Práctica 4

Sistemas de Gestión Empresarial

Adrián Condines Celada

Aula Estudio

2º Ciclo Superior - Desarrollo de Aplicaciones Multiplataforma

Curso 2025 - 2026

Índice

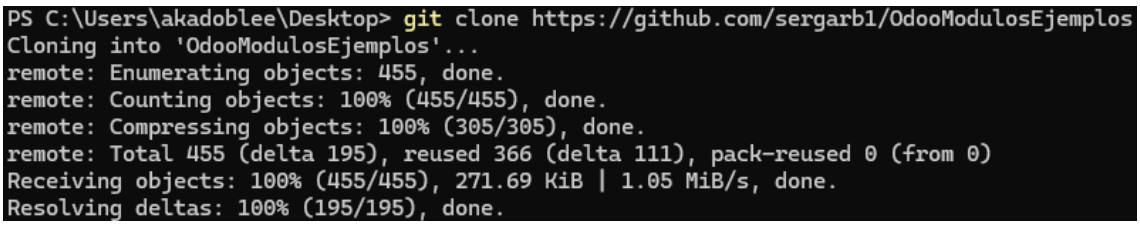
1. INTRODUCCIÓN Y OBJETIVOS

El objetivo de esta práctica es la modificación y creación de módulos personalizados en Odoo

2. PASO PREVIO PARA ACTIVIDADES 1 y 2

Para las actividades 1 y 2, previamente se clonará el repositorio que contiene los módulos de ejemplo que serán necesarios para la realización de dichas actividades. Para ello se abrirá una terminal en el directorio en el que se desea clonar el repositorio y se ejecutará el siguiente comando:

```
git clone https://github.com/sergarb1/OdooModulosEjemplos
```



3. ACTIVIDAD 1

3.1. Introducción

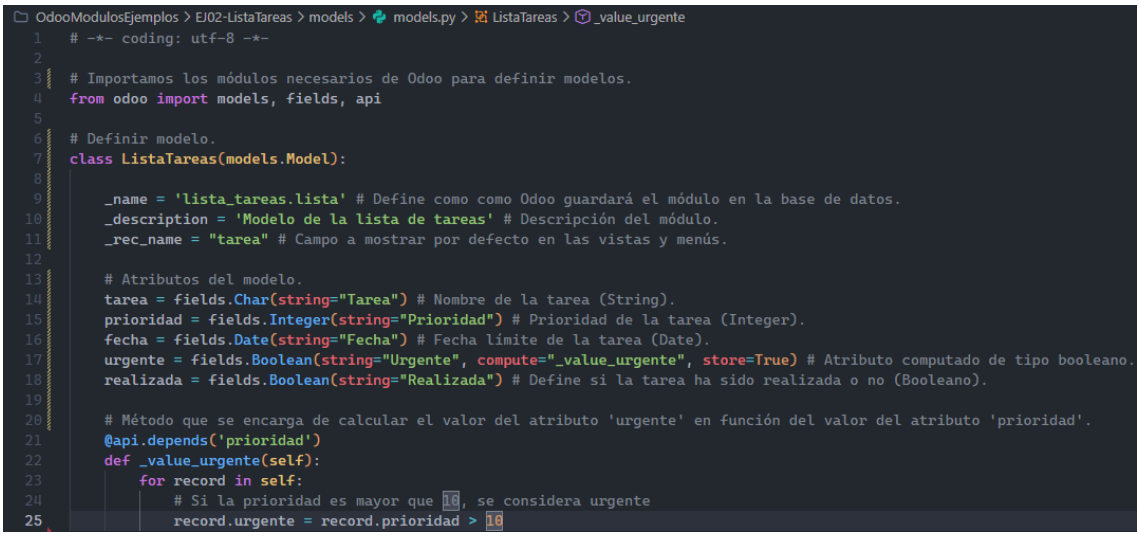
El objetivo de esta actividad es modificar el ejemplo más básico de la **Lista de Tareas** `(Directorio EJ02-ListaTareas del repositorio clonado previamente)` para que las tareas se muestren en formato **Kanban**. También se debe crear una nueva vista para visualizar las tareas en formato **Calendario** según la fecha definida.

3.2. Modelos

Partiendo del directorio mencionado previamente, el primer paso será modificar el archivo `models/models.py` para añadir el nuevo campo **Fecha** al modelo de la lista de tareas.

Para definir el modelo se crea una nueva clase de Python que hereda de la clase **models.Model** y también se han definido los siguientes elementos:

- **_name**: Define como como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **_rec_name**: Campo a mostrar por defecto en las vistas y menús.
- **tarea**: Nombre de la tarea `(String)`.
- **prioridad**: Nivel de prioridad de la tarea `(Integer)`.
- **urgente**: Atributo computado que define si la tarea es urgente o no `(Booleano)`.
- **realizada**: Define si la tarea ha sido realizada o no `(Booleano)`.
- **_value_urgente**: Método que se encarga de calcular el valor del atributo 'urgente' en función del valor del atributo 'prioridad'.



3.3. Vistas

Una vez modificado el modelo, hay que actualizar las vistas del módulo para que reflejen los cambios realizado en el modelo y también para añadir los nuevos tipos de vistas requeridos para la realización de la actividad. Para ello se modificará el archivo `views/views.xml`.

El primer paso sera definir la acción principal de módulo que será abrir el modelo `lista_tareas.lista` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list `(tabla)`, form `(formulario)`, kanban `(kanban)`, calendar `(calendario)`.

```xml
<!— Acción principal del módulo que abre el modelo lista_tareas.lista y muestra sus vistas. —>
<record model = "ir.actions.act_window" id = "action_lista_tareas">

  <!— Título visible de la ventana —>
  <field name = "name">Listado de tareas</field>

  <!— Modelo al que se refiere la acción —>
  <field name = "res_model">lista_tareas.lista</field>

  <!— Tipo de vistas que se mostrarán: list (tabla), form (formulario), kanban (kanban), calendar (calendario) —>
  <field name = "view_mode">list,form,kanban,calendar</field>

</record>
```

A continuación se definirán los menús del módulo que aparecerán en la parte superior de la interfaz de Odoo y para ello se les debe asignar un atributo **id** que será el identificador único de cada menú, un atributo **name** que será el nombre visble del menú y un atributo **parent** que será el menú padre del menú.

Para ello, se definirán los siguientes `menuitem`:

- **lista_tareas_menu_root**: Menú raíz del módulo.
- **lista_tareas_menu_1**: Submenú dentro del menú raíz que puede agrupar más opciones si se llegarán a necesitar.
- **lista_tareas_menu_1_list**: Opción del menú que ejecuta la acción principal del módulo definida en el paso anterior (se le añade el atributo **action**, que hace referencia a la acción del módulo).

```
<!— Menús del módulo que aparecerán en la barra superior. —>

<!— Menú raíz que aparecerá en la barra superior. —>
<menuitem id = "lista_tareas_menu_root" name = "Listado de tareas" sequence = "10"/>

<!— Submenú dentro del menú raíz. Permite agrupar más opciones si lo necesitas en el futuro. —>
<menuitem id = "lista_tareas_menu_1" name = "Gestión de tareas" parent = "lista_tareas_menu_root" sequence = "20"/>

<!— Opción final de menú que ejecuta la acción definida arriba. —>
<menuitem id = "lista_tareas_menu_1_list" name = "Ver tareas" parent = "lista_tareas_menu_1" action = "action_lista_tareas" sequence = "30"/>
```

Una vez definidas las acciones y los menús, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list** que mostrará la lista de tareas en una tabla y para ello se utiliza la etiqueta **record** con los atributos **id** (identificador único de la vista), **model** (modelo de la vista).

Dentro de la etiqueta record se definen los siguientes campos:

- **name**: Nombre interno de la vista.
- **model**: Modelo utilizado enla vista.
- **arch**: Define la estructura XML que tendrá la tarea.

Dentro del campo con `name`: **arch** se define el tipo de vista y su orden (se ordenará en función del valor del atributo **prioridad** y en orden descendente), que en este caso será **list** y también se definen los campos que se mostrarán en la vista:

- **tarea**: Nombre de la tarea `(String)`.
- **prioridad**: Nivel de prioridad de la tarea `(Integer)`.
- **fecha**: Fecha límite de la tarea `(Date)`.
- **urgente**: Atributo computado que define si la tarea es urgente o no `(Booleano)`.
- **realizada**: Define si la tarea ha sido realizada o no `(Booleano)`.

Así quedaría la vista de tipo **list** `(views/views.xml)`:

```
<!— Vista de lista —>
<record id = "view_lista_tareas_list" model = "ir.ui.view">

  <!— Nombre interno de la vista. —>
  <field name = "name">lista.tareas.list</field>

  <!— Modelo al que pertenece esta vista. —>
  <field name = "model">lista_tareas.lista</field>

  <!— Estructura XML de la vista. —>
  <field name = "arch" type = "xml">

    <!— Vista tipo list —>
    <list string = "Tareas" default_order = "prioridad desc">

      <!— Campos que se muestran como columnas —>
      <field name = "tarea"/>          <!— Nombre de la tarea —>
      <field name = "prioridad"/>      <!— Nivel de prioridad —>
      <field name = "fecha"/>          <!— Fecha de la tarea —>
      <field name = "urgente"/>        <!— Urgente o no (computado) —>
      <field name = "realizada"/>      <!— Finalizada o no —>

    </list>

  </field>

</record>
```

La siguiente vista a explicar es la vista de de tipo **form** que se mostrará cuando el usuario cree o edite una tarea. Para ello, al igual que con la vista anterior, se debe utilizar la etiqueta **record** con los atributos **id** (identificador único de la vista), **model** (modelo de la vista).

Dentro de la etiqueta record se definen los siguientes campos:

- **name**: Nombre interno de la vista.
- **model**: Modelo utilizado en la vista.
- **arch**: Define la estructura XML que tendrá la tarea.

Dentro del campo con `name:` **arch** se define el tipo de vista y su orden que en este caso será **form** con el atributo **name** para darle un título al formulario. En el interior de la etiqueta que define el tipo de vista se definirá una etiqueta **sheet**, que actúa como el contenedor principal del formulario.

Dentro de la etiqueta **sheet**, están definidos dos grupos de campos y en ellos se definen los siguientes atributos:

- **tarea**: Nombre de la tarea `(String)`.
- **fecha**: Fecha límite de la tarea `(Date)`.
- **realizada**: Define si la tarea ha sido realizada o no `(Booleano)`.
- **prioridad**: Nivel de prioridad de la tarea `(Integer)`.
- **urgente**: Atributo computado que define si la tarea es urgente o no `(Booleano)`.

Así quedaría la vista de tipo **form** (`views/views.xml`):

```xml
<!-- Vista de formulario utilizada para crear y editar tareas -->
<record id = "view_lista_tareas_form" model = "ir.ui.view">

  <!-- Nombre interno de la vista -->
  <field name = "name">lista.tareas.form</field>

  <!-- Modelo al que pertenece esta vista -->
  <field name = "model">lista_tareas.lista</field>

  <!-- Estructura XML de la vista -->
  <field name = "arch" type = "xml">

    <!-- Vista tipo form -->
    <form string = "Tarea">

      <!-- El contenedor visual principal del formulario -->
      <sheet>

        <!-- Primer bloque de campos (agrupados) -->
        <group>

          <field name = "tarea"/>          <!-- Nombre de tarea -->
          <field name = "fecha"/>          <!-- Fecha límite de la tarea -->
          <field name = "realizada"/>      <!-- Finalizada o no -->

        </group>

        <!-- Segundo bloque de campos (prioridad y urgencia) -->
        <group string = "Prioridad y urgencia">

          <field name = "prioridad"/>                <!-- Campo de prioridad editable -->
          <field name = "urgente" readonly = "1"/>   <!-- Campo calculado: no editable. -->

        </group>

      </sheet>

    </form>

  </field>

</record>
```

La siguiente vista a explicar es la vista de tipo **calendar** y para ello como en todas las vistas se define con la etiqueta **record** con los atributos **id** (identificador único de la vista), **model** (modelo de la vista).

Dentro de la etiqueta record se definen los siguientes campos:

- **name**: Nombre interno de la vista.
- **model**: Modelo utilizado en la vista.
- **arch**: Define la estructura XML que tendrá la tarea.

Dentro del campo con `name:` **arch** se define el tipo de vista y su orden que en este caso será **calendar** con los siguientes atributos:

- **string**: Título de la vista.
- **date_start**: Fecha de inicio de la tarea `(Date)`.
- **color**: Adignación de colores a las tareas según su prioridad.
- **mode**: Define el modo de la vista que en este caso será dividida por meses.

En la vista de calendario se mostrarán los siguientes datos de la tarea:

- **tarea**: Nombre de la tarea `(String)`.
- **prioridad**: Nivel de prioridad de la tarea `(Integer)`.

Así quedaría la vista de tipo **calendar** (`views/views.xml`):

```xml
<!-- Vista de Calendario -->
<record id = "view_lista_tareas_calendar" model = "ir.ui.view">

  <!-- Nombre interno de la vista -->
  <field name = "name">lista.tareas.calendar</field>

  <!-- Modelo al que pertenece esta vista -->
  <field name = "model">lista_tareas.lista</field>

  <!-- Estructura XML de la vista -->
  <field name = "arch" type = "xml">

    <!-- Vista tipo calendario -->
    <calendar string = "Calendario de Tareas" date_start = "fecha" color = "prioridad" mode = "month">

      <!-- Campos que se muestran -->
      <field name = "tarea"/>       <!-- Nombre de la tarea -->
      <field name = "prioridad"/>   <!-- Nivel de prioridad -->

    </calendar>

  </field>

</record>
```

La siguiente vista a explicar es la vista de tipo **kanban** y para ello como en todas las vistas se define con la etiqueta **record** con los atributos **id** (identificador único de la vista), **model** (modelo de la vista).

