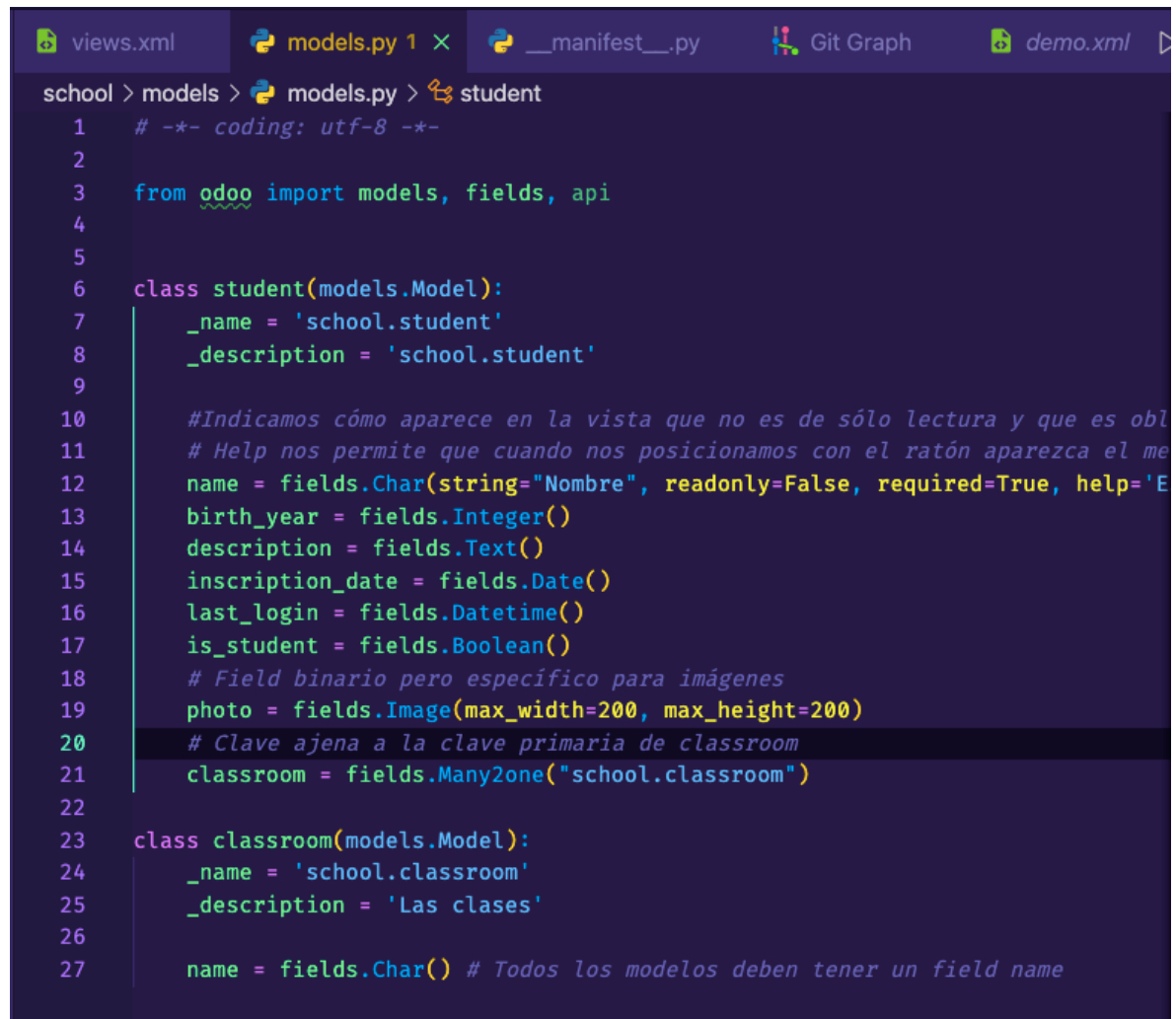


FIELDS RELACIONALES

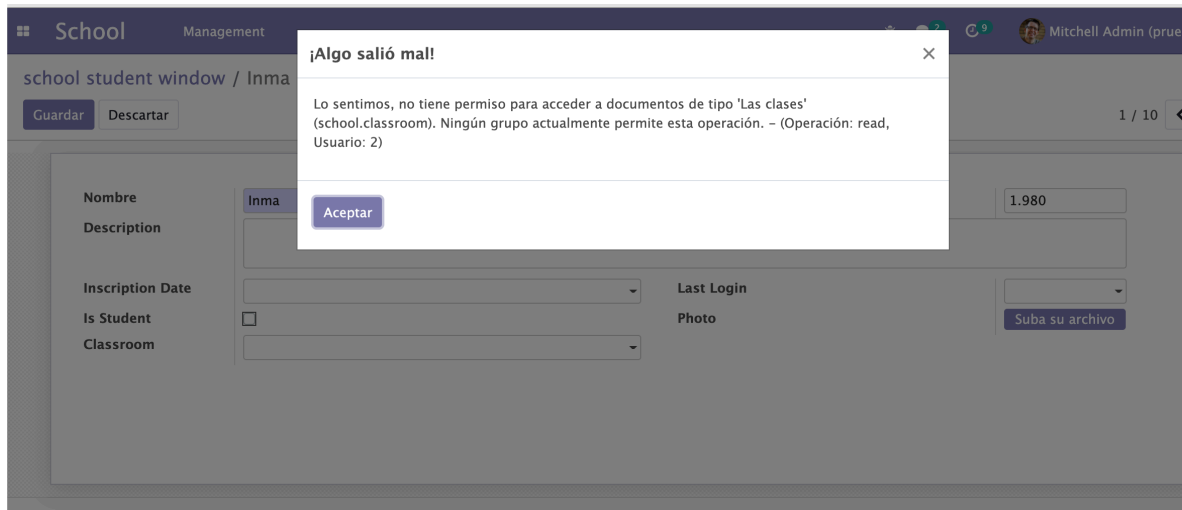
MANY2ONE

- Para crear campos relacionales, primero tenemos que tener alguna clase con la que hacer relación. Vamos a crear la clase **Classroom**.



```
school > models > models.py > student
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class student(models.Model):
7      _name = 'school.student'
8      _description = 'school.student'
9
10     #Indicamos cómo aparece en la vista que no es de sólo lectura y que es obl
11     # Help nos permite que cuando nos posicionamos con el ratón aparezca el me
12     name = fields.Char(string="Nombre", readonly=False, required=True, help='E
13     birth_year = fields.Integer()
14     description = fields.Text()
15     inscription_date = fields.Date()
16     last_login = fields.Datetime()
17     is_student = fields.Boolean()
18     # Field binario pero específico para imágenes
19     photo = fields.Image(max_width=200, max_height=200)
20     # Clave ajena a la clave primaria de classroom
21     classroom = fields.Many2one("school.classroom")
22
23     class classroom(models.Model):
24         _name = 'school.classroom'
25         _description = 'Las clases'
26
27         name = fields.Char() # Todos los modelos deben tener un field name
```

- De este modo en la clase **student** podemos crear un campo Many2one que en el ORM se traduciría como una clave ajena a la clave primaria de **Classroom**.
- Faltan muchas cosas, pero de momento aparece lo siguiente:

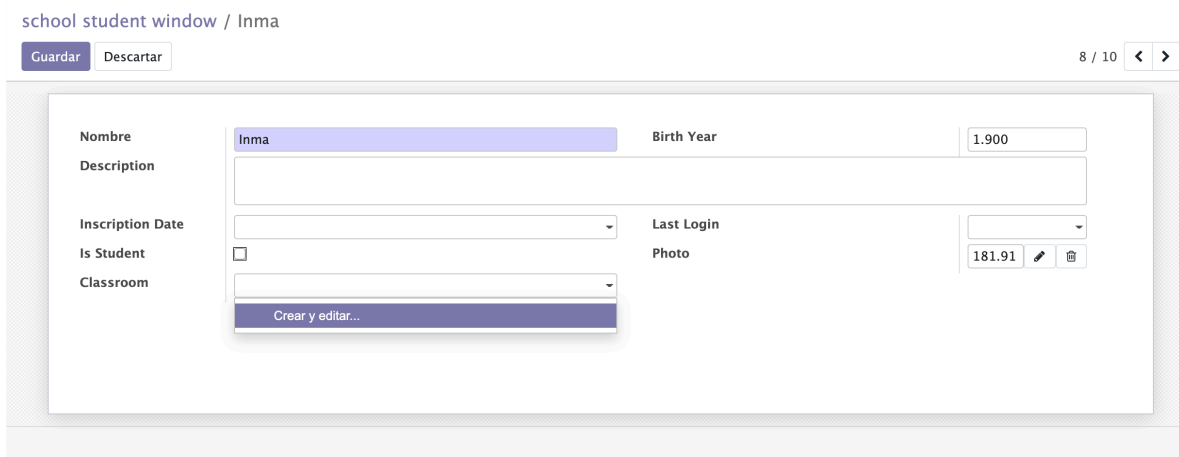


- Lo primero que vamos a hacer es darle permisos:

```

1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 access_school_student,school.student,model_school_student,base.group_user,1,1,1,1
3 access_school_classroom,school.classroom,model_school_classroom,base.group_user,1,1,1,1

```



- Voy a modificar las vistas para crearle un menú y poder acceder y crear clases, etc.

- Lo primero que haré será crear un action en **views.xml**

```

16
17      <!-- actions opening views on models -->
18
19      <record model="ir.actions.act_window" id="school.action_student_window">
20          <field name="name">school student window</field>
21          <field name="res_model">school.student</field>
22          <field name="view_mode">tree,form</field>
23      </record>
24
25      <record model="ir.actions.act_window" id="school.action_classroom_window">
26          <field name="name">school classroom window</field>
27          <field name="res_model">school.classroom</field>
28          <field name="view_mode">tree,form</field>
29      </record>
30

```

- También voy a hacer un menú:

```

<!-- Top menu item -->
<menuitem name="School" id="school.menu_root"/>
<!-- menu categories -->
<menuitem name="Management" id="school.menu_1" parent="school.menu_root"/>
<!-- actions -->
<menuitem name="Students" id="school.menu_student_list" parent="school.menu_1"
    action="school.action_student_window"/>
<menuitem name="Classrooms" id="school.menu_classroom_list" parent="school.menu_1"
    action="school.action_classroom_window"/>
</data>
</odoo>

```

192.168.203.130:8069/web?debug=1#action=362&model=school.student&view_type=list&cids=1&menu_id=258

Aplicaciones Educamos/aulas DWEC Python UML NFS

School Management Mitchell Admin (pruebas)

school student w Students Classrooms

Buscar...

▼ Filtros Agrupar por ★ Favoritos 1-10 / 10

Nombre	Birth Year
<input type="checkbox"/> Inma	1.980
<input type="checkbox"/> Lucía	0
<input type="checkbox"/> Ricardo	0
<input type="checkbox"/> Lucía2	0
<input type="checkbox"/> dsfsdfsdf	0
<input type="checkbox"/> Estudiante prueba	1.980
<input type="checkbox"/> Alumno con foto	1.900
<input type="checkbox"/> Inma	1.900
<input type="checkbox"/> a	0
<input type="checkbox"/> dfsdf	0

School

Management

school classroom window / Nuevo

Guardar

Descartar

Name

DAM2

Nombre

Inma

Birth Year

1.900

Description

Inscription Date

Last Login

Is Student

☐

Photo

181.91

Classroom

DAM2

Crear y editar...

- Tal y como lo hemos establecido, un alumno forma parte de una clase. Una clase contiene muchos alumnos.

