

Practica 4 SGE

Manuel Añel García

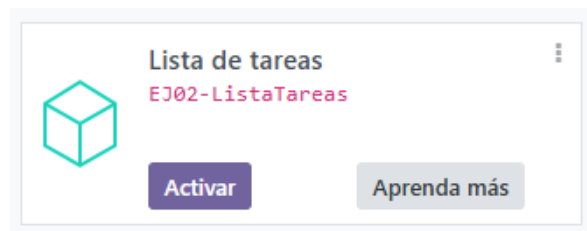
12 de diciembre de 2025

1. Introduction

En este documento se va a explicar el trabajo de SGE que consta de 4 actividades en las que se van a modificar, crear y ampliar módulos de Odoo, para poder empezar con esta práctica hay que reutilizar el contenedor que teníamos ya de Odoo y clonar el repositorio de módulos dicho en los requisitos de la práctica.

2. Actividad 01

Primero tenemos que activar el modulo en el que vamos a trabajar, buscamos "lista de tareas" activamos.



Tenemos que entrar en addons al modulo EJ02-ListaTareas y vamos a su view.xml para hacer otra vista pero en formato kanban, para empezar tenemos que agregar kanban en el view mode como se ve a continuación.

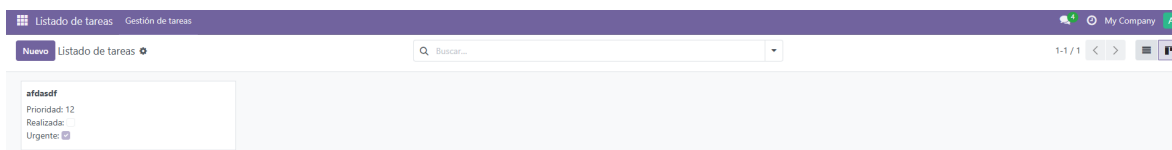
```
1 <field name="view_mode">list,form,kanban</field>
```

2.1. Vista en formato Kanban

Hay que crear un nuevo record que va a pertenecer al formato kanaban, en los comentarios del código está explicado el funcionamiento

```
1 <record id="view_lista_tareas_kanban" model="ir.ui.view">
2   <!-- Definimos el nombre interno, modelo odoo y arquitectura -->
3   <field name="name">lista.tareas.kanban</field>
4   <field name="model">lista.tareas.lista</field>
5   <field name="arch" type="xml">
6
7     <kanban>
8       <!-- Definimos los campos del modelo -->
9       <field name="tarea"/>
10      <field name="prioridad"/>
11      <field name="urgente"/>
12      <field name="realizada"/>
13
14      <templates>
15        <!-- Creamos una caja kanban -->
16        <t t-name="kanban-box">
17          <div class="oe_kanban_global_click o_kanban_record">
18
19            <!-- Título de la tarea -->
20            <div class="o_kanban_record_top">
21              <strong><t t-esc="record.tarea.value"/></strong>
22            </div>
23
24            <!-- Prioridad -->
25            <div>
26              Prioridad: <t t-esc="record.prioridad.value"/>
27            </div>
28
29            <!-- Checkbox realizada -->
30            <div>
31              Realizada: <field name="realizada"/>
32            </div>
33
34            <!-- Checkbox urgente -->
35            <div>
36              Urgente: <field name="urgente"/>
37            </div>
38
39          </div>
40        </t>
41      </templates>
42    </kanban>
43
44  </field>
45 </record>
```

Podemos ver como se ve la lista de tareas en formato kanban y como se puede alternar la vista arriba a la derecha



2.2. Fecha asignada y formato Calendar

Ahora hay que modificar las tareas para que tengan una fecha asignada y crear una nueva vista para visualizarlas en un formato Calendario según esa fecha. Empezamos poniendo en el archivo python un nuevo campo que va a ser fecha con formato Date.

```
1 fecha = fields.Date(string="Fecha")
```

Agregamos el campo fecha en la vista list y form

```
1 <field name="fecha"/>
```

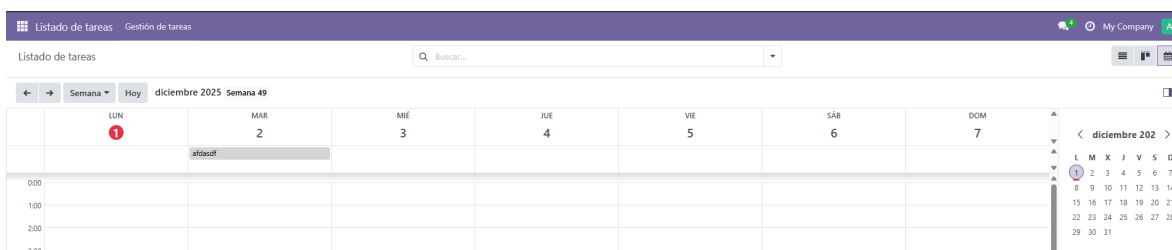
Donde antes pusimos kanban para para añadir el modo kanban hacemos lo mismo pero con calendar.

```
1 <field name="view_mode">list,form,kanban,calendar</field>
```

