UD5: Creación de un módulo para Odoo Part 2

Index

Alhassan Bouhou

1. Creación de un módulo para Odoo

1. *Campo Identificador de los alumnos:*

Añade una restricción para que el campo Identificador de los alumnos sea único. (función constraint).

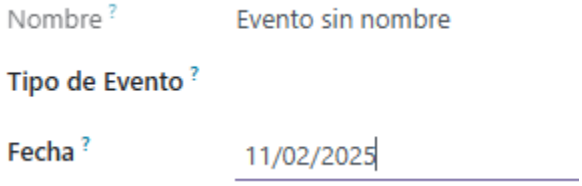
```
dni = fields.Char(string="DNI/NIE", required=True, unique=True)
```

```
@api.constrains("dni")
def _check_unique_dni(self):
    for alumno in self:
        existing_alumno = self.search(
            [("dni", "=", alumno.dni), ("id", "!=", alumno.id)]
        )
        if existing_alumno:
            raise ValidationErr("El DNI/NIE debe ser único para cada alumno.")
```

2. La fecha del evento:

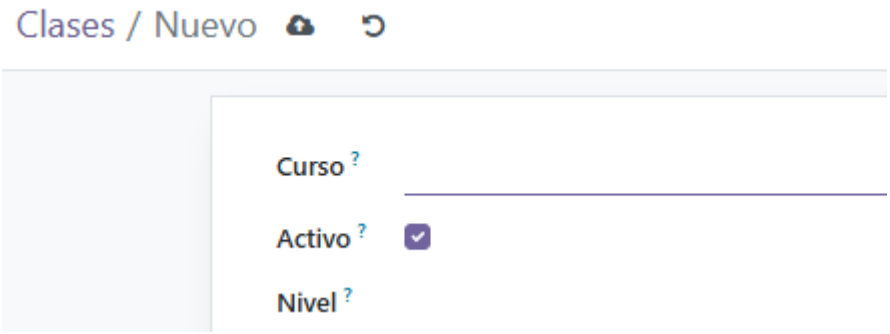
Definir el valor por defecto de la fecha del evento como día el de hoy (ver Fecha). Añadir un campo activo al modelo clases y configurar las clases como activas de manera predeterminada. (recuerda modificar la vista también).

```
fecha = fields.Date(string="Fecha", default=fields.Date.today, required=True)
descripcion = fields.Text(string="Descripción")
```




```
activo = fields.Boolean(string="Activo", default=True)
```

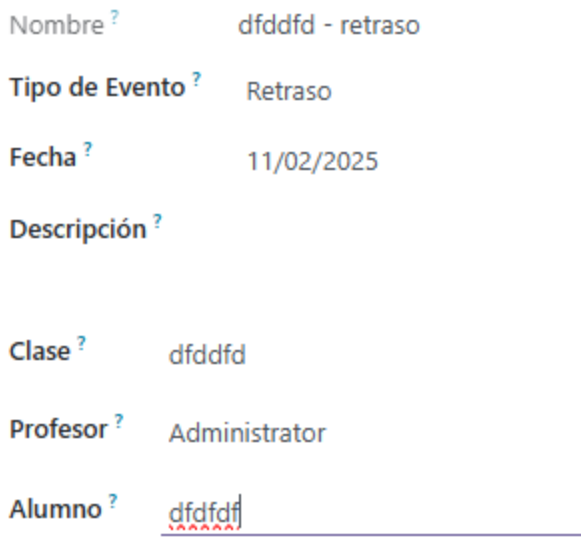
```xml
<record id="view_clase_tree" model="ir.ui.vie
    <field name="name">clase.tree</field>
    <field name="model">gestion.colegio.clase
    <field name="arch" type="xml">
        <tree>
            <field name="name" />
            <field name="activo" />
            <field name="nivel" />
            <field name="fecha_inicio" />
            <field name="fecha_final" />
            <field name="tutor_id" />
            <field name="numero_alumnos" />
        </tree>
```



3. nombre del evento como el nombre de la clase:

Definir el nombre del evento como el nombre de la clase afectada, añadiéndole el tipo de evento.

