



**ECOLE MAROCAINE DES  
SCIENCES DE L'INGENIEUR**  
*Membre de* **HONORIS UNITED UNIVERSITIES**



# PROJET ODOO

**Sous thème :**

---

## RÉSERVATION EQUIPEMENT

---

Réalise par : **Othmane SEMMAMI**

Class : **G3 Site :Maarif**

Prof : **Mohammed Ait Daoud**

Année universitaire :2025/2026

# 1 . Environnement et Déploiement

Le projet utilise Docker et Docker Compose pour isoler l'application et la base de données.

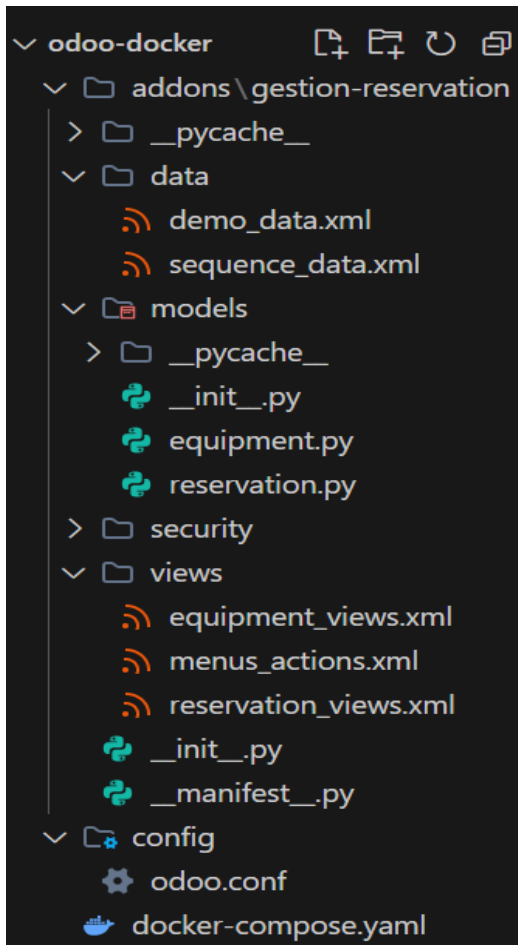
- Image Odoo : Version 17.0 Community Edition.
- Moteur de base de données : PostgreSQL 16.
- Persistence : Utilisation de volumes Docker pour les données Odoo et le dossier des modules personnalisés (/mnt/extra-addons).

## Structure du Module

Le module respecte l'architecture standard d'Odoo :

- `models/` : Définition des objets métier et de la logique serveur.
- `views/` : Définition de l'interface utilisateur (XML).
- `security/` : Gestion des droits d'accès (CSV et XML).
- `data/` : Séquences et données de démonstration.

personnalisés sont montés depuis `./addons` vers `/mnt/extra-addons` dans le conteneur.



## 2. Développement

Le module est conçu pour gérer deux entités principales : les **Équipements** et les **Réservations**.