Ahora creamos el nuevo record para hacer la nueva vista

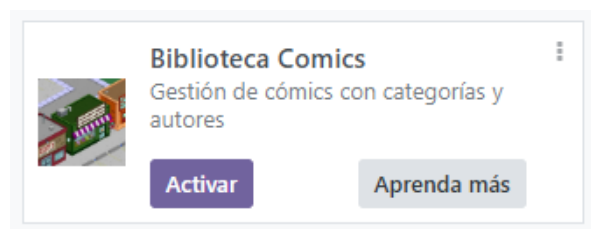
```
1 <record id="view_lista_tareas_calendar" model="ir.ui.view">
2   <!-- Definimos el nombre interno, modelo odoo y arquitectura -->
3   <field name="name">lista.tareas.calendar</field>
4   <field name="model">lista.tareas.lista</field>
5   <field name="arch" type="xml">
6     <!-- Creamos el calendario y decimos con el date_start que se coloque mediante
7       el campo fecha -->
8     <calendar string="Calendario de tareas" date_start="fecha">
9       <!-- Se muestra el nombre de la tarea -->
10      <field name="tarea"/>
11    </calendar>
12  </field>
</record>
```

Vemos como se muestra correctamente en la fecha que puse



3. Actividad 02

Primero hay que encontrar el modulo sobre el que vamos a trabajar que es y activarlo



3.1. Gestionar socios

Aquí lo que hacemos es crear un nuevo archivo .py para manejar los socios

```
1 class BibliotecaSocio(models.Model):
2     # Nombre interno del modelo en Odoo
3     _name = 'biblioteca.socio'
4     # Descripción del modelo
5     _description = 'Socio de la biblioteca'
6
7     # Campo para el nombre del socio, obligatorio
8     nombre = fields.Char(string='Nombre', required=True)
9     # Campo para el apellido del socio, obligatorio
10    apellido = fields.Char(string='Apellido', required=True)
11    # Campo para el identificador único del socio
12    identificador = fields.Char(string='ID_Socio', required=True, unique=True)
13
14    prestamo_ids = fields.One2many( # Relación One2many: un socio puede tener varios
15                                   ejemplares prestados
16                                   'biblioteca.ejemplar', # Es el modelo relacionado
17                                   'socio_id', # Es el campo en el modelo relacionado que apunta a este socio
18                                   string='Ejemplares_Prestados'
19    )
20
21    # Método para mostrar cómo se representa cada socio en campos Many2one
22    def name_get(self):
23        result = []
24        for record in self:
25            nombre_completo = f"{record.nombre}_{record.apellido}({record.
26                               identificador})"
27            result.append((record.id, nombre_completo))
28        return result
```

Ahora haay que crear una nueva vista xml para el modelo que acabamos de crear

```
1 <odoo>
2
3 <!-- Acción que abre la vista de Socios -->
4 <record id="biblioteca_socio_action" model="ir.actions.act_window">
5     <field name="name">Socios</field> <!-- Nombre de la acción -->
6     <field name="res_model">biblioteca.socio</field> <!-- Modelo que abre -->
7     <field name="view_mode">list,form</field> <!-- Tipos de vista -->
8     <field name="help" type="html">
9         <p>
10             Crea y gestiona los socios de la biblioteca.
11         </p>
12     </field>
13 </record>
14
15 <!-- Menú que apunta a la acción de Socios -->
16 <menuitem name="Socios" id="menu_biblioteca_socio" parent="biblioteca_base_menu"
17     action="biblioteca_socio_action"/>
18
19 <!-- Vista lista de socios -->
20 <record id="list" model="ir.ui.view">
21     <field name="name">Lista de Socios</field>
22     <field name="model">biblioteca.socio</field>
23     <field name="arch" type="xml">
24         <list>
25             <field name="nombre"/> <!-- Muestra el nombre -->
26             <field name="apellido"/> <!-- Muestra el apellido -->
27             <field name="identificador"/> <!-- Muestra el ID del socio -->
28         </list>
29     </field>
30 </record>
31
32 <!-- Vista formulario de socio -->
33 <record id="view_form_socio" model="ir.ui.view">
34     <field name="name">Formulario Socio</field>
35     <field name="model">biblioteca.socio</field>
36     <field name="arch" type="xml">
37         <form string="Socio de la Biblioteca">
38             <group>
39                 <field name="nombre"/> <!-- Campo nombre -->
40                 <field name="apellido"/> <!-- Campo apellido -->
41                 <field name="identificador"/> <!-- Campo ID socio -->
42                 <field name="prestamo_ids" readonly="1"/> <!-- Relación con los
43                     ejemplares prestados, solo lectura -->
44             </group>
45         </form>
46     </field>
47 </record>
48 </odoo>
```

Aquí podemos ver como se crean los socios

Aqui podemos ver un socio que tiene un ejemplar, esto ya lo veremos mas adelante

Mi biblioteca

Cómic

Categorías

Socios

Ejemplares

Nuevo

Socios

biblioteca.socio,1

1 / 1

<

>

Nombre

Manuel

Apellido

Añel

ID Socio

8944

Ejemplares Prestados

Cómic	Prestado a	Estado	Fecha de ...	Fecha pre...
El quijote	biblioteca.socio,1	Disponible	09/12/2025	11/12/2025