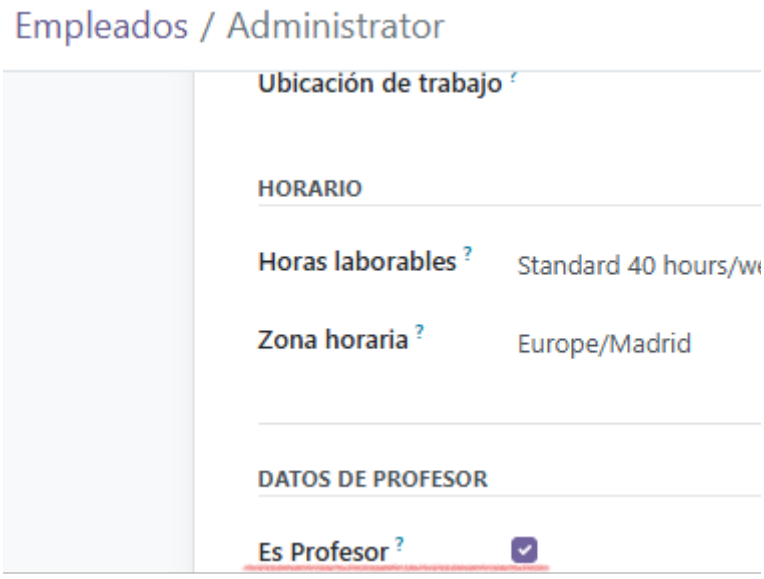
```python
fecha = fields.Date(string="Fecha", default=fields.Date.today, required=True)
descripcion = fields.Text(string="Descripción")
clase_id = fields.Many2one("gestion.colegio.clase", string="Clase", required=Tru
profesor_id = fields.Many2one(
    "hr.employee", string="Profesor", domain=[("es_profesor", "=", True)]
)
alumno_id = fields.Many2one("gestion.colegio.alumno", string="Alumno")

@api.depends("clase_id", "tipo_evento")
def _compute_name(self):
    for evento in self:
        if evento.clase_id and evento.tipo_evento:
            evento.name = f"{evento.clase_id.name} - {evento.tipo_evento}"
        else:
            evento.name = "Evento sin nombre"
        _logger.info(f"Computed event name: {evento.name}")
```



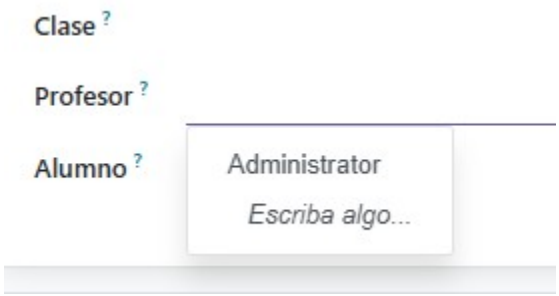
4. ***Elementos de la tabla hr.employee:***

Debemos restringir los datos de entrada al campo responsable con un dominio en campos relacionales: Al seleccionar un professor , solo aquel empleado marcado como profesor (elementos de la tabla hr.employee con el campo professor configurado como verdadero deberán estar visibles).

```
es_profesor = fields.Boolean(string="Es Profesor")
clase_ids = fields.One2many(
```