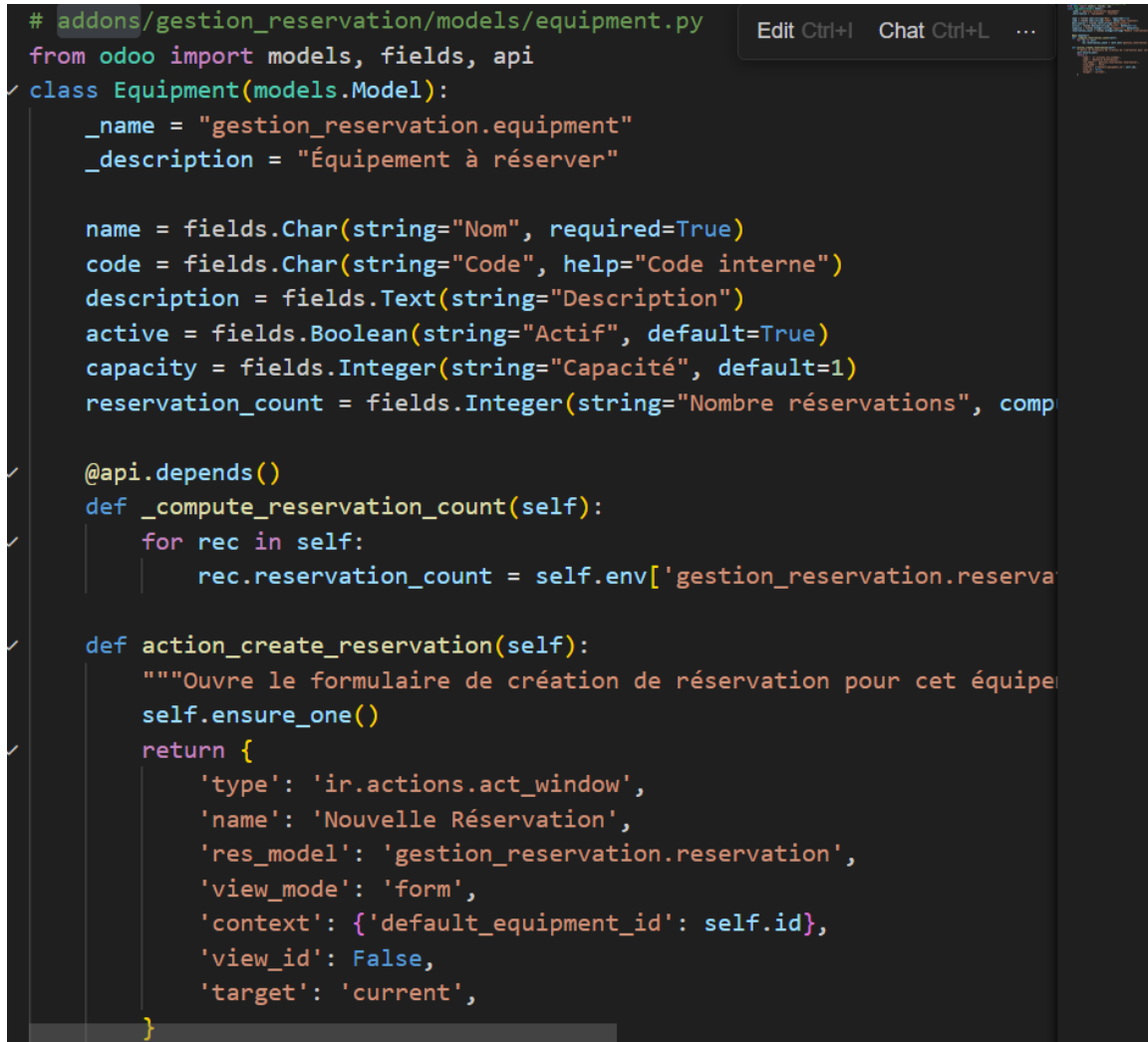
### A. Les Modèles

Le dossier **models** représente la couche "Données" et "Logique" du module. C'est ici que l'on définit la structure des tables SQL via des classes Python. Dans ce projet, il contient la définition des équipements et des réservations. C'est l'emplacement où sont programmées les règles métier, comme le calcul automatique des durées de réservation, les contraintes pour empêcher les chevauchements de dates, et les automatisations de comptage. Sans ce dossier, l'application n'aurait aucune intelligence ni base de données.

#### 1. Équipements (equipment.py) :

- Stocke les informations de base : Nom, Code, Capacité.
- **Logique** : Calcule automatiquement le nombre de réservations liées.

- **Action** : Possède une méthode `action_create_reservation` pour générer une réservation directement depuis l'équipement



```
# addons/gestion_reservation/models/equipment.py
from odoo import models, fields, api

class Equipment(models.Model):
    _name = "gestion_reservation.equipment"
    _description = "Équipement à réserver"

    name = fields.Char(string="Nom", required=True)
    code = fields.Char(string="Code", help="Code interne")
    description = fields.Text(string="Description")
    active = fields.Boolean(string="Actif", default=True)
    capacity = fields.Integer(string="Capacité", default=1)
    reservation_count = fields.Integer(string="Nombre réservations", compute=

@api.depends()
def _compute_reservation_count(self):
    for rec in self:
        rec.reservation_count = self.env['gestion_reservation.reserva

def action_create_reservation(self):
    """Ouvre le formulaire de création de réservation pour cet équipe
    self.ensure_one()
    return {
        'type': 'ir.actions.act_window',
        'name': 'Nouvelle Réservation',
        'res_model': 'gestion_reservation.reservation',
        'view_mode': 'form',
        'context': {'default_equipment_id': self.id},
        'view_id': False,
        'target': 'current',
    }
```