ONE2MANY

- De momento, en la clase no aparece la lista de alumnos, para que esto ocurra debemos añadir la relación **one2many** en la clase. Necesita la clase sobre la que se va a realizar la relación y la clave ajena de dicha clase.
- Se declara como un Field pero no se guarda en BDD, porque es simplemente una consulta, el que sí está guardado en BDD es el **many2one** y a partir de éste internamente se hace la consulta.

```
views.xml  models.py 1 x  __manifest__.py  Git Graph  ir.model.ac
school > models > models.py > student
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class student(models.Model):
7      _name = 'school.student'
8      _description = 'school.student'
9
10     #Indicamos cómo aparece en la vista que no es de sólo lectura y que es obl
11     # Help nos permite que cuando nos posicionamos con el ratón aparezca el me
12     name = fields.Char(string="Nombre", readonly=False, required=True, help='E
13     birth_year = fields.Integer()
14     description = fields.Text()
15     inscription_date = fields.Date()
16     last_login = fields.Datetime()
17     is_student = fields.Boolean()
18     # Field binario pero específico para imágenes
19     photo = fields.Image(max_width=200, max_height=200)
20     # Clave ajena a la clave primaria de classroom
21     classroom = fields.Many2one("school.classroom")
22
23     class classroom(models.Model):
24         _name = 'school.classroom'
25         _description = 'Las clases'
26
27         name = fields.Char() # Todos los modelos deben tener un field name
28         #Se declara como un field pero no se guarda en BDD porque es simplemente u
29         #consulta a partir de many2one que sí se guarda en BDD
30         students = fields.One2many("school.student", 'classroom')
```

school student window / Inma / DAM2

Editar

Crear

Acción ▾

1 / 1 < >

Name	DAM2	
Students	Nombre	Birth Year
	Inma	1.900

- Si le doy a **Editar**, desde esta vista podría crear un estudiante.

Guardar Descartar

1 / 1 <

Name	DAM2	
Students		
	Nombre	Birth Year
	Inma	1.900
	Agregar línea	

→ ↻ No es seguro | 192.168.203.130:8069/web?debug=1#id=1&model=school.classroom&view_type=form&cids=1&menu_id=258 ☆

Aplicaciones Educamos/aulas DWEC Python UML NFS

School Management

school student window / Inma / DAM2

CrearStudents

Nombre		Birth Year	0
Description			
Inscription Date		Last Login	
Is Student	<input type="checkbox"/>	Photo	Suba su archivo
Classroom			

Guardar y cerrar Guardar y Nuevo Descartar

- Si quiero añadir un estudiante ya creado, tengo que editarlo y asignarle la clase.

MANY2MANY

- Vamos a crear la clase **teacher**.

```
views.xml | models.py 1, M | __manifest__.py | Git Graph | ir.m
```

```
school > models > models.py > teacher

24     _name = 'school.classroom'
25     _description = 'Las clases'
26
27     name = fields.Char() # Todos los modelos deben tener un field name
28     #Se declara como un field pero no se guarda en BDD porque es simplemente u
29     #consulta a partir de many2one que sí se guarda en BDD
30     students = fields.One2many("school.student", 'classroom')
31
32     class teacher(models.Model):
33         _name = 'school.teacher'
34         _description = 'Los profesores'
35
36         name = fields.Char()
37         # un profesor puede dar clase en varias aulas y en un aula, varios profes
38         classrooms = fields.Many2many('school.classroom')
```

- El campo **many2many** indica que un profesor puede dar clase en varias clases y una clase puede tener varios profesores.
- Creamos el action del profesor.

```
<!-- actions opening views on models -->

<record model="ir.actions.act_window" id="school.action_student_window">
    <field name="name">school student window</field>
    <field name="res_model">school.student</field>
    <field name="view_mode">tree,form</field>
</record>

<record model="ir.actions.act_window" id="school.action_classroom_window">
    <field name="name">school classroom window</field>
    <field name="res_model">school.classroom</field>
    <field name="view_mode">tree,form</field>
</record>

<record model="ir.actions.act_window" id="school.action_teacher_window">
    <field name="name">school teacher window</field>
    <field name="res_model">school.teacher</field>
    <field name="view_mode">tree,form</field>
</record>
```