3.2. Ejemplares de comics

Ahora hay que crear un nuevo modelo para los ejemplares y tener en cuenta las fechas de préstamo.

```
1 class BibliotecaEjemplar(models.Model):
2     _name = 'biblioteca.ejemplar' # Nombre técnico del modelo
3     _description = 'Ejemplar de cómic prestable' # Descripción del modelo
4
5     # Campo Many2One que enlaza con el cómic al que pertenece este ejemplar (un comic
6     # puede tener varios ejemplares)
7     comic_id = fields.Many2one(
8         'biblioteca.comic',
9         string='Cómic', # Etiqueta que se muestra en la interfaz
10        required=True, # Campo obligatorio
11        ondelete='cascade' # Si se borra el cómic, se borran sus ejemplares
12    )
13
14    # Campo Many2One que enlaza con el socio que tiene prestado el ejemplar (un socio
15    # puede tener varios ejemplares)
16    socio_id = fields.Many2one(
17        'biblioteca.socio',
18        string='Prestado a' # Etiqueta en la interfaz
19    )
20
21    # Fecha en que se realizó el préstamo
22    fecha_prestamo = fields.Date(string='Fecha de préstamo')
23    # Fecha prevista para devolver el ejemplar
24    fecha_devolucion_prevista = fields.Date(string='Fecha prevista de devolución')
25
26    # Estado del ejemplar: disponible o prestado
27    estado = fields.Selection(
28        [('disponible', 'Disponible'),
29         ('prestado', 'Prestado')],
30        string='Estado',
31        default='disponible' # Valor por defecto
32    )
33
34    # La fecha de préstamo no puede ser futura
35    @api.constrains('fecha_prestamo')
36    def _check_fecha_prestamo(self):
37        hoy = fields.Date.today() # Fecha actual
38        for record in self:
39            if record.fecha_prestamo and record.fecha_prestamo > hoy:
40                raise ValidationError(
41                    'La fecha de préstamo no puede ser posterior al día actual.'
42                )
43
44    # La fecha prevista de devolución no puede ser anterior a hoy
45    @api.constrains('fecha_devolucion_prevista')
46    def _check_fecha_devolucion(self):
47        hoy = fields.Date.today() # Fecha actual
48        for record in self:
49            if record.fecha_devolucion_prevista and record.fecha_devolucion_prevista <
50                hoy:
51                raise ValidationError(
52                    'La fecha prevista de devolución no puede ser anterior al día
53                    actual.'
54                )
55
56    # Cambio automático del estado cuando se selecciona o quita un socio
57    @api.onchange('socio_id')
58    def _onchange_socio(self):
59        for record in self:
60            if record.socio_id: # Si hay un socio asignado
61                record.estado = 'prestado' # Cambia a prestado
62            else:
63                record.estado = 'disponible' # Si no hay socio, está disponible
```

Ahora hay que crear una nueva vista para el modelo que acabamos de crear

```
1 <odoo>
2
3 <!-- Acción para abrir la vista de Ejemplares (listado y formulario) -->
4 <record id="biblioteca_ejemplar_action" model="ir.actions.act_window">
5   <field name="name">Ejemplares de Cómic</field> <!-- Nombre que verá el
6     usuario -->
7   <field name="res_model">biblioteca.ejemplar</field> <!-- Modelo al que apunta
8     -->
9   <field name="view_mode">list,form</field> <!-- Vistas disponibles: lista y
10     formulario -->
11 </record>
12
13 <!-- Menú que abre la acción de Ejemplares dentro del menú base "Mi biblioteca"
14 -->
15 <menuitem name="Ejemplares"
16   id="menu_biblioteca_ejemplar"
17   parent="biblioteca_base_menu"
18   action="biblioteca_ejemplar_action"/>
19
20 <!-- Vista Formulario para crear o editar un ejemplar -->
21 <record id="view_form_ejemplar" model="ir.ui.view">
22   <field name="name">Formulario Ejemplar</field> <!-- Nombre interno de la vista
23     -->
24   <field name="model">biblioteca.ejemplar</field> <!-- Modelo asociado -->
25   <field name="arch" type="xml">
26     <form string="Ejemplar de Cómic">
27       <group>
28         <field name="comic_id"/> <!-- Relación con el cómic
29           correspondiente -->
30         <field name="socio_id"/> <!-- Relación con el socio que lo tiene
31           prestado -->
32         <field name="estado" readonly="1"/> <!-- Estado del ejemplar (solo
33           lectura) -->
34         <field name="fecha_prestamo"/> <!-- Fecha del préstamo -->
35         <field name="fecha_devolucion_prevista"/> <!-- Fecha prevista de
36           devolución -->
37       </group>
38     </form>
39   </field>
40 </record>
41
42 <!-- Vista Lista para ver todos los ejemplares en forma de tabla -->
43 <record id="view_list_ejemplar" model="ir.ui.view">
44   <field name="name">Lista Ejemplares</field> <!-- Nombre interno de la vista
45     -->
46   <field name="model">biblioteca.ejemplar</field> <!-- Modelo asociado -->
47   <field name="arch" type="xml">
48     <list>
49       <field name="comic_id"/> <!-- Columna: Cómic -->
50       <field name="socio_id"/> <!-- Columna: Socio -->
51       <field name="estado"/> <!-- Columna: Estado -->
52       <field name="fecha_prestamo"/> <!-- Columna: Fecha de préstamo -->
53       <field name="fecha_devolucion_prevista"/> <!-- Columna: Fecha prevista
54         de devolución -->
55     </list>
56   </field>
57 </record>
58 </odoo>
```

Aquí tuve un fallo porque hay que acordarse de los modelos nuevos meterlos aquí

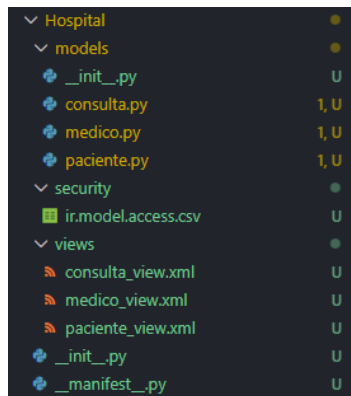
```
from . import biblioteca_comic
from . import biblioteca_comic_categoria
from . import biblioteca_socio
from . import biblioteca_ejemplar
```


Y esto lo mismo que hay que acordarse de meter las vistas nuevas

```
'data': [  
    'security/ir.model.access.csv',  
    'views/biblioteca_comic.xml',  
    'views/biblioteca_comic_categoria.xml',  
    'views/biblioteca_socio.xml',  
    'views/biblioteca_ejemplar.xml',  
],
```

4. Actividad 03

En esta actividad hay que crear un nuevo módulo desde 0 en el que gestione un hospital con sus pacientes, médicos y consultas. Para empezar hay que crear una estructura simple de carpetas y archivos para nuestro módulo, como se ve a continuación.