Dentro de la etiqueta record se definen los siguientes campos:

- **name**: Nombre interno de la vista.
- **model**: Modelo utilizado en la vista.
- **arch**: Define la estructura XML que tendrá la tarea.

Dentro del campo con `name`: **arch** se define el tipo de vista, que en este caso será **kanban** con los siguientes atributos:

- **tarea**: Título de la vista.
- **fecha**: Fecha límite de la tarea `(Date)`.
- **prioridad**: Nivel de prioridad de la tarea `(Integer)`.

```
<!— Vista de Kanban —>
<record id = "view_lista_tareas_kanban" model = "ir.ui.view">

  <!— Nombre interno de la vista —>
  <field name = "name">lista.tareas.kanban</field>

  <!— Modelo al que pertenece esta vista —>
  <field name = "model">lista_tareas.lista</field>

  <!— Estructura XML de la vista —>
  <field name = "arch" type = "xml">

    <!— Contenedor de la vista kanban —>
    <kanban>

      <!— Campos que se muestran —>
      <field name = "tarea"/>       <!— Nombre de la tarea —>
      <field name = "prioridad"/>   <!— Nivel de prioridad —>
      <field name = "fecha"/>       <!— Fecha límite de la tarea —>
```

Posteriormente se diseña una plantilla para mostrar cada una de las tareas en la vista, quedando así la plantilla para la vista de tipo **kanban**:

```
<!— Plantilla de la vista kanban —>
<templates>

  <!— Plantilla de la vista kanban —>
  <t t-name = "kanban-box">

    <!— Contenedor global de la vista kanban —>
    <div class = "oe_kanban_global_click">

      <!— Contenedor superior de la vista kanban —>
      <div class = "o_kanban_record_top">

        <!— Contenedor de la tarea —>
        <div class = "o_kanban_record_headings">

          <!— Título de la tarea —>
          <field name = "tarea"/>

        </div>

      </div>

      <!— Contenedor inferior de la vista kanban —>
      <div class = "o_kanban_record_bottom">

        <!— Contenedor izquierdo de la vista kanban —>
        <div class = "oe_kanban_bottom_left">

          <!— Fecha límite de la tarea —>
          <field name = "fecha"/>

        </div>

        <!— Contenedor derecho de la vista kanban —>
        <div class = "oe_kanban_bottom_right">

          <!— Nivel de prioridad de la tarea —>
          <span class = "badge badge-secondary">

            <!— Nivel de prioridad de la tarea —>
            <field name = "prioridad"/>

          </span>

        </div>

      </div>

    </div>

  </t>

</templates>
```

La última vista que queda por explicar es la vista de tipo **search** y para ello como en todas las vistas se define con la etiqueta **record** con los atributos **id** (identificador único de la vista), **model** (modelo de la vista).

Dentro de la etiqueta record se definen los siguientes campos:

- **name**: Nombre interno de la vista.
- **model**: Modelo utilizado en la vista.
- **arch**: Define la estructura XML que tendrá la tarea.

Dentro del campo con `name`: **arch** se define la estructura XML que tendrá la tarea. En este caso será del tipo **search** con el atributo **string** que define el título de la vista.

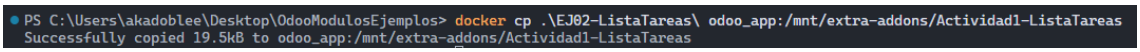
En esta vista el único campo que interesa es el nombre de la tarea ya que los demás campos no son relevantes para una búsqueda de texto y también se añaden filtros para mostrar las tareas que son urgentes y las que ya están realizadas, quedando así la vista de tipo **search**:

```xml
<!— Vista de Búsqueda —>
<record id = "view_lista_tareas_search" model = "ir.ui.view">

  <!— Nombre interno de la vista —>
  <field name = "name">lista.tareas.search</field>

  <!— Modelo al que pertenece esta vista —>
  <field name = "model">lista_tareas.lista</field>

  <!— Estructura XML de la vista —>
  <field name = "arch" type = "xml">

    <!— Vista tipo search —>
    <search string = "Buscar tareas">

      <!— Campo de búsqueda libre —>
      <field name = "tarea"/>

      <!— Filtros predefinidos —>
      <filter name = "filtro_urgente" string = "Urgente" domain = "[('urgente','=',True)]"/>
      <filter name = "filtro_realizada" string = "Realizadas" domain = "[('realizada','=',True)]"/>

    </search>

  </field>

</record>
```

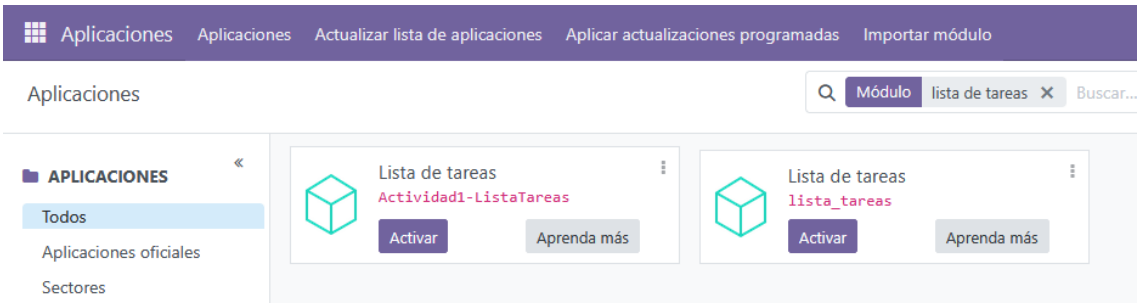
3.4. Funcionamiento

Ya terminada la explicación del modelo y las vistas del módulo, se pondrá a prueba su funcionamiento y para ello el primer paso es pasar el directorio del módulo del equipo local al contenedor de Odoo, concretamente a la carpeta mapeada para almacenar los módulos extra añadidos por el usuario y para ello se utilizará el siguiente comando:

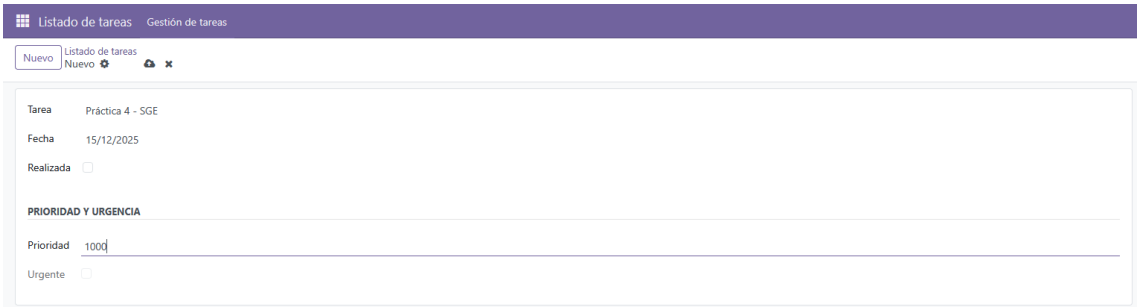
```
docker cp <ruta_módulo><contenedor_odoo>:/mnt/extra-addons/<directorio>
```



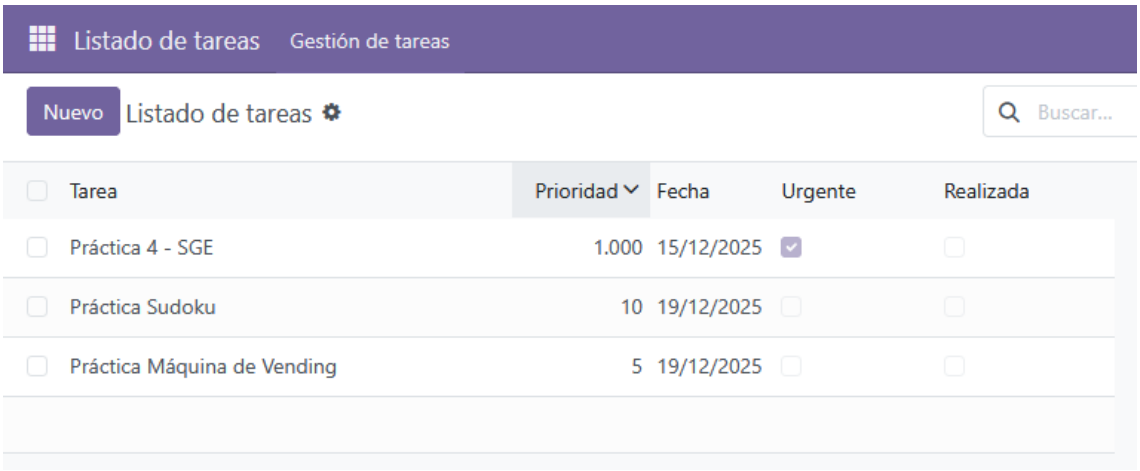
Una vez que se haya copiado el módulo al contenedor de Odoo, se debe activar el **Modo Desarrollador** y actualizar la lista de aplicaciones (explicado en la práctica anterior). Posteriormente se debe buscar el módulo en la lista de aplicaciones e instalarlo pulsando sobre el botón lila de **Activar** (se instalará el primer módulo de la lista ya que el segundo es el de la práctica anterior).



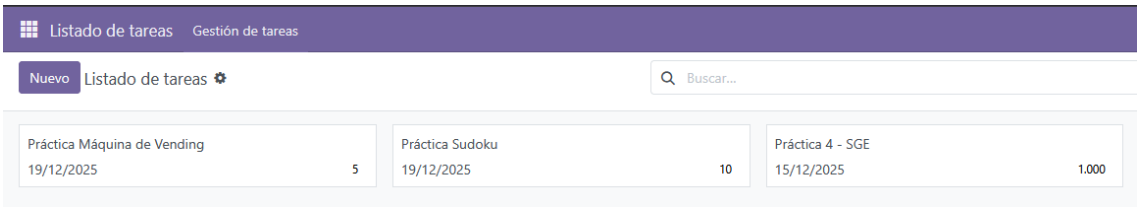
Una vez instalado el módulo, se accederá a su interfaz mediante el menú de Odoo y se crearán nuevas tareas para comprobar su funcionamiento. La vista de creación de tarea (**form**) se ve de la siguiente manera:



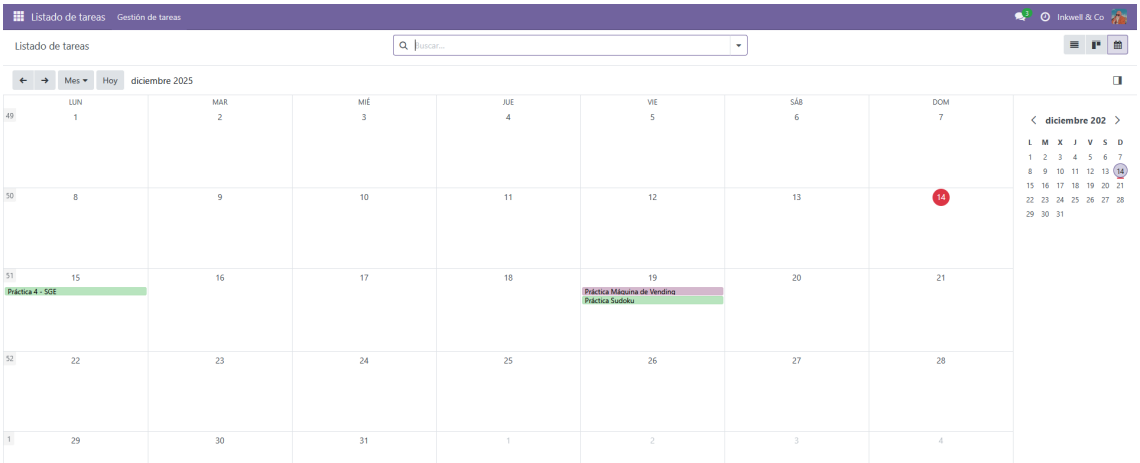
Como se puede comprobar, las tareas se crean correctamente y así se muestran en la vista de tipo **list**:



Así se mostrarían las tareas en la vista de tipo **kanban**:



Y por último, así se mostrarían las tareas en la vista de tipo **calendar**:



4. ACTIVIDAD 2

4.1. Introducción

El objetivo de esta actividad es amplicar el módulo de ejemplo de la **Biblioteca** `Directorio EJ03-ComicsSimple del repositorio clonado previamente` para que incluya la posibilidad de incluir **Socios**. También se debe implementar un sistema de gestión de **Préstamos de Cómics** para que los socios de la biblioteca puedan llevarse cómics prestados.

4.2. Modelos

4.2.1. Modelo Socio

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **_rec_name**: Campo a mostrar por defecto en las vistas y menús.
- **nombre**: Nombre del socio `(String)`.
- **apellido**: Apellido del socio `(String)`.
- **identificador**: Identificador del socio `(String)`.

