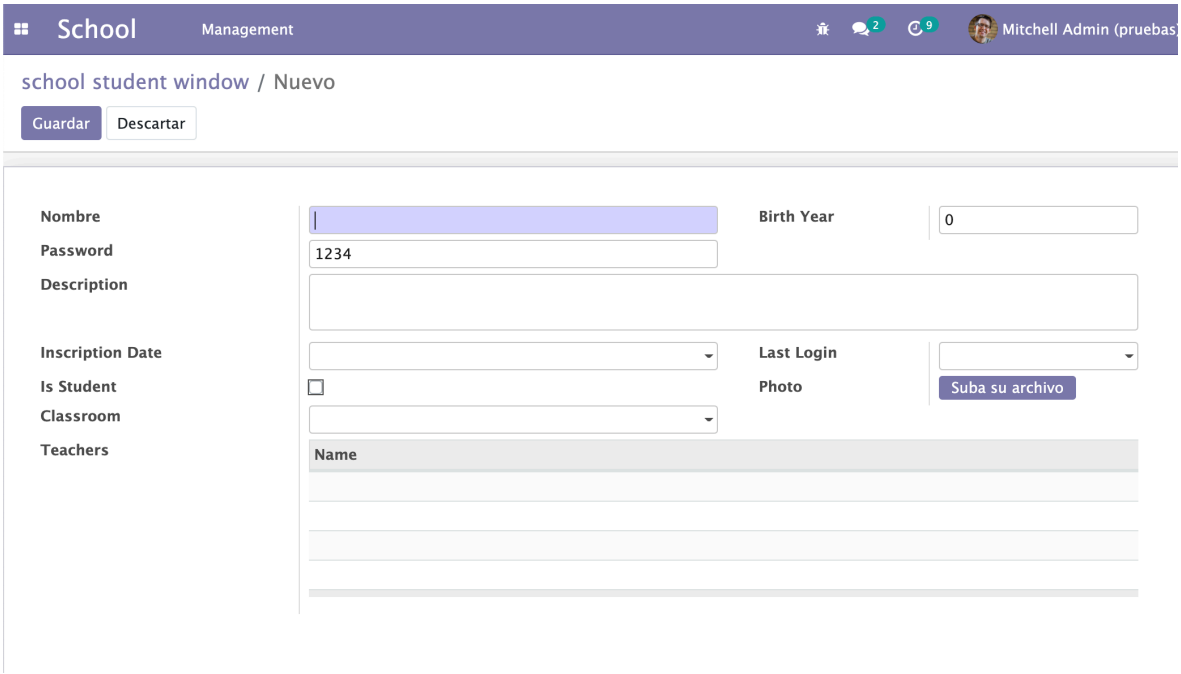
VALORES POR DEFECTO

- Todos los tipos de campos pueden tener valores por defecto para que el usuario no tenga que introducirlos.
- Vamos a cambiar el campo contraseña calculado a un campo con un valor por defecto.

```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()
    password = fields.Char(default='1234')
    description = fields.Text()
    inscription_date = fields.Date()
    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)
```

