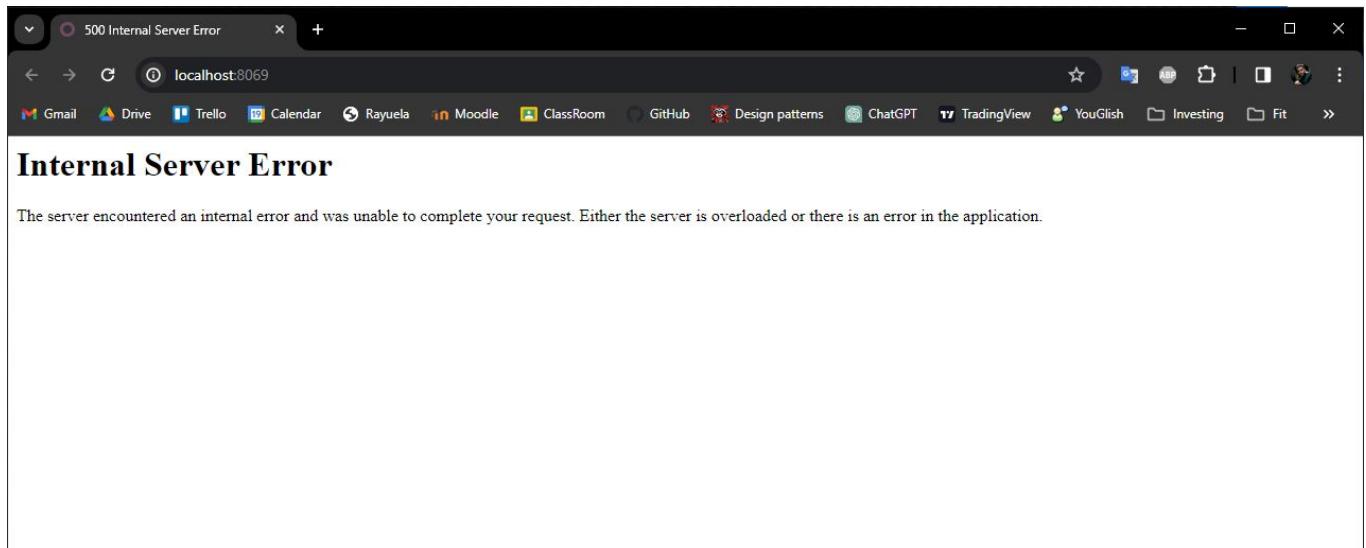
__manifest__.py
1  # -*- coding: utf-8 -*-
2  [
3      'name': "Biblioteca Comics Simple", # Titulo del módulo
4      'summary': "Gestionar comics :) (Version simple)", # Resumen de la funcionalidad
5      'description': """
6          Gestor de bibliotecas (Version Simple)
7          =====
8          """,
9
10     #Indicamos que es una aplicación
11     'application': True,
12     'author': "Sergi García",
13     'website': "http://apuntesfpinformatica.es",
14     'category': 'Tools',
15     'version': '0.1',
16     'depends': ['base'],
17
18     'data': [
19         #Estos dos primeros ficheros:
20         #1) El primero indica grupo de seguridad basado en rol
21         #2) El segundo indica la política de acceso del modelo
22         'security/groups.xml',
23         'security/ir.model.access.csv',
24         #Cargamos la vista de la biblioteca de comics
25         'views/biblioteca_comic.xml',
26         'views/biblioteca_socio.xml',
27         'views/biblioteca_comicEjemplar.xml'
28     ],
29 ]
30
```



3. Actividad 2 (Ampliación)

3.2. Desarrollo

Ahora solo quedaría probar el módulo, pero me encontré un problema a la hora de intentar iniciar el servidor Odoo en Docker. Cuando intentaba acceder a la web de Odoo me salía este error.

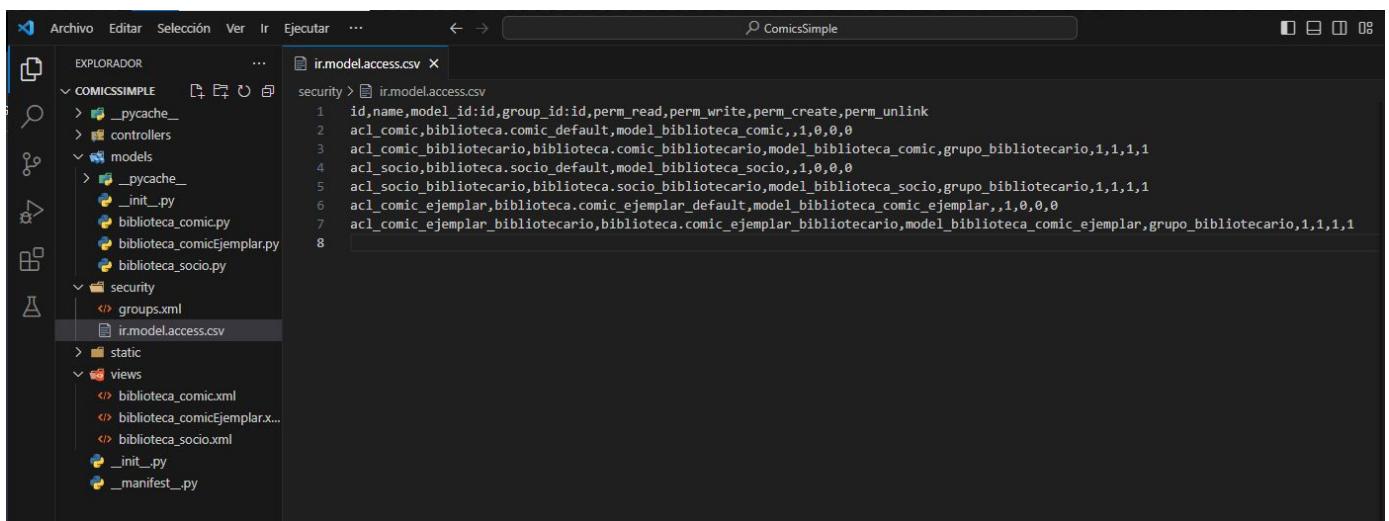




3. Actividad 2 (Ampliación)

3.2. Desarrollo

Lo primero que se me ocurrió fue actualizar el archivo `access.csv` ya que no había añadido los modelos nuevos.



```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
acl_comic,biblioteca.comic_default,model_biblioteca_comic,,1,0,0,0
acl_comic_bibliotecario,biblioteca.comic_bibliotecario,model_biblioteca_comic,grupo_bibliotecario,1,1,1,1
acl_socio,biblioteca.socio_default,model_biblioteca_socio,,1,0,0,0
acl_socio_bibliotecario,biblioteca.socio_bibliotecario,model_biblioteca_socio,grupo_bibliotecario,1,1,1,1
acl_comic_ejemplar,biblioteca.comic_ejemplar_default,model_biblioteca_comic_ejemplar,,1,0,0,0
acl_comic_ejemplar_bibliotecario,biblioteca.comic_ejemplar_bibliotecario,model_biblioteca_comic_ejemplar,grupo_bibliotecario,1,1,1,1
```



3. Actividad 2 (Ampliación)

3.2. Desarrollo

El problema persistía, por lo que me dispuse a buscar en el log del servidor web de Odoo en Docker, encontrando así el causante del problema. Como se observa en la imagen no encuentra ‘models’ porque no había realizado los import correspondientes en los modelos nuevos, y el vsCode no me saltaba el error.

Docker Scout		2024-02-06 10:00:05 File "<decorator-gen-16>", line 2, in new 2024-02-06 10:00:05 File "/usr/lib/python3/dist-packages/odoo/tools/func.py", line 87, in locked 2024-02-06 10:00:05 return func(inst, *args, **kwargs) 2024-02-06 10:00:05 File "/usr/lib/python3/dist-packages/odoo/modules/registry.py", line 90, in new 2024-02-06 10:00:05 odoo.modules.load_modules(registry, force_demo, status, update_module) 2024-02-06 10:00:05 File "/usr/lib/python3/dist-packages/odoo/modules/loading.py", line 484, in load_modules 2024-02-06 10:00:05 processed_modules += load_marked_modules(cr, graph, 2024-02-06 10:00:05 File "/usr/lib/python3/dist-packages/odoo/modules/loading.py", line 372, in load_marked_modules 2024-02-06 10:00:05 loaded, processed = load_module_graph(2024-02-06 10:00:05 File "/usr/lib/python3/dist-packages/odoo/modules/loading.py", line 188, in load_module_graph 2024-02-06 10:00:05 load_openerp_module(package.name) 2024-02-06 10:00:05 File "/usr/lib/python3/dist-packages/odoo/modules/module.py", line 471, in load_openerp_module 2024-02-06 10:00:05 __import__('odoo.addons.' + module_name) 2024-02-06 10:00:05 File "/mnt/extra-addons/ComicsSimple/__init__.py", line 4, in <module> 2024-02-06 10:00:05 from . import models 2024-02-06 10:00:05 File "/mnt/extra-addons/ComicsSimple/models/__init__.py", line 5, in <module> 2024-02-06 10:00:05 from . import biblioteca_socio 2024-02-06 10:00:05 File "/mnt/extra-addons/ComicsSimple/models/biblioteca_socio.py", line 2, in <module> 2024-02-06 10:00:05 class BibliotecaSocio(models.Model): 2024-02-06 10:00:05 NameError: name 'models' is not defined 2024-02-06 10:00:05 2024-02-06 09:00:05,852 1 INFO odoo werkzeug: 172.19.0.1 - - [06/Febr/2024 09:00:05] "GET /favicon.ico HTTP/1.1" 500 - 2024-02-06 10:00:05 11.0.003 0.044
Extensions	•	
+ Add Extensions		



3. Actividad 2 (Ampliación)

3.2. Desarrollo

Añado los import en los nuevos modelos y solucionamos el error.

```
models > biblioteca_comicEjemplar.py > BibliotecaComicEjemplar
1  from datetime import timedelta
2
3  from odoo import models, fields, api
4  from odoo.exceptions import ValidationError
5
6  # Definimos modelo Biblioteca comic ejemplar, maneja los ejemplares individuales de los cómics
7  class BibliotecaComicEjemplar(models.Model):
8      # Nombre y descripción del modelo
9      _name = 'biblioteca.comic.ejemplar'
10     _description = 'Ejemplar de Cómico para Préstamo'
11
12     # Asociación Many2One a comic
13     comic_id = fields.Many2one('biblioteca.comic', string='Cómico',
14                                required=True, help='Cómico al que pertenece este ejemplar')
15     # Asociación Many2One a socio
16     socio_id = fields.Many2one(
17         'biblioteca.socio', string='Prestado a', help='Socio al que se ha prestado el ejemplar')
18
19     # Atributos
20     # Help, se utiliza para proporcionar información adicional o descriptiva sobre el campo
21     fecha_prestamo = fields.Date(string='Fecha de Préstamo', default=fields.Date.context_today,
22                                   help='Fecha en la que se ha prestado el ejemplar')
```



3. Actividad 2 (Ampliación)

3.2. Desarrollo

A continuación muestro el resultado final.

The screenshot shows a web browser window for Odoo - Biblioteca de Comics. The URL is localhost:8069/web?action=356&model=biblioteca.comic&view_type=list&cids=1&menu_id=228. The page title is 'Mi biblioteca (Simple)'. The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and a user profile for Mitchell Admin (odoo). Below the title, there's a search bar with placeholder 'buscar...'. The main content area displays a table of comic books:

<input type="checkbox"/>	Título	Fecha publicación	Estado
<input type="checkbox"/>	Micky Maus	05/02/2024	Disponible
<input type="checkbox"/>	The Beano	02/02/2024	Disponible
<input type="checkbox"/>	SuperMan	09/11/2023	Perdido



3. Actividad 2 (Ampliación)

3.2. Desarrollo

A screenshot of a web browser displaying the Odoo application 'Socios de Biblioteca'. The URL is 'localhost:8069/web#action=359&model=biblioteca.socio&view_type=list&cid=1&menu_id=228'. The page has a purple header with tabs for 'Mi biblioteca (Simple)', 'Comics', 'Socios', and 'Ejemplares de Cómics'. On the right, there are notifications for 5 messages, 10 unread emails, and a user 'Mitchell Admin (odoo)'. The main content area shows a table with two columns: 'Nombre Completo' and 'Identificador'. There are three rows of data:

Nombre Completo	Identificador
Walter Martín Lopes	socio1
Nacho Profe	socio2

A 'NUEVO' button is visible at the top left of the table area.



3. Actividad 2 (Ampliación)

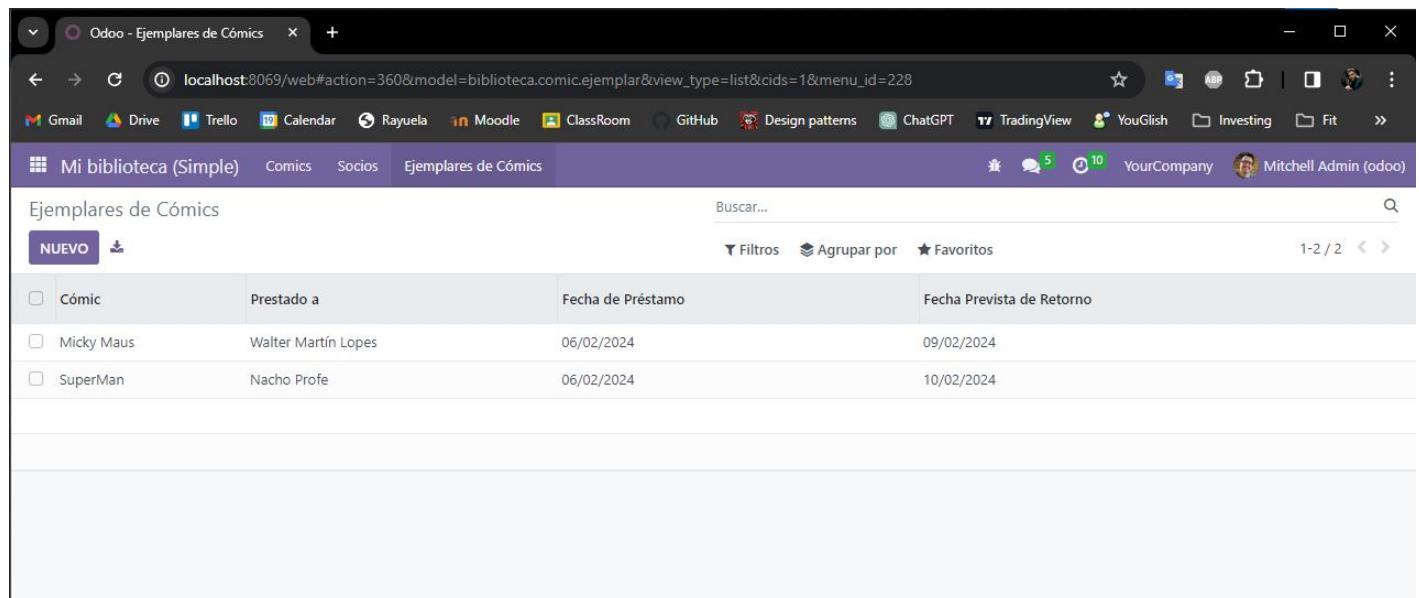
3.2. Desarrollo

Comprobamos que funcionan las restricciones de fechas que hemos controlado provocando que salten las validaciones.

The screenshot shows a web browser window for Odoo at the URL `localhost:8069/web#cid=1&menu_id=228&action=359&model=biblioteca.socio&view_type=form`. The main page displays a form for a library member ('Socios de Biblioteca') with fields for Nombre, Apellidos, Identificador, and Ejemplares Prestados. A modal dialog box is open, titled 'Error de validación', with the message 'La fecha de préstamo no puede ser posterior a la fecha de hoy.' (The loan date cannot be later than today's date). Below the message is a purple 'ACEPTAR' button. In the background, the 'Ejemplares Prestados' section shows a table with one item: 'Micky Maus' by 'Cómic' was borrowed on '09/02/2024' and is due to be returned on '05/02/2024'. There is also a link to 'Añadir línea' (Add line). At the bottom of the modal are buttons for 'GUARDAR Y CERRAR' (Save and Close) and 'DESCARTAR' (Discard).

3. Actividad 2 (Ampliación)

3.2. Desarrollo



The screenshot shows a web browser window for Odoo. The title bar says "Odoo - Ejemplares de Cómics". The address bar shows the URL "localhost:8069/web?action=360&model=biblioteca.comic.ejemplar&view_type=list&cids=1&menu_id=228". The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and a user profile for Mitchell Admin (odoo). The main menu bar has items for Mi biblioteca (Simple), Comics, Socios, and Ejemplares de Cómics. The current view is "Ejemplares de Cómics". The page displays a list of comic copies with columns for "Cómic", "Prestado a", "Fecha de Préstamo", and "Fecha Prevista de Retorno". There are two entries: one for "Micky Maus" lent to "Walter Martín Lopes" on "06/02/2024" with a return date of "09/02/2024", and another for "SuperMan" lent to "Nacho Profe" on "06/02/2024" with a return date of "10/02/2024". A "NUEVO" button is visible at the top left of the list.

Cómic	Prestado a	Fecha de Préstamo	Fecha Prevista de Retorno
Micky Maus	Walter Martín Lopes	06/02/2024	09/02/2024
SuperMan	Nacho Profe	06/02/2024	10/02/2024

Actividad 3 (Desarrollo)





4. Actividad 3 (Desarrollo)

4.1. Enunciado

Crea un módulo para gestionar los pacientes y médicos de un hospital.

Para cada paciente, tendremos un modelo con los siguientes datos:

- Nombre y apellidos del paciente.
- Síntomas.

Para cada médico, tendremos un modelo con los siguientes datos:

- Nombre y apellidos del médico.
- Número de colegiado.

Para cada vez que un médico atiende a un paciente, tendremos un modelo indicando el diagnóstico. Un paciente puede haber sido atendido por diferentes médicos y un médico puede haber atendido a diferentes pacientes. Implementa los modelos y las vistas que creas adecuados para los tres modelos.

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Comenzamos creando la estructura del módulo partiendo del módulo comics.

The screenshot shows a Python IDE interface with the following details:

- File Menu:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, ...
- Title Bar:** GestionHospital
- Explorer View (Left):** Shows the directory structure of the module:
 - GESTIONHOSPITAL
 - controllers
 - models
 - __init__.py
 - hospital_paciente.py
 - security
 - ir.model.access.csv
 - static
 - views
 - hospital_paciente.xml
 - __init__.py
 - __manifest__.py
- Code Editor (Right):** Displays the content of the __manifest__.py file.

