

## Práctica Modelos y Vistas

Ayuda para los campos Date

(Pág 203 del libro en inglés y pág 232 como operar con fechas)

Ayuda de un foro

for record in self:

```
record.x_prueba = abs((record.x_eta - record.x_farribo).days)
```

#From this code you will get the days between x\_eta and x\_farribo.

1. Instala el módulo lista de tareas, puedes estudiarlo y tomarlo como ejemplo para alguno de los ejercicios.

2. Vamos a crear un módulo para la gestión de un zoo.

```
user@kevinserver:~/docker/odoo$ docker exec -it --user root odoo_odoo_1 bash
root@bcc5ea3805f6:/# odoo scaffold gestion_zoo /opt/odoo/addons
root@bcc5ea3805f6:/#
```

```
user@kevinserver:~/docker/odoo/custom-addons/gestion_zoo$ tree
.
├── controllers
│   ├── controllers.py
│   └── __init__.py
├── demo
│   └── demo.xml
├── __init__.py
├── __manifest__.py
├── models
│   ├── __init__.py
│   └── models.py
├── security
│   └── ir.model.access.csv
└── views
    ├── templates.xml
    └── views.xml

6 directories, 10 files
user@kevinserver:~/docker/odoo/custom-addons/gestion_zoo$ _
```

Añadiremos las clases dentro del modelo models.py

## a) Modelo Cuidadores

### 1. Nombre

```
class cuidador(models.Model):
    _name = 'gestion_zoo.cuidador'
    _description = 'Cuidador del zoo'

    name = fields.Char(string='Nombre', required=True)
```

## b) Modelo Habitáculos:

### 1. Nombre.

### 2. Zona (char, en que zona del zoo se encuentra)

### 3. Superficie, real con 2 decimales.

### 4. Tipo, entre los valores “Exterior”, “Aviario”, “Granja”, “Terrario” y “Acuario”.

### 5. Fecha de limpieza. Por defecto la fecha de hoy.

### 6. Días pasados desde la última limpieza (campo calculado).

### 7. Fecha prevista de la próxima limpieza. Dependiendo del tipo de habitáculo se le suma a la fecha de la última limpieza una cantidad de días u otra (elige los valores que prefieras).

### 8. Responsable (many2one con el modelo de cuidadores)

```
class habitaculo(models.Model):
    _name = 'gestion_zoo.habitaculo'
    _description = 'Habitáculo del zoológico'

    name = fields.Char(string='Nombre', required=True)
    zone = fields.Char(string='Zona', required=True)
    area = fields.Float(string='Área (m²)', digits=(6, 2), required=True)
    habitat_type = fields.Selection([
        ('exterior', 'Exterior'),
        ('aviario', 'Aviario'),
        ('granja', 'Granja'),
        ('terrario', 'Terrario'),
        ('acuario', 'Acuario'),
    ], string='Tipo de hábitat', required=True)
    cleaning_date = fields.Date(string='Última limpieza', default=fields.Date.today)
    days_since_cleaning = fields.Integer(
        string='Días desde la última limpieza',
        compute='_compute_days_since_cleaning',
        store=False
    )
    next_cleaning_date = fields.Date(
        string='Próxima limpieza',
        compute='_compute_next_cleaning_date',
        store=False
    )
    caretaker_id = fields.Many2one(
        'gestion_zoo.cuidador',
        string='Cuidador responsable'
    )

    @api.depends('cleaning_date')
    def _compute_days_since_cleaning(self):
        """Calcula los días transcurridos desde la última limpieza."""
        for record in self:
            if record.cleaning_date:
                record.days_since_cleaning = (fields.Date.today() - record.cleaning_date).days
            else:
                record.days_since_cleaning = 0

    @api.depends('cleaning_date', 'habitat_type')
    def _compute_next_cleaning_date(self):
        """Calcula la fecha de la próxima limpieza en función del tipo de hábitat."""
        cleaning_intervals = {
            'exterior': 7,
            'aviario': 14,
            'granja': 10,
            'terrario': 21,
            'acuario': 30,
        }
        for record in self:
            interval = cleaning_intervals.get(record.habitat_type, 7)
            record.next_cleaning_date = record.cleaning_date + timedelta(days=interval) if record.cleaning_date else None
```

### c) Modelo continentes

```
class continente(models.Model):
    _name = 'gestion_zoo.continente'
    _description = 'Continente del zoo'

    name = fields.Char(string='Name', required=True)
```

