Modelos

> Los modelos pueden ser visualizados en el interfaz si el modo desarrollador está activado. Se encuentran en el menú Ajustes > Técnico > Estructura de la base de datos > Modelos.

Declaración de un nuevo modelo

Existen tres tipos de modelos en Odoo: los básicos, los transitorios y los abstractos. También podemos incluir el modelo de vista SQL como cuarto tipo. Vamos a ver cómo declarar cada uno y qué particularidades tienen.

Modelo básico

```
from odoo import models

class SocialNetwork(models.Model):
_name = 'social.network'
_description = 'Social Network'
_order = 'name, launch_date desc'

...
```

Modelo transitorio

```
from odoo import models

class SocialNetworkPrintingConfigWizard(models.TransientModel):
_name = 'social.network.printing.config.wizard'
_description = 'Social Network Printing Configuration Pop-up'

...
```

Modelo abstracto

```
from odoo import models

class Animal(models.AbstractModel):
_name = 'animal'
_description = 'Animal'

...
```

Modelo de vista SQL

```
from odoo import models

class AccountInvoiceReport(models.Model):
_name = 'account.invoice.report'
_description = 'Invoices Statistics'
_auto = False



    ...
```

Herencia

Existen tres tipos de herencia en Odoo: la básica, la básica sobre un modelo abstracto, la múltiple sobre modelos abstractos y la basada en composición. Vamos a ver cómo aplicar cada una y qué particularidades tienen.

Herencia básica

```
from odoo import models

class ResPartner(models.Model):
_inherit = 'res.partner'

...
```

Herencia única de un modelo abstracto

```
class Dog(models.Model):
_name = 'dog'
_inherit = 'animal'

...

class Cat(models.Model):
_name = 'cat'
_inherit = 'animal'

...
```

Herencia múltiple de modelos abstractos

```
from odoo import models

class ResPartner(models.Model):
_name = 'res.partner'

_inherit = ['format.address.mixin', 'image.mixin']

...
```

Herencia basada en composición

```
from odoo import models

class ProductProduct(models.Model):
      _name = 'product.product'
      _inherits = {
            'product.template': 'product_tmpl_id',
      }
      ...
      product_tmpl_id = fields.Many2one(
            comodel_name='product.template',
            string='Product Template',
            auto_join=True,
            index=True,
            ondelete='cascade',
            required=True,
      )
      ...
```

Propiedades de un modelo

Como hemos visto en los anteriores apartados, cuando se trabaja con un modelo, ya sea generando uno nuevo o heredando de uno ya creado, existen una serie de propiedades especiales establecidas por el framework de Odoo que pueden ser especificadas y que alteran completamente el significado y funcionalidad del modelo. Las posibilidades son:

- `_name`: cadena de texto, nombre técnico del modelo. Debe ser en minúsculas, y separando las palabras por puntos.

  > Si en el `_name` se especifica el nombre técnico de un modelo ya existente (y no se especifica la propiedad `_inherit`), éste será redefinido por completo.

- `_description`: cadena de texto, nombre del modelo, legible para usuarios. Este es el nombre que se visualizará en ciertas partes del interfaz.

- `_table`: cadena de texto, nombre de la tabla de PostgreSQL que se generará para almacenar los registros de este modelo. Si se especifica, debe ser en minúsculas, y separando las palabras por guiones bajos. Si no se especifica, el nombre de la tabla será idéntico al indicado en `_name`, pero con guiones bajos en lugar de puntos.

- `_order`: cadena de texto, nombre técnico del campo por el que se ordenarán los registros del modelo por defecto en el interfaz. Se puede añadir `desc` si se desea ordenar descendentemente. Se pueden especificar varios campos separados por comas, pero siempre dentro de la misma cadena de texto. Si no se especifica, se ordenará siempre por el campo que se llame `name`, en caso de existir.
- `_auto`: booleano, si es `True`, el modelo generará tabla en la base de datos, si es `False`, el modelo no generará tabla, por lo que sus campos serán calculados mediante consultas SQL a partir de otros modelos y campos que sí se almacenan en base de datos. Estas consultas SQL suelen definirse en el método `init()`.
- `_register`: booleano, si es `True`, el modelo podrá ser heredado empleando el ORM del framework de Odoo (con `_inherit`). Si es `False`, el modelo sólo podrá ser heredado a través de Python.
- `_abstract`: booleano, si es `True`, el modelo es abstracto (un esqueleto para que otros modelos hereden de él) y por tanto no se podrán crear registros asociados a él. Si es `False`, podrá ser instanciado.
- `transient`: booleano, si es `True`, el modelo es transitorio, es decir, genera tabla en base de datos, pero no se desea almacenar sus datos permanentemente y éstos son elimnados cada cierto período de tiempo.
- `_custom`: booleano, propiedad que debe ser `True` en todos aquellos modelos personalizados.
- `_inherit`: cadena de texto o lista de cadenas de texto, es el/los nombre/s técnico/s del/de los modelo/s de Odoo del/de los que está heredando nuestro modelo.

  > En el caso de especificarse una lista, se considera herencia múltiple.

  > Si a mayores se especifica la propiedad `_name`, y ésta tiene el mismo valor que `_inherit`, se considera herencia básica.

  > Si a mayores se especifica la propiedad `_name`, y ésta tiene distinto valor que `_inherit`, el resultado es la generación de un modelo nuevo completamente independiente, con el nombre indicado en `_name` y copiado totalmente del modelo especificado en `_inherit`.

- `_inherits`: diccionario cuyas claves son cadenas de texto que representan nombres técnicos de modelos ya existentes, y cuyos valores también son cadenas de texto que representan campos de tipo *Many2one* que enlazan con el modelo especificado en su clave y que existirán en nuestro modelo.

  > Esta propiedad se emplea para la herencia basada en composición.

- `_sequence`: secuencia SQL que seguirá automáticamente el modelo para rellenar el campo `id` de sus registros.
- `_sql_constraints`: lista de tuplas de tres elementos (cadenas de texto). Se trata de restricciones SQL que se aplicarán a los registros del modelo. Cada tupla se compone de:
  - Nombre de la restricción: nombre que le damos a la restricción internamente en Odoo, algo completamente personalizable.
  - Definición de la restricción SQL: código SQL en el que se define la restricción a aplicar.

    - Mensaje: texto que saltará a los usuarios cuando el registro que han generado o modificado no cumpla la restricción.

- `_rec_name`: cadena de texto, nombre técnico del campo que se desea mostrar a los usuarios en el intefaz para que éstos puedan identificar el registro. Por defecto coge el campo `name`, si existe. Si se quiere algo más complejo, habría que trabajar con el método `_name_get`.

- `_parent_name`: cadena de texto, nombre técnico del campo de tipo *Many2one* que apunta a este mismo modelo, de manera recursiva, representando al registro padre. Se define en compañía de la propiedad `_parent_store`. Por defecto toma el valor `parent_id`.

- `_parent_store`: booleano, si es `True`, el modelo debe tener un campo de tipo *Char* con el parámetro `index=True` llamado `parent_path`. En ese caso almacenará en él un valor que reflejará la estructura jerárquica construida sobre el campo establecido en la propiedad `_parent_name`. Ese valor optimizará los operadores `child_of` y `parent_of` en los dominios del modelo, agilizando las búsquedas. Se define en compañía de la propiedad `_parent_name`.

- `_date_name`: cadena de texto, nombre técnico de un campo de tipo *Date* o *Datetime* que estará en el modelo y que se empleará por defecto para las vistas de tipo calendario. Por defecto toma el valor `date`.

- `_fold_name`: cadena de texto, nombre técnico de un campo que estará en el modelo y que se empleará por defecto para las agrupaciones de las vistas de tipo kanban. Por defecto toma el valor `fold`.

- `_check_company_auto`: booleano, si es `True`, cuando se cree o modifique un registro del modelo, se comprobará la consistencia de las compañías en aquellos campos relacionales que tengan el parámetro `check_company=True`, mediante la ejecución del método `_check_company`. Por defecto toma el valor `False`.

- `_depends`: diccionario cuyas claves son cadenas de texto que representan nombres técnicos de *modelos vista SQL* (aquéllos con `_auto = False`) y cuyos valores son listas de cadenas de texto, que representan los campos de esos *modelos vista SQL* de los que depende nuestro modelo. Lo más probable es que nuestro modelo sea también de tipo *vista SQL* y emplee esos campos en la consulta SQL de su método `init()`.

- `_transient_check_count`: número entero, propiedad empleada internamente para gestionar la eliminación de registros transitorios. Por defecto toma el valor `0`.

- `_transient_max_count`: número entero que representa el número máximo de registros temporales de modelos transitorios que pueden permanecer almacenados en base de datos sin ser elimnados. Por defecto toma el valor del parámetro `osv_memory_count_limit` del archivo de configuración de la instancia (o de la línea de comandos).

- `_transient_max_hours`: número decimal que representa las horas que pueden permanecer almacenados en base de datos los registros temporales de los modelos transitorios antes de ser eliminados. Por defecto toma el valor del parámetro `osv_memory_age_limit` del archivo de configuración de la instancia (o de la línea de comandos).