```
# -*- coding: utf-8 -*-
{
    'name': "Gestión Hospital", # Titulo del módulo
    'summary': "Gestionar un hospital", # Resumen de la funcionalidad
    'description': """
Gestor de hospital (Version Simple)
=====
""",
    'application': True,
    'author': "Walter Martín Lopes",
    'website': "http://apuntesfpinformatica.es",
    'category': 'Tools',
    'version': '0.1',
    'depends': ['base'],
    'data': [
        'security/ir.model.access.csv',
        #Cargamos las vistas
        'views/hospital_paciente.xml'
    ],
}
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Creamos un modelo sencillo para gestionar los pacientes.

```
EXPLORADOR          GESTIONHOSPITAL
                      > controllers
                      > security
                      > static
                      > views
                      > hospital_paciente.xml
                      > __init__.py
                      > __manifest__.py

models > hospital_paciente.py > HospitalPaciente
models > hospital_paciente.xml > hospital_paciente.py 2 x

# -*- coding: utf-8 -*-
from datetime import timedelta
from odoo import models, fields, api
from odoo.exceptions import ValidationError

#Definimos modelo HospitalPaciente
class HospitalPaciente(models.Model):
    _name = 'hospital.paciente' # Define el nombre técnico del modelo
    _description = 'Paciente' # Descripción
    _rec_name = 'nombre' # Especifica el campo a usar como representación de nombre del registro

    nombre = fields.Char(string='Nombre y Apellidos', required=True, index=True) # Campo para el nombre, obligatorio e indexado para optimizar búsquedas
    sintomas = fields.Text(string='Síntomas') # Campo de texto para los síntomas, permite texto largo

    # Restricción de SQL para garantizar nombres únicos
    _sql_constraints = [
        ('nombre_uniq', 'UNIQUE(nombre)', 'El nombre y apellidos del paciente deben ser únicos.'),
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

A continuación desarrollamos la correspondiente vista, muy similar a las creadas en módulos anteriores.

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Vista de árbol/lista para los pacientes -->
    <record id="view_hospital_paciente_tree" model="ir.ui.view">
        <field name="name">hospital.paciente.tree</field>
        <field name="model">hospital.paciente</field>
        <field name="arch" type="xml">
            <tree string="Pacientes">
                <field name="nombre" />
                <field name="sintomas" />
            </tree>
        </field>
    </record>

    <!-- Vista de formulario para los pacientes -->
    <record id="view_hospital_paciente_form" model="ir.ui.view">
        <field name="name">hospital.paciente.form</field>
        <field name="model">hospital.paciente</field>
        <field name="arch" type="xml">
            <form string="Paciente">
                <sheet>
                    <group>
                        <field name="nombre" />
                        <field name="sintomas" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- Acción para abrir la vista de lista/formulario -->
    <record id="action_hospital_pacientes" model="ir.actions.act_window">
        <field name="name">Pacientes</field>
        <field name="res_model">hospital.paciente</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- Menú para acceder a los pacientes -->
    <menutem id="hospital_menu" name="Hospital" />
    <menutem id="hospital_paciente_menu" name="Pacientes" parent="hospital_menu"
        action="action_hospital_pacientes" />
</odoo>
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Creamos un modelo para gestionar los médicos.

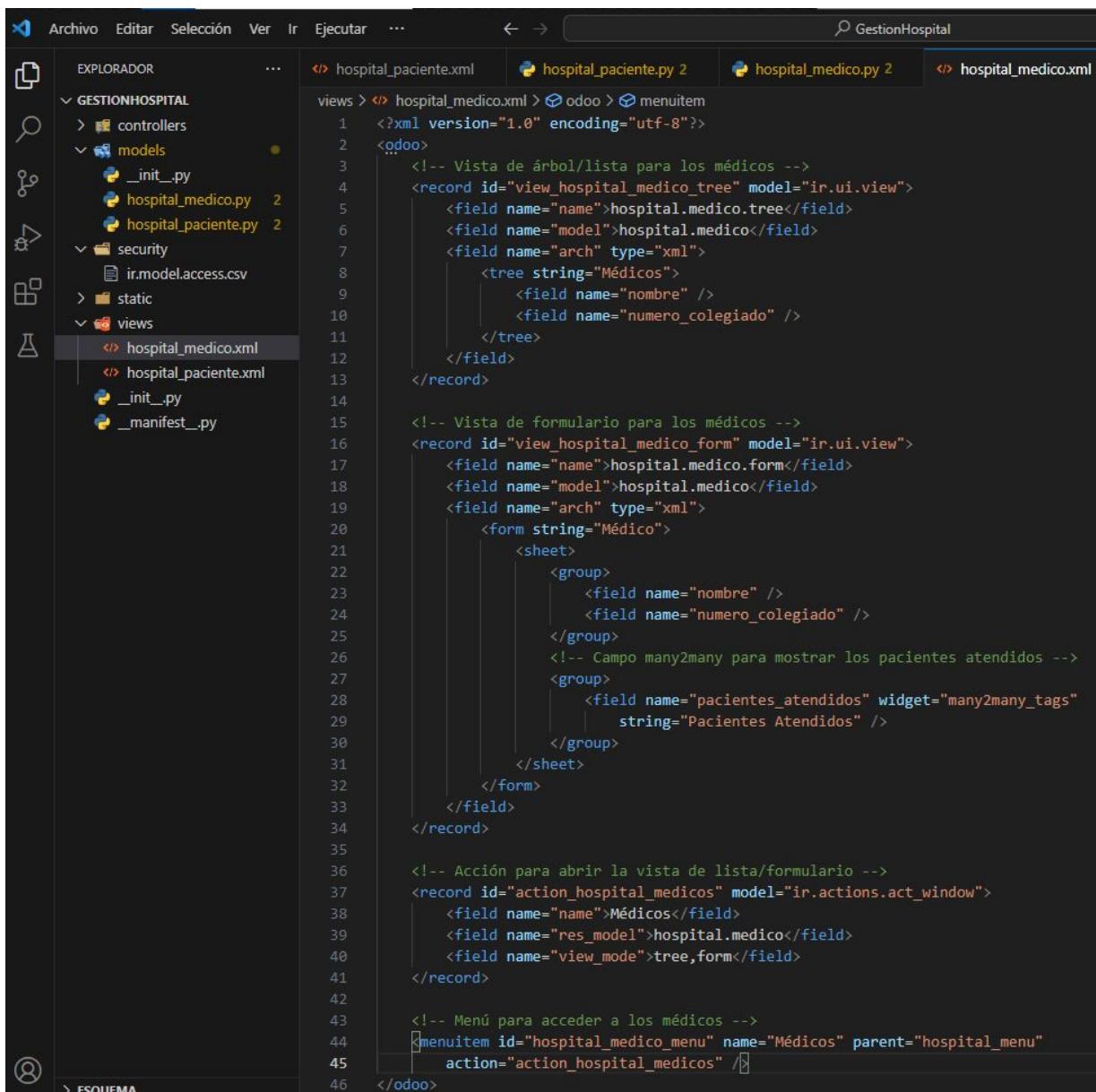
```
EXPLORADOR Archivo Editar Selección Ver Ir Ejecutar ... GestionHospital hospital_paciente.xml hospital_paciente.py 2 hospital_medico.py 2 _init_.py
models > hospital_medico.py > HospitalMedico
1 # -*- coding: utf-8 -*-
2 from datetime import timedelta
3
4 from odoo import models, fields, api
5 from odoo.exceptions import ValidationError
6
7 #Definimos modelo HospitalMedico
8 class HospitalMedico(models.Model):
9     _name = 'hospital.medico' # Define el nombre técnico del modelo
10    _description = 'Médico' # Descripción
11    _rec_name = 'nombre' # Especifica el campo a usar como representación de nombre del registro
12
13    nombre = fields.Char(string='Nombre y Apellidos', required=True, index=True) # Campo para el nombre, obligatorio e indexado para optimizar búsquedas
14    numero_colegiado = fields.Char(string='Número de Colegiado', required=True) # Campo para el número de colegiado
15
16    pacientes_atendidos = fields.Many2many('hospital.paciente', string='Pacientes Atendidos', relation='hospital_medico_paciente_rel',
17                                              column1='medico_id', column2='paciente_id', help='Pacientes atendidos por este médico') # Campo para relacionar médicos con pacientes
18
19    _sql_constraints = [
20        ('numero_colegiado_uniq', 'UNIQUE(numero_colegiado)', 'El número de colegiado debe ser único.'), # Restricción de SQL para garantizar números de colegiado únicos
21    ]
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Desarrollamos una vista sencilla para el modelo médico.



The screenshot shows the Odoo IDE interface with the following details:

- Toolbar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, ...
- Search Bar:** GestionHospital
- Left Sidebar (Explorador):**
 - GESTIONHOSPITAL
 - controllers
 - models
 - _init_.py
 - hospital_medico.py
 - hospital_paciente.py
 - security
 - static
 - views
 - hospital_medico.xml
 - hospital_paciente.xml
 - _init_.py
 - _manifest_.py
- Central Area:** Content pane displaying the XML code for `hospital_medico.xml`. The code defines views for the 'hospital_medico' model, including a tree view for 'Médicos' and a form view for 'Médico'. It also defines an action to open the window and a menu item for the 'Médicos' menu.