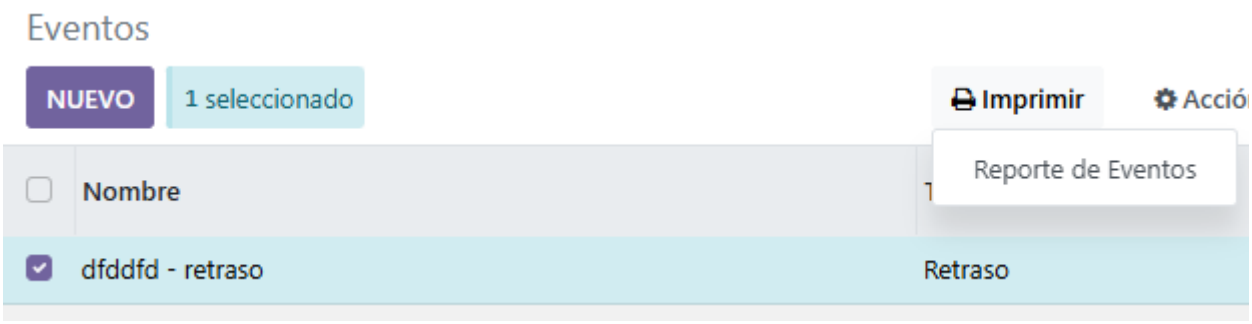
Alhassan Bouhou



5. *Informe para el modelo de Eventos:*

Necesitamos crear un informe para el modelo de Eventos: Para cada evento, tiene que mostrar la información del evento identificador, fecha, profesor asignado, descripción del evento y el listado de estudiantes asociados a ese evento.

```xml
<odoo>
    <data>
        <record id="report_evento_action" model="ir.actions.report">
            <field name="name">Reporte de Eventos</field>
            <field name="model">gestion.colegio.evento</field>
            <field name="report_name">colegio.report_evento_template</field>
            <field name="report_type">qweb-pdf</field>
            <field name="binding_model_id" ref="model_gestion_colegio_evento"></field>
            <field name="binding_type">report</field>
        </record>

        <template id="report_evento_template">
            <t t-name="colegio.report_evento_template">
                <main>
                    <t t-foreach="docs" t-as="evento">
                        <div class="page">
                            <h2>Evento Report</h2>
                            <p>
                                <strong>Tipo Evento:</strong>
                                <t t-if="evento.tipo_evento == 'ausencia'">Ausencia</t>
                                <t t-elif="evento.tipo_evento == 'retraso'">Retraso</t>
                                <t t-elif="evento.tipo_evento == 'felicitacion'">Felicitación</t>
                                <t t-elif="evento.tipo_evento == 'comportamiento'">Comportamiento</t>
                            </p>
                            <p>
                                <strong>Fecha:</strong>
                                <t t-esc="evento.fecha" />
                            </p>
                            <p>
                                <strong>Descripción:</strong>
                                <t t-esc="evento.descripcion" />
                            </p>
                            <p>
```



6. ***Wizard:***

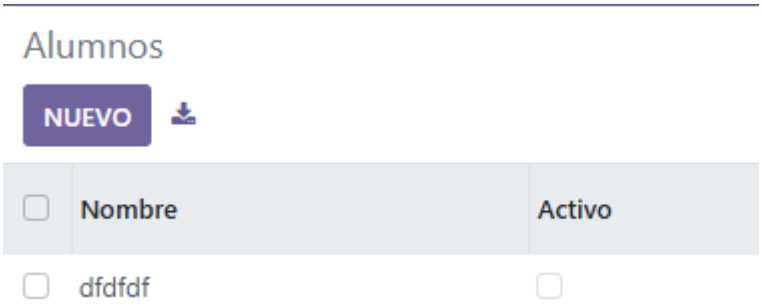
Wizard: Nos piden que creemos un wizard desde un botón dentro de la propia vista de estudiante y que nos permitirá marcar a los mismos como activos.

```xml
<odoo>
    <record id="gestion_colegio_wizard_form" model="ir.ui.view">
        <field name="name">gestion.colegio.wizard.form</field>
        <field name="model">gestion.colegio.wizard</field>
        <field name="arch" type="xml">
            <form string="activar alumnos">
                <group>
                    <field name="alumno_ids">
                        <tree editable="bottom">
                            <field name="name" />
                            <field name="activo" />

                        </tree>
                    </field>
                </group>
                <footer>
                    <button string="Registrar" type="object" name="activate_students"
                        class="btn-primary" />
                    <button string="Cancelar" class="btn-secondary"
                        special="cancel" />
                </footer>
            </form>
        </field>
    </record>
    <record id="action_gestion_colegio_wizard" model="ir.actions.act_window">
        <field name="name">activar alumnos</field>
        <field name="res_model">gestion.colegio.wizard</field>
        <field name="view_mode">form</field>
        <field name="target">new</field>
        <!-- <field name="binding_model_id" ref="model_libreria_libro" /> -->
    </record>
```

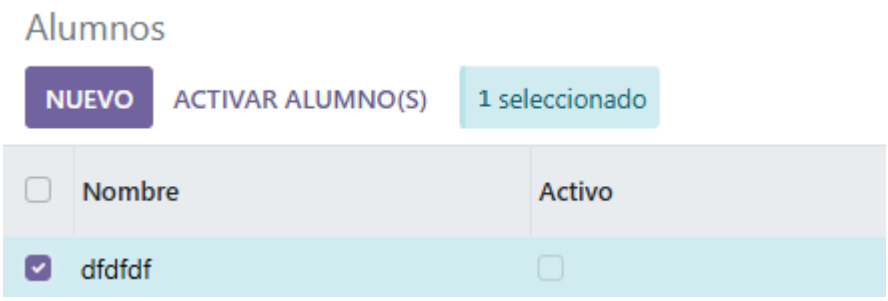
model Wizard :

