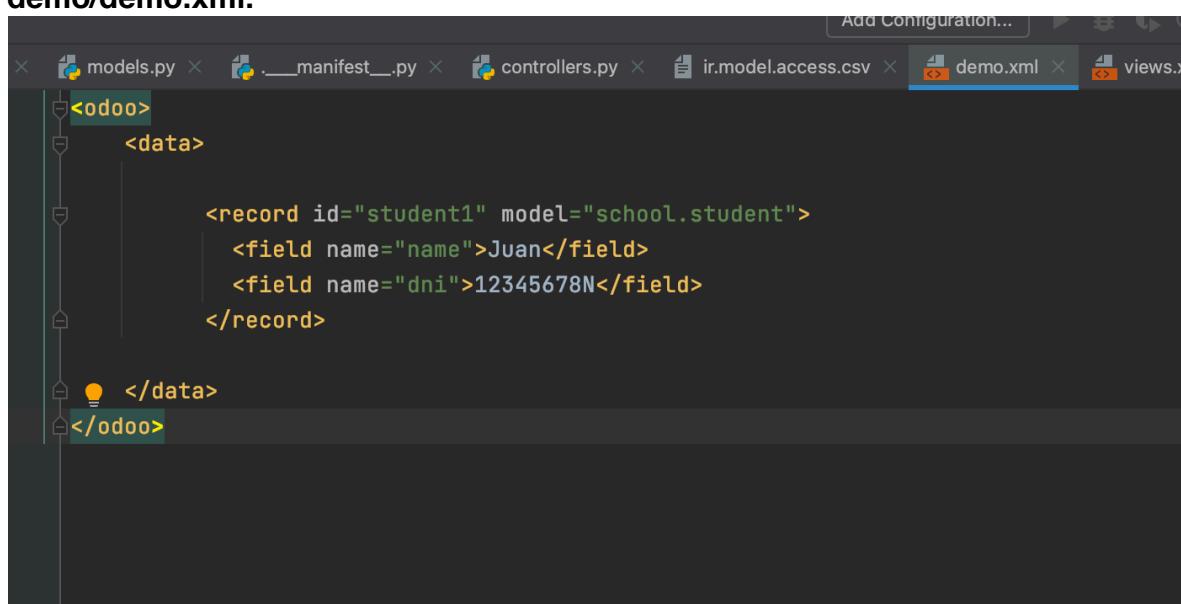


## FICHEROS DE DATOS

- Los ficheros de datos son aquellos que permiten que un módulo añada datos a la BDD.
- Un ejemplo es el **views.xml** que añade datos a tablas. Introduce datos en el modelo **ir.ui.view**. Cada field será una de las columnas que se guardan en la tabla, por tanto esto, ya es un fichero de datos.

```
<record model="ir.ui.view" id="school.student_list">
    <field name="name">school.student list</field>
    <field name="model">school.student</field>
    <field name="arch" type="xml">
        <tree>
            <field name="name"/>
            <field name="password"/>
            <field name="birth_year"/>
        </tree>
    </field>
</record>
```

- Vamos a hacer otro fichero de datos. Lo vamos a añadir en datos de demo para que si seleccionan esa opción, tengamos ya estudiantes sobre los que trabajar.
- Podemos tener un CSV que cumpla la estructura del modelo y se pueden importar directamente mediante la aplicación de Odoo, mediante la opción **importar**. Esta sería la opción de hacerlo cuando ya está en producción y queremos importar datos.
- También tenemos la posibilidad de por ejemplo hacer datos de demo. Fichero **demo/demo.xml**.



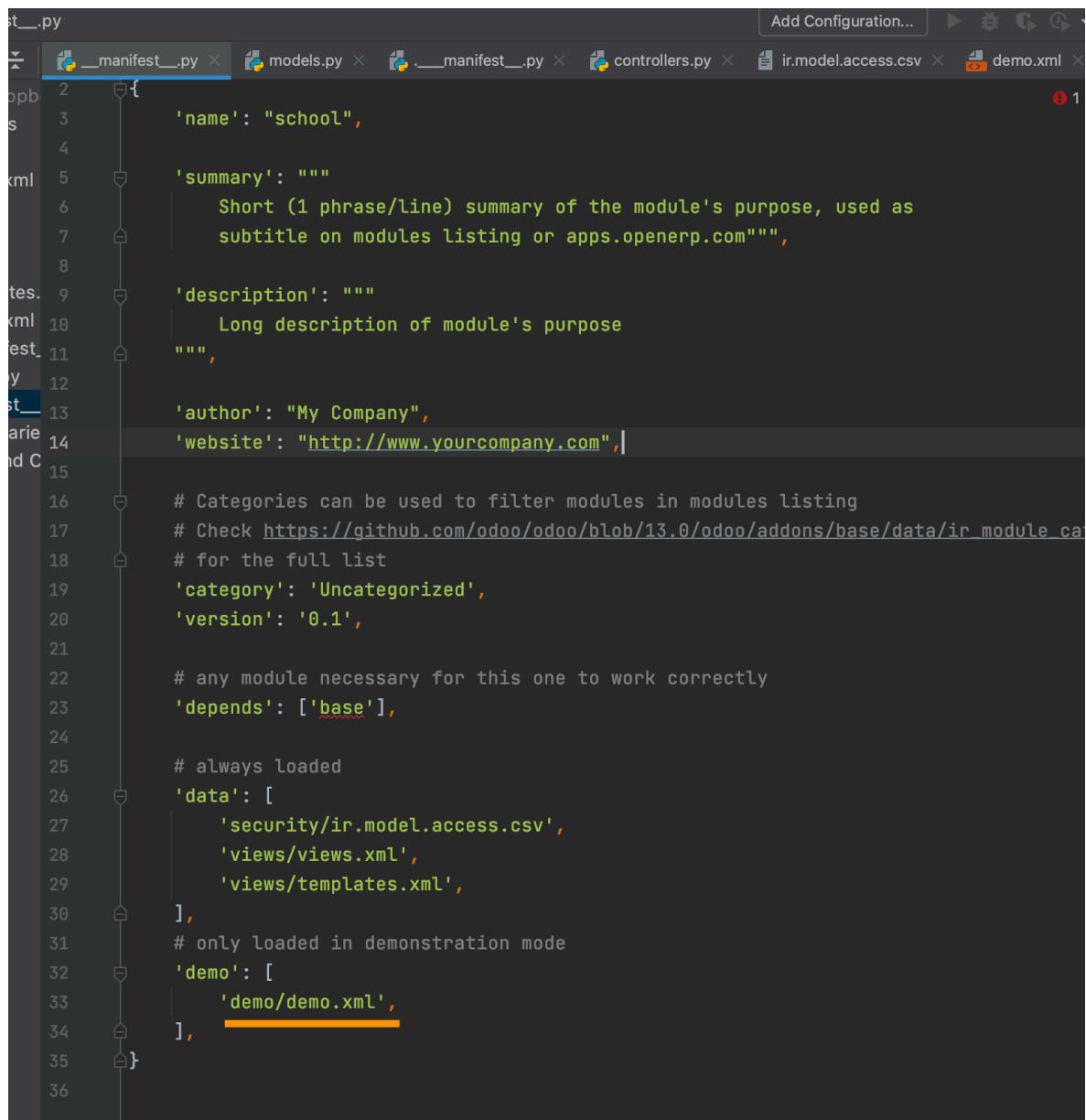
```
<odoo>
    <data>

        <record id="student1" model="school.student">
            <field name="name">Juan</field>
            <field name="dni">12345678N</field>
        </record>

    </data>
</odoo>
```

- dni tiene que cumplir las restricciones de DNI válido y único.

- Para que esto funcione lo tenemos que añadir en **\_\_manifest\_\_.py**



```
st__.py
1  __manifest__.py x  models.py x  __manifest__.py x  controllers.py x  ir.model.access.csv x  demo.xml x
2  {
3      'name': "school",
4
5      'summary': """
6          Short (1 phrase/line) summary of the module's purpose, used as
7          subtitle on modules listing or apps.openerp.com""",
8
9      'description': """
10         Long description of module's purpose
11         """,
12
13      'author': "My Company",
14      'website': "http://www.yourcompany.com",
15
16      # Categories can be used to filter modules in modules listing
17      # Check https://github.com/odoo/odoo/blob/13.0/odoo/addons/base/data/ir_module_cat
18      # for the full list
19      'category': 'Uncategorized',
20      'version': '0.1',
21
22      # any module necessary for this one to work correctly
23      'depends': ['base'],
24
25      # always loaded
26      'data': [
27          'security/ir.model.access.csv',
28          'views/views.xml',
29          'views/templates.xml',
30      ],
31      # only loaded in demonstration mode
32      'demo': [
33          'demo/demo.xml',
34      ],
35  }
```

- Si refrescamos, deberá aparecer el estudiante de prueba y veremos que le ha generado automáticamente una contraseña y una fecha de inscripción.

school student window / Juan

		Acción ▾	14 / 14 < >
<b>Nombre</b>	Juan	<b>Birth Year</b>	0
<b>DNI</b>	12345678N	<b>Password</b>	L7Q_08x7YL_uQBCe
<b>Description</b>		<b>Last Login</b>	
<b>Inscription Date</b>	16/02/2021 13:06:31	<b>Photo</b>	
<b>Is Student</b>	<input type="checkbox"/>		
<b>Classroom</b>			
<b>Teachers</b>			
<b>Name</b>			
<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>			

- Vamos a generar un csv de prueba mediante una página web que genera datos de prueba: <https://www.mockaroo.com/>
- Vamos a generar 1000 registros de prueba con los campos: **name**, **DNI** y **birthyear**.

Need some mock data to test your app? Mockaroo lets you generate up to 1,000 rows of realistic test data in CSV, JSON, SQL, and Excel formats.

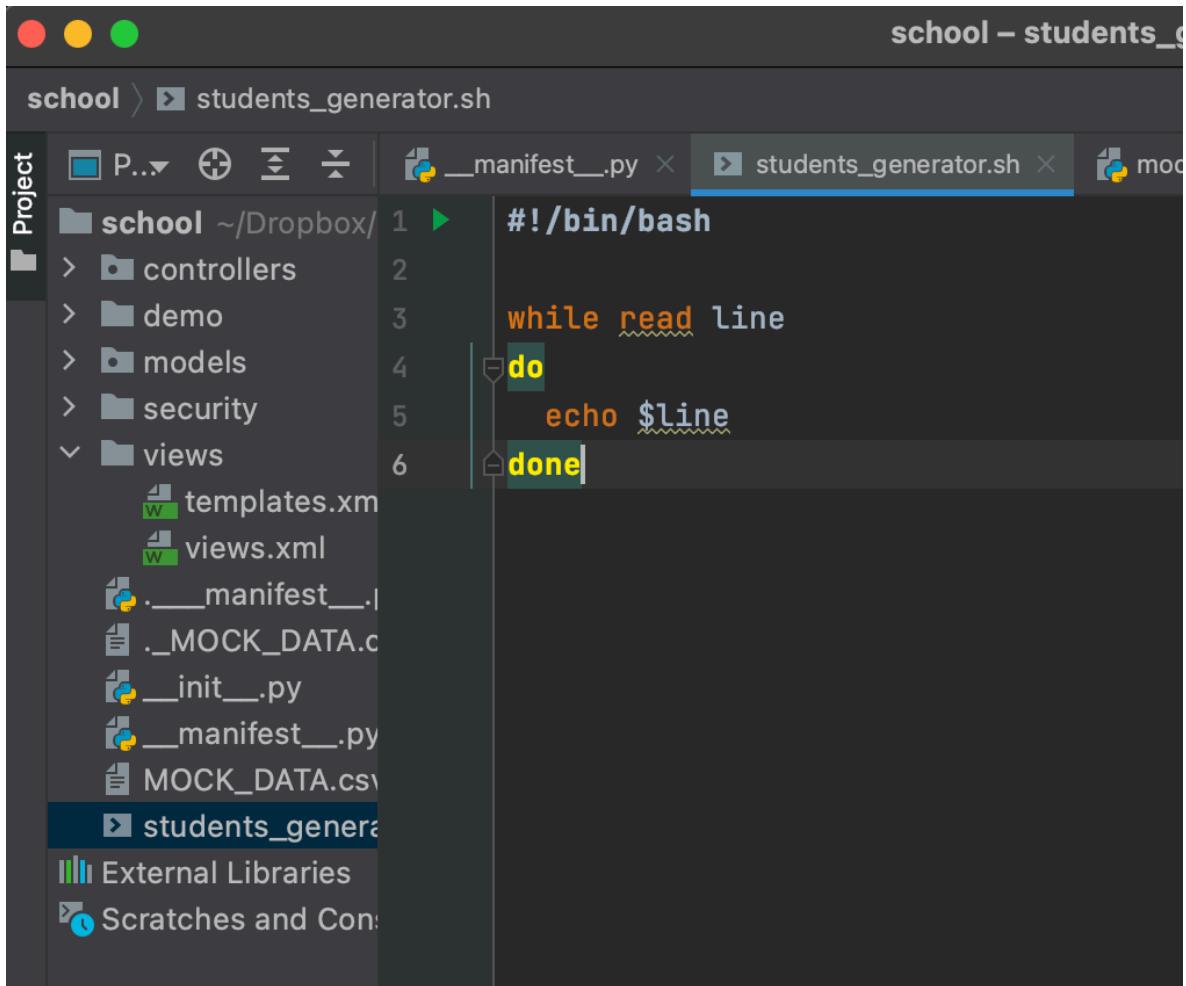
Download data using your browser or sign in and create your own [Mock APIs](#).

Need more data? Plans start at just \$50/year. Mockaroo is also available as a [docker image](#) that you can deploy in your own private cloud.

Field Name	Type	Options
<input type="text"/> name	Full Name	blank: 0 %
<input type="text"/> DNI	Character Sequence	#####^ blank: 0 %
<input type="text"/> birthyear	Car Model Year	blank: 0 %
<a href="#">Add another field</a>		
# Rows:	<input type="text"/> 1000	Format: <input type="button" value="CSV"/>
Line Ending:		<input type="button" value="Unix (LF)"/>
Include: <input checked="" type="checkbox"/> header <input type="checkbox"/> BOM		
<a href="#">Download Data</a>   <a href="#">Preview</a>   <a href="#">More ▾</a> Want to save this for later? <a href="#">Sign up for free</a> .		

MOCK_DATA.csv		
name	DNI	birthyear
Vicki Cunnell	57699468G	1967
Emlyn Mcnelly	15006838V	1998
Anni Dutt	47689063C	1995
Selig Ower	82474061Z	2010
Jarrett Holbie	45103185T	2004
Hazel Loyley	054925195	2011
Blake Sictornes	27265713D	2002
Fannie Tidbold	77516174X	2003
Dud Lewcock	86314185Z	2012
Lyon Kempton	84444777R	1985
Nickie Sapsed	97051957B	1997
Domingo Shakespeare	19491926W	1997
Cayla Iacovino	58391108V	1976
Coralyn Elwin	82917638E	2006
Farris Isaacs	35109023K	1998
Lilias Jasper	17953534A	1992
Errick Abramin	86205858U	2008
Valenka Beqgini	61655251G	1994
Roddy Linguard	52583547P	2011
Adelaida Dilloway	63769038A	1996
Hendrick Cherm	25024232J	1993
Sansone Lawther	92958331W	1992
Peggie Triggs	02394073D	2006
Alard Mearing	95728622B	2006
Odelinda Baigrie	39453993H	1984
Maxi Spellacey	19866894E	1988
Dani Harryman	16186429X	2008
Casey Easterbrook	07695927I	1997
Melanie Loy	70383461S	2007

- Ahora vamos a ver cómo generar un fichero de datos demo en xml a partir del csv.
- Para ello hay varias herramientas, nosotros vamos a crear un **.sh** que se va a llamar **students\_generator.sh** y lo vamos a guardar en el directorio raíz del proyecto.
- Vamos a generar un Shell script que a partir del csv genere los estudiantes.
- Para empezar creamos un sh que sólo imprima las líneas que lee:



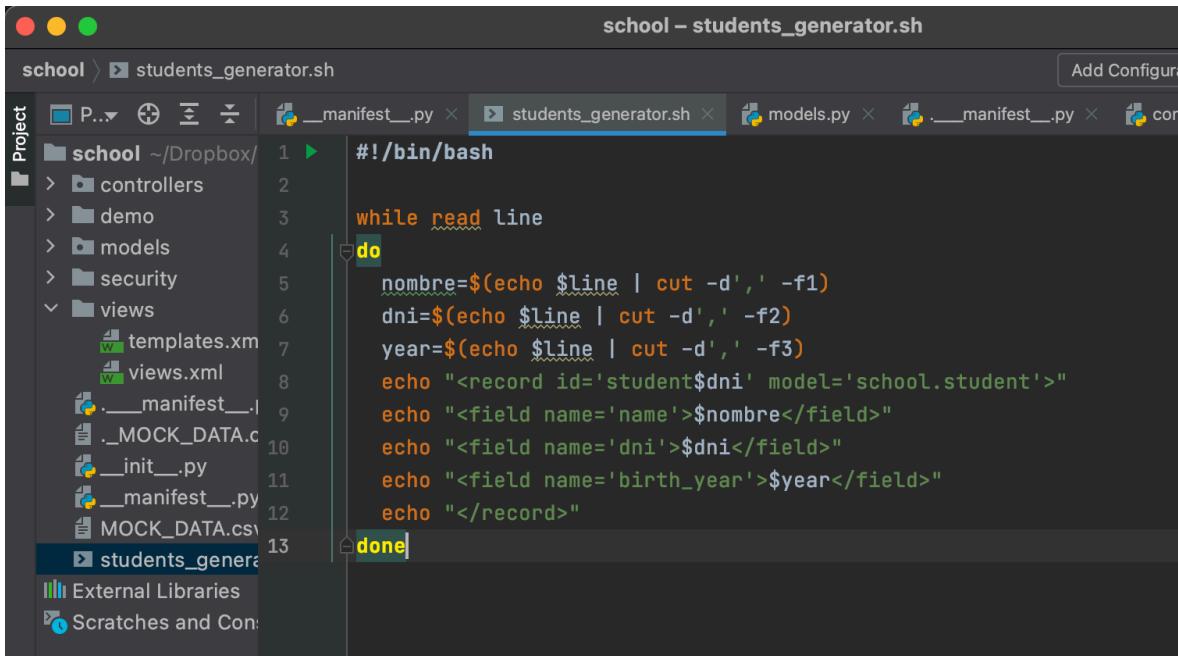
The screenshot shows a terminal window with the title "school – students\_generator.sh". The command entered is "ls". The output lists several files and directories:

```
alumno@alumnodam:/opt/odoo/odoo/modules/school$ ls
__manifest__.py  students_generator.sh
```

- Desde el terminal, le damos permisos y lo probamos. En el mismo directorio tenemos el .csv que queremos leer.

```
[alumno@alumnodam:/opt/odoo/odoo/modules/school$ chmod 777 students_generator.sh
[alumno@alumnodam:/opt/odoo/odoo/modules/school$ ./students_generator.sh < MOCK_DATA.csv
name,DNI,birthyear
Vicki Cunnell,57699468G,1967
Emlynn McNelly,15006838V,1998
Anni Dutt,47689063C,1995
Selig Ower,82474061Z,2010
Jarrett Holbie,45103185T,2004
Hazel Loyley,05492519S,2011
Blake Sictornes,27265713D,2002
Fannie Tidbold,77516174X,2003
Dud Lewcock,86314185Z,2012
Lyon Kempton,84444777R,1985
Nickie Sapshed,97051957B,1997
Domingo Shakespeare,19491926W,1997
Cayla Iacovino,58391108V,1976
Coralyn Elwin,82917638E,2006
Farris Isaacs,35109023K,1998
Liliias Jasper,17953534A,1992
Errick Abramin,86205858U,2008
Valenka Beggini,61655251G,1994
Roddy Linguard,52583547P,2011
```

- El shell script nos quedaría:



```
school - students_generator.sh
school > ./students_generator.sh < MOCK_DATA.csv > demo/students.xml
alumno@alumnodam:/opt/odoo/odoo/modules/school$ ./students_generator.sh < MOCK_DATA.csv > demo/students.xml
alumno@alumnodam:/opt/odoo/odoo/modules/school$
```

The screenshot shows a terminal window with the command `./students_generator.sh < MOCK_DATA.csv > demo/students.xml` being run. The output of the command is also visible in the terminal window.

- Vamos a redireccionar la salida a un fichero .xml.

```
[alumno@alumnodam:/opt/odoo/odoo/modules/school$ ./students_generator.sh < MOCK_DATA.csv > demo/students.xml
alumno@alumnodam:/opt/odoo/odoo/modules/school$
```

The screenshot shows the PyCharm IDE interface with the 'students.xml' file open in the center editor pane. The code is an XML document defining multiple student records. Each record contains fields for name, DNI, and birth year. The XML structure is as follows:

```
<record id='studentDNI' model='school.student'>
    <field name='name'>name</field>
    <field name='dni'>DNI</field>
    <field name='birth_year'>birthyear</field>
</record>
<record id='student57699468G' model='school.student'>
    <field name='name'>Vicki Cunnell</field>
    <field name='dni'>57699468G</field>
    <field name='birth_year'>1967</field>
</record>
<record id='student15006838V' model='school.student'>
    <field name='name'>Emlynn McNelly</field>
    <field name='dni'>15006838V</field>
    <field name='birth_year'>1998</field>
</record>
<record id='student47689063C' model='school.student'>
    <field name='name'>Anni Dutt</field>
    <field name='dni'>47689063C</field>
    <field name='birth_year'>1995</field>
</record>
<record id='student82474061Z' model='school.student'>
    <field name='name'>Selig Ower</field>
    <field name='dni'>82474061Z</field>
    <field name='birth_year'>2010</field>
</record>
<record id='student45103185T' model='school.student'>
    <field name='name'>Jarrett Holbie</field>
    <field name='dni'>45103185T</field>
    <field name='birth_year'>2004</field>
</record>
<record id='student05492519S' model='school.student'>
    <field name='name'>Hazel Loyley</field>
    <field name='dni'>05492519S</field>
    <field name='birth_year'>2011</field>
</record>
```

- Tenemos que añadir el principio y el final. Podemos mediante **PyCharm**, **Code Reformas Code** formatear nuestro código y quedaría así.

```
<odoo>
<data>
<record id='studentDNI' model='school.student'>
    <field name='name'>name</field>
    <field name='dni'>DNI</field>
    <field name='birth_year'>birthyear</field>
</record>
<record id='student57699468G' model='school.student'>
    <field name='name'>Vicki Cunnell</field>
    <field name='dni'>57699468G</field>
    <field name='birth_year'>1967</field>
</record>
<record id='student15006838V' model='school.student'>
    <field name='name'>Emlynn McNelly</field>
    <field name='dni'>15006838V</field>
    <field name='birth_year'>1998</field>
</record>
<record id='student47689063C' model='school.student'>
    <field name='name'>Anni Dutta</field>
</record>
```

- Tenemos que quitar el primer récord porque coge la fila de los títulos del CSV.
- Nos queda añadir en **\_\_manifest\_\_.py** la ruta de nuestro xml para que también lo coja como datos de demo.

```
school > __manifest__.py
Project school ~/Dropbox/
  controllers
  demo
    demo.xml
    students.xml
  models
  security
  views
    templates.xml
    views.xml
  __manifest__.py
  __init__.py
  __manifest__.py
  MOCK_DATA.csv
  students_generator.sh
  students.xml
  models.py
  co

6   Short (1 phrase/Line) summary of the module's purpose,
7   subtitle on modules listing or apps.openerp.com"""
8
9   'description': """
10      Long description of module's purpose
11      """,
12
13   'author': "My Company",
14   'website': "http://www.yourcompany.com",
15
16   # Categories can be used to filter modules in modules list
17   # Check https://github.com/odoo/odoo/blob/13.0/odoo/addons
18   # for the full list
19   'category': 'Uncategorized',
20   'version': '0.1',
21
22   # any module necessary for this one to work correctly
23   'depends': ['base'],
24
25   # always loaded
26   'data': [
27       'security/ir.model.access.csv',
28       'views/views.xml',
29       'views/templates.xml',
30   ],
31   # only loaded in demonstration mode
32   'demo': [
33       'demo/demo.xml',
34       'demo/students.xml',
35   ],
36 }
```

- Reiniciamos el servicio y comprobamos que se han añadido todos los estudiantes como datos de demo.

school student window

		Buscar...	
		Filtros	Agrupar por
		Favoritos	
		1-80 / 1014 < >	
<input type="checkbox"/>	Blake Sictornes	9AbG3dP7NGA7LVA_	2.002
<input type="checkbox"/>	Fannie Tidbold	mukAlpt_tnYn-DxE	2.003
<input type="checkbox"/>	Dud Lewcock	265Qj0n0513l8cGU	2.012
<input type="checkbox"/>	Lyon Kempton	nFKkt2chyj3jf7Zp	1.985
<input type="checkbox"/>	Nickie Sapshed	3NxbMF54-rkitVSW	1.997
<input type="checkbox"/>	Domingo Shakespeare	WxtatNP9MXN5GW	1.997
<input type="checkbox"/>	Cayla Iacovino	ShklmFtkJjeP8ZVB	1.976
<input type="checkbox"/>	Coralyn Elwin	5RfrxPRY4r5wDvdQ	2.006
<input type="checkbox"/>	Farris Isaacs	7ZeWimlTvbdVEM60	1.998
<input type="checkbox"/>	Lilias Jasper	O-ombYYlm1qLX737	1.992
<input type="checkbox"/>	Errick Abramin	Ef0etbKGu_POLRD1	2.008
<input type="checkbox"/>	Valenka Beggini	KWw5bDfjNDJ3XMSA	1.994
<input type="checkbox"/>	Roddy Linguard	2nhU5oxFFoc_kf1Z	2.011
<input type="checkbox"/>	Adelaida Dillowy	BaMAyp-MpuksaV73	1.996
<input type="checkbox"/>	Hendrick Cherm	LPCTex8JrSzINXgA	1.993
<input type="checkbox"/>	Sansone Lawther	GTd3kmODyfB7e18-	1.992
<input type="checkbox"/>	Peggie Triggs	SZ1t9V4MSIRPQkz1	2.006
<input type="checkbox"/>	Alard Mearing	XcBGzMAkOorgzsl7	2.006
<input type="checkbox"/>	Odelinda Baigrie	ETwnYoJU6ULkQ6yy	1.984
<input type="checkbox"/>	Maxi Spellacey	8hp9zhqd8dl2DCPw	1.988
<input type="checkbox"/>	Dani Harryman	yglRoxyh-dHPIV01	2.008
<input type="checkbox"/>	Casey Easterbrook	yxtGTyedO5ad99ku	1.997
<input type="checkbox"/>	Melanie Loy	u0AGWcZ4rK1Bpesk	2.007
<input type="checkbox"/>	Agna Aliso	x1dvxnaCBS3HCpfS	2.005
<input type="checkbox"/>	Linell Deamer	YQu5lyWSiVREdtQi	2.007
<input type="checkbox"/>	Arlie Jodlkowski	C-mlbOQ_UW0ivHbi	1.988
<input type="checkbox"/>	Jacqueta Healing	IKebv2fzXDhuaxM3	1.992
<input type="checkbox"/>	Goldi Laxon	5F6aoXtuBhZXumXN	1.992
<input type="checkbox"/>	Clary Dallywater	-OvUGzekH5w9iCVi	2.005

- Si no coge bien la actualización, puede que tengamos que actualizar el módulo desde **Aplicaciones**.

- Los estudiantes que hemos añadido, existen pero no están asociados a ninguna clase.
- Si no tenemos ninguna clase creada, la podemos crear por ejemplo en el fichero **demo.xml**.

school – demo – demo.xml

```

school > demo > demo.xml
Project school ~/Dropbox/ controllers students_generator.sh models.xml models.py controllers.py ir.model.access.csv demo.xml
school > controllers
school > demo
  demo.xml
  students.xml
  > models
  > security
  > views
    templates.xml
    views.xml
    __manifest__.py
    __MOCK_DATA.csv
    __init__.py
    __manifest__.py
    MOCK_DATA.csv
  > students_generator
External Libraries
Scratches and Con

```

```

<odoor>
  <data>
    <record id="student1" model="school.student">
      <field name="name">Juan</field>
      <field name="dni">12345678N</field>
    </record>

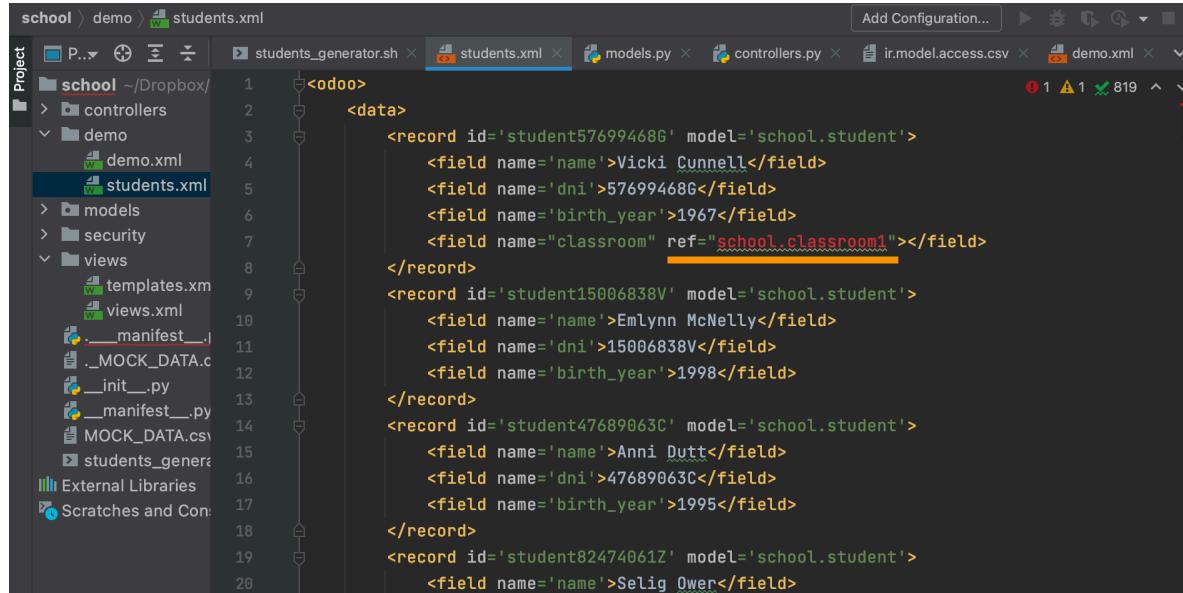
    <record id="classroom1" model="school.classroom">
      <field name="name">DAM2</field>
    </record>
  </data>
</odoor>

```

- Ahora lo que tengo que hacer es que los estudiantes pertenezcan a esa clase,

porque en el modelo **student** es donde tengo la relación **Many2one** con el modelo **classroom**.

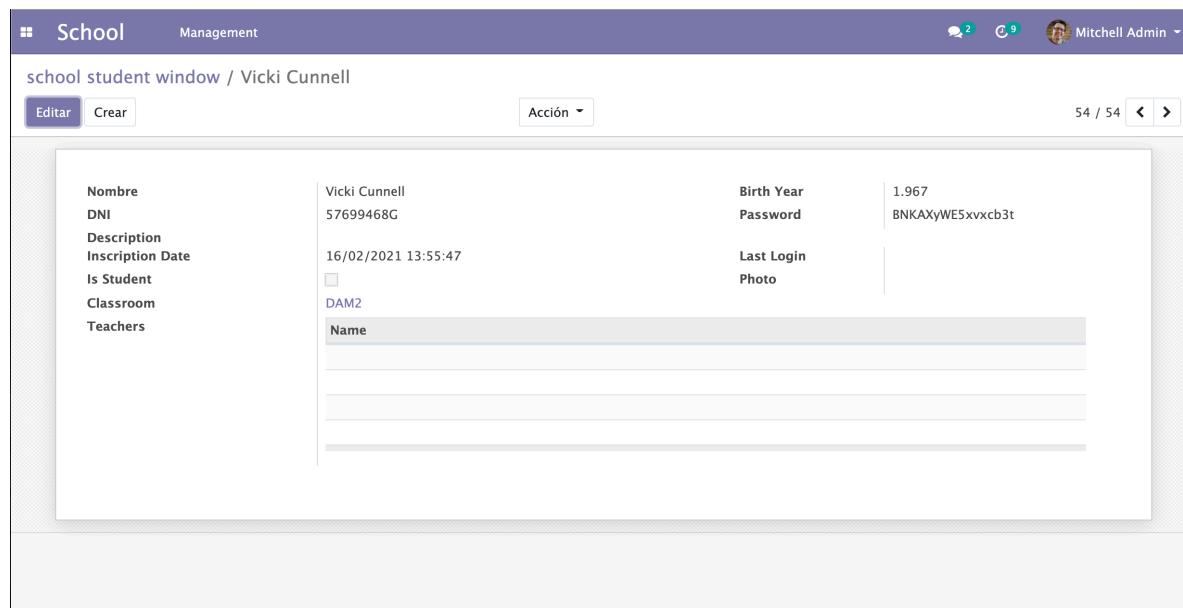
- Probamos con el primero, lo eliminamos de los estudiantes previamente añadidos, reiniciamos el servicio y vemos si lo asocia a la clase DAM2.



```

<odoo>
    <data>
        <record id='student57699468G' model='school.student'>
            <field name='name'>Vicki Cunnell</field>
            <field name='dni'>57699468G</field>
            <field name='birth_year'>1967</field>
            <field name="classroom" ref="school.classroom1"></field>
        </record>
        <record id='student15006838V' model='school.student'>
            <field name='name'>Emlynn McNelly</field>
            <field name='dni'>15006838V</field>
            <field name='birth_year'>1998</field>
        </record>
        <record id='student47689063C' model='school.student'>
            <field name='name'>Anni Dutt</field>
            <field name='dni'>47689063C</field>
            <field name='birth_year'>1995</field>
        </record>
        <record id='student82474061Z' model='school.student'>
            <field name='name'>Selig Ower</field>
        </record>
    </data>
</odoo>

```



Nombre	Vicki Cunnell	Birth Year	1.967
DNI	57699468G	Password	BNKAXyWE5vxccb3t
Description		Last Login	
Inscription Date	16/02/2021 13:55:47	Photo	
Is Student	<input type="checkbox"/>	Name	
Classroom	DAM2		
Teachers			

- Vemos que sí ha quedado añadido. Lo tenemos que hacer para todos, por lo tanto, lo añadimos al script y lo volvemos a generar.

- Si quiero ver el identificador externo de otra clase ya creada previamente, lo podría ver (en modo desarrollador) en **Ajustes > Técnico > Secuencias e identificadores > Identificadores externos**.

Ajustes      Opciones Generales      Usuarios y compañías      Traducciones      Técnico

Mitchell Admin (prue)

Identificadores externos

Nombre del modelo school Buscar...

Filtros    Agrupar por    Favoritos

1-80 / 1002

<input type="checkbox"/> Id. completo	Nombre mostrado	Nombre del modelo	ID de regi
<input type="checkbox"/> school.student20700573S	Roxana Well	school.student	
<input type="checkbox"/> school.student17953534A	Lilias Jasper	school.student	
<input type="checkbox"/> school.student86205858U	Errick Abramín	school.student	
<input type="checkbox"/> school.student61655251G	Valenka Begini	school.student	
<input type="checkbox"/> school.student52583547P	Roddy Linguard	school.student	
<input type="checkbox"/> school.student63769038A	Adelaida Dillowy	school.student	
<input type="checkbox"/> school.student25024232J	Hendrick Cherm	school.student	
<input type="checkbox"/> school.student92958331W	Sansone Lawther	school.student	
<input type="checkbox"/> school.student02394073D	Peggie Triggs	school.student	
<input type="checkbox"/> school.student95728622B	Alard Mearing	school.student	
<input type="checkbox"/> school.student39453993H	Odelinda Baigrie	school.student	
<input type="checkbox"/> school.student19866894E	Maxi Spellacey	school.student	
<input type="checkbox"/> school.student16186429X	Dani Harryman	school.student	
<input type="checkbox"/> school.student07695927T	Casey Easterbrook	school.student	
<input type="checkbox"/> school.student70383461S	Melanie Loy	school.student	
<input type="checkbox"/> school.student75839352T	Agna Allso	school.student	
<input type="checkbox"/> school.student87852575A	Linell Deamer	school.student	
<input type="checkbox"/> school.student46761663V	Arlie Jodlkowski	school.student	
<input type="checkbox"/> school.student24166642T	Jacquetta Healing	school.student	
<input type="checkbox"/> school.student49775426Q	Goldi Laxon	school.student	
<input type="checkbox"/> school.student80493333Z	Clary Dallywater	school.student	
<input type="checkbox"/> school.student31211149J	Viki McNish	school.student	
<input type="checkbox"/> school.student56255238K	Fannie Behrendsen	school.student	
<input type="checkbox"/> school.student91972224C	Damian Tuvey	school.student	
<input type="checkbox"/> school.student13180184B	Ax Piotrkowski	school.student	
<input type="checkbox"/> school.student44255919W	Adrea Vergo	school.student	

- Aquí por ejemplo podemos ver los identificadores externos de todos los elementos del modelo **school.student**. Cada cosa de Odoo: field, vista, modelo, etc tiene un id externo asociado.

- Añadimos el campo al script:

```

school > students_generator.sh
Project P.. + - + __manifest__.py > students_generator.sh < students.xml < models.py < controllers.py < ir
  school ~Dropbox
    > controllers
    > demo
      > demo.xml
      > students.xml
    > models
      > __init__.py
      > models.py
    > security
    > views
      > templates.<
      > views.xml
      > __manifest__.py
      > __init__.py
      > __manifest__.py
      > MOCK_DATA.csv
      > students_generator.sh
    > External Libraries
    > Scratches and Co

```

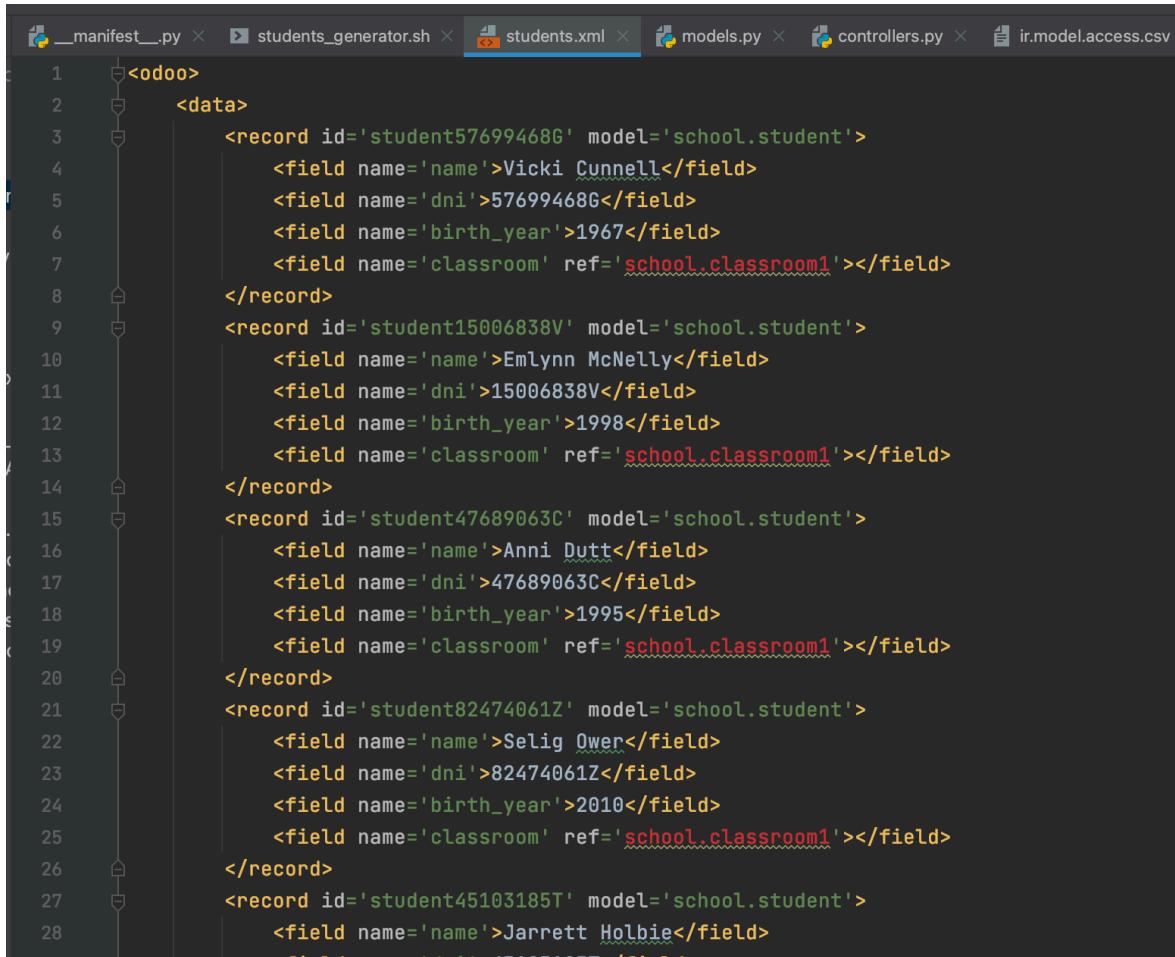
```

#!/bin/bash
while read line
do
  nombre=$(echo $line | cut -d',' -f1)
  dni=$(echo $line | cut -d',' -f2)
  year=$(echo $line | cut -d',' -f3)
  echo "<record id='student$dni' model='school.student'>"
  echo "  <field name='name'>$nombre</field>"
  echo "  <field name='dni'>$dni</field>"
  echo "  <field name='birth_year'>$year</field>"
  echo "  <field name='classroom' ref='school.classroom1'></field>"
  echo "</record>"
done

```

```
alumno@alumnodam:/opt/odoo/odoo/modules/school$ ./students_generator.sh < MOCK_DATA.csv > demo/students.xml
alumno@alumnodam:/opt/odoo/odoo/modules/school$
```

- Volvemos a hacer lo de antes, añadir la parte que falta al xml y eliminamos el primer récord.



```
<odoo>
    <data>
        <record id='student57699468G' model='school.student'>
            <field name='name'>Vicki Cunnell</field>
            <field name='dni'>57699468G</field>
            <field name='birth_year'>1967</field>
            <field name='classroom' ref='school.classroom1'></field>
        </record>
        <record id='student15006838V' model='school.student'>
            <field name='name'>Emlynn McNelly</field>
            <field name='dni'>15006838V</field>
            <field name='birth_year'>1998</field>
            <field name='classroom' ref='school.classroom1'></field>
        </record>
        <record id='student47689063C' model='school.student'>
            <field name='name'>Anni Dutt</field>
            <field name='dni'>47689063C</field>
            <field name='birth_year'>1995</field>
            <field name='classroom' ref='school.classroom1'></field>
        </record>
        <record id='student82474061Z' model='school.student'>
            <field name='name'>Selig Ower</field>
            <field name='dni'>82474061Z</field>
            <field name='birth_year'>2010</field>
            <field name='classroom' ref='school.classroom1'></field>
        </record>
        <record id='student45103185T' model='school.student'>
            <field name='name'>Jarrett Holbie</field>
            <field name='dni'>F103185T</field>
        </record>
    </data>
</odoo>
```

- También le vamos a reverenciar la clase al estudiante Juan.

```

<odoo>
    <data>
        <record id="student1" model="school.student">
            <field name="name">Juan</field>
            <field name="dni">12345678N</field>
            <field name='classroom' ref='school.classroom1'></field>
        </record>
        <record id="classroom1" model="school.classroom">
            <field name="name">DAM2</field>
        </record>
    </data>
</odoo>

```

- Como los estudiantes ya están creados, habrá que borrarlos para que se añadan correctamente.
- De este modo tenemos una clase con más de 1000 estudiantes, lo cual no tiene sentido, pero como es un ejemplo, bastaría con organizar dichos datos de demo para estructurarlos en distintas clases.
- Para que no dé error al calcular el coordinador hacemos lo siguiente:

```

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')

    # Vamos a considerar para el ejemplo que una clase puede tener un coordinador (profesor) y que un mismo profesor
    # pudiera ser coordinador de varias clases
    coordinator = fields.Many2one('school.teacher', compute='_get_coordinator')

    all_teachers = fields.Many2many('school.teacher', compute='_get_teacher')

    def _get_coordinator(self):
        for classroom in self:
            if len(classroom.teachers) > 0:
                classroom.coordinator = classroom.teachers[0].id #Para el ejemplo, vamos a establecer como coordinador al primero
            else:
                classroom.coordinator = None

    def _get_teacher(self):
        for classroom in self:
            # Para trabajar, acepta o lista de id o recordset, las dos cosas le valen. En este caso le metemos recordset.
            classroom.all_teachers = classroom.teachers + classroom.teachers_last_year

```

The screenshot shows a list of students in the DAM2 classroom. The columns are labeled 'Nombre', 'Password', and 'Birth Year'. The data includes various student names like Vicki Cunnell, Emlyn McNelly, and Nickie Saphshed, along with their birth years ranging from 1967 to 1998.

Name	Password	Birth Year
Vicki Cunnell	KYDrc0CKcxUqrYH1	1.967
Emlyn McNelly	eFnQnQTInJSZSLA7	1.998
Anni Dutt	TCOYZGeQn7dqtlI	1.995
Selig Ower	zcnTLB86vD0CfXKB7	2.010
Jarrett Holbie	XjBdnLwKnzL-0nM6	2.004
Hazel Loyley	fUxNm5CGNW5kbdw	2.011
Blake Sictornes	7q5djtXhsCU3hYw	2.002
Fannie Tidbold	W_DWB9jOn53nPfts	2.003
Dud Lewcock	9hxQ9LsBwRDQ7Xi	2.012
Lyon Kempton	_LetmeWSOUthundb	1.985
Nickie Saphshed	i6nSHoIEDJE0St6x	1.997
Domingo Shakespeare	G6uM3Q_NcSz_azLf	1.997
Cayla Iacovine	H19G3pvnNic_LYz	1.976
Coralyn Elwin	NTHSnO9d0fOUzG8	2.006
Farris Isaacs	QUPBHjm_IVi_xTwa	1.998
Lillas Jasper	iSWgBTjheyeRCfjk	1.992
Errick Abramin	9eExkdhbx-2ObQy	2.008
Valenka Beglini	wwwIRAn8ReWh1hk	1.994
Roddy Linguard	s9PLKU6baEurDMot	2.011
Adelaida Dilloway	djhNyvLa9mSmTN	1.996
Hendrick Cherm	n5p1awfJ0FOrkXem	1.993
Sansou Lawther	j-0Prf7WBMS4zG	1.992
Peggie Triggs	188cDQymVHiShwm0	2.006
Alard Mearing	J15cp11KCMYcRJDy	2.006
Odelinda Baigrie	f5dpMmgDMCalWA8d	1.984
Maxi Spellacey	F05pgkVqHvKGUne4	1.988

- Vamos a hacer ahora que la fecha de **last\_login** sea un valor calculado que sea el momento en el que se actualiza el módulo y sólo en ese momento. En este caso vamos a ver cómo hacerlo desde el XML, para ello hay una propiedad **eval** que admite sólo expresiones **Python**, **no expresiones de Odoo**.

```

<manifest__.py>
<students_generator.sh>
<students.xml> (selected)
<models.py>
<controllers.py>
<ir.model.access.csv>
<demo.xml>
<views.xml>

```

```

<odoo>
    <data>
        <record id='student57699468G' model='school.student'>
            <field name='name'>Vicki Cunnell</field>
            <field name='dni'>57699468G</field>
            <field name='birth_year'>1967</field>
            <field name='classroom' ref='school.classroom1'></field>
            <field name='last_login' eval="(datetime.now() + timedelta(-1)).strftime('%Y-%m-%d')"></field>
        </record>
        <record id='student15006838V' model='school.student'>
            <field name='name'>Emlyn McNelly</field>
            <field name='dni'>15006838V</field>
    </data>

```

- Sólo lo haremos en el ejemplo para el estudiante de nombre **Vicki Cunnell**. Para comprobar que funciona, hay que borrar dicho estudiante, parar el servicio y arrancarlo de nuevo para que lo vuelva a añadir.

The screenshot shows the student details for Vicki Cunnell. The 'Last Login' field is highlighted with a yellow bar, showing the value '16/02/2021 01:00:00'.

Nombre	Vicki Cunnell	Birth Year	1.967
DNI	57699468G	Password	QHdujUQ9NAdYUFwy
Description			
Inscription Date	17/02/2021 12:08:08	Last Login	16/02/2021 01:00:00
Is Student		Photo	
Classroom	DAM2		
Teachers	Name		

- Esto es un ejemplo de cómo podemos meter una expresión de Python que se

evalúa cuando se cargan los datos.

- Ahora vamos a crear unos profesores y los vamos a añadir al aula. En el fichero **demo.xml**, además de la clase, vamos a crear un par de profesores. También tenemos que añadir al **Many2many** de la clase esos dos profesores.
- Un Many2many está esperando una lista de ids.

```
manifest__.py x students_generator.sh x students.xml x models.py x controllers.py x ir.model.access.csv x demo.xml x views.x v
<odoo>
    <data>
        <record id="teacher1" model="school.teacher">
            <field name="name">José Alberto</field>
        </record>

        <record id="teacher2" model="school.teacher">
            <field name="name">José Luis</field>
        </record>

        <record id="classroom1" model="school.classroom">
            <field name="name">DAM2</field>
            <field name="teachers" eval="[(6,0,[ref('school.teacher1'),ref('school.teacher2')])]"/>
        </record>
    </data>
</odoo>
```

- Los valores iniciales significan lo siguiente:

- (0,\_,{'field': value}): Crea un nuevo registro y lo vincula
- (1,id,{field': value}): Actualiza los valores en un registro ya vinculado.
- (2,id,\_): Desvincula y elimina el registro.
- (3,id,\_): Desvincula pero no elimina el registro de la relación.
- (4,id,\_): Vincula un registro ya existente.
- (5,\_,\_): Desvincula pero no elimina todos los registros vinculados.
- (6,\_,[ids]): Reemplaza la lista de registros vinculados.

- El más usado es el 6 si queremos reemplazar y el 4 si no queremos reemplazar.

- En resumen, un Many2many necesita que le pasemos una tupla, en la que necesita saber qué hacer con los registros existentes y la lista de ids con los que referencia.

- Cada vez que añadimos datos nuevos, tenemos que andar eliminando los datos ya existentes (como estamos en desarrollo no pasa nada). Si queremos que eso no pase, podemos hacer datos de demo que primero se puedan eliminar. Por ejemplo, vamos a hacer que se elimine la clase 1 cada vez.

```

manifest_.py x students_generator.sh x students.xml x models.py x controllers.py x ir.model.access.csv x demo.xml x views.x
1 <odoo>
2   <data>
3
4     <record id="teacher1" model="school.teacher">
5       <field name="name">José Alberto</field>
6     </record>
7
8     <record id="teacher2" model="school.teacher">
9       <field name="name">José Luis</field>
10    </record>
11    <delete model="school.classroom" id="school.classroom1"></delete>
12    <record id="classroom1" model="school.classroom">
13      <field name="name">DAM2</field>
14      <field name="teachers" eval="[(6,0,[ref('school.teacher1'),ref('school.teacher2'))])]"/></field>
15    </record>
16
17   </data>
18 </odoo>

```

- Todo esto sería una forma de añadir datos de demo en XML, pero también se podrían importar desde el CSV si está bien formado, pero es más recomendable tenerlo en el XML.

- Nos queda cómo añadir la foto de los estudiantes. Lo primero que haremos será buscar una imagen para el ejemplo:



- La vamos a guardar por ahora en el directorio **school** con el nombre **student.png**.

- Odoo almacena las fotos en base64 porque este formato utiliza ascii y es perfectamente interpretable por el navegador, por lo tanto hacemos:

```

1  #!/bin/bash
2
3  echo "<odoo><data>"
4  while read line
5  do
6      nombre=$(echo $line | cut -d',' -f1)
7      dni=$(echo $line | cut -d',' -f2)
8      year=$(echo $line | cut -d',' -f3)
9      echo "<record id='student$dni' model='school.student'>"
10     echo "    <field name='name'>$nombre</field>"
11     echo "    <field name='dni'>$dni</field>"
12     echo "    <field name='birth_year'>$year</field>"
13     echo "    <field name='classroom' ref='school.classroom1'></field>"
14     echo "    <field name='photo'>$(base64 student.png)</field>"
15     echo "    </record>"
16 done
17 echo "</data></odoo>"
```

- Del fichero MOCK\_DATA.csv vamos a eliminar la primera fila para no tener que eliminar luego el récord en el xml.

Line	Content
1	Vicki Cunnell,57699468G,1967
2	Emlynn McNelly,15006838V,1998
3	Anni Dutt,47689063C,1995
4	Selig Ower,82474061Z,2010
5	Jarrett Holbie,45103185T,2004
6	Hazel Loyley,05492519S,2011
7	Blake Sictornes,27265713D,2002
8	Fannie Tidbold,77516174X,2003
9	Dud Lewcock,86314185Z,2012
10	Lyon Kempton,84444777R,1985
11	Nickie Sapshed,97051957B,1997
12	Domingo Shakespeare,19491926W,1997
13	Cayla Iacovino,58391108V,1976
14	Coralyn Elwin,82917638E,2006
15	Farris Isaacs,35109023K,1998
16	Lilias Jasper,17953534A,1992
17	Errick Abramin,86205858U,2008
18	Valenka Beggini,61655251G,1994
19	Roddy Linguard,52583547P,2011
20	Adelaida Dillaway,63769038A,1996
21	Hendrick Cherm,25024232J,1993

- De este modo quedará el fichero xml generado sin tener que tocarlo después.
- Antes de nada, borraremos todos los datos previos en Odoo.
- También quitamos el <delete> que habíamos puesto de ejemplo en **demo.xml**.

```

<odoo>
    <data>
        <record id="teacher1" model="school.teacher">
            <field name="name">José Alberto</field>
        </record>

        <record id="teacher2" model="school.teacher">
            <field name="name">José Luis</field>
        </record>

        <record id="classroom1" model="school.classroom">
            <field name="name">DAM2</field>
            <field name="teachers" eval="[(6,0,[ref('school.teacher1'),ref('school.teacher2')])]"/></field>
        </record>
    </data>
</odoo>

```

- Generamos el xml de nuevo.

```

alumno@alumnodam:/opt/odoo/odoo/modules/school$ ./students_generator.sh < MOCK_DATA.csv > demo/students.xml

```

- Despu s de formatear el c digo (**Code > Reformat Code** ...), obtenemos esto:

```

<data>
    <record id='student57699468G' model='school.student'>
        <field name='name'>Vicki Cunnell</field>
        <field name='dni'>57699468G</field>
        <field name='birth_year'>1967</field>
        <field name='classroom' ref='school.classroom1'></field>
        <field name='photo'>iVBORw0KGgoAAAANSUhEUgAAAtwAAAICAYAAABnDFggAAAMA...</field>
    </record>
</data>

```

- Reiniciamos el servicio.

- Se cargan todos y cada uno tiene su foto:

The screenshot shows a software application window titled "School Management". The title bar also includes "school student window / Selig Ower". Below the title bar are buttons for "Editar" and "Crear". On the right side of the header is a "Acción" dropdown menu with a downward arrow. The main content area displays student information in a grid format. The columns are labeled "Nombre", "DNI", "Description", "Inscription Date", "Is Student", "Classroom", "Teachers", "Birth Year", "Password", "Last Login", and "Photo". The "Nombre" column contains "Selig Ower". The "DNI" column contains "82474061Z". The "Description" column contains "17/02/2021 13:16:28". The "Is Student" column has a checked checkbox. The "Classroom" column contains "DAM2". The "Teachers" column lists "Name" followed by "José Alberto" and "José Luis". The "Birth Year" column contains "2.010". The "Password" column contains "qTAGDjbTpqzAwYUd". The "Last Login" column is empty. The "Photo" column contains a small thumbnail image of a person's face.

- Veremos cómo mostrar las fotos cuando veamos el tema de las vistas.
- Podríamos haber optado por hacer en lugar de un Shell script un programa en otro lenguaje de programación, es indiferente.