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Vista de árbol/lista para los médicos -->
    <record id="view_hospital_medico_tree" model="ir.ui.view">
        <field name="name">hospital.medico.tree</field>
        <field name="model">hospital.medico</field>
        <field name="arch" type="xml">
            <tree string="Médicos">
                <field name="nombre" />
                <field name="numero_colegiado" />
            </tree>
        </field>
    </record>

    <!-- Vista de formulario para los médicos -->
    <record id="view_hospital_medico_form" model="ir.ui.view">
        <field name="name">hospital.medico.form</field>
        <field name="model">hospital.medico</field>
        <field name="arch" type="xml">
            <form string="Médico">
                <sheet>
                    <group>
                        <field name="nombre" />
                        <field name="numero_colegiado" />
                    </group>
                    <!-- Campo many2many para mostrar los pacientes atendidos -->
                    <group>
                        <field name="pacientes_atendidos" widget="many2many_tags" string="Pacientes Atendidos" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- Acción para abrir la vista de lista/formulario -->
    <record id="action_hospital_medicos" model="ir.actions.act_window">
        <field name="name">Médicos</field>
        <field name="res_model">hospital.medico</field>
        <field name="view_mode">tree,form</field>
    </record>

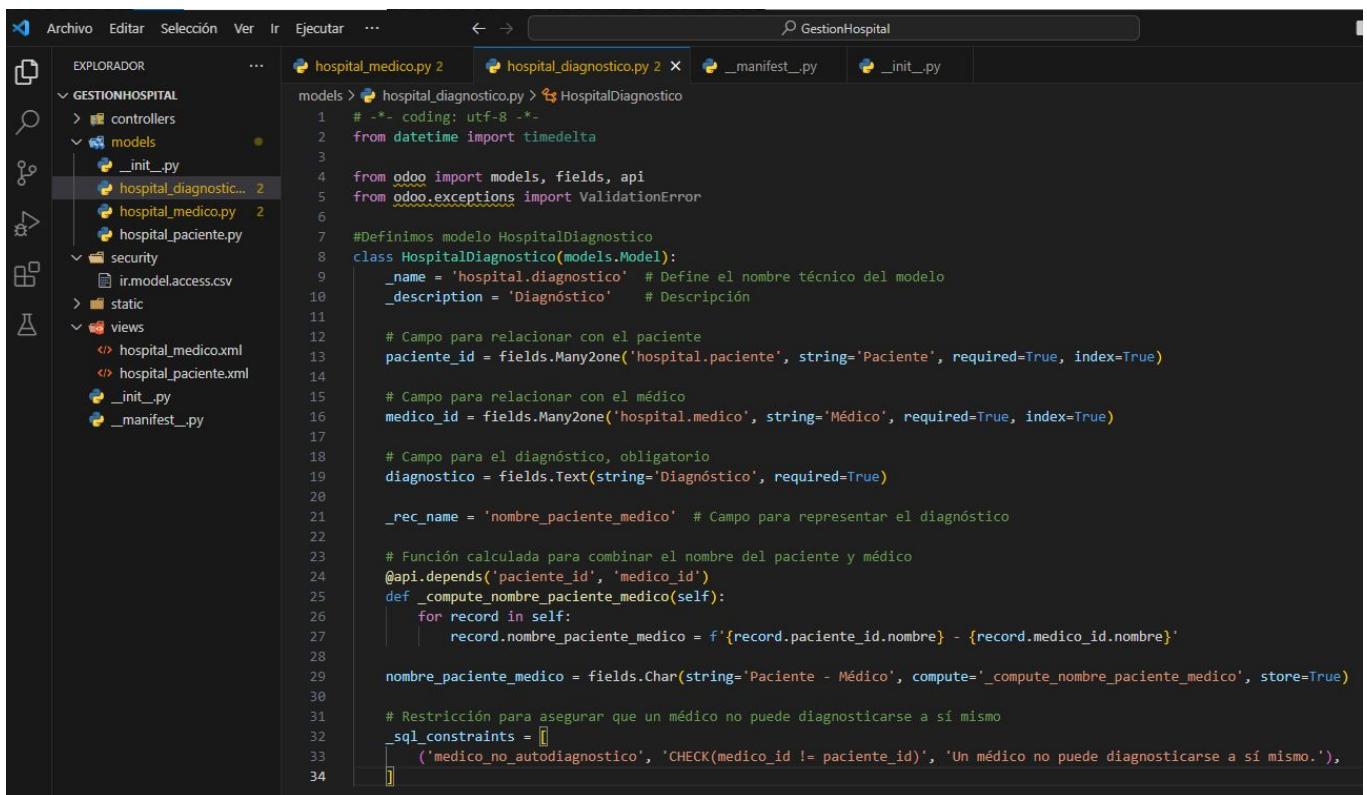
    <!-- Menú para acceder a los médicos -->
    <menuitem id="hospital_medico_menu" name="Médicos" parent="hospital_menu" action="action_hospital_medicos" />
</odoo>
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Creamos el modelo diagnóstico.



```
# -*- coding: utf-8 -*-
from datetime import timedelta
from odoo import models, fields, api
from odoo.exceptions import ValidationError

#Definimos modelo HospitalDiagnóstico
class HospitalDiagnóstico(models.Model):
    _name = 'hospital.diagnóstico' # Define el nombre técnico del modelo
    _description = 'Diagnóstico' # Descripción

    # Campo para relacionar con el paciente
    paciente_id = fields.Many2one('hospital.paciente', string='Paciente', required=True, index=True)

    # Campo para relacionar con el médico
    medico_id = fields.Many2one('hospital.medico', string='Médico', required=True, index=True)

    # Campo para el diagnóstico, obligatorio
    diagnostico = fields.Text(string='Diagnóstico', required=True)

    _rec_name = 'nombre_paciente_medico' # Campo para representar el diagnóstico

    # Función calculada para combinar el nombre del paciente y médico
    @api.depends('paciente_id', 'medico_id')
    def _compute_nombre_paciente_medico(self):
        for record in self:
            record.nombre_paciente_medico = f'{record.paciente_id.nombre} - {record.medico_id.nombre}'

    nombre_paciente_medico = fields.Char(string='Paciente - Médico', compute='_compute_nombre_paciente_medico', store=True)

    # Restricción para asegurar que un médico no puede diagnosticarse a sí mismo
    _sql_constraints = [
        ('medico_no_autodiagnóstico', 'CHECK(medico_id != paciente_id)', 'Un médico no puede diagnosticarse a sí mismo.')
    ]
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Desarrollamos una vista para el correspondiente módulo.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar labeled 'EXPLORADOR' containing a tree view of a module named 'GESTIONHOSPITAL'. The tree includes nodes for controllers, models (with files like __init__.py, hospital_diagnostico.py, hospital_medico.py, and hospital_paciente.py), security, static files, and views (with files like hospital_diagnostico.xml, hospital_medico.xml, and hospital_paciente.xml). The main pane displays the XML code for 'hospital_diagnostico.xml'. The code defines three records: one for a tree view ('view_hospital_diagnostico_tree') and two for form views ('view_hospital_diagnostico_form'). It also defines an action ('action_hospital_diagnosticos') and a menu item ('hospital_diagnostico_menu'). The XML uses Odoo's schema, including fields for 'name', 'model', 'arch', and 'view_mode'.

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Vista de árbol/lista para los diagnósticos -->
    <record id="view_hospital_diagnostico_tree" model="ir.ui.view">
        <field name="name">hospital.diagnostico.tree</field>
        <field name="model">hospital.diagnostico</field>
        <field name="arch" type="xml">
            <tree string="Diagnósticos">
                <!-- Mostrar el nombre del paciente y el médico calculado -->
                <field name="nombre_paciente_medico" />
                <field name="diagnostico" />
            </tree>
        </field>
    </record>

    <!-- Vista de formulario para los diagnósticos -->
    <record id="view_hospital_diagnostico_form" model="ir.ui.view">
        <field name="name">hospital.diagnostico.form</field>
        <field name="model">hospital.diagnostico</field>
        <field name="arch" type="xml">
            <form string="Diagnóstico">
                <sheet>
                    <group>
                        <field name="nombre_paciente_medico" readonly="1" />
                        <field name="diagnostico" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- Acción para abrir la vista de lista/formulario -->
    <record id="action_hospital_diagnosticos" model="ir.actions.act_window">
        <field name="name">Diagnósticos</field>
        <field name="res_model">hospital.diagnostico</field>
        <field name="view_mode">tree,form</field>
    </record>

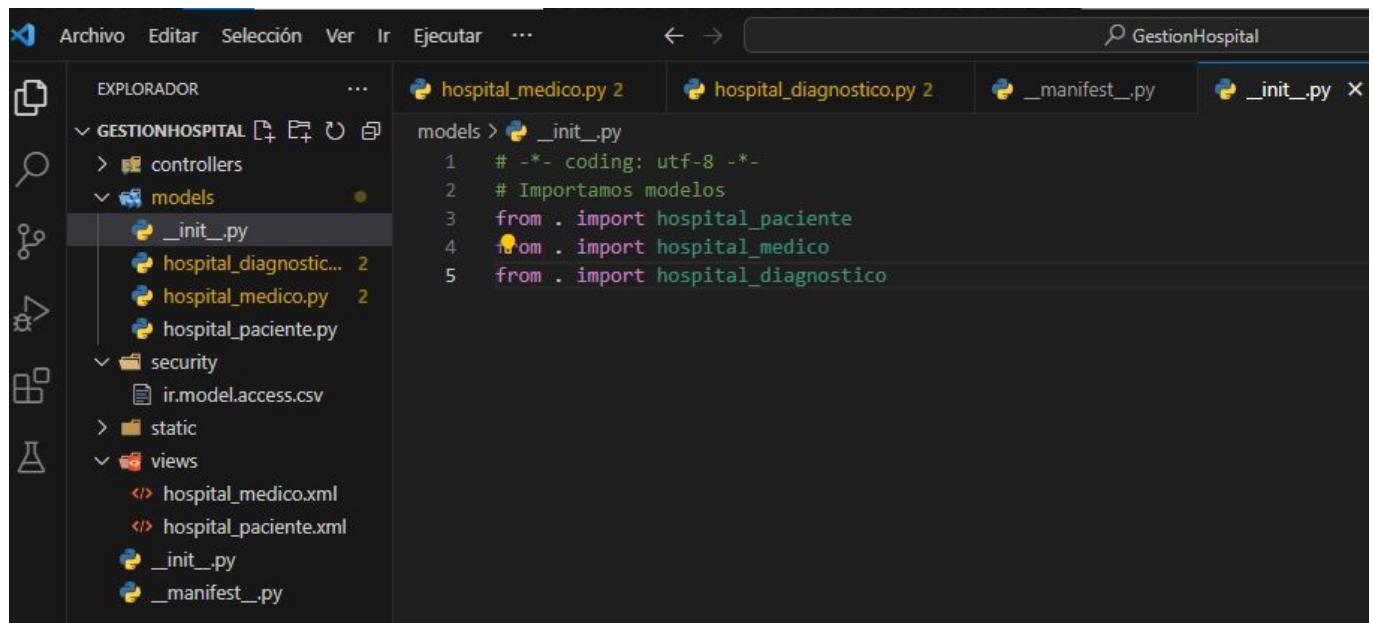
    <!-- Menú para acceder a los diagnósticos -->
    <menuitem id="hospital_diagnostico_menu" name="Diagnósticos" parent="hospital_menu"
              action="action_hospital_diagnosticos" />
</odoo>
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

No hay que olvidar importar todos los modelos creados en el `init.py`.



```
# -*- coding: utf-8 -*-
# Importamos modelos
from . import hospital_paciente
from . import hospital_medico
from . import hospital_diagnostico
```



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

También tenemos que cargar todas las vistas en el 'data' del manifest.py.

```
# -*- coding: utf-8 -*-
{
    'name': "Gestión Hospital", # Título del módulo
    'summary': "Gestionar un hospital", # Resumen de la funcionalidad
    'description': """",
    'version': '0.1',
    'depends': ['base'],
    'data': [
        'security/ir.model.access.csv',
        #Cargamos las vistas
        'views/hospital_paciente.xml',
        'views/hospital_medico.xml',
        'views/hospital_diagnostico.xml'
    ],
}
```


4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Antes de probar el módulo debemos actualizar el access.csv.

```
security > ir.model.access.csv
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 acl_paciente,hospital_paciente,model_hospital_paciente,,1,1,1,1
3 acl_medico,hospital_medico,model_hospital_medico,,1,1,1,1
4 acl_diagnostico,hospital_diagnostico,model_hospital_diagnostico,,1,1,1,1
5
```


4. Actividad 3 (Desarrollo)

4.1. Desarrollo

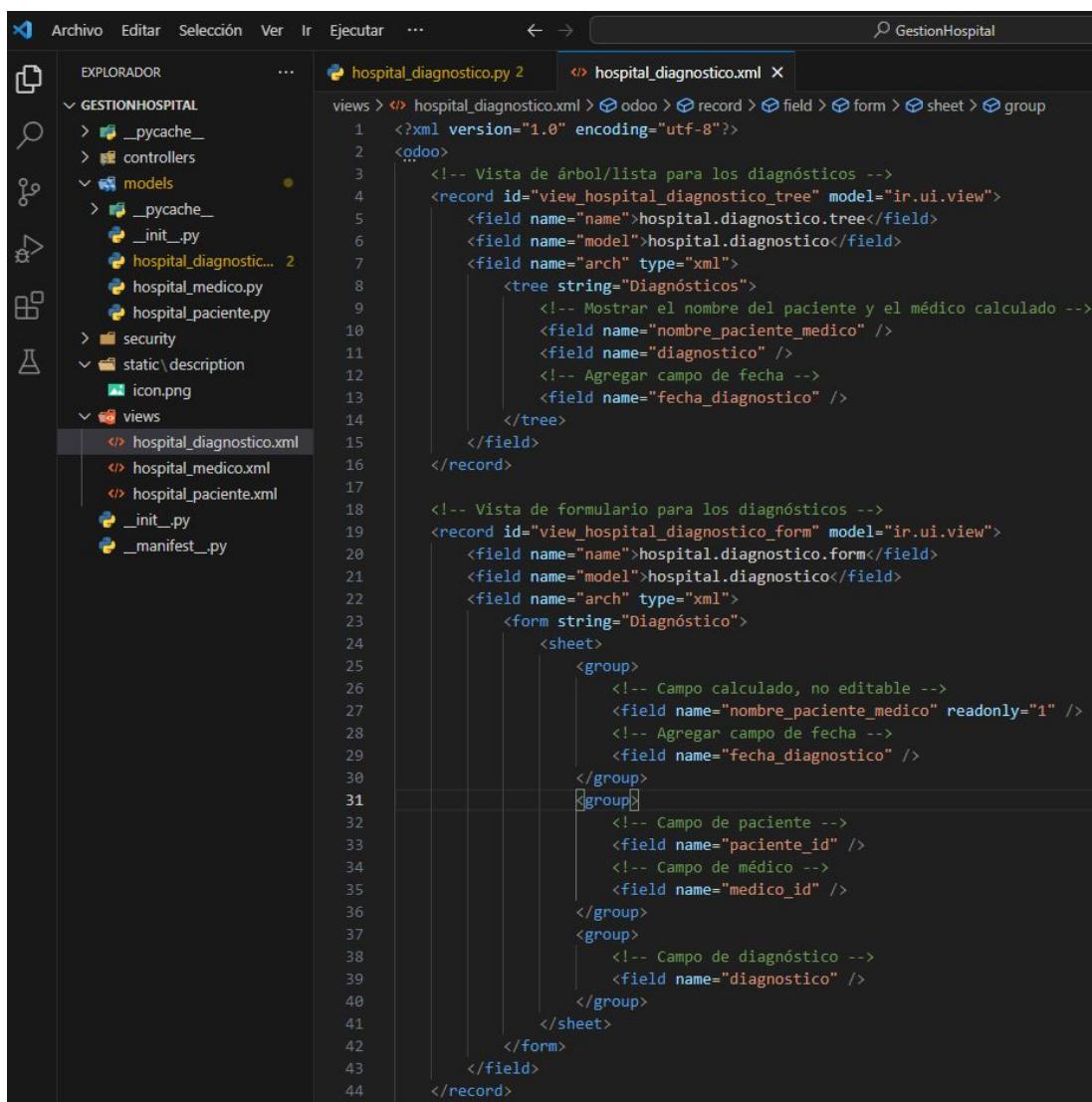
Antes de probar el módulo vamos a implementar una vista sencilla para visualizar los diagnósticos en forma de calendario. Para ello necesitamos modificar el modelo diagnóstico y añadir el campo fecha.