## 2. Réservations (reservation.py) :

- Gère le cycle de vie : Brouillon → Confirmée → Terminée.
- **Champs clés** : Équipement, Utilisateur, Dates début/fin, Durée calculée.
- **Innovations récentes** : Ajout des champs **Objet de la réservation** (purpose) et **Nombre de participants** (attendee\_count).
- **Validation** : Empêche techniquement le chevauchement (double réservation du même équipement sur le même créneau).

```

1 # addons/gestion_reservation/models/reservation.py
2 from odoo import models, fields, api, exceptions
3 from datetime import timedelta
4
5 class Reservation(models.Model):
6     _name = "gestion_reservation.reservation"
7     _description = "Réservation d'équipement"
8     _order = "start_datetime desc"
9
10    name = fields.Char(string="Référence", required=True, default="New")
11    equipment_id = fields.Many2one('gestion_reservation.equipement', string="Équipement")
12    user_id = fields.Many2one('res.users', string="Utilisateur", default="")
13    start_datetime = fields.Datetime(string="Début", required=True)
14    end_datetime = fields.Datetime(string="Fin", required=True)
15    duration_hours = fields.Float(string="Durée (h)", compute="_compute_duration")
16
17    # Nouveaux champs demandés
18    purpose = fields.Char(string="Objet de la réservation")
19    attendee_count = fields.Integer(string="Nombre de participants")
20
21    state = fields.Selection([
22        ('draft', 'Brouillon'),
23        ('confirmed', 'Confirmée'),
24        ('done', 'Terminée'),
25        ('cancel', 'Annulée')
26    ], default='draft', string="Statut")
27    notes = fields.Text(string="Notes")
28
29    @api.depends('start_datetime', 'end_datetime')
30    def _compute_duration(self):

```

```

@api.depends('start_datetime', 'end_datetime')
def _compute_duration(self):
    for rec in self:
        if rec.start_datetime and rec.end_datetime:
            delta = rec.end_datetime - rec.start_datetime
            rec.duration_hours = round(delta.total_seconds() / 3600.0)
        else:
            rec.duration_hours = 0.0

@api.constrains('start_datetime', 'end_datetime', 'equipment_id')
def _check_dates(self):
    for rec in self:
        if not rec.start_datetime or not rec.end_datetime:
            continue
        if rec.start_datetime >= rec.end_datetime:
            raise exceptions.ValidationError("La date de début doit être inférieure à la date de fin")
        # Vérifier chevauchement (ignorer annulées et s'exclure soi-même)
        domain = [
            ('equipment_id', '=', rec.equipment_id.id),
            ('id', '!=', rec.id),
            ('state', '!=', 'cancel'),
            ('start_datetime', '<', rec.end_datetime),
            ('end_datetime', '>', rec.start_datetime),
        ]
        overlapping = self.search(domain)
        if overlapping:
            raise exceptions.ValidationError("Chevauchement détecté : " + str(overlapping))

```

```

action_confirm(self):
self.write({'state': 'confirmed'})

action_done(self):
self.write({'state': 'done'})

action_cancel(self):
self.write({'state': 'cancel'})

..model_create_multi
create(self, vals_list):
for vals in vals_list:
    if vals.get('name', 'New') == 'New':
        vals['name'] = self.env['ir.sequence'].next_by_code('gestion_reservation.')
return super(Reservation, self).create(vals_list)

```

## B. L'Interface Utilisateur (Vues)

Le dossier views correspond à la couche "Présentation" (Interface Utilisateur). Il contient des fichiers XML qui décrivent comment les informations du modèle doivent être affichées à l'écran. C'est ici que l'on construit les formulaires de saisie, les listes (tree views), le calendrier et les menus de navigation. On y définit également les "Actions", qui sont les commandes permettant d'ouvrir une vue spécifique lorsqu'un utilisateur clique sur un menu ou un bouton.