Ahora hay que rellenar los archivos básicos del módulo para que funcione el módulo y este todo enlazado. Hay que tener muy en cuenta que las nuevas vistas hay que ponerlas en el manifest y los nuevos modelos en el init dentro de models.

```
Odoo > addons > Hospital > __manifest__.py  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
  
{  
    'name': 'Hospital',  
    'version': '1.0',  
    'author': 'Manuel',  
    'category': 'Tools',  
    'summary': 'Gestión de Hospital',  
    'depends': ['base'],  
    'data': [  
        'security/ir.model.access.csv',  
        'views/paciente_view.xml',  
        'views/medico_view.xml',  
        'views/consulta_view.xml',  
    ],  
    'installable': True,  
    'application': True,  
}
```

```
Odoo > addons > Hospital > __init__.py  
1 from . import models
```

```
Odoo > addons > Hospital > models > __init__.py  
1 from . import paciente  
2 from . import medico  
3 from . import consulta
```

También hay que tener en cuenta que hay que rellenar el archivo csv dentro de security para los permisos de los modelos

```
Odoo > addons > Hospital > security > ir.model.access.csv > data
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 access_paciente,paciente_access,model_hospital_paciente,,1,1,1,1
3 access_medico,medico_access,model_hospital_medico,,1,1,1,1
4 access_consulta,consulta_access,model_hospital_consulta,,1,1,1,1
```

Ahora vamos a crear los modelos y las vistas, la explicación del código está en los comentarios del mismo

4.1. Pacientes

Modelo paciente

```
1 class Paciente(models.Model):
2     _name = 'hospital.paciente'
3     _description = 'Paciente del hospital'
4
5     name = fields.Char('Nombre y Apellidos', required=True) # El char se diferencia del
6     sintomas = fields.Text('Síntomas') # text de que tiene un límite de caracteres de 255 y text no
7
8     # Es una relación varios a varios, un paciente puede tener varios médicos y viceversa
9     consulta_ids = fields.Many2many('hospital.medico', # Modelo destino (el otro modelo)
10                                     'consulta_paciente_medico', # Nombre de la tabla
11                                     'paciente_id', 'medico_id', # Columnas en la tabla
12                                     'intermedia', # intermedia en la base de datos
13                                     string='Médicos que lo atendieron')
```

Vista paciente

```

1 <odoo>
2   <record id="view_form_paciente" model="ir.ui.view">
3     <field name="name">paciente.form</field>
4     <field name="model">hospital.paciente</field>
5     <field name="arch" type="xml">
6       <form string="Paciente"> <!-- Utilizamos el form para la creacion del
7         paciente -->
8         <sheet>
9           <group>
10            <field name="name"/> <!-- Eston son los atributos que se van a
11              rellenar al crear el paciente -->
12            <field name="sintomas"/>
13            <field name="consulta_ids"/>
14          </group>
15        </sheet>
16      </form>
17    </field>
18  </record>
19
20  <record id="view_list_paciente" model="ir.ui.view"> <!-- La lista se usa para la
21    visualización de los pacientes como en una lista -->
22    <field name="name">paciente.list</field>
23    <field name="model">hospital.paciente</field>
24    <field name="arch" type="xml">
25      <list string="Pacientes">
26        <field name="name"/> <!-- La informacion que se va a ver en la lista
27          solamente va a ser el nombre -->
28      </list>
29    </field>
30  </record>
31
32  <record id="action_paciente" model="ir.actions.act_window"> <!-- Aqui se determina
33    lo que va haber en la vista que es un form y una list -->
34    <field name="name">Pacientes</field>
35    <field name="res_model">hospital.paciente</field>
36    <field name="view_mode">list,form</field>
37  </record>
38
39  <menuitem id="menu_hospital_root" name="Hospital"/> <!-- Aqui creamos básicamente
40    el boton del menu de arriba para cambiar entre pacientes, medicos etc... -->
41  <menuitem id="menu_paciente" name="Pacientes" parent="menu_hospital_root" action="
42    action_paciente"/>
43 </odoo>

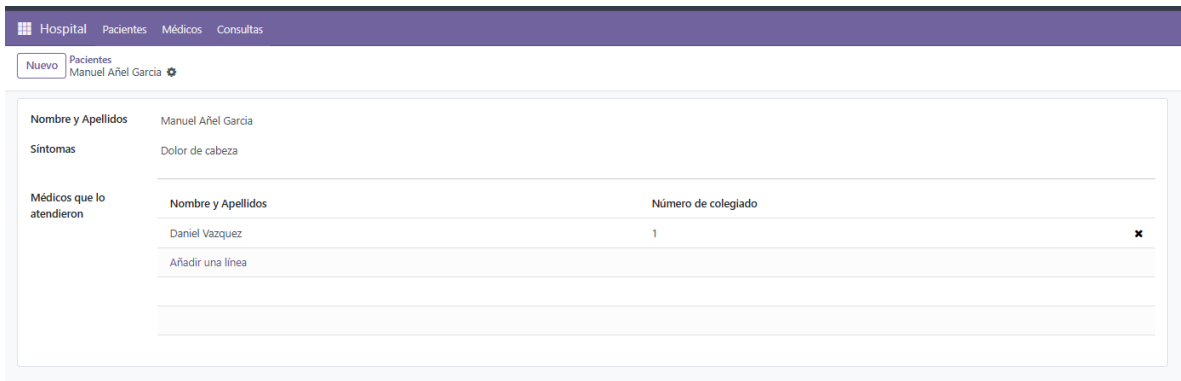
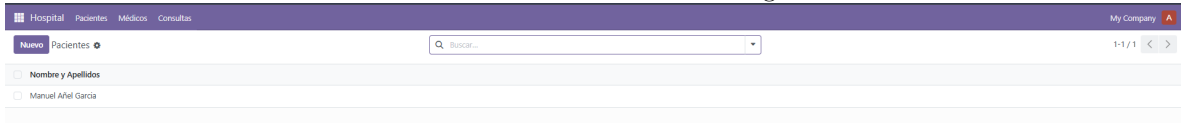
```