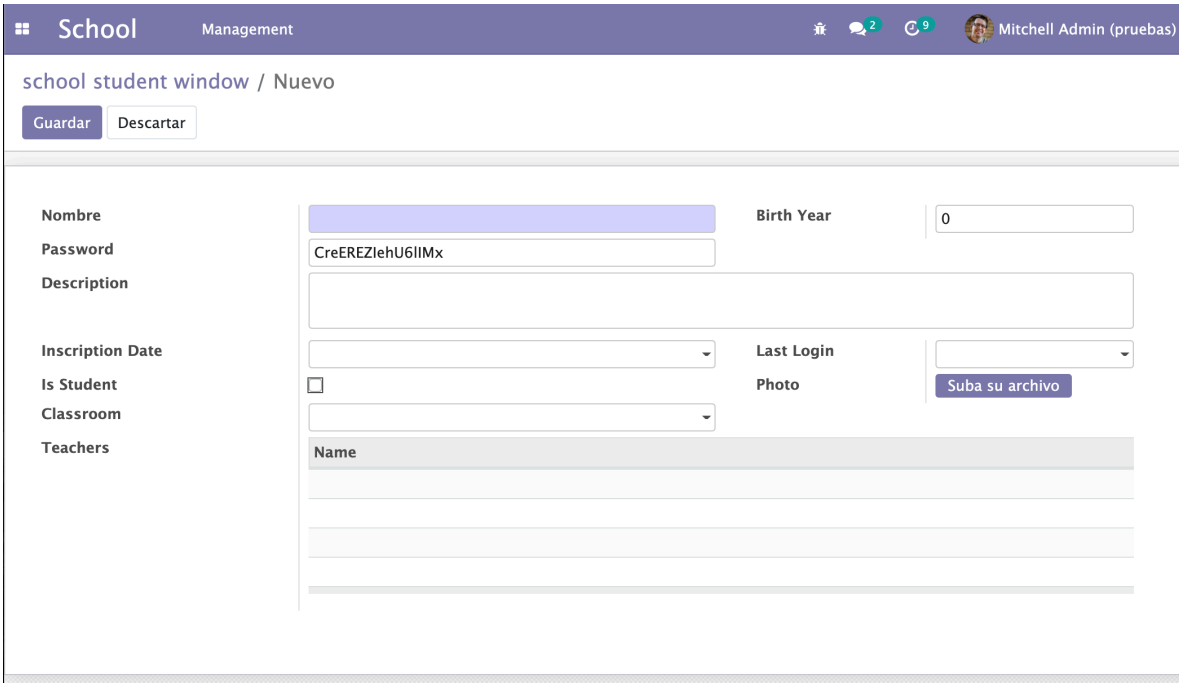


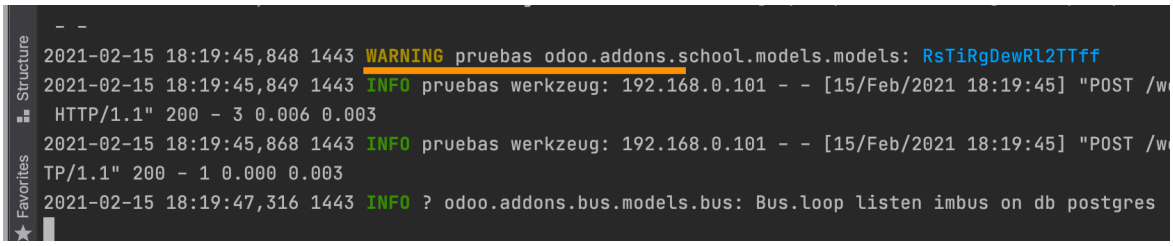
- Como podemos ver, aparece un valor por defecto, pero ese valor se podría cambiar.

- Si lo que quiero es que el valor sea por defecto, pero sea el resultado de lo que devuelve una función, haríamos:

```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()

    def _get_password(self):
        password = secrets.token_urlsafe(12)  # Generará un token de 12 bytes
        _logger.warning('\033[94m' + str(student) + '\033[0m')
        return password

    password = fields.Char(default=_get_password)

    description = fields.Text()
    inscription_date = fields.Date()
    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)
```

- Como Python es un lenguaje interpretado, la función en este caso no es parte del modelo de Odoo, sino que se ejecuta en el momento que se llama y por tanto debe estar previamente declarada. Por este motivo la hemos cambiado de posición.





- En este caso, no añadimos entre comillas la función porque queremos que sea el resultado de la ejecución de dicha función.