```
EXPLORADOR GESTIONHOSPITAL
models > hospital_diagnostico.py 2 x
models > hospital_diagnostico.py > HospitalDiagnostico
1  # -*- coding: utf-8 -*-
2  from datetime import timedelta
3
4  from odoo import models, fields, api
5  from odoo.exceptions import ValidationError
6
7  #Definimos modelo HospitalDiagnostico
8  class HospitalDiagnostico(models.Model):
9      _name = 'hospital.diagnostico' # Define el nombre técnico del modelo
10     _description = 'Diagnóstico' # Descripción
11
12     # Campo para relacionar con el paciente
13     paciente_id = fields.Many2one('hospital.paciente', string='Paciente', required=True, index=True)
14
15     # Campo para relacionar con el médico
16     medico_id = fields.Many2one('hospital.medico', string='Médico', required=True, index=True)
17
18     # Campo para el diagnóstico, obligatorio
19     diagnostico = fields.Text(string='Diagnóstico', required=True)
20
21     # Campo de fecha
22     fecha_diagnostico = fields.Date(string='Fecha del Diagnóstico', required=True, default=fields.Date.today())
23
24     _rec_name = 'nombre_paciente_medico' # Campo para representar el diagnóstico
25
26     # Función calculada para combinar el nombre del paciente y médico
27     @api.depends('paciente_id', 'medico_id')
28     def _compute_nombre_paciente_medico(self):
29         for record in self:
30             record.nombre_paciente_medico = f'{record.paciente_id.nombre} - {record.medico_id.nombre}'
31
32     nombre_paciente_medico = fields.Char(string='Paciente - Médico', compute='_compute_nombre_paciente_medico', store=True)
33
34     # Restricción para asegurar que un médico no puede diagnosticarse a sí mismo
35     _sql_constraints = [
36         ('medico_no_autodiagnostico', 'CHECK(medico_id != paciente_id)', 'Un médico no puede diagnosticarse a sí mismo.')
37     ]
```

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Acto seguido debemos actualizar también la vista `hospital_diagnostico` añadiendo el campo `fecha` que hemos añadido anteriormente en el modelo en el formulario de la vista, además añadimos los campos `paciente_id` y `medico_id` que inicialmente se me había olvidado poner en el formulario.



The screenshot shows a Python IDE interface with the following details:

- File Explorer (Left):** Shows the project structure under "GESTIONHOSPITAL".
 - `views`: Contains `hospital_diagnostico.xml`, `hospital_medico.xml`, `hospital_paciente.xml`, `_init_.py`, and `__manifest__.py`.
 - `models`: Contains `_init_.py`, `hospital_diagnostico.py`, `hospital_medico.py`, and `hospital_paciente.py`.
 - `static`: Contains `description` and `icon.png`.
- Code Editor (Right):** Displays the XML code for the `hospital_diagnostico` module.

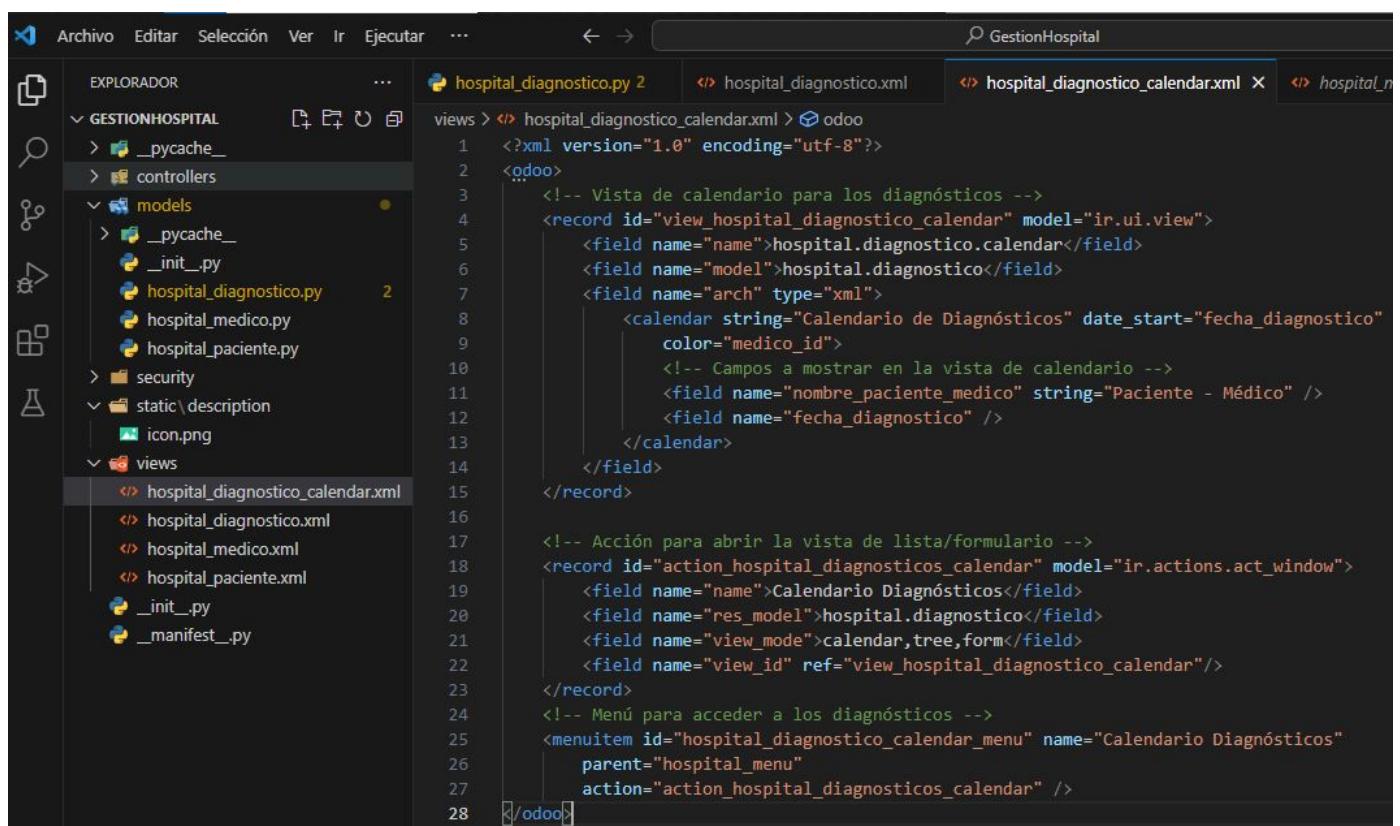
```
<?xml version="1.0" encoding="utf-8"?>
<odoor>
    <!-- Vista de árbol/lista para los diagnósticos -->
    <record id="view_hospital_diagnostico_tree" model="ir.ui.view">
        <field name="name">hospital.diagnostico.tree</field>
        <field name="model">hospital.diagnostico</field>
        <field name="arch" type="xml">
            <tree string="Diagnósticos">
                <!-- Mostrar el nombre del paciente y el médico calculado -->
                <field name="nombre_paciente_medico" />
                <field name="diagnostico" />
                <!-- Agregar campo de fecha -->
                <field name="fecha_diagnostico" />
            </tree>
        </field>
    </record>

    <!-- Vista de formulario para los diagnósticos -->
    <record id="view_hospital_diagnostico_form" model="ir.ui.view">
        <field name="name">hospital.diagnostico.form</field>
        <field name="model">hospital.diagnostico</field>
        <field name="arch" type="xml">
            <form string="Diagnóstico">
                <sheet>
                    <group>
                        <!-- Campo calculado, no editable -->
                        <field name="nombre_paciente_medico" readonly="1" />
                        <!-- Agregar campo de fecha -->
                        <field name="fecha_diagnostico" />
                    </group>
                    <group>
                        <!-- Campo de paciente -->
                        <field name="paciente_id" />
                        <!-- Campo de médico -->
                        <field name="medico_id" />
                    </group>
                    <group>
                        <!-- Campo de diagnóstico -->
                        <field name="diagnostico" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>
</odoor>
```

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

Ahora debemos crear la vista calendar para visualizar los diagnósticos en un calendario.



The screenshot shows the Odoo IDE interface with the following details:

- Toolbar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, ...
- Search Bar:** GestionHospital
- Code Editor:** The active file is `hospital_diagnostico_calendar.xml`. The code defines a calendar view for diagnostic records:<?xml version="1.0" encoding="utf-8"?>
<odoo>
 <!-- Vista de calendario para los diagnósticos -->
 <record id="view_hospital_diagnostico_calendar" model="ir.ui.view">
 <field name="name">hospital.diagnostico.calendar</field>
 <field name="model">hospital.diagnostico</field>
 <field name="arch" type="xml">
 <calendar string="Calendario de Diagnósticos" date_start="fecha_diagnostico"
 color="medico_id">
 <!-- Campos a mostrar en la vista de calendario -->
 <field name="nombre_paciente_medico" string="Paciente - Médico" />
 <field name="fecha_diagnostico" />
 </calendar>
 </field>
 </record>
 <!-- Acción para abrir la vista de lista/formulario -->
 <record id="action_hospital_diagnosticos_calendar" model="ir.actions.act_window">
 <field name="name">Calendario Diagnósticos</field>
 <field name="res_model">hospital.diagnostico</field>
 <field name="view_mode">calendar,tree,form</field>
 <field name="view_id" ref="view_hospital_diagnostico_calendar"/>
 </record>
 <!-- Menú para acceder a los diagnósticos -->
 <menuitem id="hospital_diagnostico_calendar_menu" name="Calendario Diagnósticos"
 parent="hospital_menu"
 action="action_hospital_diagnosticos_calendar" />
</odoo>
- Explorer View:** Shows the project structure under "GESTIONHOSPITAL".



4. Actividad 3 (Desarrollo)

4.1. Desarrollo

No olvidamos cargar la vista en el 'data' del manifest.py.

```
# -*- coding: utf-8 -*-
{
    'name': "Gestión Hospital", # Título del módulo
    'summary': "Gestionar un hospital", # Resumen de la funcionalidad
    'description': """",
    'category': 'Tools',
    'version': '0.1',
    'depends': ['base'],