Aqui vemos como crear un paciente

The screenshot shows the 'Nuevo' (New) patient form in the Odoo Hospital application. The form is titled 'Nuevo Paciente' and contains the following fields:

- Nombre y Apellidos:** Manuel Añel García
- Síntomas:** Dolor de cabeza
- Médicos que lo atendieron:** A table with two columns: 'Nombre y Apellidos' and 'Número de colegiado'. The table has one row with the text 'Añadir una línea'.

Aquí vemos como se visualizan los pacientes desde la lista y el formulario, en el formulario se puede ver la relacion con el medico al haber creado la consulta gracias a la funcion definida



4.2. Medicos

Modelo medico

```
1 class Medico(models.Model):
2     _name = 'hospital.medico'
3     _description = 'Medico del hospital'
4
5     name = fields.Char('Nombre y Apellidos', required=True)
6
7     numero_colegiado = fields.Char('Número de colegiado')
8
9     # Es una relacion varios a varios, un paciente puede tener varios medicos y
10    viceversa
11    consulta_ids = fields.Many2many('hospital.paciente', # Modelo destino (paciente)
12                                   'consulta_paciente_medico', # Nombre de la tabla
13                                   intermedia en la base de datos
14                                   'medico_id', 'paciente_id', # Columnas en la tabla
15                                   intermedia
16                                   string='Pacientes atendidos')
```

Vista medico

```
1 <odoo>
2   <record id="view_form_medico" model="ir.ui.view">
3     <field name="name">medico.form</field>
4     <field name="model">hospital.medico</field>
5     <field name="arch" type="xml">
6       <form string="Médico"> <!-- Utilizamos el form para la creacion del medico -->
7         <sheet>
8           <group>
9             <field name="name"/> <!-- Eston son los atributos que se van a
10              rellenar al crear el medico -->
11             <field name="numero_colegiado"/>
12             <field name="consulta_ids"/>
13           </group>
14         </sheet>
15       </form>
16     </field>
17   </record>
18
19   <record id="view_list_medico" model="ir.ui.view"> <!-- La lista se usa para la
20     visualización de los medicos como en una lista -->
21     <field name="name">medico.list</field>
22     <field name="model">hospital.medico</field>
23     <field name="arch" type="xml">
24       <list string="Médicos">
25         <field name="name"/> <!-- La informacion que se va a ver en la lista
26           solamente va a ser el nombre y el numero de colegiado -->
27         <field name="numero_colegiado"/>
28       </list>
29     </field>
30   </record>
31
32   <record id="action_medico" model="ir.actions.act_window"> <!-- Aqui se determina
33     lo que va haber en la vista que es un form y una list -->
34     <field name="name">Médicos</field>
35     <field name="res_model">hospital.medico</field>
36     <field name="view_mode">list,form</field>
37   </record>
38
39   <!-- Aqui creamos básicamente el boton del menu de arriba para cambiar entre
40     pacientes, medicos etc... -->
41   <menuitem id="menu_medico" name="Médicos" parent="menu_hospital_root" action="
42     action_medico"/>
43 </odoo>
```

Aqui vemos como crear un medico

The screenshot shows the 'Nuevo Médico' (New Doctor) form in the Odoo Hospital application. The form is titled 'Nuevo Médico' and has a 'Nuevo' button. It contains the following fields:

- Nombre y Apellidos:** Daniel Vazquez
- Número de colegiado:** 1
- Pacientes atendidos:** A table with one header row 'Nombre y Apellidos' and one data row 'Añadir una línea'.

Aquí vemos como se visualizan los medicos desde la lista y el formulario, en el formulario se puede ver la relacion con el paciente al haber creado la consulta gracias a la funcion definida

Hospital Pacientes Médicos Consultas	
<div>Nuevo Médicos</div> <div> <input type="text"/> </div>	
Nombre y Apellidos	Número de colegiado
<input type="checkbox"/> Daniel Vazquez	1