- Añadimos también el menú correspondiente:

```

<!-- Top menu item -->
<menuitem name="School" id="school.menu_root"/>
<!-- menu categories -->
<menuitem name="Management" id="school.menu_1" parent="school.menu_root"/>
<!-- actions -->
<menuitem name="Students" id="school.menu_student_list" parent="school.menu_1"
          action="school.action_student_window"/>
<menuitem name="Classrooms" id="school.menu_classroom_list" parent="school.menu_1"
          action="school.action_classroom_window"/>
<menuitem name="Teachers" id="school.menu_teacher_list" parent="school.menu_1"
          action="school.action_teacher_window"/>
</data>
</odoo>

```

- Modificamos el fichero referente a los permisos:

```

__manifest__.py x models.py x controllers.py x ir.model.access.csv x demo.xml x views.xml x templates
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 access_school_student,school.student,model_school_student,base.group_user,1,1,1,1
3 access_school_classroom,school.classroom,model_school_classroom,base.group_user,1,1,1,1
4 access_school_teacher,school.teacher,model_school_teacher,base.group_user,1,1,1,1
5

```

- Reiniciamos el servicio

School Management

school student window

Crear Importar

Students
Classrooms
Teachers

Buscar...

Filtros Agrupar por Favoritos

1-10 / 10

Nombre	Birth Year
<input type="checkbox"/> Inma	1.980
<input type="checkbox"/> Lucia	0
<input type="checkbox"/> Ricardo	0
<input type="checkbox"/> Lucia2	0
<input type="checkbox"/> dsfsdfsdf	0
<input type="checkbox"/> Estudiante prueba	1.980
<input type="checkbox"/> Alumno con foto	1.900
<input type="checkbox"/> Inma	1.900
<input type="checkbox"/> a	0
<input type="checkbox"/> dfsdf	0

- Vamos a crear un nuevo profesor para 2DAM.

school teacher window / Nuevo

Guardar Descartar

Name

Classrooms

Antonio

Name

DAM2

Agregar línea

- Ahora vamos a crear una clase:

school classroom window / Nuevo

Guardar Descartar

Name	DAM1	
Students	Nombre	Birth Y...
	Agregar línea	

- No me aparece la lista de profesores porque me faltaría crear el campo **many2many** correspondiente.

```
from odoo import models, fields, api

class student(models.Model):
    _name = 'school.student'
    _description = 'school.student'

    name = fields.Char(string="Nombre", readonly=False, required=True, help='Este es el nombre')
    birth_year = fields.Integer()
    description = fields.Text()
    inscription_date = fields.Date()
    last_login = fields.Datetime()
    is_student = fields.Boolean()
    photo = fields.Image(max_width=200, max_height=200) # Field binario pero específico para imágenes
    classroom = fields.Many2one("school.classroom")

class classroom(models.Model):
    _name = 'school.classroom'
    _description = 'Las clases'

    name = fields.Char() # Todos los modelos deben tener un field name
    students = fields.One2many("school.student", 'classroom')
    teachers = fields.Many2many('school.teacher')

class teacher(models.Model):
    _name = 'school.teacher'
    _description = 'Los profesores'

    name = fields.Char() # Todos los modelos deben tener un field name
    classrooms = fields.Many2many('school.classroom')
```

school classroom window / DAM2

Editar Crear Acción

1 / 2

Name

Students

Teachers

DAM2

Nombre	Birth Year
Inma	1.900

Name

Antonio

PROFUNDIZANDO EN LAS OPCIONES QUE TENEMOS

OPCIONES DE MANY2ONE

```
class student(models.Model):
    _name = 'school.student'
    _description = 'school.student'

    name = fields.Char(string="Nombre", readonly=False, required=True, help='Este es el nombre')
    birth_year = fields.Integer()
    description = fields.Text()
    inscription_date = fields.Date()
    last_login = fields.Datetime()
    is_student = fields.BooleanField()
    photo = fields.ImageField(max_width=200, max_height=200) # Field binario pero específico para imágenes.
    classroom = fields.Many2One('school.classroom', ondelete='set null', help='Clase a la que pertenece')
```

- **ondelete**: qué ocurre si eliminamos la clase: el estudiante se queda sin la clase, el estudiante se borra también, etc.
 - **set null**: el estudiante se queda sin la clase. Es la opción por defecto si no se establece, pero es recomendable establecerla para saber qué va a ocurrir.
 - **restrict**: no se elimina la clase en el estudiante.
- **help**: lo que mostraremos al usuario y/o desarrollador cuando pase el ratón por encima.