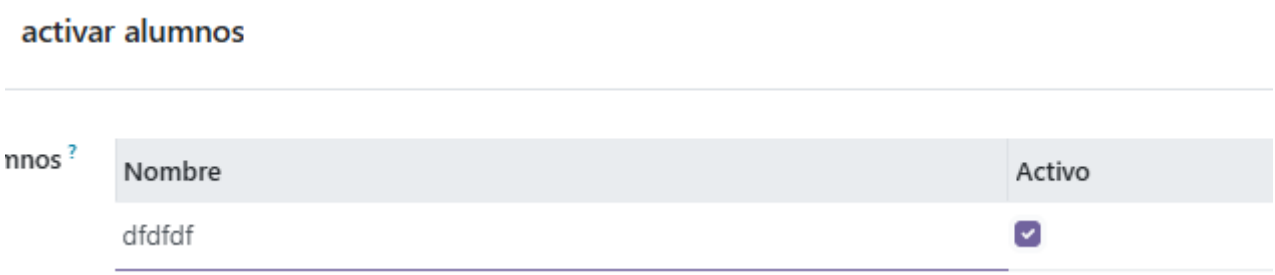
```python
class Wizard(models.Model):
    _name = "gestion.colegio.wizard"
    _description = "Wizard para activar alumnos"

    alumno_ids = fields.Many2many("gestion.colegio.alumno", string="Alumnos")

    def activate_students(self):
        if self.alumno_ids:
            self.alumno_ids.write({"activo": True})
        return {"type": "ir.actions.act_window_close"}
```

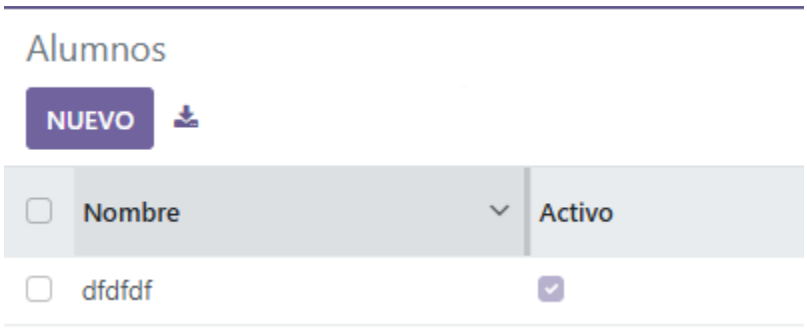
```python
class Alumno(models.Model):
    _name = "gestion.colegio.alumno"
    _description = "Alumno del Colegio"

    name = fields.Char(string="Nombre", required=True)
    apellidos = fields.Char(string="Apellidos", required=True)
    dni = fields.Char(string="DNI/NIE", required=True, unique=True)
    fecha_nacimiento = fields.Date(string="Fecha de Nacimiento", required=True)
    edad = fields.Integer(string="Edad", compute="_compute_edad")
    clase_id = fields.Many2one("gestion.colegio.clase", string="Clase")
    evento_ids = fields.One2many(
        "gestion.colegio.evento", "alumno_id", string="Eventos Relacionados"
    )
    activo = fields.Boolean(String="Activo", default=True)
```



Alhassan Bouhou







7. *Control de acceso:*

Añadir control de acceso a través de archivos de datos al módulo. Deberéis crear un grupo Manager (al cual perteneceran todos los usuarios administradores por defecto) con acceso completo a todos los modelos del módulo, un grupo profesor que puede leer todos los modelos y puede escribir en el modelo de eventos, esto será aplicable a los menús y al wizard creados en el módulo.

```xml
<odoo>
    <!-- Define the module category -->
    <record id="module_category_hr" model="ir.module.category">
        <field name="name">Human Resources</field>
        <!-- Optionally, add other fields such as sequence, description, etc. -->
    </record>

    <!-- Now reference the category in your groups -->
    <record id="group_manager" model="res.groups">
        <field name="name">Manager</field>
        <field name="category_id" ref="module_category_hr" />
    </record>

    <record id="group_profesor" model="res.groups">
        <field name="name">Profesor</field>
        <field name="category_id" ref="module_category_hr" />
    </record>
</odoo>
```

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_school_school,school.school,model_school_school,base.group_user,1,1,1,1
access_manager,Manager,model_gestion_colegio_evento,group_manager,1,1,1,1
access_profesor,Profesor,model_gestion_colegio_evento,group_profesor,1,1,0,0
```

```python
# any module necessary for this one to work correctly
"depends": ["base", "hr"],
# always loaded
"data": [
    # 'security/ir.model.access.csv',
    "security/group.xml",
    "views/views.xml",
    "views/clase_view.xml",
    "views/alumno_view.xml",
    "views/wizard.xml",
    "views/profesor_view.xml",
    "views/menu.xml",
    "reports/report_evento.xml",
],
```

Alhassan Bouhou

Alhassan Bouhou