    'data': [
        'security/ir.model.access.csv',
        #Cargamos las vistas
        'views/hospital_paciente.xml',
        'views/hospital_medico.xml',
        'views/hospital_diagnostico.xml',
        'views/hospital_diagnostico_calendar.xml'
    ],
}
```

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

A continuación mostraré el resultado final del desarrollo del módulo para gestionar el hospital.

The screenshot shows the Odoo application manager interface. The browser address bar displays the URL: `localhost:8069/web#action=36&model=ir.module.module&view_type=kanban&cids=1&menu_id=5`. The main content area shows a list of applications. On the left, a sidebar titled 'CATEGORÍAS' lists various modules: Todos, Ventas, Servicios, Contabilidad, Inventario, Fabricación, Sitio web, Marketing, Recursos Humanos, Productividad, Técnico, and Administración. The 'Todos' category is currently selected. To the right, a card for the 'Gestión Hospital' module is displayed, featuring a thumbnail image of a hospital building, the module name 'Gestión Hospital', its technical name 'GestionHospital', a large blue 'ACTIVAR' button, and a link 'APRENDA MÁS'. The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, and other Odoo modules like Aplicaciones, Tienda de aplicaciones, Actualizaciones, and Actualizar lista de aplicaciones. The search bar at the top right contains filters for 'Aplicaciones', 'Módulo hosp', and a search term 'Buscar...'. Below the search bar are buttons for 'Filtros', 'Agrupar por', and 'Favoritos'.

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

The screenshot shows the Odoo web interface for managing patients. The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and a user profile for Mitchell Admin (odoo). The main menu at the top has items for Hospital, Pacientes, Médicos, Diagnósticos, and Calendario Diagnósticos. The current view is 'Pacientes'. A search bar and filter buttons ('Filtros', 'Agrupar por', 'Favoritos') are visible. The list displays patient names and their symptoms:

Nombre y Apellidos	Síntomas
Rosangela	Pereza aguda
Israel	Dolor de cabeza
Nacho Profe	Gusto obsesivo por el jamón

The screenshot shows the Odoo web interface for managing doctors. The top navigation bar and main menu are identical to the patient interface. The current view is 'Médicos'. A search bar and filter buttons are present. The list displays doctor names and their medical registration numbers:

Nombre y Apellidos	Número de Colegiado
Walter Martin Lopes	2323
Medico de familia	2424

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

The screenshot shows a browser window for the Odoo platform. The URL is `localhost:8069/web#id=1&cids=1&menu_id=235&action=362&model=hospital.medico&view_type=form`. The page title is "Odoo - Walter Martín Lopes". The top navigation bar includes links to Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and more. The main menu has items: Hospital, Pacientes, Médicos, Diagnósticos, and Calendario Diagnósticos. The user is logged in as "Mitchell Admin (odoo)". The current view is "Médicos / Walter Martín Lopes". The form contains fields: Nombre y Apellidos (Walter Martín Lopes), Número de Colegiado (2323), and Pacientes Atendidos (Rosangela, Nacho Profe). There are buttons for Acción, Nuevo, and pagination (1 / 2).

The screenshot shows a browser window for the Odoo platform. The URL is `localhost:8069/web#action=363&model=hospital.diagnostico&view_type=list&cids=1&menu_id=235`. The page title is "Odoo - Diagnósticos". The top navigation bar and user information are identical to the previous screenshot. The main menu items are Hospital, Pacientes, Médicos, Diagnósticos, and Calendario Diagnósticos. The current view is "Diagnósticos". A search bar and filter options are available. The list table has columns: Paciente - Médico, Diagnóstico, and Fecha del Diagnóstico. The data rows are:

Paciente - Médico	Diagnóstico	Fecha del Diagnóstico
Nacho Profe - Walter Martín Lopes	El paciente tras un estudio de su caso es diagnosticado con un síndrome obsesivo con el jamón. El paciente debe comer un jamón antes de las evaluaciones.	06/02/2024
Israel - Walter Martín Lopes	Ibuprofeno, 400mg al día.	05/02/2024
Nacho Profe - Medico de familia	El paciente debe comer 40g de jamón cada 3h.	08/02/2024

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

The screenshot shows the Odoo web interface for the 'Calendario Diagnóstico' module. The top navigation bar includes links for 'Gmail', 'Drive', 'Trello', 'Calendar', 'Rayuela', 'Moodle', 'ClassRoom', 'GitHub', 'Design patterns', 'ChatGPT', 'TradingView', 'YouGlish', 'Investing', and 'Fit'. The main menu on the left lists 'Hospital', 'Pacientes', 'Médicos', 'Diagnósticos', and 'Calendario Diagnósticos'. The current view is 'Calendario Diagnósticos (febrero de 2024)'. The calendar grid shows dates from February 5 to March 3. Several events are listed in the calendar cells:

- February 5: Israel - Walter Martí... (red box)
- February 6: Nacho Profe - Walt... (red box)
- February 8: Nacho Profe - Medi... (orange box)

A sidebar on the right displays a monthly calendar for February 2024, showing days from Monday to Sunday. The 6th is highlighted with a red circle.

Lunes	Martes	miércoles	jueves	viernes	sábado	domingo
5 29	30	31	1	2	3	4
6 5 Israel - Walter Martí...	6 Nacho Profe - Walt...	7	8 Nacho Profe - Medi...	9	10	11
7 12	13	14	15	16	17	18
8 19	20	21	22	23	24	25
9 26	27	28	29	1	2	3
10 4	5	6	7	8	9	10

4. Actividad 3 (Desarrollo)

4.1. Desarrollo

The screenshot shows the Odoo Diagnostic Calendar interface. The top navigation bar includes links for Hospital, Pacientes, Médicos, Diagnósticos, and Calendario Diagnósticos. The current view is 'Calendario Diagnósticos (5 – 11 feb 2024)'. The calendar grid shows appointments for three doctors: Israel - Walter Martin Lopes (Monday), Nacho Profe - Walter Martin Lopes (Tuesday), and Nacho Profe - Médico de familia (Wednesday). The sidebar on the right displays a monthly calendar for February 2024, with the 6th highlighted in red.

L	M	X	J	V	S	D
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	1	2	3

Actividad 4 (Desarrollo)





5. Actividad 4 (Desarrollo)

5.1. Enunciado

Haz un módulo de Odoo que represente nuestros estudios de ciclos formativos en un instituto:

- Modelo ciclo formativo. Cada instancia representa a un ciclo formativo en un instituto. Un ciclo tiene uno o más módulos asociados.
- Modelo módulo. Cada “módulo” estará relacionado con ciclos formativos (al que pertenece), alumnos matriculados y profesor que lo imparte.
- Modelo alumno. Relacionado con los módulos en los que está matriculado.
- Modelo profesor. Relacionado con los módulos que imparte.

Implementa los modelos, las relaciones necesarias y las vistas que creas adecuadas para los 4 modelos.

Una vez en funcionamiento la aplicación, queremos que implementes la siguiente configuración de seguridad:

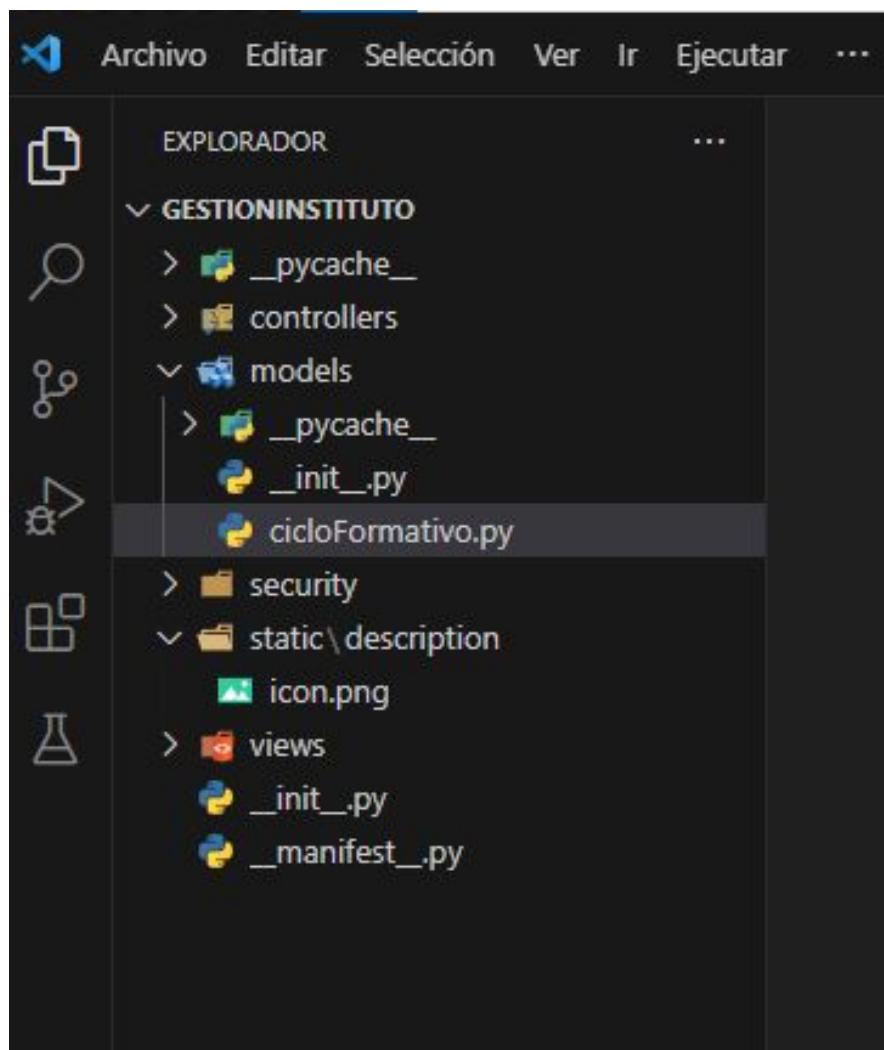
- Los usuarios con el rol de “Director” podrán modificar los registros de los modelos anteriores.
- Además del director, los únicos usuarios que podrán ver los datos de los profesores (en modo lectura) serán los usuarios con el rol de “Profesor”.



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

En primer lugar, creamos la estructura básica del proyecto partiendo del módulo de hospital creado anteriormente, eliminando las cosas que no necesito y modificando el manifest.

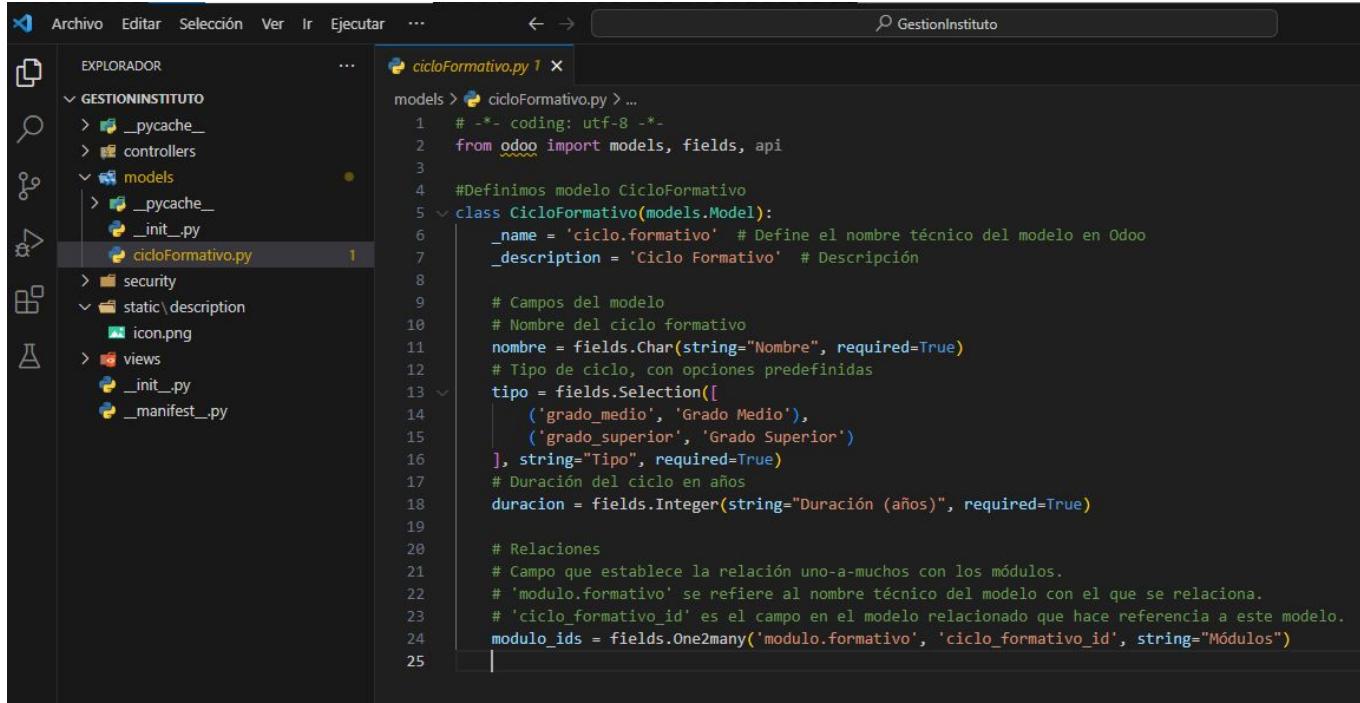




5. Actividad 4 (Desarrollo)

5.2. Desarrollo

A continuación comenzamos a desarrollar los modelos necesarios expuestos en el enunciado, creando sus respectivas vistas como hemos hecho en módulos anteriores. Comenzaremos con el módulo ciclo formativo.



```
models > cicloFormativo.py > ...
1  # -*- coding: utf-8 -*-
2  from odoo import models, fields, api
3
4  #Definimos modelo CicloFormativo
5  class CicloFormativo(models.Model):
6      _name = 'ciclo.formativo' # Define el nombre técnico del modelo en Odoo
7      _description = 'Ciclo Formativo' # Descripción
8
9      # Campos del modelo
10     # Nombre del ciclo formativo
11     nombre = fields.Char(string="Nombre", required=True)
12     # Tipo de ciclo, con opciones predefinidas
13     tipo = fields.Selection([
14         ('grado_medio', 'Grado Medio'),
15         ('grado_superior', 'Grado Superior')
16     ], string="Tipo", required=True)
17     # Duración del ciclo en años
18     duracion = fields.Integer(string="Duración (años)", required=True)
19
20     # Relaciones
21     # Campo que establece la relación uno-a-muchos con los módulos.
22     # 'modulo.formativo' se refiere al nombre técnico del modelo con el que se relaciona.
23     # 'ciclo_formativo_id' es el campo en el modelo relacionado que hace referencia a este modelo.
24     modulo_ids = fields.One2many('modulo.formativo', 'ciclo_formativo_id', string="Módulos")
25
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Creamos una vista para este modelo similar a las creadas en módulos anteriores.

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Vista de Lista para Ciclo Formativo -->
    <record id="view_ciclo_formativo_list" model="ir.ui.view">
        <field name="name">ciclo.formativo.list</field>
        <field name="model">ciclo.formativo</field>
        <field name="arch" type="xml">
            <tree string="Ciclo Formativo Lista">
                <field name="nombre" />
                <field name="tipo" />
                <field name="duracion" />
            </tree>
        </field>
    </record>

    <!-- Vista de Formulario para Ciclo Formativo -->
    <record id="view_ciclo_formativo_form" model="ir.ui.view">
        <field name="name">ciclo.formativo.form</field>
        <field name="model">ciclo.formativo</field>
        <field name="arch" type="xml">
            <form string="Ciclo Formativo">
                <sheet>
                    <group>
                        <field name="nombre" />
                        <field name="tipo" />
                        <field name="duracion" />
                        <field name="modulo_ids" widget="many2many_tags" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- Acción para abrir la vista de lista -->
    <record id="action_ciclo_formativo" model="ir.actions.act_window">
        <field name="name">Ciclos Formativos</field>
        <field name="res_model">ciclo.formativo</field>
        <field name="view_mode">tree,form</field>
    </record>

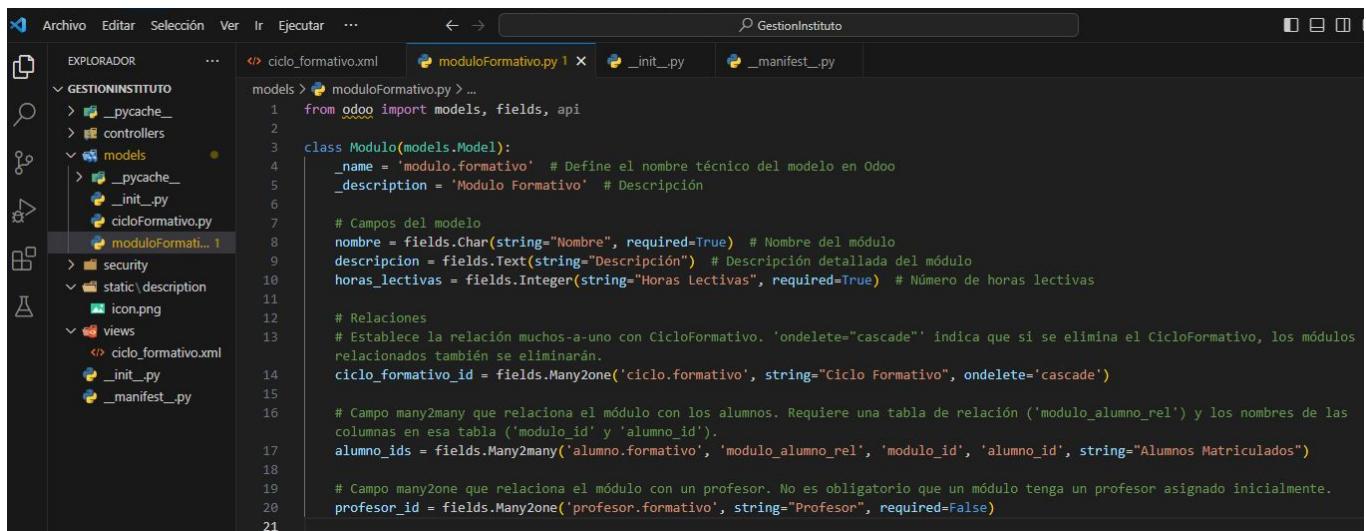
    <!-- Menú para acceder a los Ciclos Formativos -->
    <menuitem id="instituto_menu" name="Instituto" />
    <menuitem id="ciclo_formativo_menu" name="Ciclos Formativos" action="action_ciclo_formativo" parent="instituto_menu" />
</odoo>
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Ahora crearemos el modelo módulo formativo.