### d) Modelo Especies

1. Nombre
2. Continente, many2many (hay especies que habitan en varios continentes)
3. Atractivo para el público (es un número entero, debe de controlarse que el valor está entre 1 y 10). Valor por defecto 5.
4. Imagen de la especie

```
class especie(models.Model):
    _name = 'gestion_zoo.especie'
    _description = 'Especies del zoológico'

    name = fields.Char(string='Nombre', required=True)
    continent_ids = fields.Many2many('gestion_zoo.continente', string='Continentes')
    public_attraction = fields.Integer(string='Atracción para el público', default=5, required=True)
    image = fields.Image(string='Imagen de la especie')

    _sql_constraints = [
        ('check_attraction_range', 'CHECK(public_attraction BETWEEN 1 AND 10)',
         'La atracción para el público debe estar entre 1 y 10.')
    ]
```

### e) Modelo Animales

1. Nombre
2. Especies
3. Habitación
4. Fecha llegada zoo
5. Fecha nacimiento. Comprobar que a fecha de nacimiento no puede ser anterior a la fecha de llegada al zoo.
6. Nacido en zoo. Campo calculado booleano. Es True si la fecha de nacimiento y la de llegada coinciden.
7. Cuidador responsable(campo related). Es el cuidador del habitáculo
8. Imagen de la especie (campo related).

```

class animal(models.Model):
    _name = 'gestion_zoo.animal'
    _description = 'Animales del zoológico'

    name = fields.Char(string='Nombre', required=True)
    species_id = fields.Many2one('gestion_zoo.especie', string='Especie', required=True)
    habitat_id = fields.Many2one('gestion_zoo.habitaculo', string='Hábitat', required=True)
    arrival_date = fields.Date(string='Fecha de llegada', required=True)
    birth_date = fields.Date(string='Fecha de nacimiento')
    born_in_zoo = fields.Boolean(
        string='Nacido en el zoológico',
        compute='_compute_born_in_zoo',
        store=True
    )
    caretaker_id = fields.Many2one(
        related='habitat_id.caretaker_id',
        string='Cuidador responsable',
        readonly=True
    )
    species_image = fields.Image(
        related='species_id.image',
        string='Imagen de la especie',
        readonly=True
    )

    @api.depends('arrival_date', 'birth_date')
    def _compute_born_in_zoo(self):
        """Calcula si el animal nació en el zoológico."""
        for record in self:
            record.born_in_zoo = (
                record.birth_date and
                record.arrival_date and
                record.birth_date <= record.arrival_date
            )

    @api.constrains('birth_date', 'arrival_date')
    def _check_birth_date(self):
        """Valida que la fecha de nacimiento no sea posterior a la fecha de llegada."""
        for record in self:
            if record.birth_date and record.birth_date > record.arrival_date:
                raise ValidationError('La fecha de nacimiento no puede ser posterior a la fecha de llegada.')

```

## Modificamos el fichero views.xml

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <data>
    <!-- Action Windows with Unique IDs -->
    <record model="ir.actions.act_window" id="gestion_zoo.action_window_continente">
      <field name="name">Ventana Continentes</field>
      <field name="res_model">gestion_zoo.continente</field>
      <field name="view_mode">tree,form</field>
    </record>

    <record model="ir.actions.act_window" id="gestion_zoo.action_window_especie">
      <field name="name">Ventana Especies</field>
      <field name="res_model">gestion_zoo.especie</field>
      <field name="view_mode">tree,form</field>
    </record>

    <record model="ir.actions.act_window" id="gestion_zoo.action_window_animal">
      <field name="name">Ventana Animales</field>
      <field name="res_model">gestion_zoo.animal</field>
      <field name="view_mode">tree,form</field>
    </record>

    <record model="ir.actions.act_window" id="gestion_zoo.action_window_cuidador">
      <field name="name">Ventana Cuidadores</field>
      <field name="res_model">gestion_zoo.cuidador</field>
      <field name="view_mode">tree,form</field>
    </record>

    <record model="ir.actions.act_window" id="gestion_zoo.action_window_habitaculo">
      <field name="name">Ventana Habitáculos</field>
      <field name="res_model">gestion_zoo.habitaculo</field>
      <field name="view_mode">tree,form</field>
    </record>

    <!-- Top menu item -->
    <menuitem name="Gestión Zoo" id="gestion_zoo.menu_root"/>

    <!-- Menu categories with action references -->
    <menuitem
      name="Continentes"
      id="gestion_zoo.menu_continente"
      parent="gestion_zoo.menu_root"
      action="gestion_zoo.action_window_continente"/>

    <menuitem
      name="Especies"
      id="gestion_zoo.menu_especie"
      parent="gestion_zoo.menu_root"
      action="gestion_zoo.action_window_especie"/>

    <menuitem
      name="Animales"
      id="gestion_zoo.menu_animal"
      parent="gestion_zoo.menu_root"
      action="gestion_zoo.action_window_animal"/>

    <menuitem
      name="Cuidadores"
      id="gestion_zoo.menu_cuidador"
      parent="gestion_zoo.menu_root"
      action="gestion_zoo.action_window_cuidador"/>

    <menuitem
      name="Habitáculos"
      id="gestion_zoo.menu_habitaculo"
      parent="gestion_zoo.menu_root"
      action="gestion_zoo.action_window_habitaculo"/>
  </data>
</odoo>
```

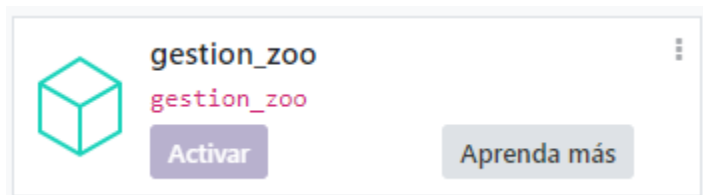
Descomentamos security en el archivo `__manifest__.py`

```
# always loaded
'data': [
    'security/ir.model.access.csv',
    'views/views.xml',
    'views/templates.xml',
],
# not loaded in demonstration mode
```

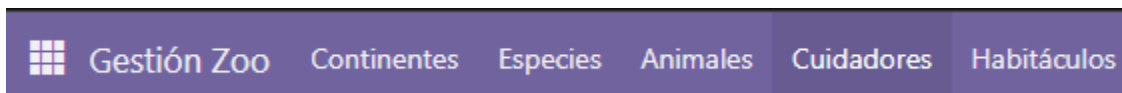
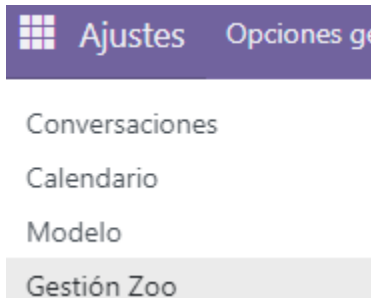
Añadimos los modelos en `ir.model.access.csv`

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_gestion_zoo_cuidador,gestion_zoo.cuidador,model_gestion_zoo_cuidador,base.group_user,1,1,1,1
access_gestion_zoo_animal,gestion_zoo.animal,model_gestion_zoo_animal,base.group_user,1,1,1,1
access_gestion_zoo_continente,gestion_zoo.continente,model_gestion_zoo_continente,base.group_user,1,1,1,1
access_gestion_zoo_especie,gestion_zoo.especie,model_gestion_zoo_especie,base.group_user,1,1,1,1
access_gestion_zoo_habitaculo,gestion_zoo.habitaculo,model_gestion_zoo_habitaculo,base.group_user,1,1,1,1
```

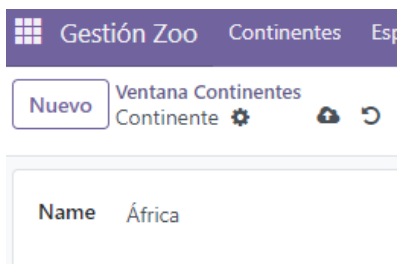
Reiniciamos el contenedor y activamos el modulo




En el menú sale la Gestión del zoo



Creación de continente



Creación de cuidador




Gestión Zoo

Continentes

Especies

Nuevo

Ventana Cuidadores

Kevin Gómez Valderas 




Nombre

Kevin Gómez Valderas

Creación de habitáculo

Nuevo

Ventana Habitáculos

Nuevo   

Nombre

Animales salvajes

Área (m²)

300,00

Última limpieza

04/12/2024

Próxima limpieza

11/12/2024

Zona

1

Tipo de hábitat

Exterior

Días desde la última limpieza

0

Cuidador responsable

Kevin Gómez Valderas

Creación de especie

Nuevo

Ventana Especies  
León

Nombre	León				
Continentes	<table><tr><td>Name</td></tr><tr><td>África</td></tr><tr><td>Agregar una línea</td></tr><tr><td></td></tr></table>	Name	África	Agregar una línea	
Name					
África					
Agregar una línea					
Atracción para el público	5				
Imagen de la especie	10.37 Kb				

Creación de animal

Nuevo

Ventana Animales  
Nuevo

Nombre	Wilson
Hábitat	Animales salvajes
Fecha de nacimiento	17/02/2010
Cuidador responsable	Kevin Gómez Valderas
Especie	León
Fecha de llegada	11/01/2012
Nacido en el zoológico	<input checked="" type="checkbox"/>
Imagen de la especie	