OPCIONES DE ONE2MANY

```
class classroom(models.Model):
    _name = 'school.classroom'
    _description = 'Las clases'

    name = fields.Char() # Todos los modelos deben tener un field name
    students = fields.One2many(string='Alumnos', comodel_name='school.student', inverse_name='classroom')
    teachers = fields.Many2many('school.teacher')
```

- **comodel_name**: modelo con el que establecemos la relación. Una clase tiene muchos estudiantes, el modelo serían los estudiantes.
- **inverse_name**: clave ajena de la clase con la que relacionamos. En este caso,

una clase tiene varios estudiantes, y el campo que será la clave ajena del modelo **student** es **classroom**.

- Si no ponemos más atributos, no es necesario especificarlo, pero si ponemos más tenemos que decir a qué se corresponde cada uno.

OPCIONES DE MANY2MANY

```
class classroom(models.Model):
    _name = 'school.classroom'
    _description = 'Las clases'

    name = fields.Char() # Todos los modelos deben tener un field name
    students = fields.One2many(string='Alumnos', comodel_name='school.student', inverse_name='classroom')
    teachers = fields.Many2many(comodel_name='school.teacher',
                               relation='teachers_classroom',
                               column1='classroom_id',
                               column2='teacher_id')

class teacher(models.Model):
    _name = 'school.teacher'
    _description = 'Los profesores'

    name = fields.Char() # Todos los modelos deben tener un field name
    classrooms = fields.Many2many(comodel_name='school.classroom',
                                  relation='teachers_classroom',
                                  column1='teacher_id',
                                  column2='classroom_id')
```

- **relation**: puedo establecer el nombre de la tabla intermedia que si no indico, Odoo establece por defecto para realizar esta relación.

- **column1**: establece el nombre de la columna que va a hacer referencia al modelo de la clase actual, en este caso **classroom** para el campo teachers de classroom.

- **column2**: establece el nombre de la columna que va a hacer referencia al modelo de la clase con la que referenciamos, en este caso **teacher** para el campo teachers de classroom.

- Para el campo classrooms de teacher sería igual pero las columnas se intercambiarían.

- Tenemos que establecerlo en las dos implicadas para que tiren de la misma tabla.

- Guardamos, reiniciamos el servicio y probamos. Como la tabla intermedia es otra que le hemos establecido, a priori no habrá datos guardados en esas relaciones.

- Si le asigno a Antonio las dos clases DAM1 y DAM2, si luego accedo a una clase, debe aparecer Antonio como teacher.

School Management Mitchell Admin (prueb

school teacher window / Antonio

Guardar Descartar 1 / 1 <

Name
Classrooms

Antonio

Name	
DAM2	✕
DAM1	✕
Agregar línea	
<div></div>	

School Management Mitchell Admin (prueb

school classroom window / DAM2

Editar

Crear

Acción ▾

1 / 2 <

Name
Alumnos

Teachers

DAM2

Nombre	Birth Year
Inma	1.900
<div></div>	
<div></div>	

Name
Antonio
<div></div>
<div></div>

- Si realizo una consulta en el terminal de psql:

```
[pruebas=# select * from teachers_classroom
;
 classroom_id | teacher_id
-----+-----
            1 |          1
            2 |          1
(2 rows)
```