Hospital Pacientes Médicos Consultas

Nuevo Médicos

Daniel Vazquez

Nombre y Apellidos
Daniel Vazquez

Número de colegiado
1

Pacientes atendidos

Nombre y Apellidos

Manuel Afel Garcia

Añadir una línea

4.3. Consulta

Modelo consulta, en este modelo es importante la funcion de crear los registros porque al hacer la consulta no se actualiza automatico en la relacion de paciente medico así que lo hacemos con la funcion

```

1 class Consulta(models.Model):
2     _name = 'hospital.consulta'
3     _description = 'Consulta de un paciente con un médico'
4
5     paciente_id = fields.Many2one('hospital.paciente', string='Paciente', required=
6         True) # Relacion de que un paciente puede tener varias consultas y al reves no
7     medico_id = fields.Many2one('hospital.medico', string='Médico', required=True) #
8         Relacion de que un medico puede tener varias consultas y al reves no
9     diagnostico = fields.Text('Diagnóstico') # Texto en el que se pone el diagnóstico
10    fecha = fields.Datetime('Fecha', default=fields.Datetime.now) # Ponemos la fecha
11        para el momento en el que se crea la consulta
12
13    @api.model
14    def create(self, vals):
15        # Crear la consulta
16        record = super().create(vals)
17        # Actualizar Many2many del paciente-medico
18        if record.paciente_id and record.medico_id: # Verifica si estan los valores
19            llenos
20            record.paciente_id.write({'consulta_ids': [(4, record.medico_id.id)]}) #
21                Creamos los registros, el 4 es para añadir sin borrar nada
22            record.medico_id.write({'consulta_ids': [(4, record.paciente_id.id)]})
23        return record

```

Vista consulta

```
1 <odoo>
2   <record id="view_form_consulta" model="ir.ui.view">
3     <field name="name">consulta.form</field>
4     <field name="model">hospital.consulta</field>
5     <field name="arch" type="xml">
6       <form string="Consulta"> <!-- Utilizamos el form para la creacion de la
7         consulta -->
8         <sheet>
9           <group>
10             <field name="paciente_id"/> <!-- Eston son los atributos que
11               se van a rellenar al crear la consulta -->
12             <field name="medico_id"/>
13             <field name="diagnostico"/>
14             <field name="fecha"/> <!-- La fecha se pone sola -->
15           </group>
16         </sheet>
17       </form>
18     </field>
19   </record>
20
21   <record id="view_list_consulta" model="ir.ui.view"> <!-- La lista se usa para la
22     visualización de las consultas como en una lista -->
23     <field name="name">consulta.list</field>
24     <field name="model">hospital.consulta</field>
25     <field name="arch" type="xml">
26       <list string="Consultas">
27         <field name="paciente_id"/> <!-- La informacion que se va a ver en la
28           lista -->
29         <field name="medico_id"/>
30         <field name="fecha"/>
31       </list>
32     </field>
33   </record>
34
35   <record id="action_consulta" model="ir.actions.act_window"> <!-- Aqui se determina
36     lo que va haber en la vista que es un form y una list -->
37     <field name="name">Consultas</field>
38     <field name="res_model">hospital.consulta</field>
39     <field name="view_mode">list,form</field>
40   </record>
41
42   <!-- Aqui creamos básicamente el boton del menu de arriba para cambiar entre
43     pacientes, medicos y consultas -->
44   <menuitem id="menu_consulta" name="Consultas" parent="menu_hospital_root" action="
45     action_consulta"/>
46 </odoo>
```

Aqui vemos como crear una consulta con los pacientes y medicos creados gracias a las relaciones

Hospital Pacientes Médicos Consultas

Nuevo Consultas

Nuevo

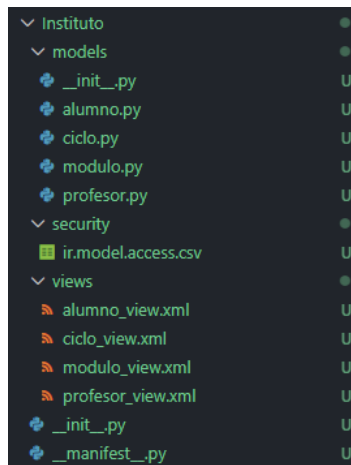
Paciente	Manuel Afel Garcia
Médico	Daniel Vazquez
Diagnóstico	Paracetamol 1g
Fecha	11/12/2025 17:36:17

Aquí podemos ver las consultas creadas

Hospital Pacientes Médicos Consultas			My Company	
Nuevo Consultas		Buscar...		1-1/1 < >
Paciente		Médico		Fecha
Manuel Añel García		Daniel Vazquez		11/12/2023 18:17:59

5. Actividad 04

En esta actividad como la anterior hay que crear un módulo desde cero que gestione un instituto, con sus profesores, alumnos, ciclos y módulos. Para empezar tenemos que crear la estructura de carpetas de nuestro módulo con sus modelos y sus vistas correspondientes además de sus archivos de configuración necesarios.



Archivos de configuración necesarios para el funcionamiento e instalación del módulo: Este es el manifest, aquí para que funcione es fundamental poner todas las vistas que tengamos

```
Odoo > addons > Instituto > __manifest__.py
1
2 'name': 'Instituto',
3 'version': '1.0',
4 'author': 'Manuel',
5 'category': 'Tools',
6 'summary': 'Gestión de ciclos, módulos, alumnos y profesore',
7 'depends': ['base'],
8 'data': [
9     'security/ir.model.access.csv',
10     'views/ciclo_view.xml',
11     'views/modulo_view.xml',
12     'views/alumno_view.xml',
13     'views/profesor_view.xml',
14 ],
15 'installable': True,
16 'application': True,
17
```

Este es el init, aquí hacemos que se refiera a la carpeta de models que tiene otro init que se ejecutara

```
Odoo > addons > Instituto > __init__.py
1 from . import models
```

Este es el init dentro de models que apunta a todos los modelos que tengamos

```
Odoo > addons > Instituto > models > __init__.py
1 from . import alumno
2 from . import modulo
3 from . import ciclo
4 from . import profesor
```


