

# Arquitectura Aplicacion Movil

Manuel Añel García

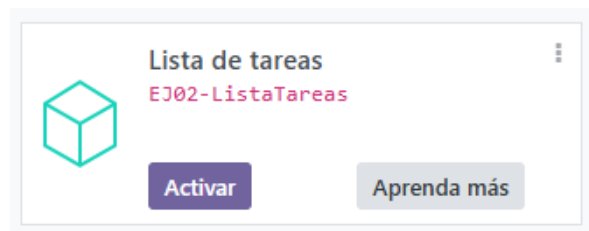
11 de diciembre de 2025

## 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>
```

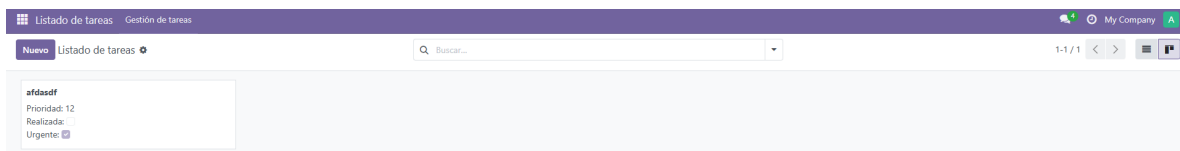
Hay que crear un nuevo record que va a pertenecer al formato kanaban, en los comentarios del código está explicado el funcionamiento

```

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



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"/>
```

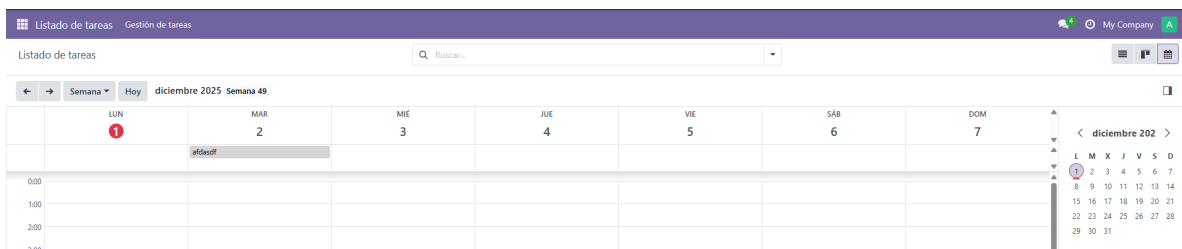
Donde antes pusimos kanban para para añadir el modo kanban hacemos lo mismo pero con calendar.

```
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>
```

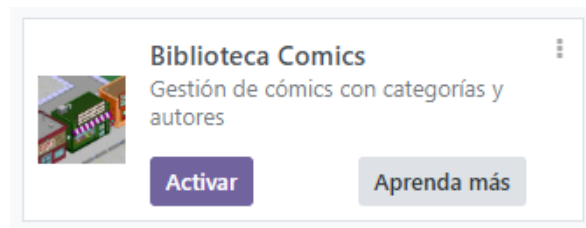
Vemos como se muestra correctamente en la fecha que puse



	LUN	MAR	MIE	JUE	VIE	SAB	DOM
	1	2	3	4	5	6	7
	atascos						
0:00							
1:00							
2:00							
3:00							

### 3. Actividad 02

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



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

Nombre

Apellido

ID Socio

Ejemplares Prestados	Cómic	Prestado a	Estado	Fecha de ... Fecha pre...

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

Apellido

ID Socio

Ejemplares Prestados

Manuel

Añel

8944

Cómic

Prestado a

Estado

Fecha de ...

Fecha pre...

El quijote

biblioteca.socio,1

Disponible

09/12/2025

11/12/2025

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='Cómico', # 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
```

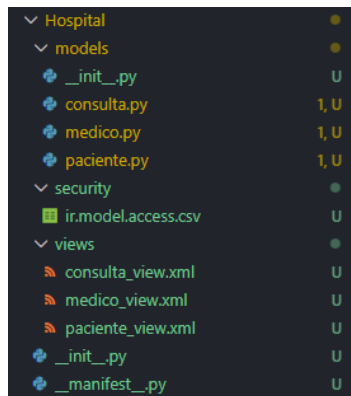


Y esto lo mismo que hay que acordarse de meter las vistas nuevas

```
'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
```

Tambien 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 explicacion del código está en los comentarios del mismo

#### 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('Sintomas') # text de que tiene un limite de caracteres de 255 y text no
7
8     # Es una relacion varios a varios, un paciente puede tener varios medicos 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
13                                     string='Médicos que lo atendieron')
```

#### 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                                    'medico_id', 'paciente_id', # Columnas en la tabla
14                                    'intermedia', # intermedia
15                                    string='Pacientes atendidos')
```

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 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>
```

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

## 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 un paciente

Hospital Pacientes Médicos Consultas

Nuevo Pacientes

Nombre y Apellidos Manuel Ariel Garcia

Síntomas Dolor de cabeza

Médicos que lo atendieron

Nombre y Apellidos	Número de colegiado
Añadir una línea	

Aqui vemos como crear un medico

The screenshot shows the 'Nuevo Medico' form. The top navigation bar includes 'Hospital', 'Pacientes', 'Médicos', and 'Consultas'. The form has a 'Nuevo' button and a 'Médicos' dropdown. The form fields are:

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

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

The screenshot shows the 'Nueva Consulta' form. The top navigation bar includes 'Hospital', 'Pacientes', 'Médicos', and 'Consultas'. The form has a 'Nuevo' button and a 'Consultas' dropdown. The form fields are:

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

Aqui 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

The screenshot shows the 'Pacientes' list view. The top navigation bar includes 'Hospital', 'Pacientes', 'Médicos', and 'Consultas'. The list has a 'Nuevo' button and a 'Pacientes' dropdown. The list items are:

- ☐ Nombre y Apellidos
- ☐ Manuel Añel Garcia

The screenshot shows the 'Pacientes' form. The top navigation bar includes 'Hospital', 'Pacientes', 'Médicos', and 'Consultas'. The form has a 'Nuevo' button and a 'Pacientes' dropdown. The form fields are:

- Nombre y Apellidos:** Manuel Añel Garcia
- Síntomas:** Dolor de cabeza
- Médicos que lo atendieron:** A table with one row containing 'Nombre y Apellidos' and 'Número de colegiado'.

Aqui 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

The screenshot shows the 'Médicos' list view. The top navigation bar includes 'Hospital', 'Pacientes', 'Médicos', and 'Consultas'. The list has a 'Nuevo' button and a 'Médicos' dropdown. The list items are:

- ☐ Nombre y Apellidos
- ☐ Daniel Vazquez

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

Aqui podemos ver las consultas creadas

Hospital Pacientes Médicos Consultas			My Company
<div>Nuevo Consultas</div> <div> <div>Buscar...</div> <div></div> </div>			1-1/1 < >
Paciente	Médico	Fecha	
<input type="checkbox"/> Manuel Afel Garcia	<input type="checkbox"/> Daniel Vazquez	11/12/2025 18:17:59	

## 5. Actividad 04