- Imaginemos que queremos hacer una referencia a profesores del año pasado y clases del año pasado.

```
class classroom(models.Model):
    _name = 'school.classroom'
    _description = 'Las clases'

    name = fields.Char() # Todos los modelos deben tener un field name
    students = fields.One2many(string='Alumnos', comodel_name='school.student', inverse_name='classroom')
    teachers = fields.Many2many(comodel_name='school.teacher',
                                relation='teachers_classroom',
                                column1='classroom_id',
                                column2='teacher_id')
    teachers_last_year = fields.Many2many(comodel_name='school.teacher',
                                           relation='teachers_classroom_ly',
                                           column1='classroom_id',
                                           column2='teacher_id')

class teacher(models.Model):
    _name = 'school.teacher'
    _description = 'Los profesores'

    name = fields.Char() # Todos los modelos deben tener un field name
    classrooms = fields.Many2many(comodel_name='school.classroom',
                                   relation='teachers_classroom',
                                   column1='teacher_id',
                                   column2='classroom_id')
    classrooms_last_year = fields.Many2many(comodel_name='school.classroom',
                                              relation='teachers_classroom_ly',
                                              column1='teacher_id',
                                              column2='classroom_id')
```

Name	DAM2	
Alumnos	Nombre	Birth Year
	Inma	1.900
Teachers	Name	
	Antonio	
Teachers Last Year	Name	

school teacher window / Antonio

1 / 1

Name
Classrooms

Antonio
Name
DAM2
DAM1

Classrooms Last Year

Name

- De este modo podríamos tener 2 relaciones Many2many entre los mismos modelos pero distintas y con tablas intermedias, por lo tanto, distintas.

CAMPOS RELACIONADOS NO ALMACENADOS EN BDD

```
class student(models.Model):
    _name = 'school.student'
    _description = 'school.student'

    name = fields.Char(string="Nombre", readonly=False, required=True, help='Este es el nombre')
    birth_year = fields.Integer()
    description = fields.Text()
    inscription_date = fields.Date()
    last_login = fields.Datetime()
    is_student = fields.Boolean()
    photo = fields.Image(max_width=200, max_height=200) # Field binario pero específico para imágenes.
    classroom = fields.Many2one('school.classroom', ondelete='set null', help='Clase a la que pertenece')
    teachers = fields.Many2many('school.teacher', related='classroom.teachers', readonly=True)
```

- Imaginemos que queremos que el estudiante muestre la lista de profesores que le dan clase, esto no tiene sentido a nivel de BDD normalizadas ya que en BDD está almacenada la relación entre la clase a la que pertenece el alumno y los profesores que tienen esa clase asignada. Pero puede tener sentido mostrar información de este tipo en el modelo. Esto se haría del siguiente modo.

- En este caso no quiero que la relación sea una nueva tabla, quiero que tire de la tabla intermedia creada entre classroom y teachers, por eso utilizo el atributo **related**. Es importante que el campo destino del **related** sea igual que el campo al que le estamos estableciendo la relación, es decir, **classroom.teachers** hace referencia al atributo **teachers** de la clase **classroom** y esta ya tira de la tabla correspondiente.

- Si añadiésemos el atributo **store=True**, se almacenaría en BDD pero sería información redundante aunque Odoo la gestionaría correctamente. Este atributo sería obligatorio si quisiésemos hacer búsquedas sobre los profesores de un estudiante.

- Debemos establecer este campo como de sólo lectura, porque desde los estudiantes no tiene sentido que se modifique.

```

class classroom(models.Model):
    _name = 'school.classroom'
    _description = 'Las clases'


    name = fields.Char() # Todos los modelos deben tener un field name
    students = fields.One2many(string='Alumnos', comodel_name='school.student', inverse_name='classroom')
    teachers = fields.Many2many(comodel_name='school.teacher',
                                relation='teachers_classroom',
                                column1='classroom_id',
                                column2='teacher_id')
    teachers_last_year = fields.Many2many(comodel_name='school.teacher',
                                           relation='teachers_classroom_ly',
                                           column1='classroom_id',
                                           column2='teacher_id')

```

- Si guardamos y reiniciamos el servicio podemos ver:

school student window / Inma

[Editar](#)
[Crear](#)
Acción ▾
8 / 10
◀ ▶

Nombre	Inma	Birth Year	1.900
Description		Last Login	
Inscription Date	<input type="checkbox"/>	Photo	
Is Student	DAM2		
Classroom	<div>Name</div> <div>Antonio</div> <div></div> <div></div>		
Teachers			