Ahora vamos a rellenar los archivos de los modelos y vistas para que funcione, la explicación del código está en los comentarios de El

Modulo ciclo

```
1 class Ciclo(models.Model):
2     _name = 'instituto.ciclo'
3     _description = 'Ciclo'
4
5     # Estos son los atributos del ciclo
6     name = fields.Char(string='Nombre del ciclo', required=True)
7     descripcion = fields.Text(string='Descripción')
8
9     # Relacion con modulos, un ciclo puede tener varios modulos y un modulo solo puede
10    estar en un ciclo
11    modulo_ids = fields.One2many('instituto.modulo', 'ciclo_id', string='Módulos')
```

Modulo modulo formativo

```
1 class Modulo(models.Model):
2     _name = 'instituto.modulo'
3     _description = 'Módulo del ciclo formativo'
4
5     # Estos son los atributos del modulo
6     name = fields.Char(string='Nombre del módulo', required=True)
7     codigo = fields.Char(string='Código')
8
9     # Relacion con ciclo, un ciclo puede tener varios modulos y un modulo solo puede
10    estar en un ciclo
11    ciclo_id = fields.Many2one('instituto.ciclo', string='Ciclo formativo', ondelete='cascade')
12
13    # Relacion con profesor, un profesor puede impartir varios modulos y un modulo
14    solo puede ser impartido por un profesor
15    profesor_id = fields.Many2one('instituto.profesor', string='Profesor')
16
17    # Relacion varios a varios con modulos-alumnos
18    alumno_ids = fields.Many2many(
19        'instituto.alumno',
20        'modulo_alumno_rel', # Nueva tabla generada
21        'modulo_id', # Atributo de la nueva tabla
22        'alumno_id', # Atributo de la nueva tabla
23        string='Alumnos matriculados'
24    )
```

Modulo alumno

```
1 class Alumno(models.Model):
2     _name = 'instituto.alumno'
3     _description = 'Alumno'
4
5     # Estos son los atributos del modulo
6     name = fields.Char(string='Nombre y apellidos', required=True)
7     dni = fields.Char(string='DNI/NIE')
8     fecha_nacimiento = fields.Date(string='Fecha de nacimiento')
9
10    # Relacion varios a varios con alumnos-modulos
11    modulo_ids = fields.Many2many(
12        'instituto.modulo',
13        'modulo_alumno_rel', # Nueva tabla generada
14        'alumno_id', # Atributo de la nueva tabla
15        'modulo_id', # Atributo de la nueva tabla
16        string='Módulos matriculados'
17    )
```

Modulo profesor

```
1 class Profesor(models.Model):
2     _name = 'instituto.profesor'
3     _description = 'Profesor'
4
5     # Estos son los atributos del modulo
6     name = fields.Char(string='Nombre y apellidos', required=True)
7     departamento = fields.Char(string='Departamento')
8     numero_colegiado = fields.Char(string='Número de identificación laboral')
9
10    # Relacion con modulos, un profesor puede impartir varios modulos y un modulo solo
11        puede ser impartido por un profesor
12    modulo_ids = fields.One2many('instituto.modulo', 'profesor_id', string='Módulos impartidos')
```

Vista ciclo

```
1 <odoo>
2 <!-- Aqui se hace un formulario para crear nuevos ciclos -->
3 <record id="view_ciclo_form" model="ir.ui.view">
4   <field name="name">instituto.ciclo.form</field>
5   <field name="model">instituto.ciclo</field>
6   <field name="arch" type="xml">
7     <form string="Ciclo Formativo">
8       <sheet>
9         <!-- Atributos para crear ciclo -->
10        <group>
11          <field name="name"/>
12          <field name="descripcion"/>
13        </group>
14        <!-- Esto es para poder crear un nuevo modulo desde el ciclo que
15          estemos creando o editando -->
16        <notebook>
17          <page string="Módulos">
18            <field name="modulo_ids">
19              <list>
20                <field name="name"/>
21                <field name="codigo"/>
22                <field name="profesor_id"/>
23              </list>
24            </field>
25          </page>
26        </notebook>
27      </sheet>
28    </form>
29  </field>
30 </record>
31 <!-- Aqui se hace la visualización en lista -->
32 <record id="view_ciclo_list" model="ir.ui.view">
33   <field name="name">instituto.ciclo.list</field>
34   <field name="model">instituto.ciclo</field>
35   <field name="arch" type="xml">
36     <list string="Ciclos">
37       <!-- Atributos que se ven en la lista, el many2many_tags es una forma
38         de visualizacion sencilla para poder ver varias modulos en un ciclo
39         -->
40       <field name="name"/>
41       <field name="modulo_ids" widget="many2many_tags"/>
42     </list>
43   </field>
44 </record>
45 <!-- Aqui se determina lo que va haber en la vista que es un form y una list -->
46 <record id="action_ciclo" model="ir.actions.act_window">
47   <field name="name">Ciclos formativos</field>
48   <field name="res_model">instituto.ciclo</field>
49   <field name="view_mode">list,form</field>
50 </record>
51 <!-- Aqui creamos el boton del menu de arriba para cambiar entre vistas -->
52 <menuitem id="menu_instituto_root" name="Instituto"/>
53 <menuitem id="menu_instituto_ciclos" name="Ciclos Formativos" parent="
54   menu_instituto_root" action="action_ciclo"/>
55 </odoo>
```