Así quedaría el modelo **Socio** `(models/biblioteca_socio.py)`:

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

# Se importan los módulos necesarios de Odoo para definir modelos.
from odoo import models, fields

# Definir modelo.
class BibliotecaSocio(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'biblioteca.socio'

    # Descripción del modelo.
    _description = 'Socio de la biblioteca'

    # Campo a mostrar por defecto en las vistas y menús.
    _rec_name = 'nombre'

    # Atributos del modelo.
    nombre = fields.Char('Nombre', required = True)                 # Nombre del socio.
    apellido = fields.Char('Apellido', required = True)             # Apellido del socio.
    identificador = fields.Char('Identificador', required = True)   # Identificador del socio.
```

4.2.2. Modelo Ejemplar

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **_rec_name**: Campo a mostrar por defecto en las vistas y menús.
- **comic_id**: Relación con el modelo de Comic `(Many2one)`.
- **socio_id**: Relación con el modelo de Socio `(Many2one)`.
- **fecha_inicio**: Fecha de inicio del préstamo `(Date)`.
- **fecha_fin**: Fecha de fin del préstamo `(Date)`.

Además, se añade una restricción (`_check_fechas`) para comprobar que las fechas son coherentes: la fecha de inicio no puede ser posterior al día actual y la fecha de devolución no puede ser anterior al día actual.

Así quedaría el modelo **Ejemplar** `(models/biblioteca_ejemplar.py)`:

```python
# Se importan los módulos necesarios de Odoo para definir el modelo.
from datetime import timedelta
from odoo import models, fields, api
from odoo.exceptions import ValidationError

# Definir modelo.
class BibliotecaEjemplar(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'biblioteca.ejemplar'

    # Define la descripción del modelo.
    _description = 'Ejemplar de Comic'

    # Define el campo a mostrar por defecto en las vistas y menús.
    _rec_name = 'comic_id'

    # Atributos del modelo.
    comic_id = fields.Many2one('biblioteca.comic', string = 'Comic', required = True)    # Relación con el modelo de Comic.
    socio_id = fields.Many2one('biblioteca.socio', string = 'Socio Préstamo')            # Relación con el modelo de Socio.
    fecha_inicio = fields.Date('Fecha Inicio Préstamo')                                  # Fecha de inicio del préstamo.
    fecha_fin = fields.Date('Fecha Fin Préstamo')                                        # Fecha de fin del préstamo.

    # Método que se encarga de validar las fechas del préstamo.
    @api.constrains('fecha_inicio', 'fecha_fin')
    def _check_fechas(self):

        # Recorre todos los registros del modelo.
        for record in self:

            # Si la fecha de inicio es posterior al día actual, se lanza una error.
            if record.fecha_inicio and record.fecha_inicio > fields.Date.today():
                raise ValidationError('La fecha de inicio del prestamo no puede ser posterior al dia actual.')

            # Si la fecha de fin es anterior al día actual, se lanza una error.
            if record.fecha_fin and record.fecha_fin < fields.Date.today():
                raise ValidationError('La fecha prevista de devolucion no puede ser anterior al dia actual.')
```

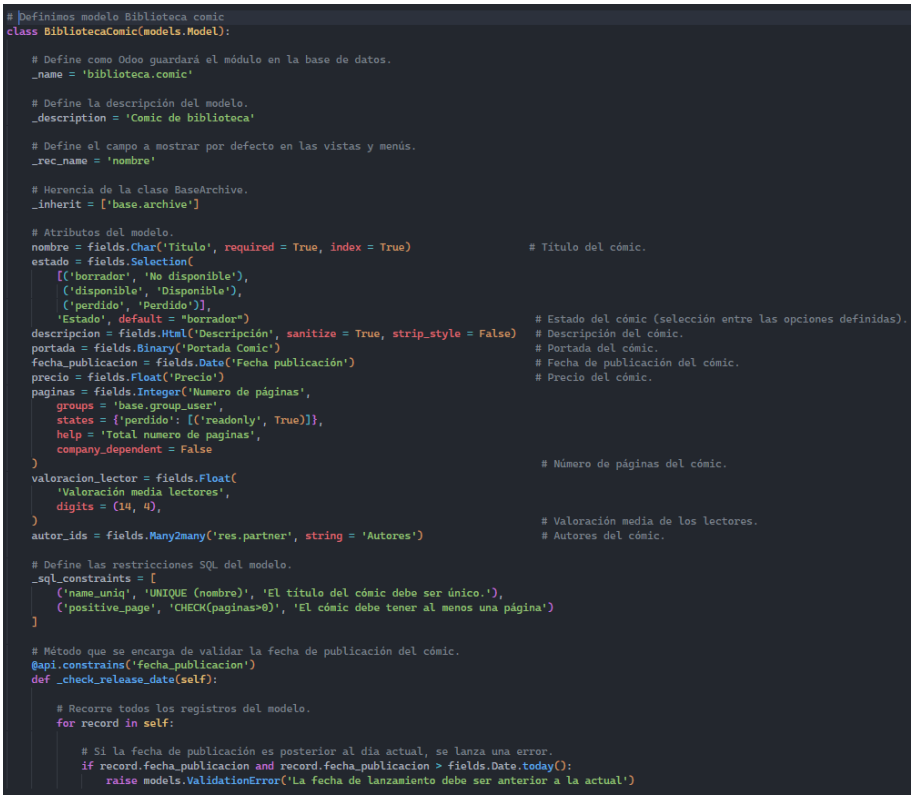
4.2.3. Modelo Cómic

Se ha modificado el modelo de **biblioteca.comic** para añadir nuevos campos y funcionalidades. Se definen los siguientes elementos:

- **estado**: Estado del cómic (borrador, disponible, perdido) `(Selection)`.
- **descripcion**: Descripción del cómic `(Html)`.
- **portada**: Imagen de la portada `(Binary)`.
- **fecha_publicacion**: Fecha de publicación `(Date)`.
- **precio**: Precio del cómic `(Float)`.
- **paginas**: Número de páginas `(Integer)`.
- **valoracion_lector**: Valoración media de los lectores `(Float)`.
- **autor_ids**: Autores del cómic `(Many2many)`.

También se incluyen restricciones SQL para asegurar que el título sea único y que el número de páginas sea positivo, así como una restricción de python para validar la fecha de publicación.

Así quedaría el modelo **Biblioteca Comic** (`models/biblioteca_comic.py`):



```python
# Definimos modelo Biblioteca comic
class BibliotecaComic(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'biblioteca.comic'

    # Define la descripción del modelo.
    _description = 'Comic de biblioteca'

    # Define el campo a mostrar por defecto en las vistas y menús.
    _rec_name = 'nombre'

    # Herencia de la clase BaseArchive.
    _inherit = ['base.archive']

    # Atributos del modelo.
    nombre = fields.Char('Titulo', required = True, index = True)                # Título del cómic.
    estado = fields.Selection(
        [('borrador', 'No disponible'),
         ('disponible', 'Disponible'),
         ('perdido', 'Perdido')],
        'Estado', default = "borrador")                                          # Estado del cómic (selección entre las opciones definidas).
    descripcion = fields.Html('Descripción', sanitize = True, strip_style = False)   # Descripción del cómic.
    portada = fields.Binary('Portada Comic')                                     # Portada del cómic.
    fecha_publicacion = fields.Date('Fecha publicación')                         # Fecha de publicación del cómic.
    precio = fields.Float('Precio')                                              # Precio del cómic.
    paginas = fields.Integer('Numero de páginas',
        groups = 'base.group_user',
        states = {'perdido': [('readonly', True)]},
        help = 'Total numero de paginas',
        company_dependent = False
    )                                                                            # Número de páginas del cómic.
    valoracion_lector = fields.Float(
        'Valoración media lectores',
        digits = (14, 4),
    )                                                                            # Valoración media de los lectores.
    autor_ids = fields.Many2many('res.partner', string = 'Autores')              # Autores del cómic.

    # Define las restricciones SQL del modelo.
    _sql_constraints = [
        ('name_uniq', 'UNIQUE (nombre)', 'El titulo del cómic debe ser único.'),
        ('positive_page', 'CHECK(paginas>0)', 'El cómic debe tener al menos una página')
    ]

    # Método que se encarga de validar la fecha de publicación del cómic.
    @api.constrains('fecha_publicacion')
    def _check_release_date(self):

        # Recorre todos los registros del modelo.
        for record in self:

            # Si la fecha de publicación es posterior al día actual, se lanza una error.
            if record.fecha_publicacion and record.fecha_publicacion > fields.Date.today():
                raise models.ValidationError('La fecha de lanzamiento debe ser anterior a la actual')
```

4.3. Vistas

4.3.1. Vistas de Cómic

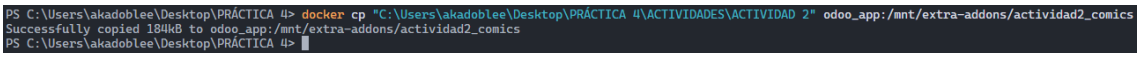
4.3.2. Vistas de Socio

4.3.3. Vistas de Préstamo

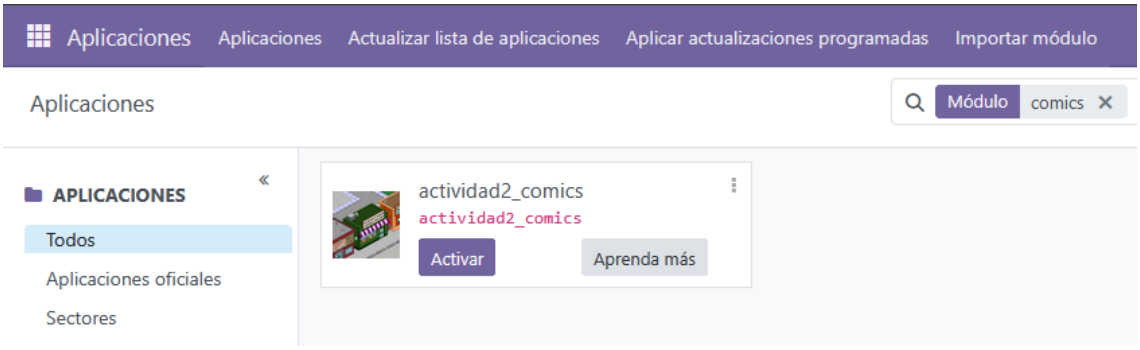
4.4. Funcionamiento

Ya terminada la explicación de los modelos y las vistas del módulo, se pondrá a prueba su funcionamiento y para ello el primer paso es pasar el directorio del módulo del equipo local al contenedor de Odoo, concretamente a la carpeta mapeada para almacenar los módulos extra añadidos por el usuario y para ello se utilizará el siguiente comando:

```
docker cp <ruta_módulo><contenedor_odoo>:/mnt/extra-addons/<directorio>
```

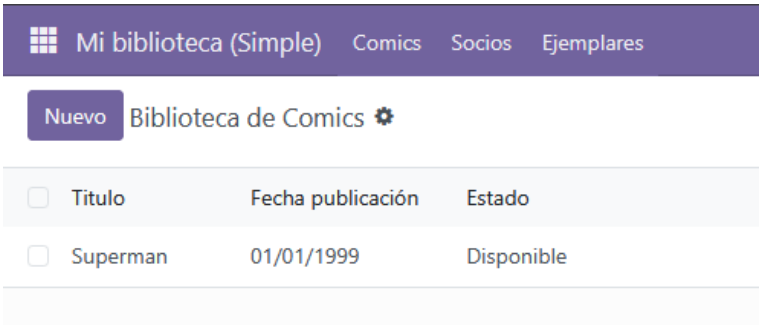


Una vez que se haya copiado el módulo al contenedor de Odoo, se debe activar el **Modo Desarrollador** y actualizar la lista de aplicaciones (explicado en la práctica anterior). Posteriormente se debe buscar el módulo en la lista de aplicaciones e instalarlo pulsando sobre el botón lila de **Activar**.

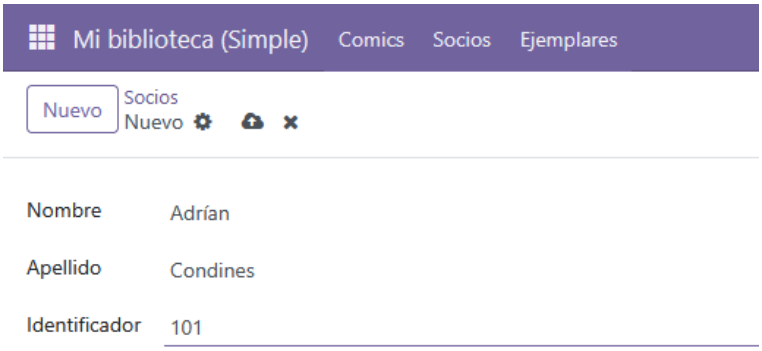


Una vez instalado el módulo, se accederá a su interfaz mediante el menú de Odoo y se creará un nuevo cómic. La vista de creación y edición de cómic (**form**) se ve de la siguiente manera:

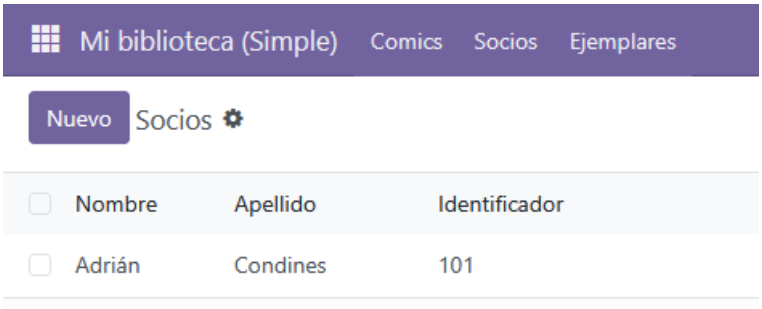
Al crear el cómic, se muestra de la siguiente manera en la vista de tipo **list**:



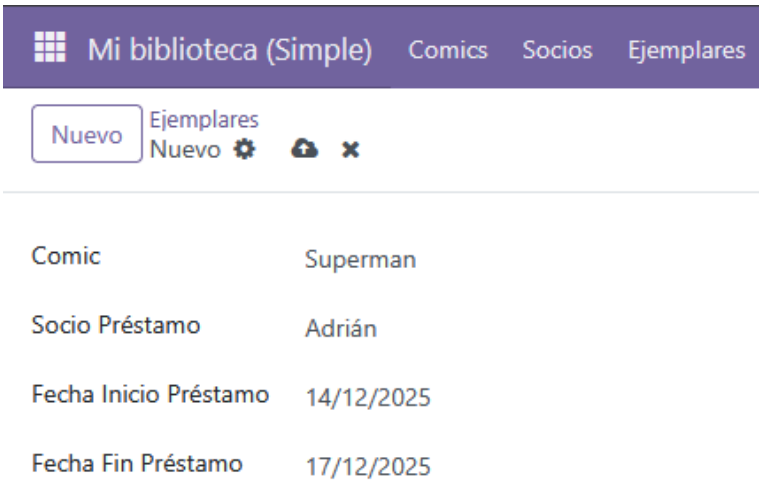
Una vez creado un cómic, es hora de crear un nuevo socio de la biblioteca. La vista de creación y edición de socio **(form)** se ve de la siguiente manera:



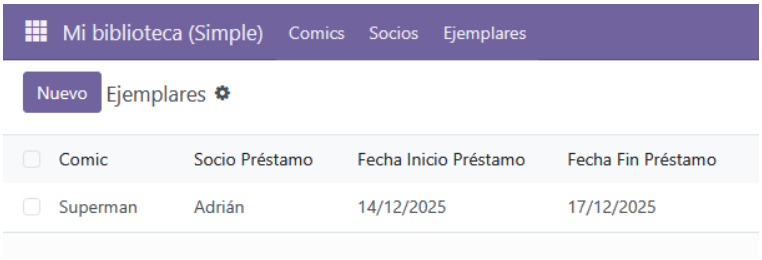
Al crear el nuevo socio, se muestra de la siguiente manera en la vista de tipo **list**:



Ahora que ya hay definidos un ejemplar de un cómic y un socio, es hora de simular un préstamo de ese mismo cómic a ese mismo socio. La vista de creación y edición de préstamo **(form)** se ve de la siguiente manera:



Al crear el nuevo préstamo, se muestra de la siguiente manera en la vista de tipo **list**:



5. ACTIVIDAD 3

5.1. Introducción

El objetivo de esta actividad es crear un módulo para simular un **Sistema de Consultas** de un hospital en el que intervienen **Médicos** y **Pacientes**.

5.2. Base del Módulo

Para ello el primer paso es crear la base del módulo utilizando **Odoo Scaffold** y para ello se utilizará el siguiente comando:

```
docker exec -it <contenedor_odoo>odoo scaffold
        <nombre_directorio><ruta_addons>
```

```
PS C:\Users\akadoblee\Desktop\PRÁCTICA 4> docker exec -it odoo_app odoo scaffold actividad3_hospital /mnt/extra-addons
```

Una vez creada la base de módulo, se debe pasar el directorio del módulo del contenedor de Odoo al equipo local para poder trabajar con él y para ello se utilizará el siguiente comando:

```
docker cp <contenedor_odoo>:<directorio_módulo><ruta_directorio_local>
```

```
PS C:\Users\akadoblee\Desktop\PRÁCTICA 4> docker cp odoo_app:/mnt/extra-addons/actividad3_hospital "C:\Users\akadoblee\Desktop\PRÁCTICA 4\ACTIVIDADES\ACTIVIDAD 3"
Successfully copied 18.4kB to C:\Users\akadoblee\Desktop\PRÁCTICA 4\ACTIVIDADES\ACTIVIDAD 3
```

5.3. Modelos

5.3.1. Modelo Paciente

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **_rec_name**: Campo a mostrar por defecto en las vistas y menús.
- **symptoms**: Atributo que define los síntomas del paciente `(String)`.
- **consulta_ids**: Atributo que define las consultas del paciente `(Many2one)`.

Así quedaría el modelo **Paciente** (`models/paciente.py`):

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

# Se importan los módulos necesarios para definir el modelo.
from odoo import models, fields, api

class Paciente(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'hospital.paciente'

    # Define la descripción del modelo.
    _description = 'Paciente del Hospital'

    # Campo obligatorio para nombre completo
    name = fields.Char(string = 'Nombre y Apellidos', required = True, help = 'Nombre completo del paciente')

    # Síntomas del paciente
    symptoms = fields.Text(string = 'Síntomas', help = 'Descripción de los síntomas del paciente')

    # Relación con consultas: un paciente puede tener muchas consultas
    consulta_ids = fields.One2many(comodel_name = 'hospital.consulta', inverse_name = 'paciente_id', string = 'Consultas', help = 'Lista de consultas')
```

5.3.2. Modelo Médico

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **name**: Campo a mostrar por defecto en las vistas y menús.
- **numero_colegiado**: Atributo que define el número de colegiado del médico (`String`).
- **consulta_ids**: Atributo que define las consultas del médico (`Many2one`).

Así quedaría el modelo **Médico** (`models/medico.py`):

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

# Se importan los módulos necesarios para definir el modelo.
from odoo import models, fields

class Medico(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'hospital.medico'

    # Define la descripción del modelo.
    _description = 'Médico del Hospital'

    # Campo obligatorio para nombre completo.
    name = fields.Char(string = 'Nombre y Apellidos', required = True, help = 'Nombre completo del médico')

    # Número de colegiado, campo obligatorio y único.
    numero_colegiado = fields.Char(string = 'Número de Colegiado', required = True, help = 'Número de colegiado del médico')

    # Relación con consultas: un médico puede tener muchas consultas.
    consulta_ids = fields.One2many(comodel_name = 'hospital.consulta', inverse_name = 'medico_id', string = 'Consultas', help = 'Lista de consultas' )
```

5.3.3. Modelo Consulta

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **paciente_id**: Atributo que define el paciente de la consulta (`Many2one`).
- **medico_id**: Atributo que define el médico de la consulta (`Many2one`).
- **diagnostico**: Atributo que define el diagnostico de la consulta (`Text`).
- **fecha_hora**: Atributo que define la fecha y hora de la consulta (`Datetime`).

Así quedaría el modelo **Consulta** (`models/consulta.py`):

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

# Se importan las los modulos necesarios para definir el modelo
from odoo import models, fields

class Consulta(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'hospital.consulta'

    # Define la descripción del modelo.
    _description = 'Consulta Médica del Hospital'

    # Relación con paciente: Many2one (muchas consultas pueden ser de un paciente).
    paciente_id = fields.Many2one(comodel_name = 'hospital.paciente', string = 'Paciente', required = True, help = 'Paciente atendido en la consulta')
    .
    # Relación con médico: Many2one (muchas consultas pueden ser de un médico)
    medico_id = fields.Many2one(comodel_name = 'hospital.medico', string = 'Médico', required = True, help = 'Médico que atendió la consulta')

    # Diagnóstico de la consulta.
    diagnostico = fields.Text(string = 'Diagnóstico', required = True, help = 'Diagnóstico realizado por el médico')

    # Fecha y hora de la consulta (por defecto, fecha/hora actual).
    fecha_hora = fields.Datetime(string = 'Fecha y Hora', default = fields.Datetime.now, help = 'Fecha y hora de la consulta')
```

5.4. Vistas

5.4.1. Vistas de Paciente

El primer paso sera definir la acción principal que será abrir el modelo `hospital.paciente` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list (`tabla`), form (`formulario`).

```xml
<!-- Acción principal del módulo que abre el modelo hospital.paciente y muestra sus vistas. -->
<record id = "action_paciente" model = "ir.actions.act_window">

    <!-- Título visible de la ventana -->
    <field name = "name">Pacientes</field>

    <!-- Modelo al que se refiere la acción -->
    <field name = "res_model">hospital.paciente</field>

    <!-- Tipo de vistas que se mostrarán: list (tabla), form (formulario) -->
    <field name = "view_mode">list,form</field>

</record>
```

Una vez definida la acción principal, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list**, que mostrará los pacientes en una tabla. Para ello se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **list** y también se definen los campos que se mostrarán en la vista:

- **name**: Nombre del paciente `(String)`.
- **symptoms**: Síntomas del paciente `(Text)`.

Así quedaría la vista de tipo **list** `(views/paciente_list.xml)`:

```xml
<!-- Vista de lista de pacientes -->
<record id = "view_paciente_list" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">hospital.paciente.list</field>

    <!-- Modelo al que pertenece esta vista -->
    <field name = "model">hospital.paciente</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo lista -->
        <list string = "Pacientes">

            <!-- Campos visibles en la vista de lista -->
            <field name = "name" string = "Nombre"/>          <!-- Nombre del paciente -->
            <field name = "symptoms" string = "Síntomas"/>    <!-- Síntomas del paciente -->

        </list>

    </field>

</record>
```

La última vista a explicar de este modelo será la vista de tipo **form**, utilizada para crear y editar pacientes. Se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **form**. Se utliza una etiqueta **sheet** como contenedor principal del formulario y también se definen los campos que se mostrarán en la vista para poder ser rellenados:

- **name**: Nombre del paciente `(String)`.
- **symptoms**: Síntomas del paciente `(Text)`.

Así quedaría la vista de tipo **form** `(views/paciente_form.xml)`:

```xml
<!-- Vista de formulario para pacientes -->
<record id = "view_paciente_form" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">hospital.paciente.form</field>

    <!-- Modelo al que pertenece esta vista -->
    <field name = "model">hospital.paciente</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo formulario -->
        <form string = "Paciente">

            <!-- Contenedor principal del formulario -->
            <sheet>

                <!-- Título del formulario -->
                <div class = "oe_title">

                    <h1> <field name = "name" placeholder = "Nombre completo del paciente"/> </h1> <!-- Nombre del paciente -->

                </div>

                <!-- Síntomas del paciente (textarea para texto largo) -->
                <group>


                    <field name = "symptoms" widget = "textarea" string = "Síntomas" placeholder = "Escribe los síntomas aquí ..."/>

                </group>

            </sheet>

        </form>

    </field>

</record>
```

5.4.2. Vistas de Médico

El primer paso sera definir la acción principal que será abrir el modelo `hospital.medico` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list `(tabla)`, form `(formulario)`.

```
<!— Acción para abrir la vista de médicos —>
<record id = "action_medico" model = "ir.actions.act_window">

    <!— Título visible de la ventana —>
    <field name = "name">Médicos</field>

    <!— Modelo al que se refiere la acción —>
    <field name = "res_model">hospital.medico</field>

    <!— Tipo de vistas que se mostrarán: list (tabla), form (formulario) —>
    <field name = "view_mode">list,form</field>

</record>
```

Una vez definida la acción principal, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list**, que mostrará los médicos en una tabla. Para ello se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **list** y también se definen los campos que se mostrarán en la vista:

- **name**: Nombre del médico `(String)`.
- **numero_colegiado**: Número de colegiado `(String)`.

Así quedaría la vista de tipo **list** (`views/medico_views.xml`):

```xml
<!-- Vista de lista de médicos -->
<record id = "view_medico_list" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">hospital.medico.list</field>

    <!-- Modelo al que pertenece esta vista -->
    <field name = "model">hospital.medico</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo lista -->
        <list string = "Médicos">

            <!-- Campos visibles en la vista de lista -->
            <field name = "name" string = "Nombre"/>                    <!-- Nombre del médico -->
            <field name = "numero_colegiado" string = "N° Colegiado"/>  <!-- Número de colegiado -->

        </list>

    </field>

</record>
```

La última vista a explicar de este modelo será la vista de tipo **form**, utilizada para crear y editar médicos. Se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **form**. Se utliza una etiqueta **sheet** como contenedor principal del formulario y también se definen los campos que se mostrarán en la vista para poder ser rellenados:

- **name**: Nombre del médico (`String`).
- **numero_colegiado**: Número de colegiado (`String`).