- **Vues Équipements :**
  - Liste (Tree) avec bouton "Réserver" pour une ergonomie rapide.
  - Formulaire complet pour la création/modification.

```
<odoo>
<record id="view_equipment_tree" model="ir.ui.view">
  <field name="arch" type="xml">
    <tree>
      <field name="reservation_count"/>
      <button name="action_create_reservation" string="Réserver" type="object" />
    </tree>
  </field>
</record>

<record id="view_equipment_form" model="ir.ui.view">
  <field name="name">gestion_reservation.equipment.form</field>
  <field name="model">gestion_reservation.equipment</field>
  <field name="arch" type="xml">
    <form>
      <sheet>
        <group>
          <field name="name"/>
          <field name="code"/>
          <field name="capacity"/>
          <field name="active"/>
        </group>
        <group>
          <field name="description"/>
        </group>
      </sheet>
    </form>
  </field>
</record>
</odoo>
```

- **Vues Réservations :**

- **Calendrier :** Permet de visualiser les réservations de manière graphique par équipement.
- **Statut (Badge) :** Identification visuelle immédiate du cycle de vie (Brouillon, Confirmé, etc.).

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <!-- Tree view -->
  <record id="view_reservation_tree" model="ir.ui.view">
    <field name="name">gestion_reservation.reservation.tree</field>
    <field name="model">gestion_reservation.reservation</field>
    <field name="arch" type="xml">
      <tree>
        <field name="name"/>
        <field name="equipment_id"/>
        <field name="user_id"/>
        <field name="start_datetime"/>
        <field name="end_datetime"/>
        <field name="state" widget="badge" decoration-info="state == 'draft'" dec
      </tree>
    </field>
  </record>

  <!-- Form view -->
  <record id="view_reservation_form" model="ir.ui.view">
    <field name="name">gestion_reservation.reservation.form</field>
    <field name="model">gestion_reservation.reservation</field>
    <field name="arch" type="xml">
      <form>
        <header>
          <button name="action_confirm" string="Confirmer" type="object" invisibl
          <button name="action_done" string="Terminer" type="object" invisible="s
          <button name="action_cancel" string="Annuler" type="object" invisible="
          <field name="state" widget="statusbar" statusbar_visible="draft,confirm
        </header>
      </form>
    </field>
  </record>
</odoo>
```

```

<form>
  <sheet>
    <div class="oe_title">
      <h1>
        <field name="name" readonly="1"/>
      </h1>
    </div>
    <group>
      <group>
        <field name="equipment_id"/>
        <field name="user_id"/>
        <field name="purpose"/>
        <field name="attendee_count"/>
      </group>
      <group>
        <field name="start_datetime"/>
        <field name="end_datetime"/>
        <field name="duration_hours"/>
      </group>
    </group>
    <notebook>
      <page string="Notes" name="notes">
        <field name="notes" placeholder="Notes sur la réservation..."/>
      </page>
    </notebook>
  </sheet>
</form>

```

```

<!-- Calendar view -->
<record id="view_reservation_calendar" model="ir.ui.view">
  <field name="name">gestion_reservation.reservation.calendar</field>
  <field name="model">gestion_reservation.reservation</field>
  <field name="arch" type="xml">
    <calendar string="Calendrier de réservation" date_start="start_datetime" date_stop="end_datetime" color="equipme
      <field name="equipment_id"/>
      <field name="user_id"/>
    </calendar>
  </field>
</record>
</odoo>

```

## C. Sécurité et Visibilité

Le dossier security gère le contrôle des accès et la protection des données. Il contient deux éléments essentiels : d'une part, la définition des groupes d'utilisateurs (comme "Utilisateurs" et "Managers") dans des fichiers XML, et d'autre part, les droits d'accès détaillés dans un fichier CSV (ir.model.access.csv). Ce dernier spécifie précisément pour chaque groupe si la lecture, l'écriture, la création ou la suppression est autorisée sur chaque modèle. C'est le garant de la sécurité du système.

- **Groupes** : Définit deux niveaux (Utilisateur et Manager).
- **Accessibilité** : Configuration optimisée pour que tous les utilisateurs internes puissent voir et utiliser l'application sans erreur de droits ("Access Error").
- **Menu** : Un menu racine dédié "Gestion Réservation" avec une icône distinctive dans le tableau de bord Odoo.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- addons/gestion_reservation/security/security.xml -->
<odoo>
  <data noupdate="1">
    <record id="module_category_gestion_reservation" model="ir.module.category">
      <field name="name">Gestion Réservation</field>
      <field name="description">Gérer les réservations d'équipements</field>
      <field name="sequence">10</field>
    </record>

    <record id="group_gestion_reservation_user" model="res.groups">
      <field name="name">Utilisateur Réservation</field>
      <field name="category_id" ref="module_category_gestion_reservation"/>
      <field name="implied_ids" eval="[(4, ref('base.group_user'))]"/>
    </record>

    <record id="group_gestion_reservation_manager" model="res.groups">
      <field name="name">Manager Réservation</field>
      <field name="implied_ids" eval="[(4, ref('group_gestion_reservation_user'))]"/>
      <field name="category_id" ref="module_category_gestion_reservation"/>
    </record>
  </data>
</odoo>
```