The screenshot shows the Odoo IDE interface with the following details:

- File Menu:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, ...
- Title Bar:** GestionInstituto
- Explorer View (Left):**
 - GESTIONINSTITUTO
 - __pycache__
 - controllers
 - models
 - __pycache__
 - __init__.py
 - cicloFormativo.py
 - moduloFormativo.py (selected)
 - security
 - static\description
 - icon.png
 - views
 - ciclo_formativo.xml
 - __init__.py
 - __manifest__.py
- Code Editor (Right):**

```
EXPLORADOR      ciclo_formativo.xml  moduloFormativo.py  __init__.py  __manifest__.py
GESTIONINSTITUTO
models > moduloFormativo.py > ...
1   from odoo import models, fields, api
2
3 class Modulo(models.Model):
4     _name = 'modulo.formativo' # Define el nombre técnico del modelo en Odoo
5     _description = 'Modulo Formativo' # Descripción
6
7     # Campos del modelo
8     nombre = fields.Char(string="Nombre", required=True) # Nombre del módulo
9     descripcion = fields.Text(string="Descripción") # Descripción detallada del módulo
10    horas_lectivas = fields.Integer(string="Horas Lectivas", required=True) # Número de horas lectivas
11
12    # Relaciones
13    # Establece la relación muchos-a-uno con CicloFormativo. 'ondelete="cascade"' indica que si se elimina el CicloFormativo, los módulos relacionados también se eliminarán.
14    ciclo_formativo_id = fields.Many2one('ciclo.formativo', string="Ciclo Formativo", ondelete='cascade')
15
16    # Campo many2many que relaciona el módulo con los alumnos. Requiere una tabla de relación ('modulo_alumno_rel') y los nombres de las
17    # columnas en esa tabla ('modulo_id' y 'alumno_id').
18    alumno_ids = fields.Many2many('alumno.formativo', 'modulo_alumno_rel', 'modulo_id', 'alumno_id', string="Alumnos Matriculados")
19
20    # Campo many2one que relaciona el módulo con un profesor. No es obligatorio que un módulo tenga un profesor asignado inicialmente.
21    profesor_id = fields.Many2one('profesor.formativo', string="Profesor", required=False)
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Creamos la correspondiente vista para el modelo módulo formativo.

The screenshot shows the Odoo IDE interface with the following details:

- File Menu:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, ...
- Search Bar:** GestionInstituto
- Explorer View (Left):**
 - GESTIONINSTITUTO
 - _pycache_
 - controllers
 - models
 - _pycache_
 - _init_.py
 - cicloFormativo.py
 - moduloFormativo... 1
 - security
 - static\description
 - icon.png
 - views
 - ciclo_formativo.xml
 - modulo_formativo.x...
 - _init_.py
 - _manifest_.py
 - ESQUEMA
 - LÍNEA DE TIEMPO
 - XML DOCUMENT
- Code Editor (Right):** moduloFormativo.py 1
- Content of moduloFormativo.py:**

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Vista de Lista para Modulo Formativo -->
    <record id="view_modulo_formativo_list" model="ir.ui.view">
        <field name="name">modulo.formativo.list</field>
        <field name="model">modulo.formativo</field>
        <field name="arch" type="xml">
            <tree string="Lista de Módulos">
                <field name="nombre" />
                <field name="descripcion" />
                <field name="horas_lectivas" />
                <field name="ciclo_formativo_id" />
                <field name="profesor_id" />
            </tree>
        </field>
    </record>

    <!-- Vista de Formulario para Modulo Formativo -->
    <record id="view_modulo_formativo_form" model="ir.ui.view">
        <field name="name">modulo.formativo.form</field>
        <field name="model">modulo.formativo</field>
        <field name="arch" type="xml">
            <form string="Detalle del Módulo">
                <sheet>
                    <group>
                        <field name="nombre" />
                        <field name="descripcion" />
                        <field name="horas_lectivas" />
                        <field name="ciclo_formativo_id" />
                        <field name="profesor_id" />
                        <field name="alumno_ids" widget="many2many_tags" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- Acción para abrir las vistas de Módulo Formativo -->
    <record id="action_modulo_formativo" model="ir.actions.act_window">
        <field name="name">Módulos Formativos</field>
        <field name="res_model">modulo.formativo</field>
        <field name="view_mode">tree,form</field>
    </record>

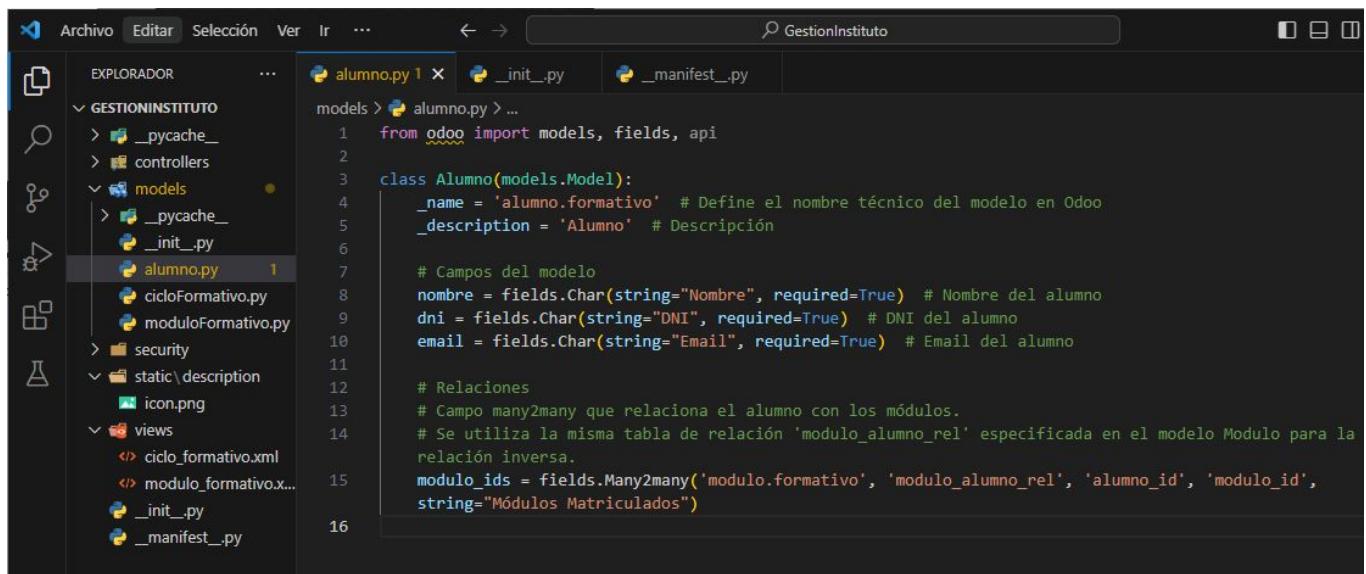
    <!-- Menú para acceder a los Módulos Formativos -->
    <menuitem id="modulo_formativo_menu" name="Módulos Formativos"
              action="action_modulo_formativo" parent="instituto_menu" />
</odoo>
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Ahora desarrollaremos el modelo alumno.



The screenshot shows a code editor interface with the title bar "GestionInstituto". The left sidebar is labeled "EXPLORADOR" and lists the project structure:

- GESTIONINSTITUTO
 - __pycache__
 - controllers
 - models
 - __pycache__
 - __init__.py
 - alumno.py 1
 - cicloFormativo.py
 - moduloFormativo.py
 - security
 - static\description
 - icon.png
 - views
 - ciclo_formativo.xml
 - modulo_formativo.x...
 - __init__.py
 - __manifest__.py

```
alumno.py 1 | __init__.py | __manifest__.py
models > alumno.py > ...
1   from odoo import models, fields, api
2
3   class Alumno(models.Model):
4       _name = 'alumno.formativo' # Define el nombre técnico del modelo en Odoo
5       _description = 'Alumno' # Descripción
6
7       # Campos del modelo
8       nombre = fields.Char(string="Nombre", required=True) # Nombre del alumno
9       dni = fields.Char(string="DNI", required=True) # DNI del alumno
10      email = fields.Char(string="Email", required=True) # Email del alumno
11
12      # Relaciones
13      # Campo many2many que relaciona el alumno con los módulos.
14      # Se utiliza la misma tabla de relación 'modulo_alumno_rel' especificada en el modelo Modulo para la
15      # relación inversa.
16      modulo_ids = fields.Many2many('modulo.formativo', 'modulo_alumno_rel', 'alumno_id', 'modulo_id',
string="Módulos Matriculados")
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Desarrollamos una vista para el modelo alumno.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows the project structure under 'GESTIONINSTITUTO'. It includes 'controllers', 'models' (with 'alumno.py'), 'static\description' (with 'icon.png'), and 'views' (with 'alumno.xml', 'ciclo_formativo.xml', 'modulo_formativo.xml', '_init_.py', and '_manifest_.py').
- Code Editor:** The 'alumno.xml' file is open and displayed. The code defines three XML records for the 'alumno' model:
 - Record 1:** A tree view for the list of students, showing fields for name, model (alumno.formativo), and arch (containing a tree node with fields for nombre, dni, and email).
 - Record 2:** A form view for the student details, showing fields for name, model (alumno.formativo), and arch (containing a form node with a sheet and group nodes for nombre, dni, email, and modulo_ids).
 - Record 3:** An action record for opening the student views, with fields for name (Alumnos), res_model (alumno.formativo), and view_mode (tree,form).
- Status Bar:** Shows options for 'ESQUEMA' and 'LÍNEA DE TIEMPO'.



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Por último crearemos el modelo profesor.

The screenshot shows the Odoo IDE interface with the project name "GestionInstituto". The left sidebar displays the file structure:

- EXPLORADOR
- GESTIONINSTITUTO
 - __pycache__
 - controllers
 - models
 - __pycache__
 - __init__.py
 - alumno.py
 - cicloFormativo.py
 - moduloFormativo.py
 - profesor.py
 - security
 - static\description
 - icon.png
 - views
 - alumno.xml
 - ciclo_formativo.xml
 - modulo_formativo.xml
 - __init__.py
 - __manifest__.py

```
alumno.py 1 alumno.xml profesor.py 1 __init__.py __manifest__.py

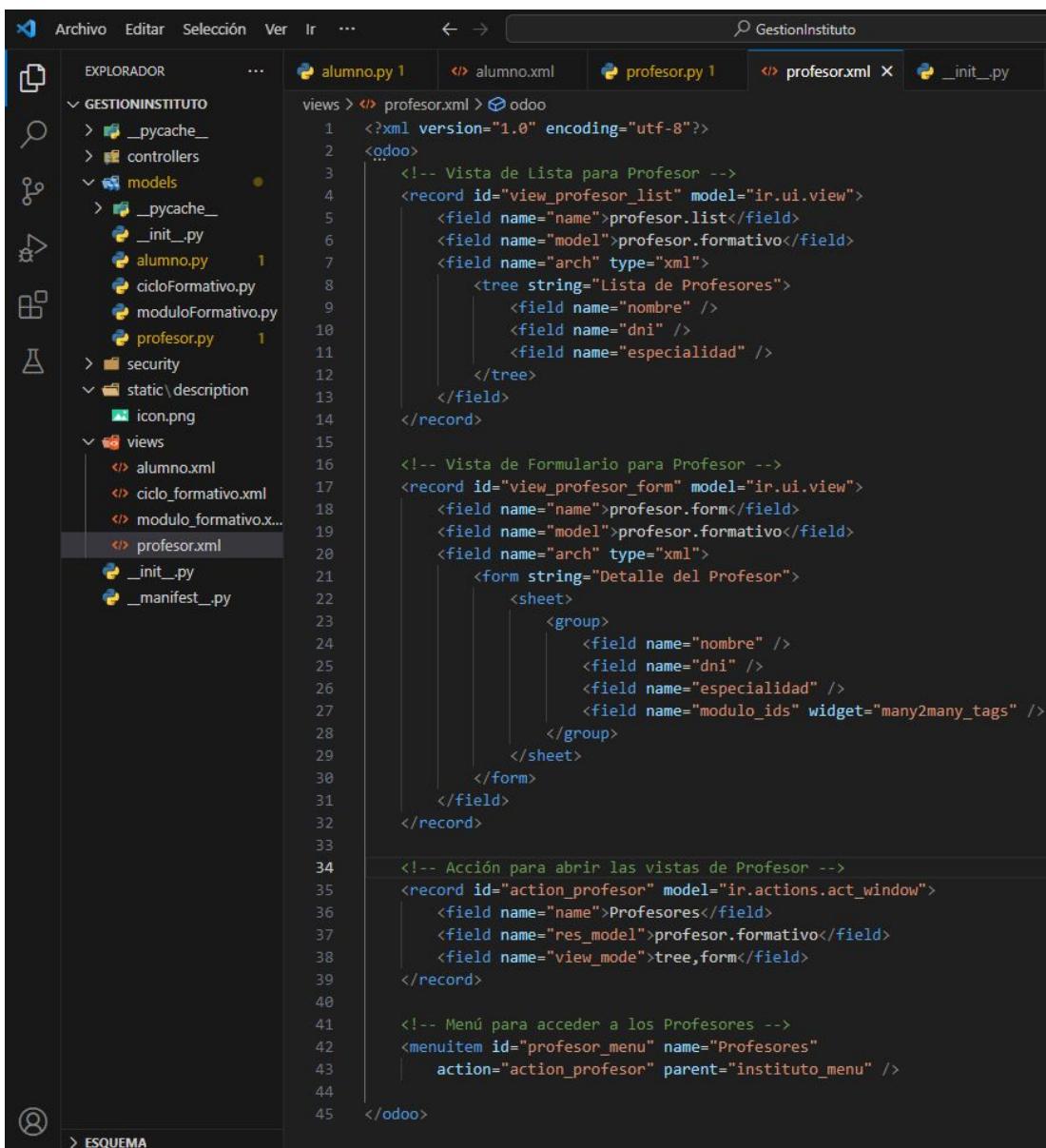
models > profesor.py > Profesor
1   from odoo import models, fields, api
2
3
4   class Profesor(models.Model):
5       # Define el nombre técnico del modelo en Odoo.
6       _name = 'profesor.formativo'
7       _description = 'Profesor' # Descripcion
8
9       # Campos del modelo
10      # Nombre del profesor.
11      nombre = fields.Char(string="Nombre", required=True)
12      # DNI del profesor, identificador único.
13      dni = fields.Char(string="DNI", required=True)
14      # Especialidad académica o profesional del profesor.
15      especialidad = fields.Char(string="Especialidad", required=True)
16
17      # Relaciones
18      # Campo One2many que relaciona el profesor con los módulos que imparte.
19      # 'modulo.formativo' es el modelo relacionado y 'profesor_id' es el campo en ese modelo que establece la
20      # conexión de vuelta.
21      modulo_ids = fields.One2many(
22          'modulo.formativo', 'profesor_id', string="Módulos Impartidos")
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Definimos la vista correspondiente para el modelo profesor.



The screenshot shows the Odoo IDE interface with the following details:

- Toolbar:** Archivo, Editar, Selección, Ver, Ir, ...
- Search Bar:** GestionInstituto
- Explorer View (Left):**
 - GESTIONINSTITUTO
 - _pycache_
 - controllers
 - models
 - _pycache_
 - _init_.py
 - alumno.py (marked with a red exclamation)
 - cicloFormativo.py
 - moduloFormativo.py
 - profesor.py (marked with a red exclamation)
 - security
 - static\description
 - icon.png
 - views
 - alumno.xml
 - ciclo_formativo.xml
 - modulo_formativo.xml
 - profesor.xml (selected and highlighted in grey)
 - _init_.py
 - _manifest_.py
- Code Editor (Right):** The code editor displays the XML configuration for the Profesor model's views. The selected tab is 'profesor.xml'. The code includes:

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Vista de Lista para Profesor -->
    <record id="view_profesor_list" model="ir.ui.view">
        <field name="name">profesor.list</field>
        <field name="model">profesor.formativo</field>
        <field name="arch" type="xml">
            <tree string="Lista de Profesores">
                <field name="nombre" />
                <field name="dni" />
                <field name="especialidad" />
            </tree>
        </field>
    </record>

    <!-- Vista de Formulario para Profesor -->
    <record id="view_profesor_form" model="ir.ui.view">
        <field name="name">profesor.form</field>
        <field name="model">profesor.formativo</field>
        <field name="arch" type="xml">
            <form string="Detalle del Profesor">
                <sheet>
                    <group>
                        <field name="nombre" />
                        <field name="dni" />
                        <field name="especialidad" />
                        <field name="modulo_ids" widget="many2many_tags" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- Acción para abrir las vistas de Profesor -->
    <record id="action_profesor" model="ir.actions.act_window">
        <field name="name">Profesores</field>
        <field name="res_model">profesor.formativo</field>
        <field name="view_mode">tree,form</field>
    </record>

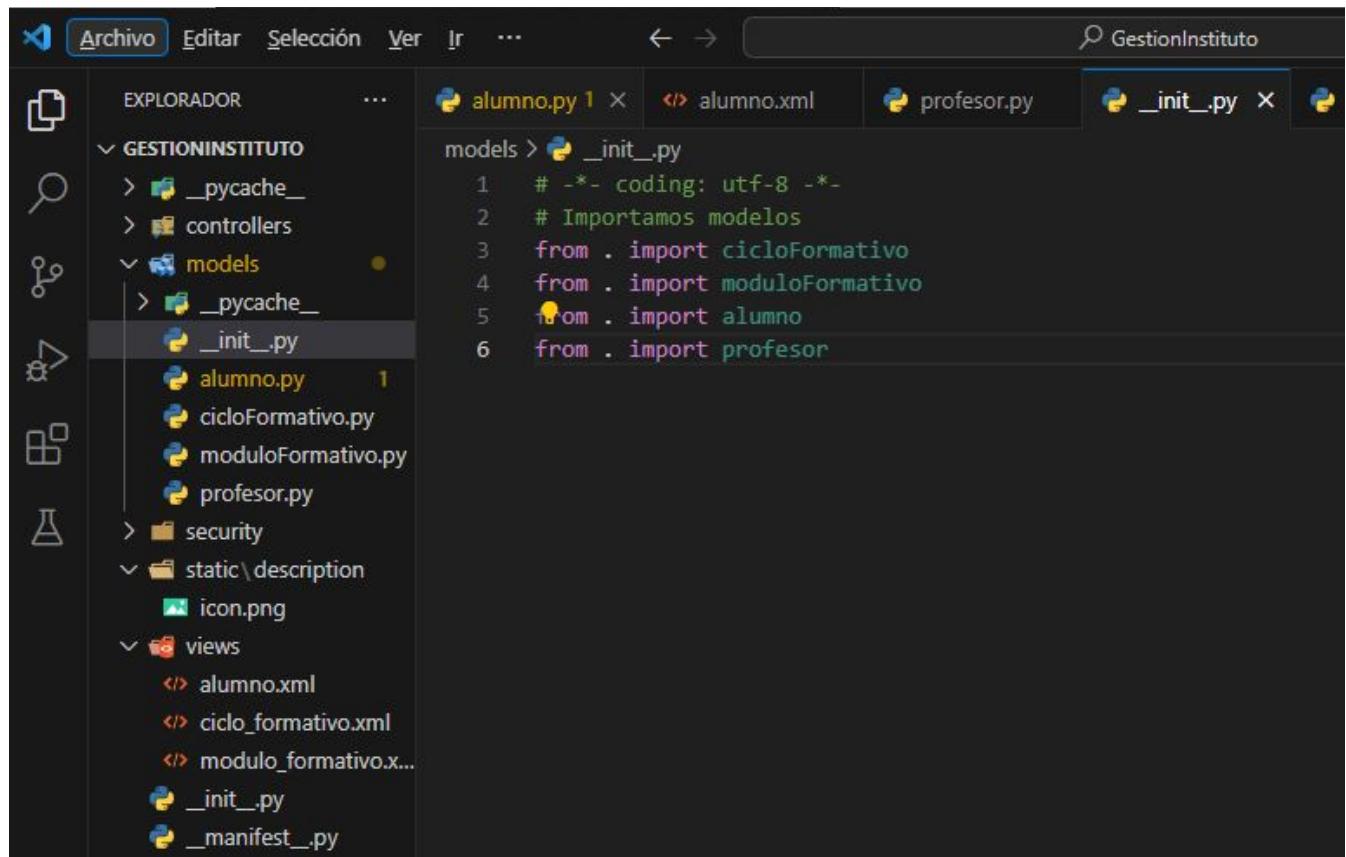
    <!-- Menú para acceder a los Profesores -->
    <menuitem id="profesor_menu" name="Profesores"
              action="action_profesor" parent="instituto_menu" />
</odoo>
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Una vez todos los modelos creados, debemos importarlos en el `init.py`



```
EXPLORADOR ... alumno.py 1 × alumno.xml profesor.py __init__.py
GESTIONINSTITUTO
  > __pycache__
  > controllers
  > models
    > __pycache__
    > __init__.py
      alumno.py 1
      cicloFormativo.py
      moduloFormativo.py
      profesor.py
    > security
    > static\description
      icon.png
  > views
    alumno.xml
    ciclo_formativo.xml
    modulo_formativo.x...
    __init__.py
    __manifest__.py

models > __init__.py
1  # -*- coding: utf-8 -*-
2  # Importamos modelos
3  from . import cicloFormativo
4  from . import moduloFormativo
5  from . import alumno
6  from . import profesor
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

De igual manera, las vistas las debemos cargar en el `manifest.py`

```
# -*- coding: utf-8 -*-
{
    'name': "Gestión Instituto", # Título del módulo
    # Resumen de la funcionalidad
    'summary': "Gestionar un instituto, módulos, ciclos...",
    'description': """
Gestor de institutos (Version Simple)
=====
""",
    # Indicamos que es una aplicación
    'application': True,
    'author': "Walter Martín Lopes",
    'website': "http://apuntesfpinformatica.es",
    'category': 'Tools',
    'version': '0.1',
    'depends': ['base'],

    'data': [
        'security/ir.model.access.csv',
        # Cargamos las vistas
        'views/ciclo_formativo.xml',
        'views/modulo_formativo.xml',
        'views/alumno.xml',
        'views/profesor.xml'
    ],
}
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

A continuación nos disponemos a instalar el módulo desarrollado.

The screenshot shows the Odoo application store interface. The URL in the browser is `localhost:8069/web#action=36&model=ir.module.module&view_type=kanban&cids=1&menu_id=5`. The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and more. The user is logged in as Mitchell Admin (odoo). The main menu on the left lists categories: Todos, Ventas, Servicios, Contabilidad, Inventario, Fabricación, Sitio web, Marketing, Recursos Humanos, Productividad, Técnico, and Administración. The search bar at the top right contains filters for 'Aplicaciones', 'Módulo instituto', and a search field. A single module card is displayed for 'Gestión Instituto' (GestionInstituto), featuring a thumbnail of a school building, the module name, its code, an 'ACTIVAR' button, and a 'APRENDA MÁS' link. The status bar at the bottom indicates page 1-1 of 1.



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Pero nos da error al instalarlo. El error que especifica nos da a entender que hay un problema con los permisos de acceso, lo que nos lleva a revisar el access.csv

The screenshot shows a browser window for 'Odoo - Aplicaciones' at 'localhost:8069'. The URL in the address bar is 'localhost:8069/web?action=36&model=ir.module.module&view_type=kanban&cids=1&menu_id='.

The main content is a modal dialog titled 'Error de Odoo' with the message 'Se ha producido un error'. It includes a button 'COPIAR EL ERROR AL COMPLETO EN EL PORTAPAPELES' (Copy full error to clipboard). Below this, there's a section 'Ver detalles' (View details) showing 'RPC_ERROR' and 'Odoo Server Error'. A large red box highlights the detailed error trace, which starts with:

```
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/odoo/http.py", line 1591, in _serve_db
    return service_model.retrying(self._serve_ir_http, self.env)
  File "/usr/lib/python3/dist-packages/odoo/service/model.py", line 133, in retrying
    result = func()
  File "/usr/lib/python3/dist-packages/odoo/http.py", line 1618, in _serve_ir_http
    response = self.dispatcher.dispatch(rule.endpoint, args)
  File "/usr/lib/python3/dist-packages/odoo/http.py", line 1822, in dispatch
    result = self.request.registry['ir.http']._dispatch(endpoint)
  File "/usr/lib/python3/dist-packages/odoo/addons/base/models/ir_http.py", line 154, in _dispatch
    result = endpoint(**request.params)
  File "/usr/lib/python3/dist-packages/odoo/http.py", line 697, in route_wrapper
    result = endpoint(*args, **params)
  File "/usr/lib/python3/dist-packages/odoo/addons/web/controllers/dataset.py", line 46, in call_button
    action = self._call_kw(model, method, args, kwargs)
  File "/usr/lib/python3/dist-packages/odoo/addons/web/controllers/dataset.py", line 33, in _call_kw
    return call_kw(request.env[model], method, args, kwargs)
  File "/usr/lib/python3/dist-packages/odoo/api.py", line 468, in call_kw
    result = _call_kw_multi(method, model, args, kwargs)
  File "/usr/lib/python3/dist-packages/odoo/api.py", line 453, in _call_kw_multi
    result = method(recs, *args, **kwargs)
  File "<decorator-gen-72>", line 2, in button_immediate_install
  File "/usr/lib/python3/dist-packages/odoo/addons/base/models/ir_module.py", line 74, in check_and_log
    return method(self, *args, **kwargs)
  File "/usr/lib/python3/dist-packages/odoo/addons/base/models/ir_module.py", line 476, in button_immediate_install
    return self._button_immediate_function(self.env.registry[self._name].button_install)
  File "/usr/lib/python3/dist-packages/odoo/addons/base/models/ir_module.py", line 600, in _button_immediate_function
    registry = modules.registry.Registry.new(self._cr dbname, update_module=True)
  File "<decorator-gen-16>", line 2, in new
  File "/usr/lib/python3/dist-packages/odoo/tools/func.py", line 87, in locked
```

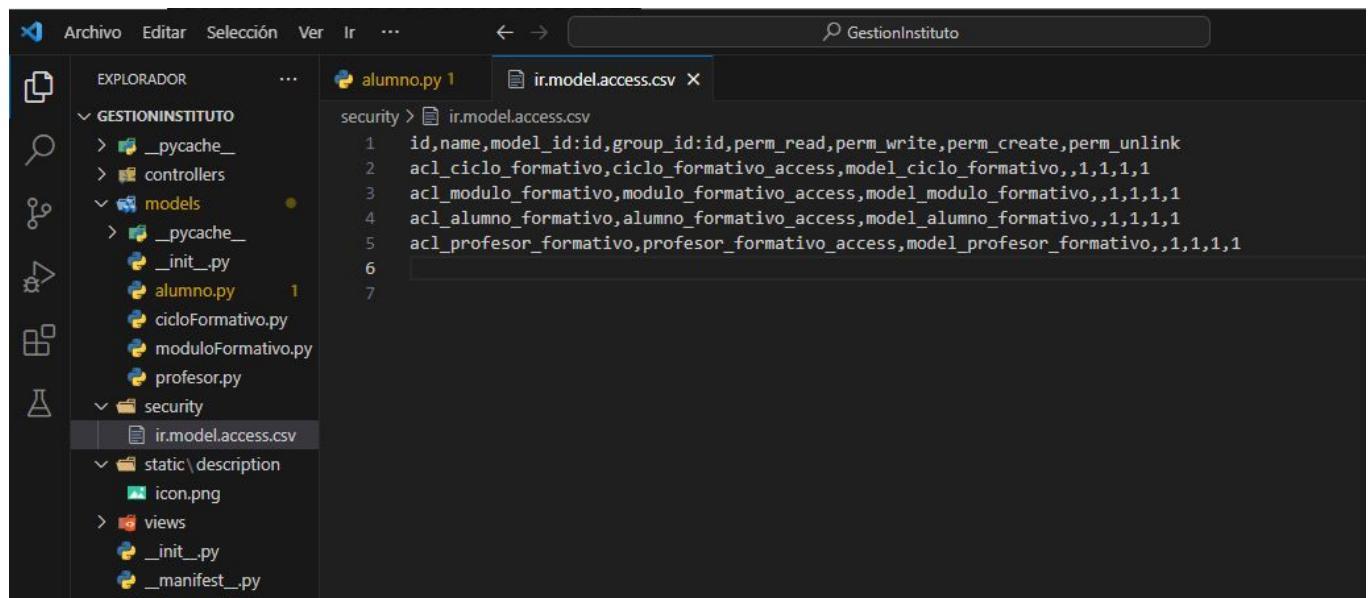
At the bottom of the dialog is a 'ACEPTAR' (Accept) button.



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

El problema venía de no actualizar correctamente el `access.csv` ya que al reutilizar la estructura del módulo hospital este tenía todo los modelos de hospital. Nos disponemos a actualizarlo.