Vista alumno

```
1 <odoo>
2 <!-- Aqui se hace un formulario para crear nuevos alumnos -->
3 <record id="view_alumno_form" model="ir.ui.view">
4   <field name="name">instituto.alumno.form</field>
5   <field name="model">instituto.alumno</field>
6   <field name="arch" type="xml">
7     <form string="Alumno">
8       <sheet>
9         <group>
10          <!-- Atributos para crear alumnos -->
11          <field name="name"/>
12          <field name="dni"/>
13          <field name="fecha_nacimiento"/>
14        </group>
15        <group>
16          <!-- Es un atributo para añadir modulos a los alumnos, se ven
17            de la forma many2many -->
18          <field name="modulo_ids" widget="many2many_tags"/>
19        </group>
20      </sheet>
21    </form>
22  </field>
23 </record>
24 <!-- Aqui se hace la visualización en lista -->
25 <record id="view_alumno_list" model="ir.ui.view">
26   <field name="name">instituto.alumno.list</field>
27   <field name="model">instituto.alumno</field>
28   <field name="arch" type="xml">
29     <list string="Alumnos">
30       <!-- Atributos que se ven en la lista -->
31       <field name="name"/>
32       <field name="dni"/>
33     </list>
34   </field>
35 </record>
36
37 <!-- Aqui se determina lo que va haber en la vista que es un form y una list -->
38 <record id="action_alumno" model="ir.actions.act_window">
39   <field name="name">Alumnos</field>
40   <field name="res_model">instituto.alumno</field>
41   <field name="view_mode">list,form</field>
42 </record>
43
44 <!-- Aqui creamos el boton del menu de arriba para cambiar entre vistas -->
45 <menuitem id="menu_instituto_alumnos" name="Alumnos" parent="menu_instituto_root"
46   action="action_alumno"/>
47 </odoo>
```

Vista profesor

```
1 <odoo>
2 <!-- Aqui se hace un formulario para crear nuevos profesores -->
3 <record id="view_profesor_form" model="ir.ui.view">
4   <field name="name">instituto.profesor.form</field>
5   <field name="model">instituto.profesor</field>
6   <field name="arch" type="xml">
7     <form string="Profesor">
8       <sheet>
9         <group>
10          <!-- Atributos para crear profesores -->
11          <field name="name"/>
12          <field name="departamento"/>
13          <field name="numero_colegiado"/>
14        </group>
15        <group>
16          <!-- Es un atributo para añadir modulos a los profesores, se
17           ven de la forma many2many -->
18          <field name="modulo_ids" widget="many2many_tags" string="Módulos que imparte"/>
19        </group>
20      </sheet>
21    </form>
22  </field>
23 </record>
24 <!-- Aqui se hace la visualización en lista -->
25 <record id="view_profesor_list" model="ir.ui.view">
26   <field name="name">instituto.profesor.list</field>
27   <field name="model">instituto.profesor</field>
28   <field name="arch" type="xml">
29     <list string="Profesores">
30       <!-- Atributos que se ven en la lista -->
31       <field name="name"/>
32       <field name="departamento"/>
33     </list>
34   </field>
35 </record>
36
37 <!-- Aqui se determina lo que va haber en la vista que es un form y una list -->
38 <record id="action_profesor" model="ir.actions.act_window">
39   <field name="name">Profesores</field>
40   <field name="res_model">instituto.profesor</field>
41   <field name="view_mode">list,form</field>
42 </record>
43
44 <!-- Aqui creamos el boton del menu de arriba para cambiar entre vistas -->
45 <menuitem id="menu_instituto_profesores" name="Profesores" parent="
46   menu_instituto_root" action="action_profesor"/>
46 </odoo>
```

Vista modulo

```
1 <odoo>
2     <!-- Aqui se hace un formulario para crear nuevos modulos -->
3     <record id="view_modulo_form" model="ir.ui.view">
4         <field name="name">instituto.modulo.form</field>
5         <field name="model">instituto.modulo</field>
6         <field name="arch" type="xml">
7             <form string="Módulo">
8                 <sheet>
9                     <group>
10                        <!-- Atributos para crear modulos -->
11                        <field name="name"/>
12                        <field name="codigo"/>
13                    </group>
14                    <group>
15                        <!-- Atributos para crear modulos que son relaciones -->
16                        <field name="ciclo_id"/>
17                        <field name="profesor_id"/>
18                    </group>
19                    <group>
20                        <!-- Es un atributo para añadir alumnos a los modulos, se ven
21                         de la forma many2many -->
22                        <field name="alumno_ids" widget="many2many_tags"/>
23                    </group>
24                </sheet>
25            </form>
26        </field>
27    </record>
28
29    <!-- Aqui se hace la visualización en lista -->
30    <record id="view_modulo_list" model="ir.ui.view">
31        <field name="name">instituto.modulo.list</field>
32        <field name="model">instituto.modulo</field>
33        <field name="arch" type="xml">
34            <list string="Módulos">
35                <!-- Atributos que se ven en la lista -->
36                <field name="name"/>
37                <field name="codigo"/>
38                <field name="ciclo_id"/>
39                <field name="profesor_id"/>
40            </list>
41        </field>
42    </record>
43
44    <!-- Aqui se determina lo que va haber en la vista que es un form y una list -->
45    <record id="action_modulo" model="ir.actions.act_window">
46        <field name="name">Módulos</field>
47        <field name="res_model">instituto.modulo</field>
48        <field name="view_mode">list,form</field>
49    </record>
50
51    <!-- Aqui creamos el boton del menu de arriba para cambiar entre vistas -->
52    <menuitem id="menu_instituto_modulos" name="Módulos" parent="menu_instituto_root"
53        action="action_modulo"/>
54 </odoo>
```