RESTRICCIONES

- Vamos a añadir un DNI para explicar esto. Va a ser de tipo Char. Los DNI tienen que cumplir un patrón.
- Para ello vamos a crear una función con el decorador **@api.constrains('dni')** y debemos de tener en cuenta que las funciones con este decorador, reciben una lista de estudiantes, no un estudiante únicamente.
- Para hacer este tipo de chequeos, vamos a utilizar las expresiones regulares en python. Podemos utilizar el terminal de python para probar.

```
>>> import re
>>> regex = re.compile('[0-9]{8}[a-z]\Z',re.I)
>>> regex.match('12345678N')
<re.Match object; span=(0, 9), match='12345678N'>
>>> regex.match('12345678Nsdf')
>>> regex.match('123456asdf78N')
>>> regex.match('d12345678N')
>>>
```

- re.I indica que ignore case.
- Con match comprobamos las cadenas. Sólo el primer match es válido según el patrón. Esta es la expresión regular que vamos a probar.

```
from odoo import models, fields, api
from odoo.exceptions import ValidationError
import secrets
import logging
import re

_logger = logging.getLogger(__name__)
```

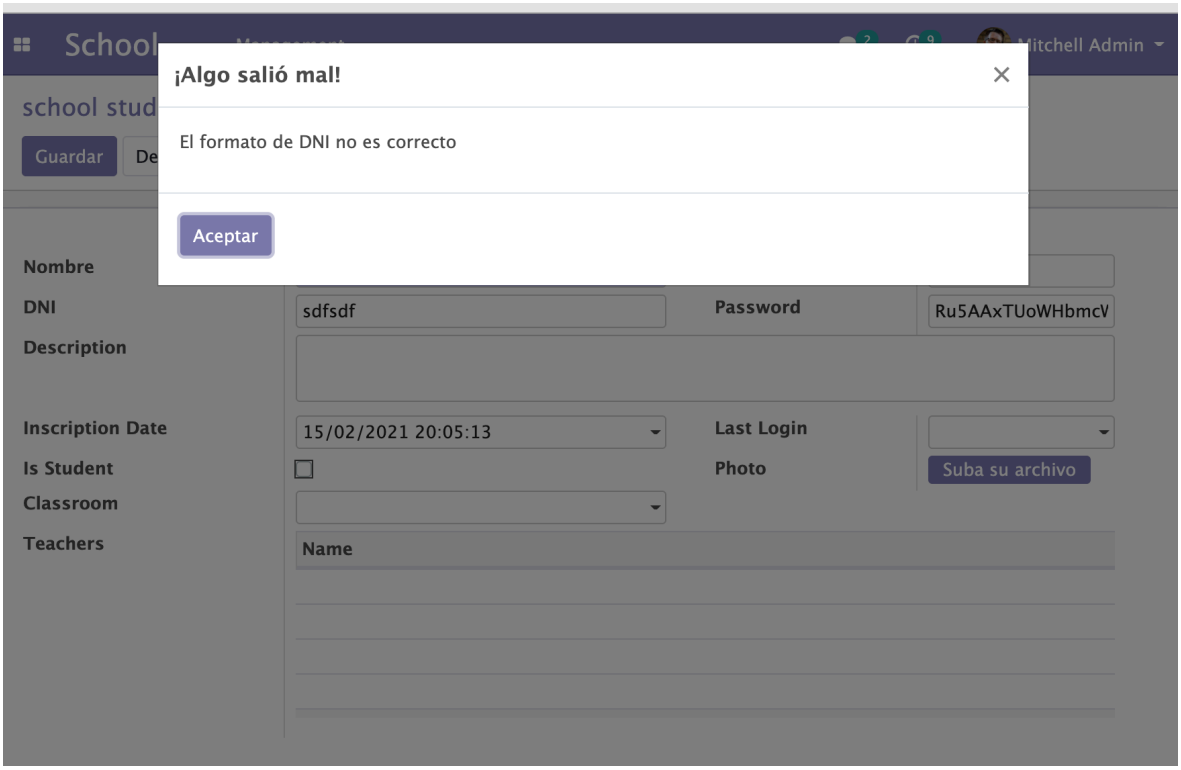
```python
class student(models.Model):
    _name = 'school.student'
    _description = 'Los alumnos'

    name = fields.Char(string="Nombre", readonly=False, required=True, help='Este es el nombre')
    birth_year = fields.Integer()
    dni = fields.Char(string="DNI")
    password = fields.Char(default=lambda p: secrets.token_urlsafe(12))

    description = fields.Text()
    inscription_date = fields.Datetime(default=lambda d: fields.Datetime.now())
    last_login = fields.Datetime()
    is_student = fields.Boolean()
    photo = fields.Image(max_width=300, max_height=300) # 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)

    @api.constrains('dni')
    def _check_dni(self):
        regex = re.compile('[0-9]{8}[a-z]\Z', re.I)
        for student in self:
            #Ahora vamos a validar si se cumple la condición
            if regex.match(student.dni):
                _logger.info('El dni fue correcto')
            else:
                #No coinciden por lo que tenemos que informar e impedir que se guarde
                raise ValidationError('El formato de DNI no es correcto')
```

- Si el DNI no es válido, no debe permitirme guardar.



- Este chequeo también impediría crear estudiantes si no tienen DNI válidos, aunque la creación la hiciésemos desde una función. Va a chequear el campo antes de guardarlo **siempre.**
- Por otro lado, el DNI debe ser único. Podríamos hacerlo algorítmicamente desde

Python haciendo una búsqueda para ver si ya existe, pero también se puede establecer esa unicidad en BDD.
- El modelo, tiene una variable privada **_sql_constraints** que por defecto es un array vacío. Permite en cada posición recibir una tupla. El primer valor, será el nombre de la constraint, el segundo valor será la restricción en postgresql y por último el mensaje).
- **TODOS LOS MENSAJES LOS DEBERÍAMOS DE PONER EN INGLÉS Y LUEGO TRADUCIR.**

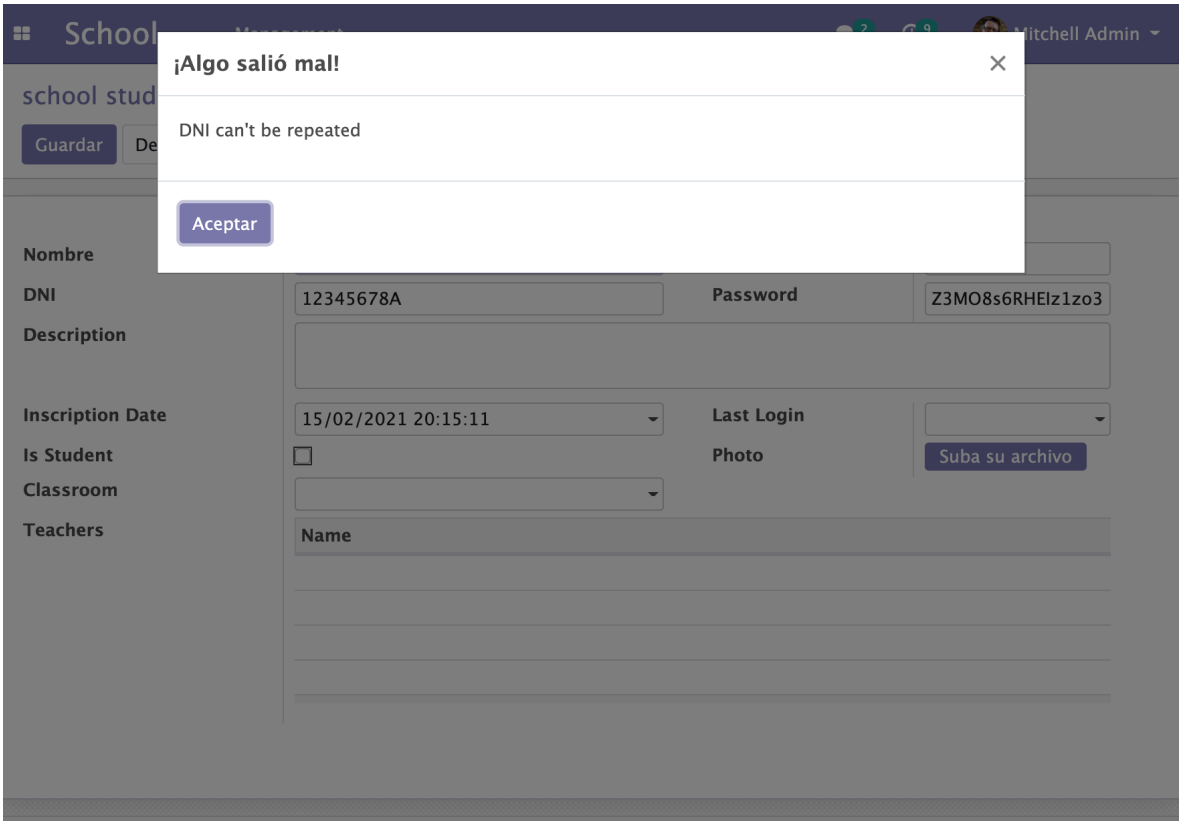
```python
class student(models.Model):
    _name = 'school.student'
    _description = 'Los alumnos'

    name = fields.Char(string="Nombre", readonly=False, required=True, help='Este es el nombre')
    birth_year = fields.Integer()
    dni = fields.Char(string="DNI")
    password = fields.Char(default=lambda p: secrets.token_urlsafe(12))

    description = fields.Text()
    inscription_date = fields.Datetime(default=lambda d: fields.Datetime.now())
    last_login = fields.Datetime()
    is_student = fields.Boolean()
    photo = fields.Image(max_width=300, max_height=300) # 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)

    @api.constrains('dni')
    def _check_dni(self):
        regex = re.compile('[0-9]{8}[a-z]\Z', re.I)
        for student in self:
            #Ahora vamos a validar si se cumple la condición
            if regex.match(student.dni):
                _logger.info('DNI correct')
            else:
                #No coinciden por lo que tenemos que informar e impedir que se guarde
                raise ValidationError('DNI format incorrect')

    _sql_constraints = [('dni_uniq','unique(dni)','DNI can\'t be repeated')]
```


School
Mitchell Admin
¡Algo salió mal!
DNI can't be repeated
Aceptar
school stud
Guardar
Nombre
DNI
12345678A
Password
Z3MO8s6RHEIz1zo3
Description
Inscription Date
15/02/2021 20:15:11
Last Login
Is Student
Photo
Suba su archivo
Classroom
Teachers
Name