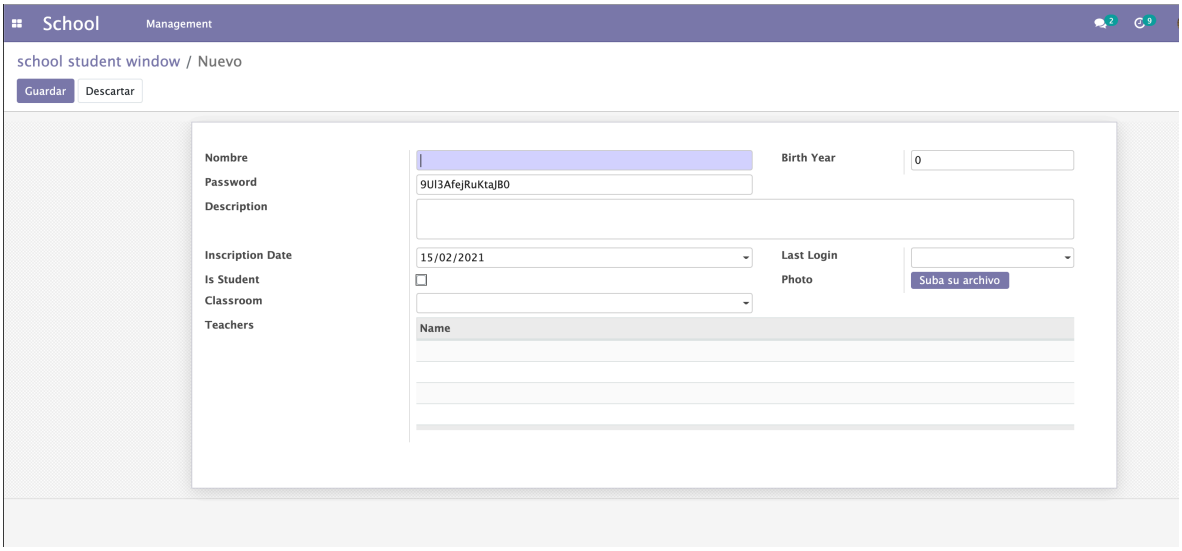
- En este caso, **self** es la instancia del modelo, no un resultaste de estudiantes como en el caso de los campos calculados.

- En este caso, esto mismo lo podríamos hacer con una función **lambda**. Estas funciones, son funciones que sólo aceptan una línea de código aunque dicha línea de código llame a otra función. Por ejemplo:

```
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()

    password = fields.Char(default=lambda p: secrets.token_urlsafe(12))

    description = fields.Text()
    inscription_date = fields.Date()
    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)
```

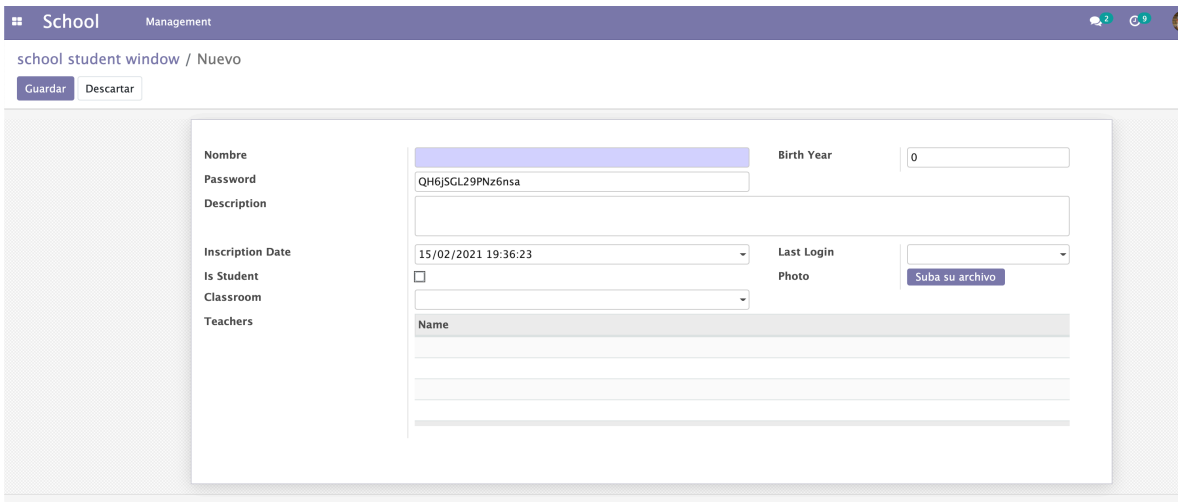
- También es muy útil establecer como valor por defecto en los campos Date, la fecha de hoy.

```
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()

    password = fields.Char(default=lambda p: secrets.token_urlsafe(12))

    description = fields.Text()
    inscription_date = fields.Date(default=lambda d: fields.Date.today())
    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)
```



- Si en lugar de un tipo **Date** tenemos un tipo **Datetime,** sería:

```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()

    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)
```



- Tiene que ser una función lambda, porque si directamente ponemos Fields.Datetime(default=fields.Datetime.now()), se carga al iniciar el servicio, pero no se calcularía cada vez que se crea un alumno, por ejemplo. Deben ser punteros

a funciones como hemos hecho antes con _get_password o una función lambda, si no el comportamiento lo genera únicamente al cargar el servicio.