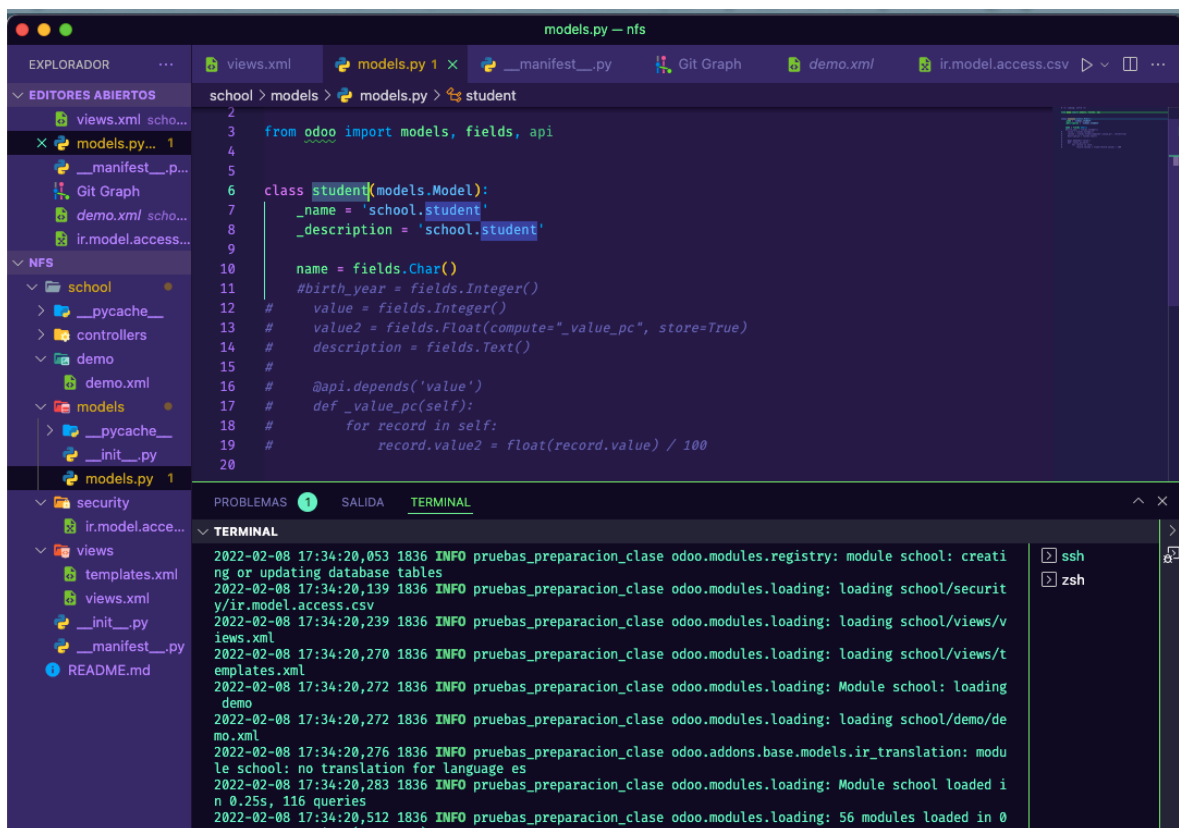


INICIAR SERVICIO

- No nos olvidemos que cuando reiniciamos la máquina se carga el servicio de odoo con el archivo de configuración por defecto y queremos arrancar con nuestra configuración con los dos paths para los módulos que están en producción y el path con los módulos que estamos desarrollando.
- Debemos parar el servicio como root, cambiarnos al usuario odoo y arrancarlo con dicho usuario para que coja la configuración que establecimos en su momento con los dos paths incorporados al addons-path.
- Tendremos dos terminales abiertos, uno para arrancar y parar el servicio con el usuario odoo y otro con el cual editaremos los distintos ficheros.

- Podemos abrir el terminal en VSCode y arrancar el servicio con ssh con el usuario odoo (habiendo parado antes el servicio oficial con root).



```
models.py — nfs
EXPLORADOR ... views.xml models.py 1 __manifest__.py Git Graph demo.xml ir.model.access.csv
EDITORES ABIERTOS school > models > models.py > student
views.xml scho...
models.py... 1
__manifest__.p...
Git Graph
demo.xml scho...
ir.model.access...
NFS
school
__pycache__
controllers
demo
demo.xml
models
__pycache__
__init__.py
models.py 1
security
ir.model.acce...
views
templates.xml
views.xml
__init__.py
__manifest__.py
README.md
PROBLEMAS 1 SALIDA TERMINAL
TERMINAL
2022-02-08 17:34:20,053 1836 INFO pruebas_preparacion_clase odoo.modules.registry: module school: creati
ng or updating database tables
2022-02-08 17:34:20,139 1836 INFO pruebas_preparacion_clase odoo.modules.loading: loading school/securit
y/ir.model.access.csv
2022-02-08 17:34:20,239 1836 INFO pruebas_preparacion_clase odoo.modules.loading: loading school/views/v
iews.xml
2022-02-08 17:34:20,270 1836 INFO pruebas_preparacion_clase odoo.modules.loading: loading school/views/t
emplates.xml
2022-02-08 17:34:20,272 1836 INFO pruebas_preparacion_clase odoo.modules.loading: Module school: loading
demo
2022-02-08 17:34:20,272 1836 INFO pruebas_preparacion_clase odoo.modules.loading: loading school/demo/de
mo.xml
2022-02-08 17:34:20,276 1836 INFO pruebas_preparacion_clase odoo.addons.base.models.ir_translation: modu
le school: no translation for language es
2022-02-08 17:34:20,283 1836 INFO pruebas_preparacion_clase odoo.modules.loading: Module school loaded i
n 0.25s, 116 queries
2022-02-08 17:34:20,512 1836 INFO pruebas_preparacion_clase odoo.modules.loading: 56 modules loaded in 0
.48s, 116 queries (10 extra)
```

TIPOS DE DATOS EN ODOO

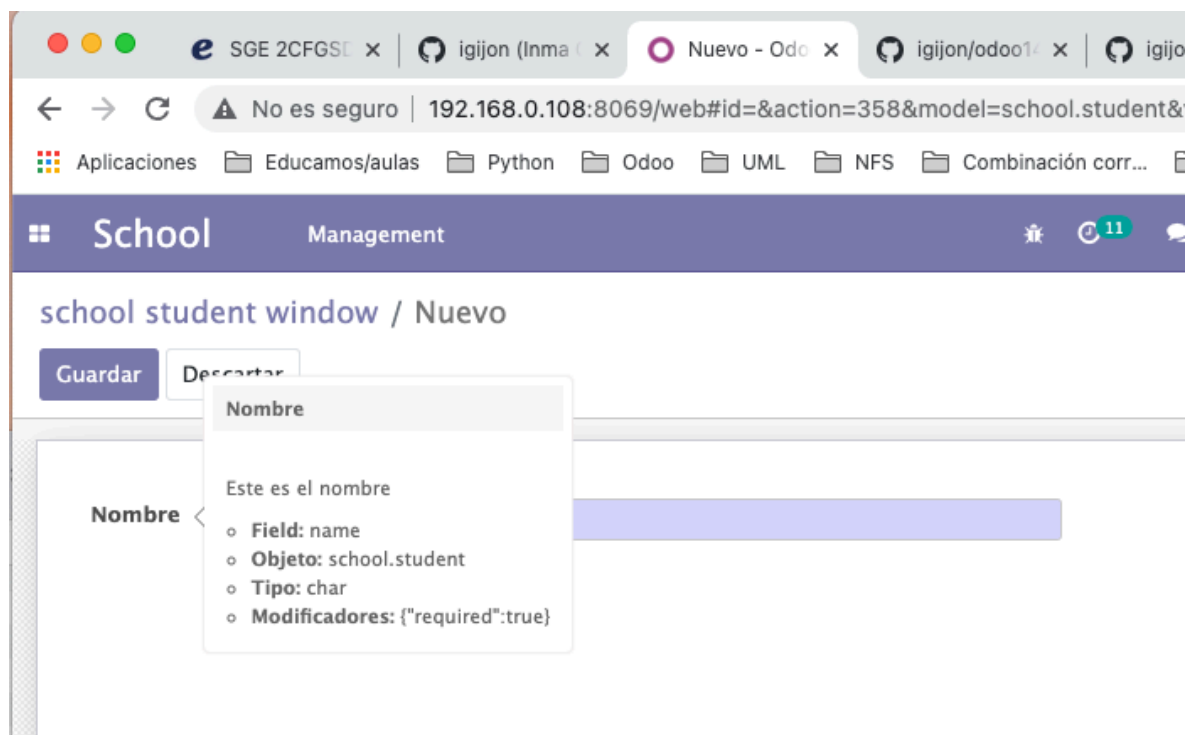
- Podemos acceder a la documentación de Odoo mediante:

<https://www.odoo.com/documentation/14.0/reference/orm.html>

Podemos indicar algunos atributos a los **Fields**

```
views.xml | models.py 1, M | __manifest__.py | Git Graph | demo.xml | ir.model.acce...
school > models > models.py > ...
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     name = fields.Char(string="Nombre", readonly=False, required=True, help='Este es el nombre')
11     #birth_year = fields.Integer()
12     #value = fields.Integer()
13     #value2 = fields.Float(compute="_value_pc", store=True)
14     #description = fields.Text()
15     #
16     # @api.depends('value')
17     # def _value_pc(self):
18     #     for record in self:
19     #         record.value2 = float(record.value) / 100
20
```

- En el primer caso estamos indicando cómo aparecerá en la vista, que no va a ser de sólo lectura y que es obligatorio. De este modo, si reiniciamos el servicio y vemos la vista de Student, observaremos que el campo del nombre ha cambiado porque ahora es obligatorio.
- help nos permite que cuando el ratón se posiciona encima aparece ese mensaje.



- Hay varios tipos de **fields**. Por ejemplo:

```
views.xml | models.py 1, M | __manifest__.py | Git Graph | demo.xml | ir.model.acce
school > models > models.py > ...
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 obligatorio
11    # Help nos permite que cuando nos posicionamos con el ratón aparezca el mensaje
12    name = fields.Char(string="Nombre", readonly=False, required=True, help='Este es el nombre')
13    birth_year = fields.Integer()
14    #birth_year = fields.Integer()
15    # value = fields.Integer()
16    # value2 = fields.Float(compute="_value_pc", store=True)
17    # description = fields.Text()
18    #
19    # @api.depends('value')
20    # def _value_pc(self):
```

- De este modo, en el formulario debe aparecer la opción de modificar el año pero en la lista no, que es la vista tree, porque no tiene incorporado mostrar ese campo.

school student window / Lucía

Guardar Descartar

Nombre

Lucía

Birth Year

0

School

Management

school student window

Buscar...

▼ Filtros ▾ ≡ Agrupar por ▾ ★ Favoritos ▾

☐ Nombre

☐ Inma

☐ Lucía

☐ Ricardo

☐ Lucía2

☐ dsfsdfsdf

☐ Estudiante prueba

- Para añadirlo en las vistas, tengo que modificar **views/views.xml**. Al reiniciar el servidor ya aparece:

The image shows a development environment with two parts: a code editor at the top and a web browser at the bottom.

Code Editor (views.xml):

```
1 <odoo>
2   <data>
3     <!-- explicit list view definition -->
4
5     <record model="ir.ui.view" id="school.student_list">
6       <field name="name">school.student list</field>
7       <field name="model">school.student</field>
8       <field name="arch" type="xml">
9         <tree>
10          <field name="name" />
11          <field name="birth_year" />
12        </tree>
13      </field>
14    </record>
15
16
```

Web Interface (School Management):

The browser shows the 'School' management interface. The title is 'school student window'. There is a search bar and buttons for 'Crear', 'Importar', and a download icon. Below the buttons are filters: 'Filtros', 'Agrupar por', and 'Favoritos'. The page shows 1-6 / 6 items.

<input type="checkbox"/>	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

- Si nos fijamos, si yo en el modelo añado los Fields con la sintaxis **snake case**, propia de python con palabras en minúscula separadas por `_`, automáticamente la vista detecta que debe ponerlos así sin poner el atributo **string** como parámetro.
- Es importante codificar en inglés.

```
views.xml  models.py 1, M x  __manifest__.py  Git Graph  den ▾
school > models > models.py > ...
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     photo = fields.Binary()
19     # Este es el formato tradicional para guardar fotos, aunque ahora hay uno
20     # imagen en las versiones actuales de Odoo que veremos después
21
```

School Management 2 9 Mitchell Admin ▾

school student window / Nuevo

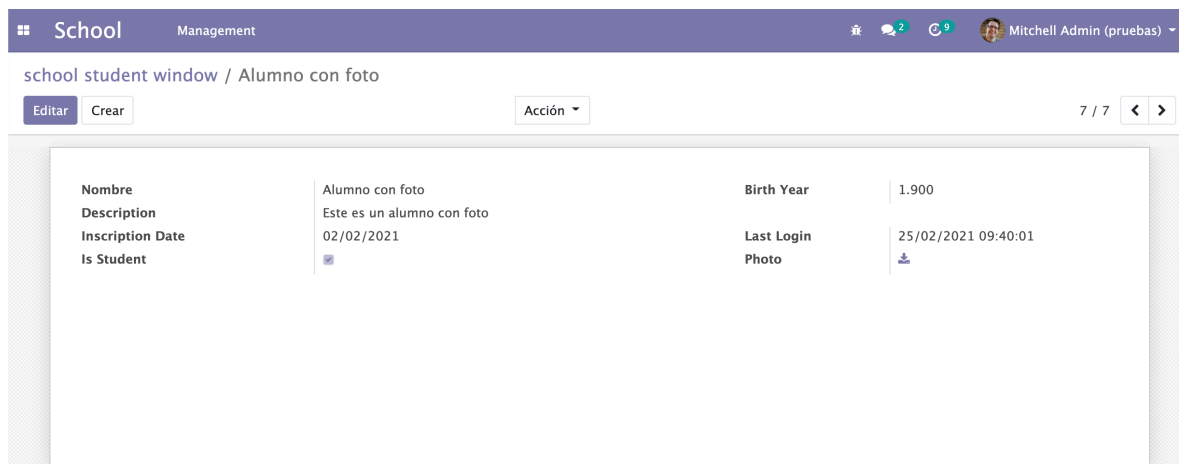
Guardar Descartar

Nombre	<input type="text"/>	Birth Year	<input type="text" value="0"/>
Description	<input type="text"/>		
Inscription Date	<input type="text"/>	Last Login	<input type="text"/>

febrero 2021

#	lu	ma	mi	ju	vi	sá	do
5	1	2	3	4	5	6	7
6	8	9	10	11	12	13	14
7	15	16	17	18	19	20	21
8	22	23	24	25	26	27	28
9	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14

✓



- Para cada tipo de campo, Odoo establece un Widget. En este caso, el tipo de campo Binary no muestra una vista previa de la foto. Si lo cambiamos a Image, sin cambiar la vista tampoco se muestra la vista previa. Con este tipo de Field, puedo establecer parámetros propios de una imagen.

```

views.xml  models.py 1 x  __manifest__.py  Git Graph  demo.xml
school > models > models.py > ...
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     photo = fields.Image(max_width=200, max_height=200)
19     # Field binario pero específico para imágenes

```

- De este modo se guarda, como máximo con este tamaño.

- Estos serían en líneas generales los tipos de campos básicos. Ahora vamos a comenzar con los tipos de datos relacionales que nos permitirán establecer relaciones del tipo: alumnos que pertenecen a una clase, etc. Establecen relaciones entre distintos modelos.