```
security > ir.model.access.csv
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 acl_ciclo_formativo,ciclo_formativo_access,model_ciclo_formativo,,1,1,1,1
3 acl_modulo_formativo,modulo_formativo_access,model_modulo_formativo,,1,1,1,1
4 acl_alumno_formativo,alumno_formativo_access,model_alumno_formativo,,1,1,1,1
5 acl_profesor_formativo,profesor_formativo_access,model_profesor_formativo,,1,1,1,1
6
7
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Una vez instalado y comprobando su funcionamiento nos damos cuenta de que no nos muestra los nombres de los profesores y ciclos formativos, sino su id.

<input type="checkbox"/> Nombre	Descripción	Horas Lectivas	Ciclo Formativo	Profesor
<input type="checkbox"/> Seguridad Informática	Seguridad Informática	6	ciclo.formativo,3	profesor.formativo,2
<input type="checkbox"/> Programación Multimedia y Dispositivos Móviles	Android	8	ciclo.formativo,1	profesor.formativo,4
<input type="checkbox"/> Redes Locales	Redes	7	ciclo.formativo,3	profesor.formativo,1
<input type="checkbox"/> Acceso a Datos	Bases de datos y demás	25	ciclo.formativo,1	profesor.formativo,2
<input type="checkbox"/> Sistemas de Gestión Empresarial	Nacho da Odoo	45	ciclo.formativo,1	profesor.formativo,1
<input type="checkbox"/> Pinta y colorea	Esas cosas que hacen los de Daw	5	ciclo.formativo,2	profesor.formativo,5



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Para solucionar esto debemos implementar un método para obtener el nombre, en la siguiente imagen se explica la implementación de este método, que es el mismo para todos los modelos, ya que todos tienen un campo nombre.

The screenshot shows a code editor interface with the title bar "GestionInstituto". The left sidebar displays a file tree for a Python application named "GESTIONINSTITUTO". The "models" directory contains several files: alumno.py, moduloFormativo.py, and profesor.py. The "moduloFormativo.py" file is currently open and shows the following Python code:

```
models > moduloFormativo.py > name_get
    descripcion = fields.Text(string='Descripción')
    horas_lectivas = fields.Integer(
        string="Horas Lectivas", required=True) # Número de horas lectivas

    # Relaciones
    # Establece la relación muchos-a-uno con CicloFormativo. 'ondelete="cascade"' indica que si se elimina el CicloFormativo, los módulos relacionados también se eliminarán.
    ciclo_formativo_id = fields.Many2one(
        'ciclo.formativo', string="Ciclo Formativo", ondelete='cascade')

    # Campo many2many que relaciona el módulo con los alumnos. Requiere una tabla de relación ('modulo_alumno_rel') y los nombres de las columnas en esa tabla ('modulo_id' y 'alumno_id').
    alumno_ids = fields.Many2many('alumno.formativo', 'modulo_alumno_rel',
        'modulo_id', 'alumno_id', string="Alumnos Matriculados")

    # Campo many2one que relaciona el módulo con un profesor. No es obligatorio que un módulo tenga un profesor asignado inicialmente.
    profesor_id = fields.Many2one(
        'profesor.formativo', string="Profesor", required=False)

# Indica que el método debe ser llamado cuando cambie el valor del campo 'nombre'.
@api.depends('nombre')
def name_get(self):
    # Inicializa una lista vacía para almacenar los nombres personalizados de los registros.
    result = []

    # Itera sobre cada registro en el conjunto actual ('self' representa el conjunto de registros del modelo).
    for record in self:
        # Obtiene el valor del campo 'nombre' del registro actual para usarlo como parte del nombre mostrado.
        name = record.nombre

        # Añade una tupla al resultado, donde el primer elemento es el ID del registro
        # y el segundo es el nombre personalizado que definiste.
        result.append((record.id, name))

    # Devuelve la lista de tuplas que representa los nombres personalizados de los registros.
    # Cada tupla contiene el ID del registro y el nombre personalizado para mostrar.
    return result
```



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Ahora sí, mostramos el resultado final del desarrollo del módulo para gestionar un instituto.

Nombre	Tipo	Duración (años)
Desarrollo de Aplicaciones Multiplataforma (DAM)	Grado Superior	2
Desarrollo de Aplicaciones Web (DAW)	Grado Superior	2
Sistemas Microinformáticos y Redes (SMR)	Grado Medio	2

Ciclos Formativos / Desarrollo de Aplicaciones Multiplataforma (DAM)

Nombre ?	Desarrollo de Aplicaciones Multiplataforma (DAM)
Tipo ?	Grado Superior
Duración (años) ?	2
Módulos ?	(Programación Multimedia y Dispositivos) (Acceso a Datos) (Sistemas de Gestión Empresarial)



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

The screenshot shows a web browser window for Odoo Módulos Formativos. The URL is `localhost:8069/web#action=366&model=modulo.formativo&view_type=list&cids=1&menu_id=241`. The page title is "Odoo - Módulos Formativos". The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and a user profile for Mitchell Admin (odoo). The main content area has a purple header with tabs for Instituto, Ciclos Formativos, Módulos Formativos, Alumnos, and Profesores. Below the header is a search bar and filter options (Filtros, Agrupar por, Favoritos). A table lists training modules:

NOMBRE	DESCRIPCIÓN	HORAS LECTIVAS	CICLO FORMATIVO	PROFESOR
Seguridad Informática	Seguridad Informática	6	Sistemas Microinformáticos y Redes (SMR)	Antonio
Programación Multimedia y Dispositivos Móviles	Android	8	Desarrollo de Aplicaciones Multiplataforma (DAM)	Pepe
Redes Locales	Redes	7	Sistemas Microinformáticos y Redes (SMR)	Nacho
Acceso a Datos	Bases de datos y demás	25	Desarrollo de Aplicaciones Multiplataforma (DAM)	Antonio
Sistemas de Gestión Empresarial	Nacho da Odoo	45	Desarrollo de Aplicaciones Multiplataforma (DAM)	Nacho
Pinta y colorea	Esas cosas que hacen los de Daw	5	Desarrollo de Aplicaciones Web (DAW)	Marcos

The screenshot shows a web browser window for Odoo Módulos Formativos. The URL is `localhost:8069/web#id=6&cids=1&menu_id=241&action=366&model=modulo.formativo&view_type=form`. The page title is "Odoo - modulo.formativo,6". The top navigation bar and header are identical to the previous screenshot. The main content area has a purple header with tabs for Instituto, Ciclos Formativos, Módulos Formativos, Alumnos, and Profesores. Below the header is a form for editing a module:

Módulos Formativos / modulo.formativo,6

Nombre ?	Pinta y colorea
Descripción ?	Esas cosas que hacen los de Daw
Horas Lectivas ?	5
Ciclo Formativo ?	Desarrollo de Aplicaciones Web (DAW)
Profesor ?	Marcos
Alumnos Matriculados ?	(Walter Martin Lopes ✕) (Rosángela de la Rosa ✕)



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Nombre	DNI	Email
Walter Martin Lopes	45134320V	wmartinl01@informatica.es
Rosángela de la Rosa	45125456R	rosa@informatica.es

Nombre ?	Walter Martin Lopes
DNI ?	45134320V
Email ?	wmartinl01@informatica.es
Módulos Matriculados ?	(Seguridad Informática) (Redes Locales) (Pinta y colorear)



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Nombre	DNI	Especialidad
Nacho	12345678A	Odoo
Antonio	23456789B	Todo
Daniel	34567891C	Android
Pepe	56789123Y	Dedin
Marcos	96325874G	Todo

Profesores / Antonio

Nombre ?	Antonio
DNI ?	23456789B
Especialidad ?	Todo
Módulos Impartidos ?	(Seguridad Informática ✕) (Acceso a Datos ✕)

5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Una vez desarrollado el módulo, debemos implementar las restricciones de seguridad para los distintos roles que expone el enunciado del ejercicio. Para ello necesitamos crear los roles Director y Profesor.

Para crear estos roles debemos ir a la pestaña Ajustes/Usuarios y compañías/Grupos

Nombre del grupo	Filtros	Agrupar por	Favoritos
Técnico / Acceso a direcciones privadas			
Técnico / Acceso a la función de exportación			
Ventas / Administrador			
Proyecto / Administrador			
Facturación / Administrador de Facturación			
Técnico / Administrar Variantes de Productos			
Técnico / Administrar envasado de los productos			
Administración / Ajustes			
Bibliotecario			
Bibliotecario			
Técnico / Bloquear pedidos confirmados			
Permisos extra / Características técnicas			
Técnico / Contabilidad analítica			
Permisos extra / Creación de contactos			
Técnico / Descuentos en líneas			
Técnico / Dirección de entrega			
Técnico / Editor de plantillas de correo			
Evitar la depuración de campos HTML			
Facturación / Facturación			
Técnico / Facturas proforma			
Técnico / Gestionar múltiples unidades de medida			



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Una vez allí debemos crear un nuevo grupo Director, y dar los permisos de acceso que expone el enunciado para los modelos específicos del módulo gestión de instituto.

The screenshot shows the Odoo web interface at localhost:8069/web#cid=1&menu_id=4&action=69&model=res.groups&view_type=form. The page title is "Odoo - Nuevo". The top navigation bar includes links to Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and more. The main menu bar has items like Ajustes, Opciones generales, Usuarios y compañías, Traducciones, and Técnico. The current view is "Grupos / Nuevo". The main form has fields for "Nombre" (set to "Director") and "Compartir grupo" (unchecked). Below the form is a table titled "Permisos de acceso" with columns: Nombre, Modelo, Permiso para leer, Permiso de escritura, Acceso para crear, and Permiso para eliminar. The table lists four models: Modulo Formativo, Ciclo Formativo, Alumno, and Profesor, each with checked checkboxes in all columns. There is also a row for "Agregar línea" at the bottom of the table.

Nombre	Modelo	Permiso para leer	Permiso de escritura	Acceso para crear	Permiso para eliminar
EditarModulo	Modulo Formativo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
EditarCiclo	Ciclo Formativo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
EditarAlumno	Alumno	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
EditarProfesor	Profesor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Agregar línea					



5. Actividad 4 (Desarrollo)

5.2. Desarrollo

Por último creamos el grupo Profesor restringiendo los permisos donde se nos pide.

The screenshot shows the Odoo web interface for creating a new group. The browser address bar indicates the URL is `localhost:8069/web#cid=1&menu_id=4&action=69&model=res.groups&view_type=form&id=50`. The page title is "Odoo - Herramientas / Profesor". The top navigation bar includes links for Gmail, Drive, Trello, Calendar, Rayuela, Moodle, ClassRoom, GitHub, Design patterns, ChatGPT, TradingView, YouGlish, Investing, Fit, and more. The main menu has sections for Ajustes, Opciones generales, Usuarios y compañías, Traducciones, and Técnico. The current view is "Grupos / Herramientas / Profesor". The form fields for the new group include:

- Aplicación?: Herramientas
- Nombre?: Profesor
- Compartir grupo?:

The "Permisos de acceso" tab is selected, showing a grid of permissions for various models:

Nombre	Modelo	Permiso para leer	Permiso de escritura	Acceso para crear	Permiso para eliminar
EditarAlumno	Alumno	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
EditarCiclo	Ciclo Formativo	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
EditarModulo	Modulo Formativo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
EditarProfesor	Profesor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

At the bottom of the grid, there is a link "Agregar línea" (Add line) and a scrollbar.

BIBLIOGRAFÍA





6. Bibliografía

- <https://github.com/sergarb1/ApuntesSistemasGestionEmpresarial>
- <https://github.com/sergarb1/OdooModulosEjemplos>
- https://www.odoo.com/documentation/15.0/es/administration/odoo_sh/getting_started/first_module.html
- <https://vauxoo.github.io/odoo/howtos/backend.html>
- <https://www.odoo.com/documentation/14.0/developer/reference/addons/orm.html#fields>
- <https://www.odoo.com/documentation/14.0/developer/reference/addons/actions.html>
- https://github.com/odoo/odoo/blob/14.0/odoo/addons/base/data/ir_module_category_data.xml
- https://www.odoo.com/es_ES/forum/ayuda-1/how-defined-display-name-in-customer-many2one-91657
- https://www.odoo.com/documentation/14.0/es/developer/howtos/rdtraining/05_securityintro.html
- https://github.com/alex55fc/DAM2-SGE-OdooModules/tree/main/modulo_original_mio
- <https://www.youtube.com/watch?v=us2bBp5RD78>