Así quedaría la vista de tipo **form** (`views/medico_views.xml`):

```xml
<!-- Vista de formulario para médicos -->
<record id = "view_medico_form" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">hospital.medico.form</field>

    <!-- Modelo al que pertenece esta vista -->
    <field name = "model">hospital.medico</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo formulario -->
        <form string = "Médico">

            <!-- Contenedor principal del formulario -->
            <sheet>

                <!-- Título del formulario -->
                <div class = "oe_title">

                    <h1> <field name = "name" placeholder = "Nombre completo del médico"/> </h1>

                </div>

                <!-- Campo para número de colegiado -->
                <group>

                    <field name = "numero_colegiado" string = "Número de Colegiado"/>       <!--

                </group>

            </sheet>

        </form>

    </field>
```

5.4.3. Vistas de Consulta

El primer paso sera definir la acción principal que será abrir el modelo `hospital.consulta` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list `(tabla)`, form `(formulario)`.

```xml
<!-- Acción principal del módulo que abre el modelo hospital.consulta y muestra sus vistas. -->
<record id = "action_consulta" model = "ir.actions.act_window">

    <!-- Título visible de la ventana -->
    <field name = "name">Consultas</field>

    <!-- Modelo al que se refiere la acción -->
    <field name = "res_model">hospital.consulta</field>

    <!-- Tipo de vistas que se mostrarán: list (tabla), form (formulario) -->
    <field name = "view_mode">list,form</field>

</record>
```

Una vez definida la acción principal, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list**, que mostrará las consultas en una tabla. Para ello se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **list** y también se definen los campos que se mostrarán en la vista:

- **paciente_id**: Paciente a atender `(String)`.
- **medico_id**: Médico que atiende al paciente `(String)`.
- **fecha_hora**: Fecha y hora de la consulta `(Datetime)`.
- **diagnostico**: Diagnóstico `(Text)`.

Así quedaría la vista de tipo **list** (`views/consulta_views.xml`):

```xml
<!-- Vista de lista de consultas -->
<record id = "view_consulta_list" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">hospital.consulta.list</field>

    <!-- Modelo al que pertenece esta vista -->
    <field name = "model">hospital.consulta</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo lista -->
        <list string = "Consultas">

            <!-- Campos visibles en la vista de lista -->
            <field name = "paciente_id" string = "Paciente"/>          <!-- Relación con paciente -->
            <field name = "medico_id" string = "Médico"/>              <!-- Relación con médico -->
            <field name = "fecha_hora" string = "Fecha y Hora"/>       <!-- Fecha y hora de la consulta -->
            <field name = "diagnostico" string = "Diagnóstico"/>       <!-- Diagnóstico -->

        </list>

    </field>

</record>
```

5.4.4. Vistas de Menús

A continuación se definirán los menús del módulo que aparecerán en la parte superior de la interfaz de Odoo y para ello se les debe asignar un atributo **id** que será el identificador único de cada menú, un atributo **name** que será el nombre visble del menú y un atributo **parent** que será el menú padre del menú.

Para ello, se definirán los siguientes `menuitem`:

- **menu_hospital**: Menú principal que aparece en la barra de navegación.
- **menu_hospital_pacientes**: Submenú del modelo paciente que aparece en el menú principal.
- **menu_hospital_medicos**: Submenú del modelo médico que aparece en el menú principal.
- **menu_hospital_consultas**: Submenú del modelo consulta que aparece en el menú principal.

Así quedarían los menús (`views/menu_views.xml`):

```
<!-- Menú principal que aparece en la barra de navegación -->
<menuitem id = "menu_hospital" name = "Hospital" sequence = "10"/>

<!-- Submenús dentro del menú Hospital -->

<!-- Menú para pacientes -->
<menuitem id = "menu_hospital_pacientes" name = "Pacientes" parent = "menu_hospital" action = "action_paciente" sequence = "10"/>

<!-- Menú para médicos -->
<menuitem id = "menu_hospital_medicos" name = "Médicos" parent = "menu_hospital" action = "action_medico" sequence = "20"/>

<!-- Menú para consultas -->
<menuitem id = "menu_hospital_consultas" name = "Consultas" parent = "menu_hospital" action = "action_consulta" sequence = "30"/>
```

5.5. Otros archivos

Se debe modificar el archivo `models/__init__.py` para importar los modelos que se han creado.

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

# Se importan todos los modelos del módulo
from . import paciente
from . import medico
from . import consulta
```

Se debe modificar el archivo `__manifest__.py` para importar las vistas que se han creado.

```python
# -*- coding: utf-8 -*-
{
    'name': 'Actividad 3 - Hospital',
    'version': '1.0',
    'summary': 'Gestión básica de pacientes, médicos y consultas',
    'category': 'Healthcare',
    'author': 'Adriano',
    'website': 'https://staytuned.com.es',
    'depends': ['base'],
    'data': [
        'security/ir.model.access.csv',
        'views/paciente_views.xml',      # Vistas de pacientes
        'views/medico_views.xml',        # Vistas de médicos
        'views/consulta_views.xml',      # Vistas de consultas
        'views/menu_views.xml',          # Menús
    ],
    'installable': True,
    'application': True,
}
```

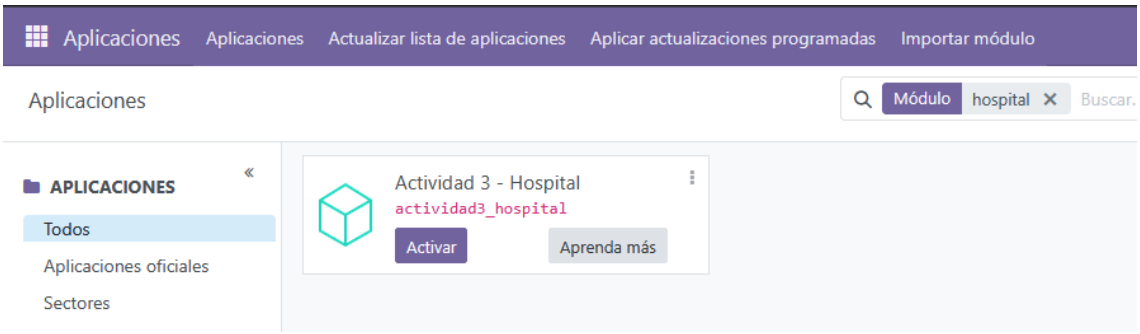
5.6. Funcionamiento

Ya terminada la explicación de los modelos y las vistas del módulo, se pondrá a prueba su funcionamiento y para ello el primer paso es pasar el directorio del módulo del equipo local al contenedor de Odoo, concretamente a la carpeta mapeada para almacenar los módulos extra añadidos por el usuario y para ello se utilizará el siguiente comando:

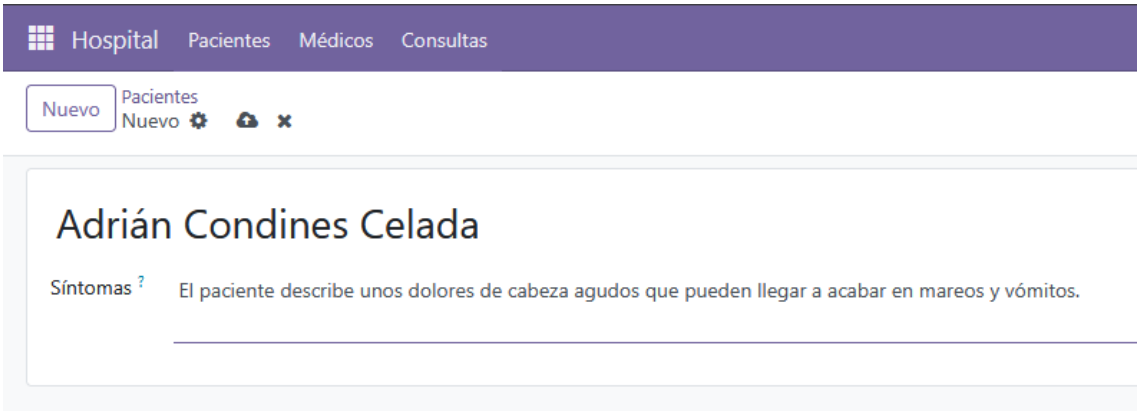
```
docker cp <ruta_módulo><contenedor_odoo>:/mnt/extra-addons/<directorio>
```

```
PS C:\Users\akadoblee\Desktop\PRÁCTICA 4> docker cp "C:\Users\akadoblee\Desktop\PRÁCTICA 4\ACTIVIDADES\ACTIVIDAD 3" odoo_app:/mnt/extra-addons/actividad3_hospital
Successfully copied 30.7kB to odoo_app:/mnt/extra-addons/actividad3_hospital
PS C:\Users\akadoblee\Desktop\PRÁCTICA 4>
```

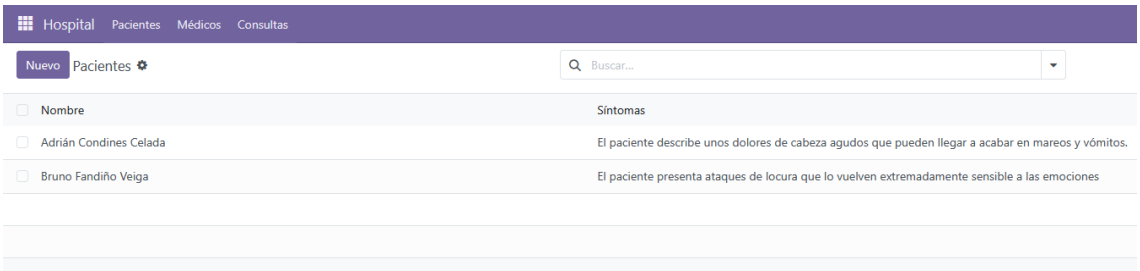
Una vez que se haya copiado el módulo al contenedor de Odoo, se debe activar el **Modo Desarrollador** y actualizar la lista de aplicaciones (explicado en la práctica anterior). Posteriormente se debe buscar el módulo en la lista de aplicaciones e instalarlo pulsando sobre el botón lila de **Activar**.



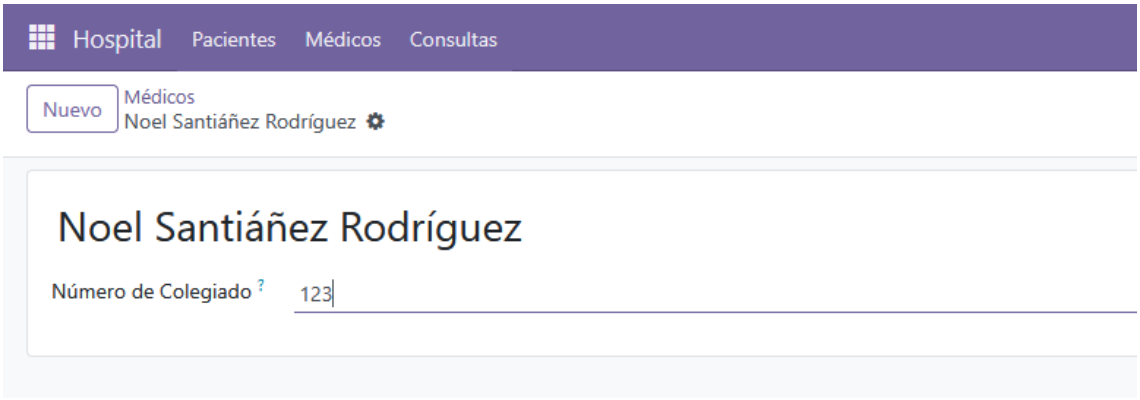
Una vez instalado el módulo, se accederá a su interfaz mediante el menú de Odoo y se crearán nuevos pacientes. La vista de creación y edición de paciente (**form**) se ve de la siguiente manera:



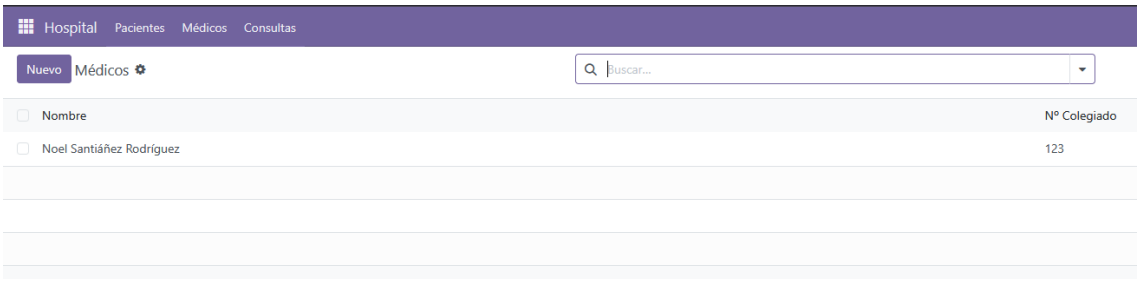
Una vez creado el paciente, se podrá ver su información en la vista de tipo **list** que se ve de la siguiente manera:



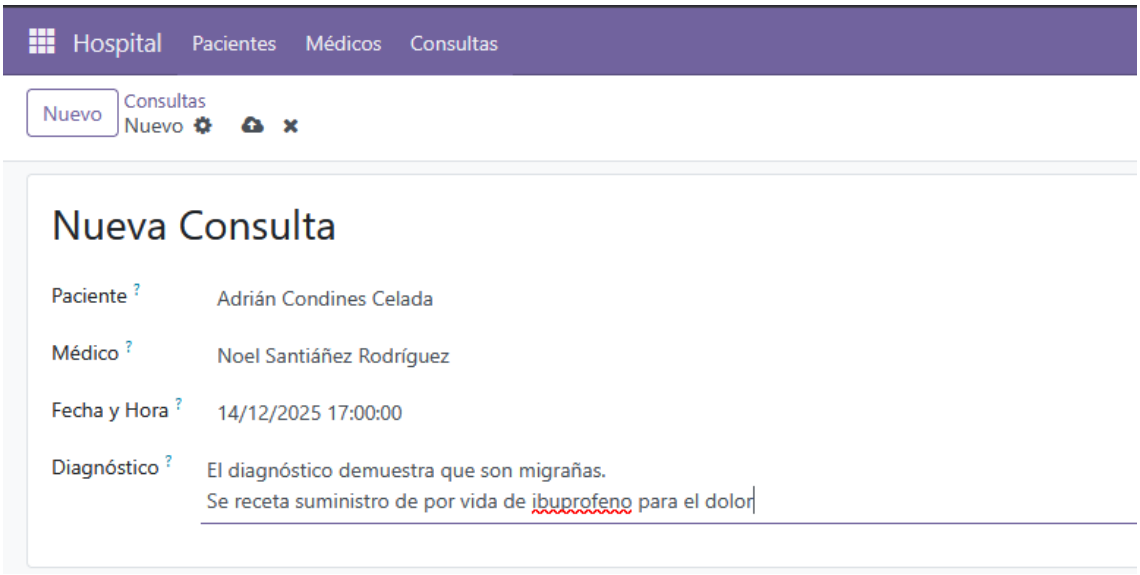
Ya creados un par de pacientes, se procederá a crear un nuevo médico para atenderlos. La vista de creación y edición de médico (**form**) se ve de la siguiente manera:



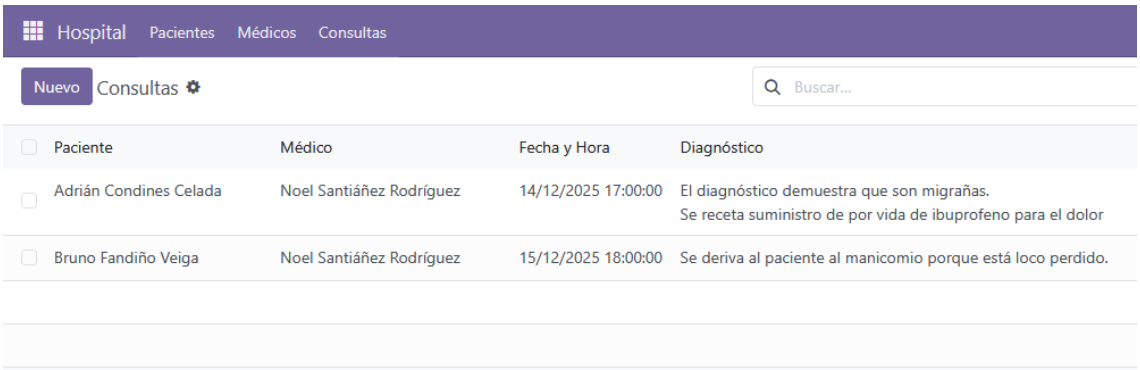
Una vez creado el médico, se podrá ver su información en la vista de tipo **list** que se ve de la siguiente manera:



Ahora que se han creado los pacientes y los médicos, se procederá a crear la consulta para cada paciente. La vista de creación y edición de consulta (**form**) se ve de la siguiente manera:



Una vez creado la consulta, se podrá ver su información en la vista de tipo **list** que se ve de la siguiente manera:



6. ACTIVIDAD 4

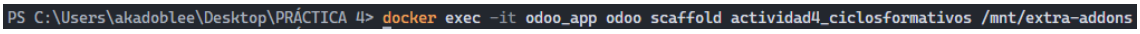
6.1. Introducción

El objetivo de esta actividad es desarrollar un módulo para simular la gestión de **Ciclos Formativos** en un instituto en el que se tendrán en cuenta los mismos Ciclos Formativos, los **Módulos** del ciclo, los **Profesores** que imparten dichos ciclos y los **Alumnos** matriculados.

6.2. Base del Módulo

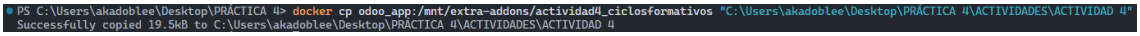
Para ello el primer paso es crear la base del módulo utilizando **Odoo Scaffold** y para ello se utilizará el siguiente comando:

```
docker exec -it <contenedor_odoo>odoo scaffold
        <nombre_directorio><ruta_addons>
```



Una vez creada la base de módulo, se debe pasar el directorio del módulo del contenedor de Odoo al equipo local para poder trabajar con él y para ello se utilizará el siguiente comando:

```
docker cp <contenedor_odoo>:<directorio_módulo><ruta_directorio_local>
```



6.3. Modelos

6.3.1. Modelo Alumno

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **name**: Atributo que define el nombre completo del alumno `(String)`.
- **dni**: Atributo que define el DNI del alumno `(String)`.
- **modulo_ids**: Atributo que define los módulos que cursa el alumno `(Many2many)`.

Así quedaría el modelo **Alumno** `(models/alumno.py)`:

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

# Se importan los módulos necesarios para definir el model
from odoo import models, fields

class Alumno(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'ciclos.alumno'

    # Define la descripción del modelo.
    _description = 'Alumno'

    # Campos obligatorios para nombre y DNI.
    name = fields.Char('Nombre', required = True)
    dni = fields.Char('DNI', required = True)

    # Relación con módulos: varios alumnos pueden estar en varios módulos
    modulo_ids = fields.Many2many('ciclos.modulo', string = 'Módulos')
```

6.3.2. Modelo Profesor

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **name**: Atributo que define el nombre completo del profesor `(String)`.
- **dni**: Atributo que define el DNI del profesor `(String)`.
- **modulo_ids**: Atributo que define los módulos que imparte el profesor `(One2many)`.

Así quedaría el modelo **Profesor** `(models/profesor.py)`:

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

# Se importan los módulos necesarios para definir el modelo.
from odoo import models, fields

class Profesor(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'ciclos.profesor'

    # Define la descripción del modelo.
    _description = 'Profesor'

    # Campos obligatorios para nombre y DNI.
    name = fields.Char('Nombre', required = True)
    dni = fields.Char('DNI', required = True)

    # Relación con módulos: un profesor imparte varios módulos.
    modulo_ids = fields.One2many('ciclos.modulo', 'profesor_id', 'Módulos')
```

6.3.3. Modelo Ciclo Formativo

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **name**: Atributo que define el nombre completo del ciclo formativo `(String)`.
- **codigo**: Atributo que define el código del ciclo formativo `(String)`.
- **modulo_ids**: Atributo que define los módulos que forman el ciclo formativo `(One2many)`.

Así quedaría el modelo **Ciclo Formativo** `(models/ciclo_formativo.py)`:

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

# Se importan los módulos necesarios para definir el model
from odoo import models, fields

class CicloFormativo(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'ciclos.ciclo'

    # Define la descripción del modelo.
    _description = 'Ciclo Formativo'

    # Campos obligatorios para nombre y código del ciclo,
    name = fields.Char('Nombre', required = True)
    codigo = fields.Char('Código', required = True)

    # Relación con módulos: un ciclo puede tener varios módulos.
    modulo_ids = fields.One2many('ciclos.modulo', 'ciclo_id', 'Módulos')
```

6.3.4. Modelo Módulo

El primer paso para crear el modelo es definir el nombre de la clase que definirá al modelo y que heredará de la clase **models.Model**. Una vez definida la clase, se definen los siguientes elementos:

- **_name**: Define como Odoo guardará el modelo en la base de datos.
- **_description**: Descripción del modelo.
- **name**: Atributo que define el nombre completo del módulo `(String)`.
- **codigo**: Atributo que define el código del módulo `(String)`.
- **ciclo_ids**: Atributo que define los módulos que imparte el profesor `(Many2One)`.
- **profesor_ids**: Atributo que define el profesor que imparte el módulo `(Many2One)`.
- **alumno_ids**: Atributo que define los alumnos que cursan el módulo `(Many2many)`.

Así quedaría el modelo **Módulo** `(models/modulo.py)`:

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

# Se importan los módulos necesarios para definir el model
from odoo import models, fields

class Modulo(models.Model):

    # Define como Odoo guardará el módulo en la base de datos.
    _name = 'ciclos.modulo'

    # Define la descripción del modelo.
    _description = 'Módulo'

    # Campos obligatorios para nombre y código del módulo.
    name = fields.Char('Nombre', required = True)
    codigo = fields.Char('Código', required = True)

    # Relación con Ciclos: Un módulo pertenece a un solo ciclo
    ciclo_id = fields.Many2one('ciclos.ciclo', 'Ciclo', required = True)

    # Relación con Profesores: Un módulo puede tener un solo profesor
    profesor_id = fields.Many2one('ciclos.profesor', 'Profesor')

    # Relación con Alumnos: Un módulo puede tener varios alumnos
    alumno_ids = fields.Many2many('ciclos.alumno', string = 'Alumnos')
```

6.4. Vistas

6.4.1. Vistas de Alumno

El primer paso sera definir la acción principal que será abrir el modelo `ciclos.alumno` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list `(tabla)`, form `(formulario)`.

```xml
<!— Acción principal del módulo que abre el modelo ciclos.alumno y muestra sus vistas. —>
<record id = "action_alumno" model = "ir.actions.act_window">

    <!— Título visible de la ventana —>
    <field name = "name">Alumnos</field>

    <!— Modelo al que se refiere la acción —>
    <field name = "res_model">ciclos.alumno</field>

    <!— Tipo de vistas que se mostrarán: list (tabla), form (formulario) —>
    <field name = "view_mode">list,form</field>

</record>
```

Una vez definida la acción principal, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list**, que mostrará los alumnos en una tabla. Para ello se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **list** y también se definen los campos que se mostrarán en la vista:

- **name**: Nombre completo del alumno `(String)`.
- **dni**: DNI del alumno `(String)`.

Así quedaría la vista **list** (`views/alumno_views.xml`):

```xml
<!-- Vista de lista de alumnos -->
<record id = "view_alumno_list" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.alumno.list</field>

    <!-- Modelo al que pertenece esta vista -->
    <field name = "model">ciclos.alumno</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo lista -->
        <list string = "Alumnos">

            <!-- Campos visibles en la vista de lista -->
            <field name = "name" string = "Nombre Completo del Alumno"/>    <!-- Nombre completo del alumno -->
            <field name = "dni" string = "DNI del Alumno"/>                 <!-- DNI del alumno -->

        </list>

    </field>

</record>
```

La última vista a explicar de este modelo será la vista de tipo **form**, utilizada para crear y editar alumnos. Se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **form**. Se utliza una etiqueta **sheet** como contenedor principal del formulario y también se definen los campos que se mostrarán en la vista para poder ser rellenados:

- **name**: Nombre completo del alumno (`String`).
- **dni**: DNI del alumno (`String`).
- **modulo_ids**: Relación con módulos (`One2many`).

El campo `modulo_ids` es una vista de lista para mostrar los datos de los módulos que cursa el alumno. Se mostrarán los siguientes datos:

- **name**: Nombre completo del módulo (`String`).
- **codigo**: Código del módulo (`String`).
- **ciclo_ids**: Ciclo Formativo del módulo (`One2Many`).
- **profesor_ids**: Profesor del módulo (`One2Many`).

Así quedaría la vista **form** (`views/alumno_views.xml`):

```xml
<!-- Vista de formulario para crear o editar alumnos -->
<record id = "view_alumno_form" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.alumno.form</field>

    <!-- Modelo al que pertenece esta vista -->
    <field name = "model">ciclos.alumno</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo formulario -->
        <form string = "Alumno">

            <!-- Contenedor principal del formulario -->
            <sheet>

                <!-- Agrupación de campos -->
                <group>

                    <field name = "name" string = "Nombre Completo del Alumno"/>    <!-- Nombre completo del alumno -->
                    <field name = "dni" string = "DNI del Alumno"/>                 <!-- DNI del alumno -->

                </group>

                <!-- Relación con módulos -->
                <field name = "modulo_ids">

                    <!-- Vista tipo lista -->
                    <list string = "Módulos">

                        <!-- Campos visibles en la vista de lista -->
                        <field name = "name" string = "Nombre del Módulo"/>     <!-- Nombre del módulo -->
                        <field name = "codigo" string = "Código del Módulo"/>   <!-- Código del módulo -->
                        <field name = "ciclo_id" string = "Ciclo Formativo"/>   <!-- Ciclo Formativo del módulo -->
                        <field name = "profesor_id" string = "Profesor"/>       <!-- Profesor del módulo -->

                    </list>

                </field>

            </sheet>

        </form>

    </field>

</record>
```

6.4.2. Vistas de Profesor

El primer paso sera definir la acción principal que será abrir el modelo `ciclos.profesor` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list `(tabla)`, form `(formulario)`.

