

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.

```
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)
```

School Management 2 9 Mitchell Admin (pruebas)

school student window / Nuevo

Guardar Descartar

Nombre	<input type="text"/>	Birth Year	<input type="text" value="0"/>					
Password	<input type="password" value="1234"/>							
Description	<input type="text"/>							
Inscription Date	<input type="text"/>	Last Login	<input type="text"/>					
Is Student	<input type="checkbox"/>	Photo	<input type="button" value="Suba su archivo"/>					
Classroom	<input type="text"/>							
Teachers	<table><thead><tr><th>Name</th></tr></thead><tbody><tr><td><input type="text"/></td></tr><tr><td><input type="text"/></td></tr><tr><td><input type="text"/></td></tr><tr><td><input type="text"/></td></tr></tbody></table>			Name	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Name								
<input type="text"/>								
<input type="text"/>								
<input type="text"/>								
<input type="text"/>								

- 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:

```

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.

The screenshot shows the Odoo School Management interface. At the top, there's a header bar with 'School Management' and user information. Below it, the breadcrumb 'school student window / Nuevo' is visible. The main area contains a form with several fields: 'Nombre' (Name), 'Password' (highlighted with a blue box), 'Description', 'Inscription Date', 'Is Student' (checkbox), 'Classroom' (dropdown), 'Birth Year' (text input with '0'), 'Last Login' (dropdown), 'Photo' (button 'Suba su archivo'), and 'Teachers' (table with 'Name' column). At the bottom, there are 'Guardar' (Save) and 'Descartar' (Discard) buttons.

```

--
2021-02-15 18:19:45,848 1443 WARNING pruebas odoo.addons.school.models.models: RsTiRgDewRL2TTff
2021-02-15 18:19:45,849 1443 INFO pruebas werkzeug: 192.168.0.101 - - [15/Feb/2021 18:19:45] "POST /w
HTTP/1.1" 200 - 3 0.006 0.003
2021-02-15 18:19:45,868 1443 INFO pruebas werkzeug: 192.168.0.101 - - [15/Feb/2021 18:19:45] "POST /w
TP/1.1" 200 - 1 0.000 0.003
2021-02-15 18:19:47,316 1443 INFO ? odoo.addons.bus.models.bus: Bus.loop listen imbus on db postgres

```

- 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.CharField(string="Nombre", readonly=False, required=True, help='Este es el nombre')
    birth_year = fields.IntegerField()

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

    description = fields.TextField()
    inscription_date = fields.DateField()
    last_login = fields.DateTimeField()
    is_student = fields.BooleanField()
    photo = fields.ImageField(max_width=300, max_height=300) # Field binario pero específico para imágenes.
    classroom = fields.ManyToManyField('school.classroom', ondelete='set null', help='Clase a la que pertenece')
    teachers = fields.ManyToManyField('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.CharField(string="Nombre", readonly=False, required=True, help='Este es el nombre')
    birth_year = fields.IntegerField()

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

    description = fields.TextField()
    inscription_date = fields.DateField(default=lambda d: fields.DateField().today())
    last_login = fields.DateTimeField()
    is_student = fields.BooleanField()
    photo = fields.ImageField(max_width=300, max_height=300) # Field binario pero específico para imágenes.
    classroom = fields.ManyToManyField('school.classroom', ondelete='set null', help='Clase a la que pertenece')
    teachers = fields.ManyToManyField('school.teacher', related='classroom.teachers', readonly=True)
```

School Management

school student window / Nuevo

Guardar Descartar

Nombre:

Password:

Description:

Inscription Date:

Is Student: ☐

Classroom:

Teachers:

Birth Year:

Last Login:

Photo:

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

```
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)
```

School Management

school student window / Nuevo

Guardar Descartar

Nombre:

Password:

Description:

Inscription Date:

Is Student: ☐

Classroom:

Teachers:

Birth Year:

Last Login:

Photo:

- 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.