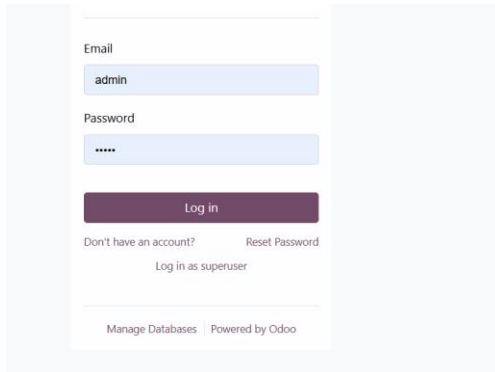
```
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 access_equipment_user,access_equipment_user,model_gestion_reservation_equipment,base.group_user,1,0,0,0
3 access_equipment_manager,access_equipment_manager,model_gestion_reservation_equipment,base.group_user,1,1,1,1
4 access_reservation_user,access_reservation_user,model_gestion_reservation_reservation,base.group_user,1,1,1,0
5 access_reservation_manager,access_reservation_manager,model_gestion_reservation_reservation,base.group_user,1,1,1,1
6
```

## 1. Réalisation

### Démarrage Container :

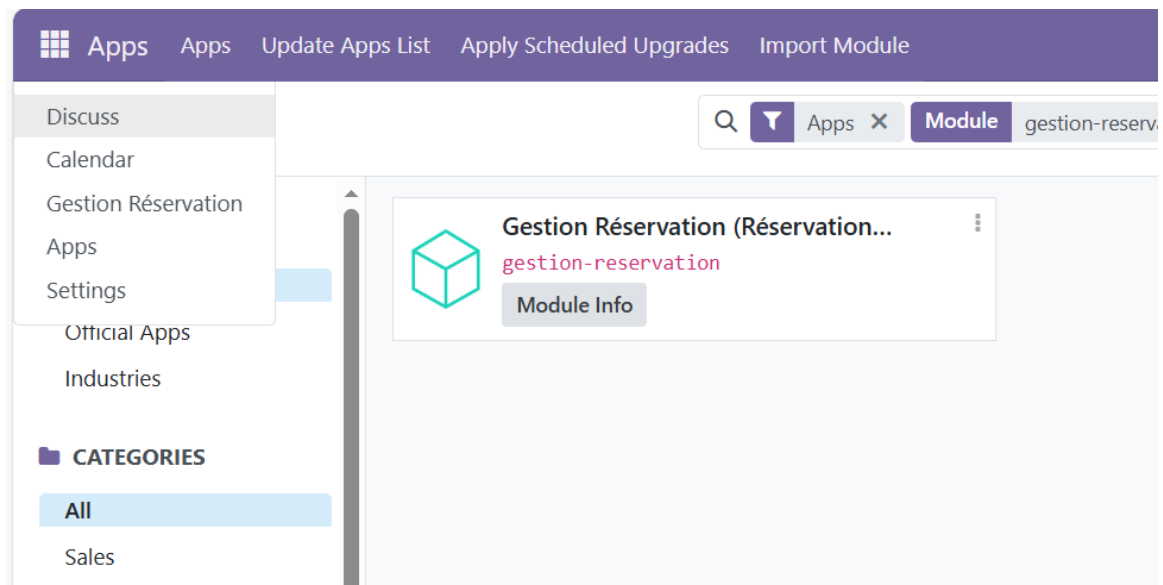
<input type="checkbox"/>	▼	odoo-docker	-	-	-	0.02%	38 min	<input type="checkbox"/>
<input checked="" type="checkbox"/>		odoo_app	afb1f7358351	<a href="#">odoo:17.0</a>	<a href="#">8069:8069</a>	0.02%	38 min	<input type="checkbox"/>
<input type="checkbox"/>		odoo_db	c5662dca5c5e	<a href="#">postgres:16</a>	<a href="#">5432:5432</a>	0%	38 min	<input type="checkbox"/>

## Authentication :



The image shows the Odoo authentication login interface. It features a light gray background with a white login box in the center. Inside the box, there are two input fields: 'Email' with the text 'admin' and 'Password' with masked characters '\*\*\*\*\*'. Below these fields is a dark purple 'Log in' button. Underneath the button, there are two links: 'Don't have an account?' and 'Reset Password'. At the bottom of the login box, there is a link 'Log in as superuser'. At the very bottom of the page, outside the login box, are the links 'Manage Databases' and 'Powered by Odoo'.

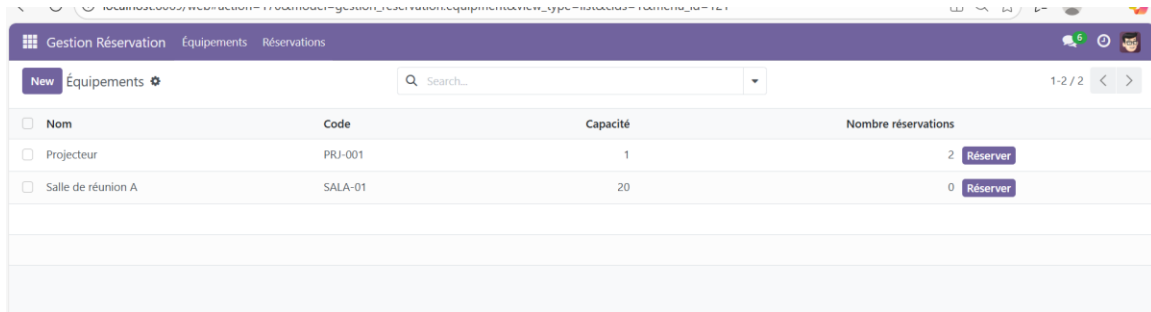
## Gestion réservation :



## Etape Travail

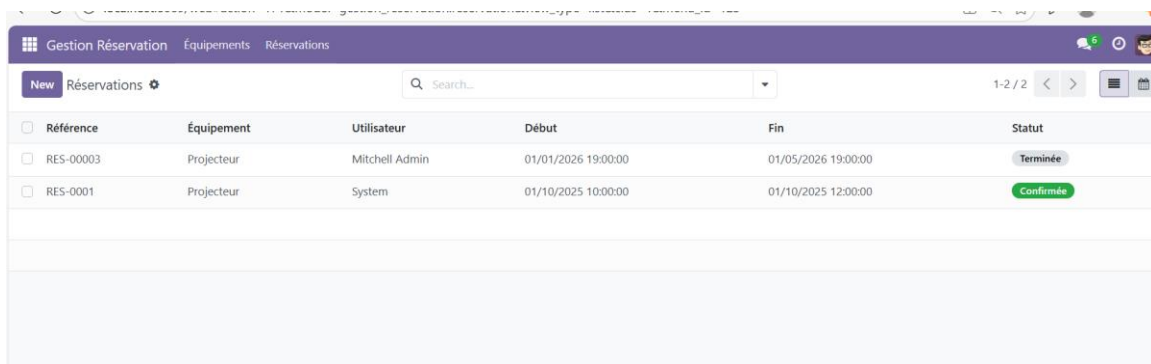
1. L'utilisateur consulte la liste des **Équipements**.
2. Il clique sur "**Réserver**" sur un équipement spécifique.
3. Le formulaire de **Réservation** s'ouvre avec l'équipement déjà sélectionné.
4. Il saisit l'objet, le nombre de participants et les dates.
5. Odoo calcule la durée et vérifie la disponibilité.
6. La réservation passe de **Brouillon** à **Confirmée** via les boutons dans l'en-tête (Header).