```xml
<!-- Acción principal del módulo que abre el modelo ciclos.profesor y muestra sus vistas. -->
<record id = "action_profesor" model = "ir.actions.act_window">

    <!-- Título visible de la ventana -->
    <field name = "name">Profesores</field>

    <!-- Modelo al que se refiere la acción -->
    <field name = "res_model">ciclos.profesor</field>

    <!-- Tipo de vistas que se mostrarán: list (tabla), form (formulario) -->
    <field name = "view_mode">list,form</field>

</record>
```

Una vez definida la acción principal, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list**, que mostrará los profesores en una tabla. Para ello se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **list** y también se definen los campos que se mostrarán en la vista:

- **name**: Nombre completo del profesor `(String)`.
- **dni**: DNI del profesor `(String)`.

Así quedaría la vista **list** (`views/profesor_views.xml`):

```xml
<!-- Vista de lista de profesores -->
<record id = "view_profesor_list" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.profesor.list</field>

    <!-- Modelo al que se refiere la vista -->
    <field name = "model">ciclos.profesor</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo lista -->
        <list string = "Profesores">

            <!-- Campos visibles en la vista de lista -->
            <field name = "name" string = "Nombre del Profesor"/>   <!-- Nombre del Profesor -->
            <field name = "dni" string = "DNI del Profesor"/>       <!-- DNI del Profesor -->

        </list>

    </field>

</record>
```

La última vista a explicar de este modelo será la vista de tipo **form**, utilizada para crear y editar profesores. Se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **form**. Se utliza una etiqueta **sheet** como contenedor principal del formulario y también se definen los campos que se mostrarán en la vista para poder ser rellenados:

- **name**: Nombre completo del profesor (`String`).
- **dni**: DNI del profesor (`String`).
- **modulo_ids**: Relación con módulos (`One2many`).

El campo `modulo_ids` es una vista de lista para mostrar los datos de los módulos que imparte el profesor. Se mostrarán los siguientes datos:

- **name**: Nombre completo del módulo (`String`).
- **codigo**: Código del módulo (`String`).
- **ciclo_ids**: Ciclo Formativo del módulo (`One2Many`).

Así quedaría la vista **form** (`views/profesor_views.xml`):

```xml
<!-- Vista de formulario para crear o editar profesores -->
<record id = "view_profesor_form" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.profesor.form</field>

    <!-- Modelo al que se refiere la vista -->
    <field name = "model">ciclos.profesor</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo formulario -->
        <form string = "Profesores">

            <!-- Contenedor principal del formulario -->
            <sheet>

                <!-- Agrupación de campos -->
                <group>

                    <!-- Campos visibles en la vista de formulario -->
                    <field name = "name" string = "Nombre del Profesor"/>   <!-- Nombre del Profesor -->
                    <field name = "dni" string = "DNI del Profesor"/>       <!-- DNI del Profesor -->

                </group>

                <!-- Relación con módulos -->
                <field name = "modulo_ids">

                    <!-- Vista tipo lista -->
                    <list string = "Módulos">

                        <!-- Campos visibles en la vista de lista -->
                        <field name = "name"/>          <!-- Nombre del Módulo -->
                        <field name = "codigo"/>        <!-- Código del Módulo -->
                        <field name = "ciclo_id"/>      <!-- Ciclo Formativo del Módulo -->

                    </list>

                </field>

            </sheet>

        </form>

    </field>

</record>
```

6.4.3. Vistas de Ciclo Formativo

El primer paso sera definir la acción principal que será abrir el modelo `ciclos.ciclo` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list `(tabla)`, form `(formulario)`.

```
<!— Acción principal del módulo que abre el modelo ciclos.ciclo y muestra sus vistas. —>
<record id = "action_ciclo" model = "ir.actions.act_window">

    <!— Título visible de la ventana —>
    <field name = "name">Ciclos</field>

    <!— Modelo al que se refiere la acción —>
    <field name = "res_model">ciclos.ciclo</field>

    <!— Tipo de vistas que se mostrarán: list (tabla), form (formulario) —>
    <field name = "view_mode">list,form</field>

</record>
```

Una vez definida la acción principal, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list**, que mostrará las consultas en una tabla. Para ello se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **list** y también se definen los campos que se mostrarán en la vista:

- **name**: Nombre completo del ciclo formativo `(String)`.
- **codigo**: Código identificador del ciclo formativo `(String)`.

Así quedaría la vista **list** (`views/ciclo_formativo_views.xml`):

```xml
<!-- Vista de lista de ciclos -->
<record id = "view_ciclo_list" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.ciclo.list</field>

    <!-- Modelo al que se refiere la vista -->
    <field name = "model">ciclos.ciclo</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo lista -->
        <list>

            <!-- Campos que se mostrarán en la lista -->
            <field name = "codigo" string = "Código del Ciclo"/>    <!-- Código del Ciclo -->
            <field name = "name" string = "Nombre del Ciclo"/>      <!-- Nombre del Ciclo -->

        </list>

    </field>

</record>
```

La última vista a explicar de este modelo será la vista de tipo **form**, utilizada para crear y editar los ciclos formativos. Se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **form**. Se utliza una etiqueta **sheet** como contenedor principal del formulario y también se definen los campos que se mostrarán en la vista para poder ser rellenados:

- **name**: Nombre completo del ciclo (`String`).
- **codigo**: Código identificador del ciclo (`String`).
- **modulo_ids**: Relación con módulos (`One2many`).

El campo `modulo_ids` es una vista de lista para mostrar los datos de los módulos que pertenecen al ciclo. Se mostrarán los siguientes datos:

- **name**: Nombre completo del módulo (`String`).
- **codigo**: Código del módulo (`String`).
- **profesor_id**: Profesor del módulo (`One2Many`).
- **alumno_ids**: Alumnos del módulo (`Many2Many`).

Así quedaría la vista **form** (`views/ciclo_formativo_views.xml`):

```xml
<!-- Vista de formulario para crear o editar ciclos -->
<record id = "view_ciclo_form" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.ciclo.form</field>

    <!-- Modelo al que se refiere la vista -->
    <field name = "model">ciclos.ciclo</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo formulario -->
        <form>

            <!-- Contenedor principal del formulario -->
            <sheet>

                <!-- Agrupación de campos -->
                <group>

                    <field name = "name" string = "Nombre del Ciclo"/>      <!-- Nombre del Ciclo -->
                    <field name = "codigo" string = "Código del Ciclo"/>    <!-- Código del Ciclo -->

                </group>

                <!-- Relación con módulos -->
                <field name = "modulo_ids">

                    <!-- Vista tipo lista -->
                    <list string = "Módulos">

                        <field name = "name" string = "Nombre del Módulo"/>                <!-- Nombre del Módulo -->
                        <field name = "codigo" string = "Código del Módulo"/>              <!-- Código del Módulo -->
                        <field name = "profesor_id" string = "Profesor del Módulo"/>       <!-- Profesor del Módulo -->
                        <field name = "alumno_ids" string = "Alumnos del Módulo"/>         <!-- Alumnos del Módulo -->

                    </list>

                </field>

            </sheet>

        </form>

    </field>

</record>
```

6.4.4. Vistas de Módulo

El primer paso sera definir la acción principal que será abrir el modelo `ciclos.modulo` y muestra sus vistas. Se deben de definir los siguientes campos:

- **name**: Título visible de la ventana.
- **res_model**: Modelo al que se refiere la acción.
- **view_mode**: Tipo de vistas que se mostrarán: list (`tabla`), form (`formulario`).

```
<!-- Acción principal del módulo que abre el modelo ciclos.modulo y muestra sus vistas. -->
<record id = "action_modulo" model = "ir.actions.act_window">

    <!-- Título visible de la ventana -->
    <field name = "name">Módulos</field>

    <!-- Modelo al que se refiere la acción -->
    <field name = "res_model">ciclos.modulo</field>

    <!-- Tipo de vistas que se mostrarán: list (tabla), form (formulario) -->
    <field name = "view_mode">list,form</field>

</record>
```

Una vez definida la acción principal, se procederá a crear y explicar todas las vistas del modelo. La primera de ellas será la vista de tipo **list**, que mostrará las consultas en una tabla. Para ello se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **list** y también se definen los campos que se mostrarán en la vista:

- **name**: Nombre completo del módulo (`String`).
- **codigo**: Código identificador del módulo (`String`).
- **ciclo_id**: Ciclo formativo al que pertenece el módulo (`Many2one`).
- **profesor_id**: Profesor que imparte el módulo (`Many2one`).
- **alumno_ids**: Alumnos que cursan el módulo (`Many2many`).

Así quedaría la vista **list** (`views/modulo_views.xml`):

```xml
<!-- Vista de lista de módulos -->
<record id = "view_modulo_list" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.modulo.list</field>

    <!-- Modelo al que se refiere la vista -->
    <field name = "model">ciclos.modulo</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo lista -->
        <list string = "Módulos">

            <!-- Campos visibles en la vista de lista -->
            <field name = "codigo" string = "Código del Módulo"/>              <!-- Código del Módulo -->
            <field name = "name" string = "Nombre del Módulo"/>                <!-- Nombre del Módulo -->
            <field name = "ciclo_id" string = "Ciclo Formativo del Módulo"/>   <!-- Ciclo Formativo del Módulo -->
            <field name = "profesor_id" string = "Profesor del Módulo"/>       <!-- Profesor del Módulo -->
            <field name = "alumno_ids" string = "Alumnos del Módulo"/>         <!-- Alumnos del Módulo -->

        </list>

    </field>

</record>
```

La última vista a explicar de este modelo será la vista de tipo **form**, utilizada para crear y editar módulos pertenecientes a un ciclo formativo. Se definen los siguientes elementos:

- **name**: Título visible de la ventana.
- **model**: Modelo al que pertenece esta vista.
- **arch**: Estructura XML de la vista.

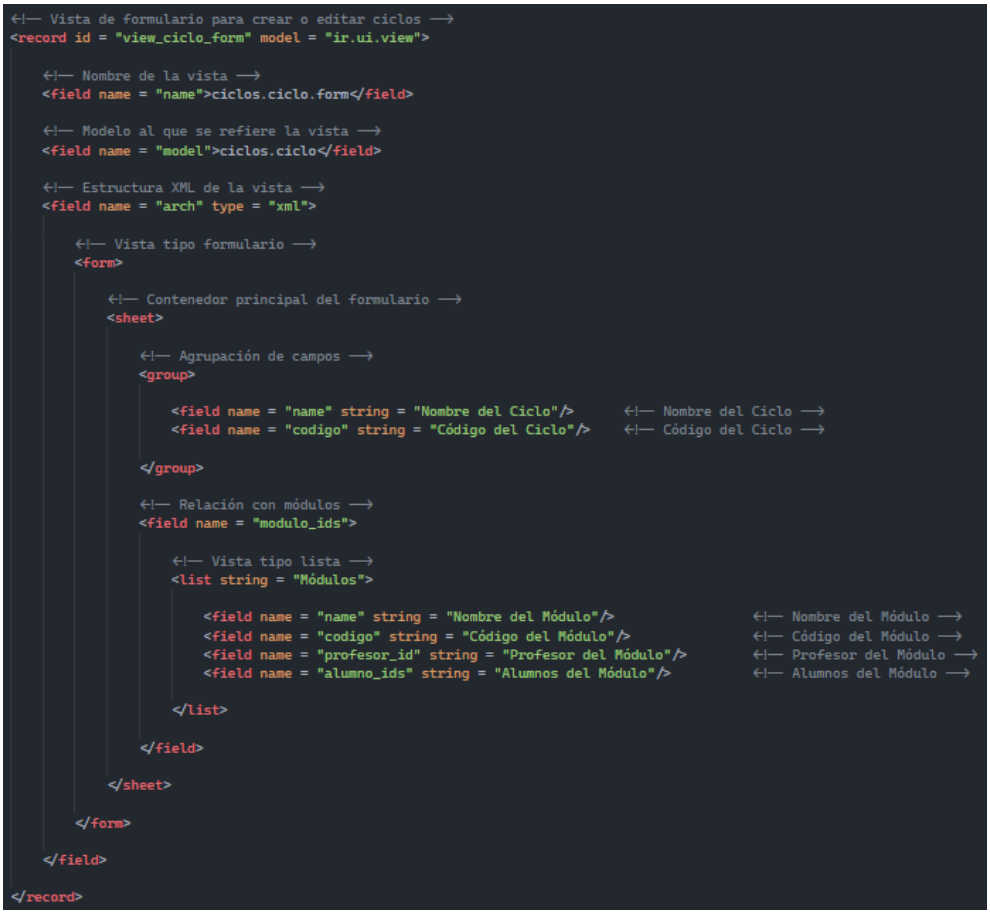
Dentro del campo con `name: arch` se define el tipo de vista que en este caso será de tipo **form**. Se utliza una etiqueta **sheet** como contenedor principal del formulario y también se definen los campos que se mostrarán en la vista para poder ser rellenados:

- **name**: Nombre completo del módulo (`String`).
- **codigo**: Código identificador del módulo (`String`).
- **ciclo_id**: Ciclo formativo al que pertenece el módulo (`Many2one`).
- **profesor_id**: Profesor que imparte el módulo (`Many2one`).
- **alumno_ids**: Alumnos que cursan el módulo (`Many2many`).

El campo `alumno_ids` es una vista de lista para mostrar los datos de los alumnos que cursan el módulo. Se mostrarán los siguientes datos:

- **name**: Nombre completo del alumno (`String`).
- **dni**: DNI del alumno (`String`).

Así quedaría la vista **form** (`views/modulo_views.xml`):



```xml
<!-- Vista de formulario para crear o editar ciclos -->
<record id = "view_ciclo_form" model = "ir.ui.view">

    <!-- Nombre de la vista -->
    <field name = "name">ciclos.ciclo.form</field>

    <!-- Modelo al que se refiere la vista -->
    <field name = "model">ciclos.ciclo</field>

    <!-- Estructura XML de la vista -->
    <field name = "arch" type = "xml">

        <!-- Vista tipo formulario -->
        <form>

            <!-- Contenedor principal del formulario -->
            <sheet>

                <!-- Agrupación de campos -->
                <group>

                    <field name = "name" string = "Nombre del Ciclo"/>      <!-- Nombre del Ciclo -->
                    <field name = "codigo" string = "Código del Ciclo"/>    <!-- Código del Ciclo -->

