Ola Mundo

@api.constraints (Restriccións)

@api.constraints

Definimos o campo peso, con un valor por defecto (por exemplo 2.7):

```
peso = fields.Float(digits=(6, 2), string="Peso en Kg.s", default=2.7)
```

O decorador @api.constraints, utilizase para facer validacións a hora de actualizar a BD por exemplo se quixesemos controlar que o peso estivese nun determinado intervalo (entre 1 e 4):

```
@api.constrains('peso')  # Ao usar ValidationError temos que importar a libreria Validati
    def _constrain_peso(self):   # from odoo.exceptions import ValidationError
        for rexistro in self:
            if rexistro.peso < 1 or rexistro.peso > 4:
                raise ValidationError('Os peso de %s ten que ser entre 1 e 4 ' % rexistro
```

Cada vez que cambie o valor do campo peso se executa o anterior código.

Como vemos, xa que utiliza o método ValidationError necesitamos importar a libreria ValidationError de odoo.exceptions

Restriccións a nivel SQL

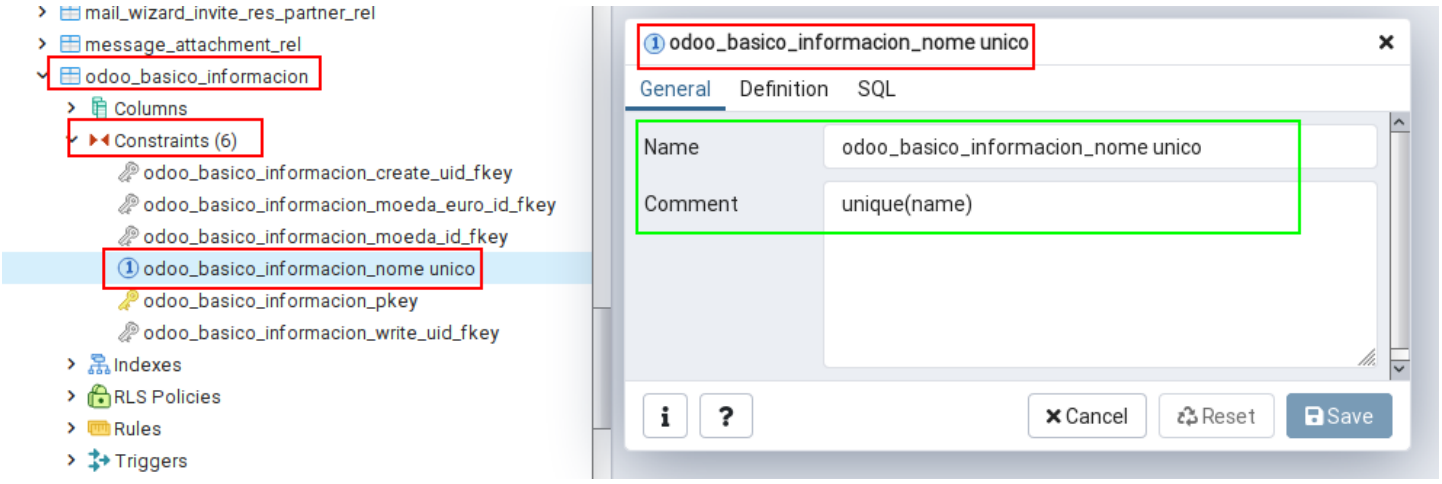
Se queremos facer restriccións a nivel de BD. Evitar inserción con valores duplicados nun campo:

Por exemplo, no campo *name* e que nos informe por pantalla coa advertencia "Non se pode repetir o nome"

Podemos utilizar a definición **_sql_constraints**

```
_sql_constraints = [('nomeUnico', 'unique(name)', 'Non se pode repetir o nome')]
```

Que creará unha restricción na táboa na BD. Como podemos ver:



Mediante a definición de **_order** podemos determinar en que orden aparecen os datos na vista tree.

Por exemplo ordeado de xeito descendente polo campo "descripcion"

```
_order = "descripcion desc"
```