## Pour Gérer Equipement :



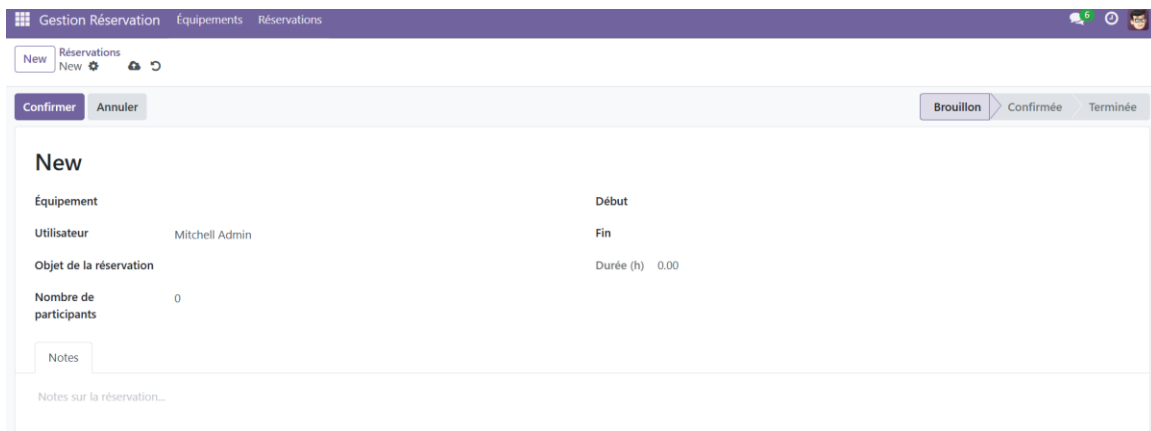
<input type="checkbox"/> Nom	Code	Capacité	Nombre réservations
<input type="checkbox"/> Projecteur	PRJ-001	1	2 <a href="#">Réserver</a>
<input type="checkbox"/> Salle de réunion A	SALA-01	20	0 <a href="#">Réserver</a>

## Pour Gérer Réservation :



<input type="checkbox"/> Référence	Équipement	Utilisateur	Début	Fin	Statut
<input type="checkbox"/> RES-0003	Projecteur	Mitchell Admin	01/01/2026 19:00:00	01/05/2026 19:00:00	<a href="#">Terminée</a>
<input type="checkbox"/> RES-0001	Projecteur	System	01/10/2025 10:00:00	01/10/2025 12:00:00	<a href="#">Confirmée</a>

## Pour Ajouter une réservation :



**New**

Équipement

Utilisateur Mitchell Admin

Objet de la réservation

Nombre de participants 0

Notes

Notes sur la réservation...

[Confirmer](#) [Annuler](#) [Brouillon](#) [Confirmée](#) [Terminée](#)

## Clic Sur Button Confirmer pour Passer à étape suivant

New

Réervations

New

🔍

🔔

🔄

Confirmer

Annuler

Brouillon

Confirmée

Terminée

### New

Équipement	Projecteur	Début	01/01/2026 20:00:00
Utilisateur	Joel Willis	Fin	01/17/2026 20:00:00
Objet de la réservation	2	Durée (h)	384.00
Nombre de participants	2		

Notes

Notes sur la réservation...

## Maintenant la réservation Confirmer Click sur Terminer Pour Terminer La réservation (finition réservation et rendre équipement)

New

Réervations

RES-00006

🔍

1 / 1

<

>

Terminer

Annuler

Brouillon

Confirmée

Terminée

### RES-00006

Équipement	Salle de réunion A	Début	01/01/2026 20:00:00
Utilisateur	Joel Willis	Fin	01/16/2026 20:00:00
Objet de la réservation	2	Durée (h)	360.00
Nombre de participants	2		

Notes

Notes sur la réservation...

## Ajout d'un équipement

New

Équipements

Microphone

🔍

Nom

Microphone

Code ?

MI7625

Capacité

1

Actif

☒

Description