                </group>

                <!-- Relación con módulos -->
                <field name = "modulo_ids">

                    <!-- Vista tipo lista -->
                    <list string = "Módulos">

                        <field name = "name" string = "Nombre del Módulo"/>                <!-- Nombre del Módulo -->
                        <field name = "codigo" string = "Código del Módulo"/>              <!-- Código del Módulo -->
                        <field name = "profesor_id" string = "Profesor del Módulo"/>       <!-- Profesor del Módulo -->
                        <field name = "alumno_ids" string = "Alumnos del Módulo"/>         <!-- Alumnos del Módulo -->

                    </list>

                </field>

            </sheet>

        </form>

    </field>

</record>
```

6.4.5. Vistas de Menús

A continuación se definirán los menús del módulo que aparecerán en la parte superior de la interfaz de Odoo y para ello se les debe asignar un atributo **id** que será el identificador único de cada menú, un atributo **name** que será el nombre visble del menú y un atributo **parent** que será el menú padre del menú.

Para ello, se definirán los siguientes `menuitem`:

- **menu_ciclos_principal**: Menú principal que aparece en la barra de navegación.
- **menu_alumnos**: Submenú del modelo Alumno que aparece en el menú principal.
- **menu_profesores**: Submenú del modelo médico que aparece en el menú principal.
- **menu_ciclos**: Submenú del modelo consulta que aparece en el menú principal.

Así quedarían los menús (`views/menu_views.xml`):

```
<!-- Menú principal -->
<menuitem id = "menu_ciclos_principal" name = "Ciclos Formativos" sequence = "10"/>

<!-- Submenús -->

<!-- Menú de ciclos -->
<menuitem id = "menu_ciclos" name = "Ciclos" parent = "menu_ciclos_principal" action = "action_ciclo" sequence = "10"/>

<!-- Menú de módulos -->
<menuitem id = "menu_modulos" name = "Módulos" parent = "menu_ciclos_principal" action = "action_modulo" sequence = "20"/>

<!-- Menú de alumnos -->
<menuitem id = "menu_alumnos" name = "Alumnos" parent = "menu_ciclos_principal" action = "action_alumno" sequence = "30"/>

<!-- Menú de profesores -->
<menuitem id = "menu_profesores" name = "Profesores" parent = "menu_ciclos_principal" action = "action_profesor" sequence = "40"/>
```

6.5. Otros archivos

Se debe modificar el archivo `models/__init__.py` para importar los modelos que se han creado.

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

# Se importan todos los modelos del módulo
from . import ciclo_formativo
from . import modulo
from . import alumno
from . import profesor
```

Se debe modificar el archivo `__manifest__.py` para importar las vistas que se han creado.

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

{
    'name': 'Gestión de Ciclos Formativos',
    'version': '1.0',
    'summary': 'Gestión de ciclos formativos, módulos, alumnos y profesores',
    'category': 'Education',
    'author': 'Adriano',
    'website': 'https://staytuned.com.es',
    'depends': ['base'],
    'data': [
        'security/ir.model.access.csv',
        'views/ciclo_formativo_views.xml',  # Vistas de los ciclos
        'views/modulo_views.xml',           # Vistas de los módulos
        'views/alumno_views.xml',           # Vistas de los alumnos
        'views/profesor_views.xml',         # Vistas de los profesores
        'views/menu_views.xml',             # Vistas de los menús
    ],
    'installable': True,
    'application': True,
}
```

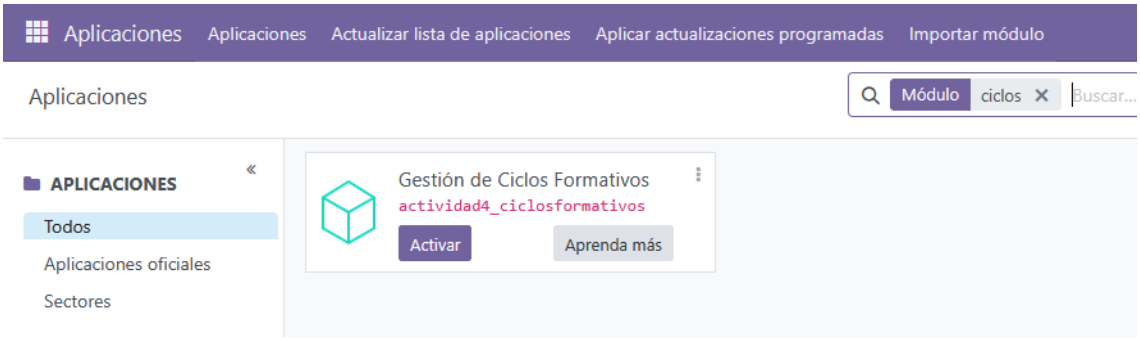
6.6. Funcionamiento

Ya terminada la explicación de los modelos y las vistas del módulo, se pondrá a prueba su funcionamiento y para ello el primer paso es pasar el directorio del módulo del equipo local al contenedor de Odoo, concretamente a la carpeta mapeada para almacenar los módulos extra añadidos por el usuario y para ello se utilizará el siguiente comando:

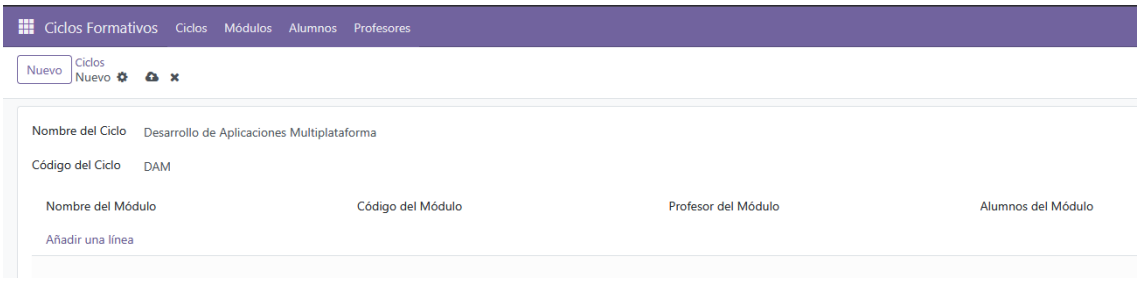
```
docker cp <ruta_módulo><contenedor_odoo>:/mnt/extra-addons/<directorio>
```

```
PS C:\Users\akadoblee\Desktop\PRÁCTICA 4> docker cp "C:\Users\akadoblee\Desktop\PRÁCTICA 4\ACTIVIDADES\ACTIVIDAD 4" odoo_app:/mnt/extra-addons/actividad4_ciclosformativos
Successfully copied 37.9kB to odoo_app:/mnt/extra-addons/actividad4_ciclosformativos
```

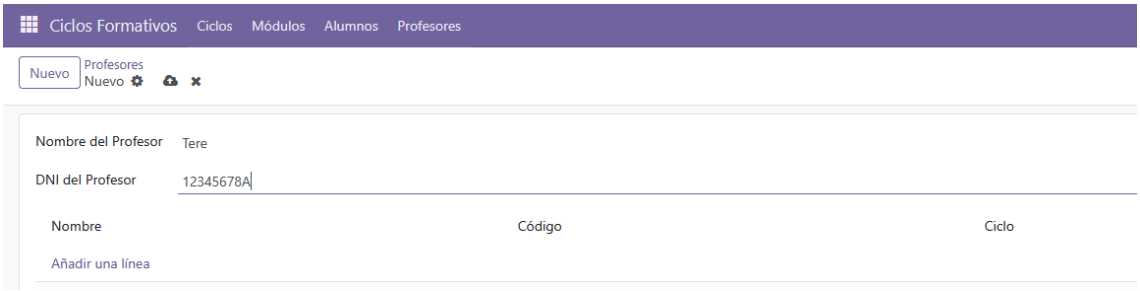
Una vez que se haya copiado el módulo al contenedor de Odoo, se debe activar el **Modo Desarrollador** y actualizar la lista de aplicaciones (explicado en la práctica anterior). Posteriormente se debe buscar el módulo en la lista de aplicaciones e instalarlo pulsando sobre el botón lila de **Activar**.



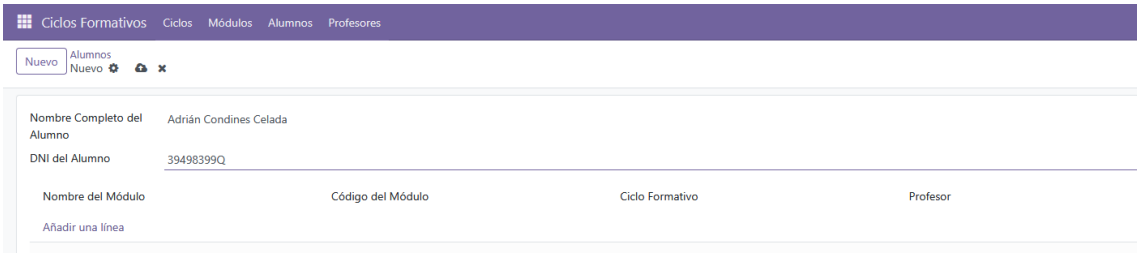
Una vez instalado el módulo, se accederá a él desde la barra de navegación de Odoo y se empezará por crear un nuevo ciclo formativo. La vista de creación y edición de ciclos formativos **(form)**, se ve de la siguiente manera (aunque se puedan crear los módulos del ciclo desde esta vista, de momento se creará el ciclo sin ningún módulo):



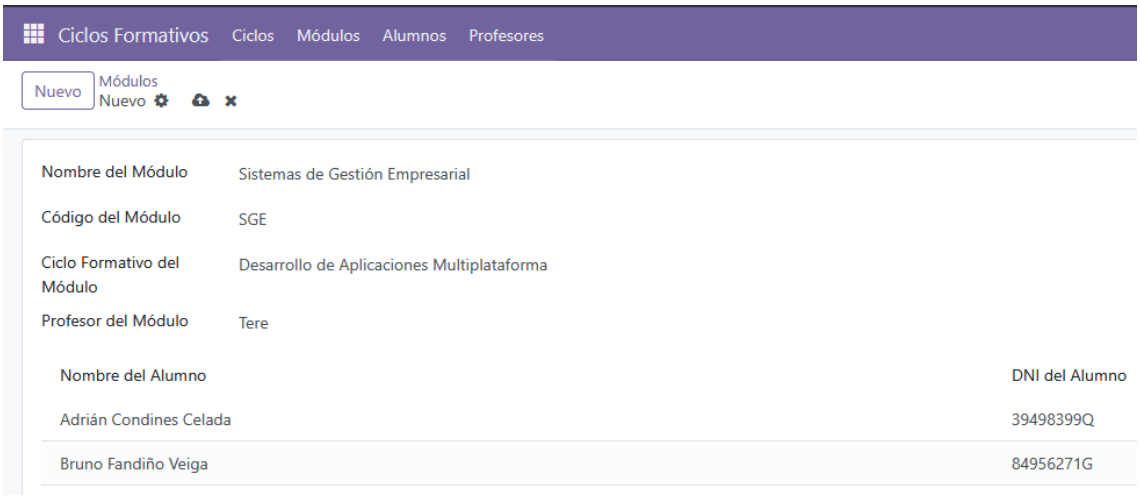
Una vez creado el ciclo formativo, se puede crearán nuevos profesores para impartir módulos en este ciclo. La vista de creación y edición de profesores **(form)**, se ve de la siguiente manera (aunque se puedan crear los módulos que imparten los profesores desde esta vista, de momento se creará el profesor sin ningún módulo asignado):



Ya creados los docentes que impartirán los ciclos, se crearán los alumnos que asistirán a los módulos impartidos por los profesores. La vista de creación y edición de alumnos **(form)**, se ve de la siguiente manera (aunque se puedan crear los módulos a los que asisten los alumnos desde esta vista, de momento se creará el alumno sin ningún módulo asignado):



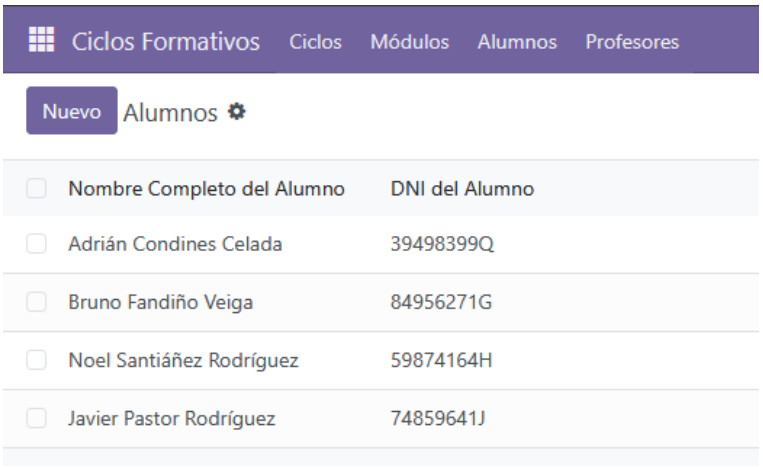
Ahora se crearán los módulos que se impartirán en el ciclo formativo. La vista de creación y edición de módulos **(form)**, se ve de la siguiente manera:



Una vez creados todos los datos, así se verá la lista de módulos en el ciclo formativo:

Así se ve la vista de lista de los profesores:

Así se verá la lista de los alumnos:



Y por último, así se verá la lista de ciclos formativos: