

Rapport de Projet :

Module de Gestion de Réservation de Salles sous Odoo 17

Réalisé par : Yassine BEKKOUCH

Encadré par : M. Mohammed AITDAOUD

*5ème Année en Ingénierie Informatique et Réseaux spécialité
MIAGE*



Année universitaire : 2023/2024

Table des matières

1. Introduction	4
2. Environnement Technique et Initialisation	4
2.1 Structure du Module.....	4
3. Développement Backend (Modèles de Données)	5
3.1 Code models	6
3.2 Code security.....	6
3.3 Code views.....	7
4. Développement Frontend (Vues XML et Menus).	7
4.1 Gestion des Salles	7
4.2 Gestion des Réservations	8
4.3 Suivi des Statuts (Workflow)	9
5. Sécurité et Déploiement.....	10
6. Conclusion.....	10

1. Introduction

Ce projet vise à développer un module personnalisé pour l'ERP Odoo (version 17) permettant la gestion complète de salles de réunion et de leurs réservations. Le module a été développé dans un environnement conteneurisé Docker, respectant l'architecture MVC (Modèle-Vue-Contrôleur) propre au framework Odoo.

Objectifs du module :

- Gérer un parc de salles (capacité, équipements).
- Planifier des réservations via un calendrier.
- Suivre le statut des demandes (Brouillon, Confirmé, Annulé).

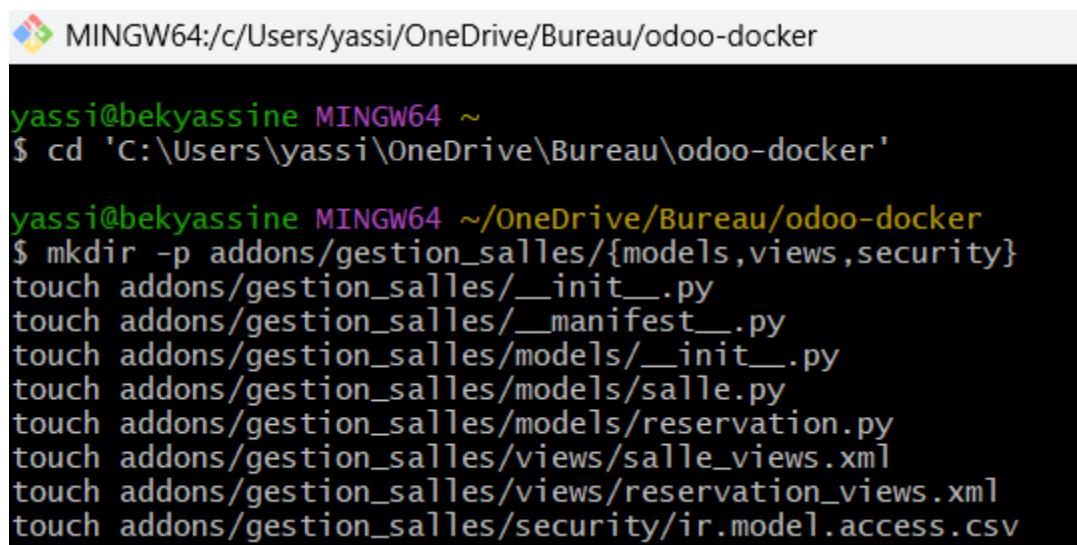
2. Environnement Technique et Initialisation

Le projet repose sur l'utilisation de Docker pour l'orchestration des conteneurs (PostgreSQL et Odoo) et de Visual Studio Code pour le développement.

2.1 Structure du Module

La première étape a consisté à créer l'arborescence standard d'un module Odoo. La logique métier (models), les interfaces (views) et la sécurité (security) ont été séparées.

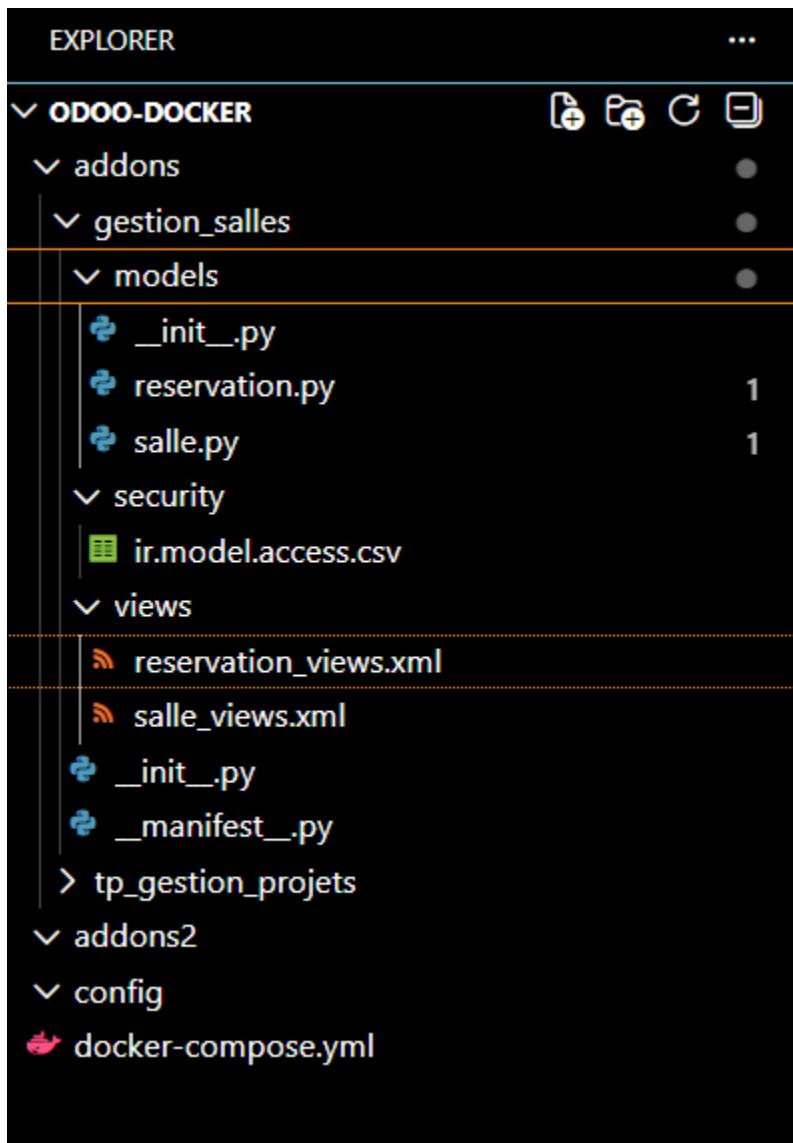
Illustration : Création de la structure via le terminal



```
MINGW64:/c/Users/yassi/OneDrive/Bureau/odoo-docker
yassi@bekyassine MINGW64 ~
$ cd 'C:\Users\yassi\OneDrive\Bureau\odoo-docker'

yassi@bekyassine MINGW64 ~/OneDrive/Bureau/odoo-docker
$ mkdir -p addons/gestion_salles/{models,views,security}
touch addons/gestion_salles/__init__.py
touch addons/gestion_salles/__manifest__.py
touch addons/gestion_salles/models/__init__.py
touch addons/gestion_salles/models/salle.py
touch addons/gestion_salles/models/reservation.py
touch addons/gestion_salles/views/salle_views.xml
touch addons/gestion_salles/views/reservation_views.xml
touch addons/gestion_salles/security/ir.model.access.csv
```

Illustration : Vue de l'explorateur de fichiers



3. Développement Backend (Modèles de Données)

Le cœur du module repose sur deux classes Python principales définies dans le dossier `models` :

- `gestion.salle` : Définit les caractéristiques d'une salle (Nom, Capacité, Projecteur, Climatisation).
- `gestion.reservation` : Gère les événements (Date de début, Date de fin, Salle liée, Responsable).

Les fichiers `_manifest__.py` et `_init__.py` ont été configurés pour déclarer le module et charger les fichiers Python.

3.1 Code models

```
salle.py
addons > gestion_salles > models > salle.py > Salle
1 from odoo import models, fields
2
3 class Salle(models.Model):
4     _name = "gestion.salle"
5     _description = "Salle de réunion"
6
7     name = fields.Char(string="Nom de la salle", required=True)
8     capacite = fields.Integer(string="Capacité (personnes)")
9     a_projecteur = fields.Boolean(string="Projecteur vidéo disponible ?")
10    a_climatisation = fields.Boolean(string="Climatisation ?")
11    description = fields.Text(string="Description / Notes")
```

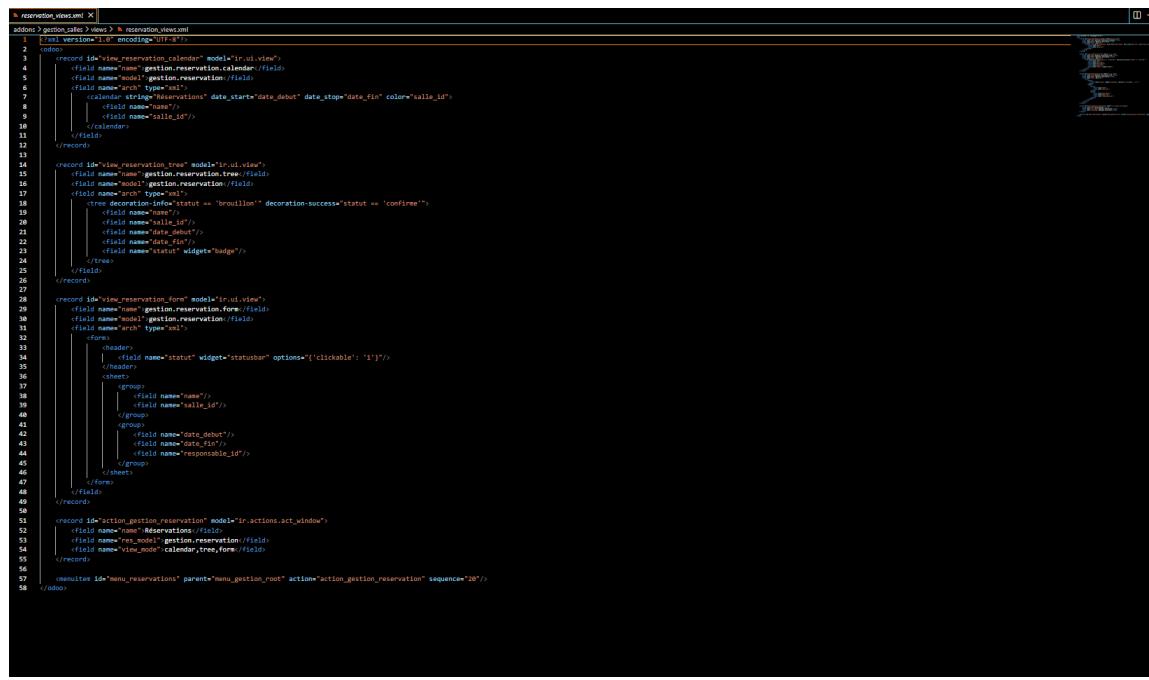
```
__init__.py
addons > gestion_salles > models > __init__.py
1 from . import salle
2 from . import reservation
```

```
reservation.py
addons > gestion_salles > models > reservation.py > Reservation
1 from odoo import models, fields
2
3 class Reservation(models.Model):
4     _name = "gestion.reservation"
5     _description = "Réservation de salle"
6
7     name = fields.Char(string="Objet de la réunion", required=True)
8     salle_id = fields.Many2one("gestion.salle", string="Salle", required=True)
9     date_debut = fields.Datetime(string="Début", required=True)
10    date_fin = fields.Datetime(string="Fin", required=True)
11    responsable_id = fields.Many2one("res.users", string="Organisateur", default=lambda self: self.env.user)
12    statut = fields.Selection([
13        ('brouillon', 'Brouillon'),
14        ('confirme', 'Confirmé'),
15        ('annule', 'Annulé'),
16    ], string="Statut", default='brouillon')
```

3.2 Code security

```
ir.model.access.csv
addons > gestion_salles > security > ir.model.access.csv
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 access_gestion_salle,gestion.salle,model_gestion_salle,,1,1,1,1
3 access_gestion_reservation,gestion.reservation,model_gestion_reservation,,1,1,1,1
```

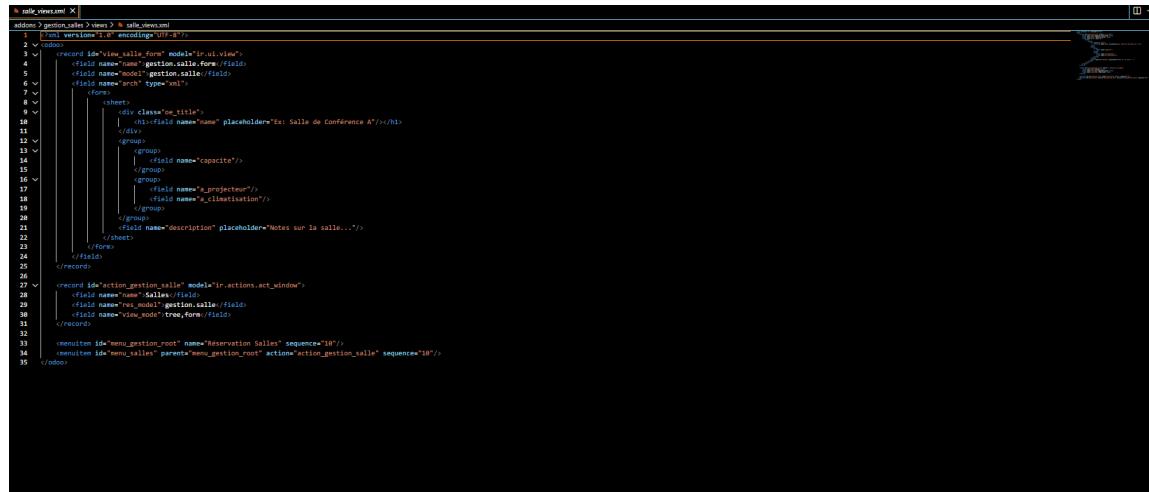
3.3 Code views



```

<?xml version="1.0" encoding="UTF-8"?>
<odoo>
<record id="view_reservation_calendar" model="ir.ui.view">
<field name="name">gestion.reservation.calendar</field>
<field name="model">gestion.reservation</field>
<field name="arch" type="xml">
<calendar string="Reservations" date_start="date_debut" date_stop="date_fin" color="salles_id" />
<field name="salles_id"/>
</calendar>
</field>
</record>
<record id="view_reservation_tree" model="ir.ui.view">
<field name="name">gestion.reservation.tree</field>
<field name="model">gestion.reservation</field>
<field name="arch" type="xml">
<tree string="Liste des réservations" decoration-success="status == 'confirmé'" decoration-error="status == 'brouillon'" decoration-warning="status == 'en cours'"/>
<field name="salles_id"/>
<field name="date_debut"/>
<field name="date_fin"/>
<field name="status" widget="badge"/>
</tree>
</field>
</record>
<record id="view_reservation_form" model="ir.ui.view">
<field name="name">gestion.reservation.form</field>
<field name="model">gestion.reservation</field>
<field name="arch" type="xml">
<form string="Formulaire de réservation">
<header>
<field name="status" widget="statusbar" options="{'clickable': 1}"/>
</header>
<sheet>
<group>
<field name="name"/>
<field name="salles_id"/>
</group>
<group>
<field name="date_debut"/>
<field name="date_fin"/>
<field name="status" widget="badge"/>
</group>
</sheet>
</form>
</field>
</record>
<record id="action_gestion_reservation" model="ir.actions.act_window">
<field name="name">Réservation</field>
<field name="res_model">gestion.reservation</field>
<field name="view_mode">calendar,tree,form</field>
</record>
<menuitem id="menu_reservations" parent="menu_gestion_root" action="action_gestion_reservation" sequence="20"/>
</odoo>

```



```

<?xml version="1.0" encoding="UTF-8"?>
<odoo>
<record id="view_salle" model="ir.ui.view">
<field name="name">gestion.salle.form</field>
<field name="model">gestion.salle</field>
<field name="arch" type="xml">
<form string="Formulaire de gestion des salles">
<sheet>
<div class="oe_title"><field name="name" placeholder="Salle de Conférence A"/></div>
<div>
<group>
<field name="capacite"/>
</group>
<group>
<field name="a_projecteur"/>
<field name="a_climatisation"/>
</group>
<group>
<field name="description" placeholder="Notes sur la salle..."/>
</group>
</div>
</sheet>
</form>
</field>
</record>
<record id="action_gestion_salle" model="ir.actions.act_window">
<field name="name">Salles</field>
<field name="res_model">gestion.salle</field>
<field name="view_mode">tree,form</field>
</record>
<menuitem id="menu_gestion_root" name="Réservation Salles" sequence="10"/>
<menuitem id="menu_salles" parent="menu_gestion_root" action="action_gestion_salle" sequence="10"/>
</odoo>

```

4. Développement Frontend (Vues XML et Menus)

L'interface utilisateur a été conçue en XML pour offrir plusieurs modes de visualisation : Liste (Tree), Formulaire (Form) et Calendrier.

4.1 Gestion des Salles

Une interface permet de lister les salles et de consulter leurs détails techniques (équipements disponibles).

Illustrations prévues :

- Vue liste des salles.

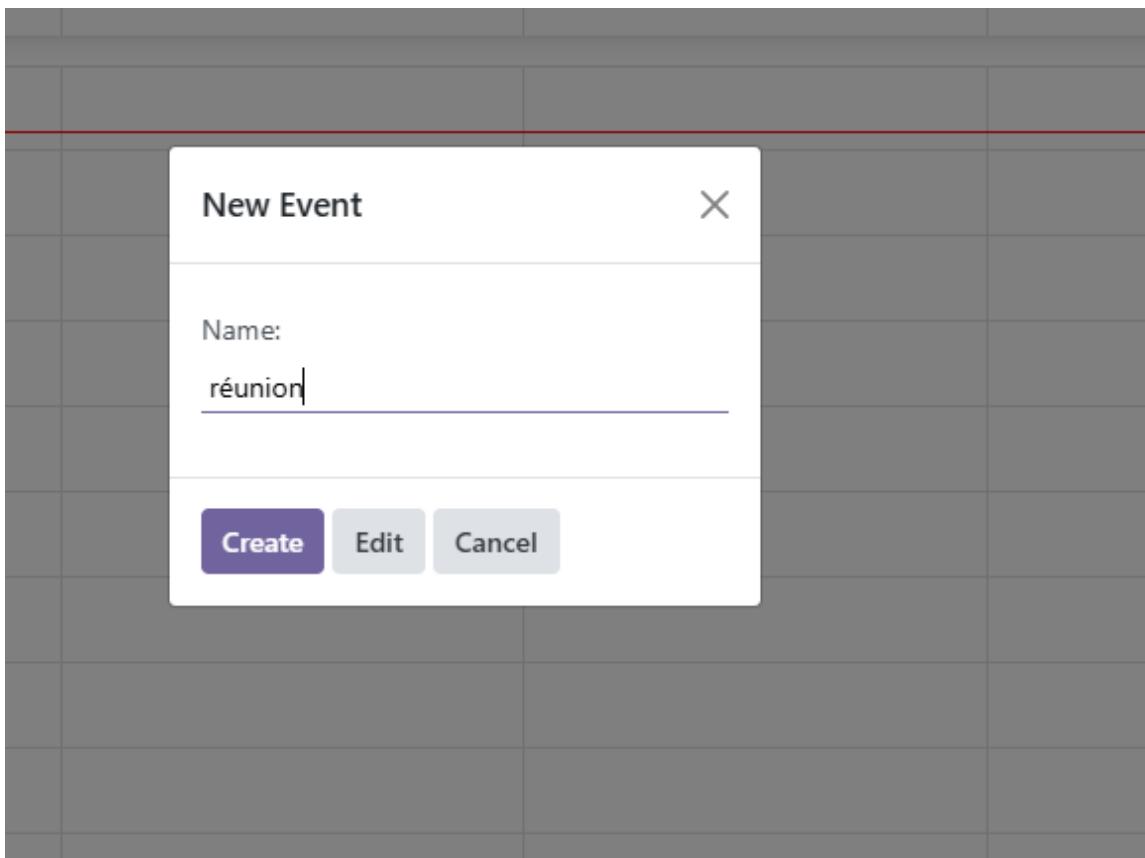
The screenshot shows a software interface for managing rooms. At the top, there's a navigation bar with 'New' and 'Salles'. A search bar is present, and the page number '1-2 / 2' with navigation arrows is at the top right. Below the header, a list of rooms is shown with checkboxes next to their names: 'Nom de la salle', 'Salle de conférence A', and 'Salle de conférence B'. The 'Salle de conférence B' row is highlighted with a grey background. Below this, a large, semi-transparent light-grey box covers the main content area. Inside this box, there's a sub-header 'Salle de conférence B', a field 'Capacité (personnes)' with the value '40', and two checkboxes: 'Projecteur vidéo disponible ?' and 'Climatisation ?'. At the bottom left of this box, it says 'Salle avec table ronde'.

4.2 Gestion des Réservations

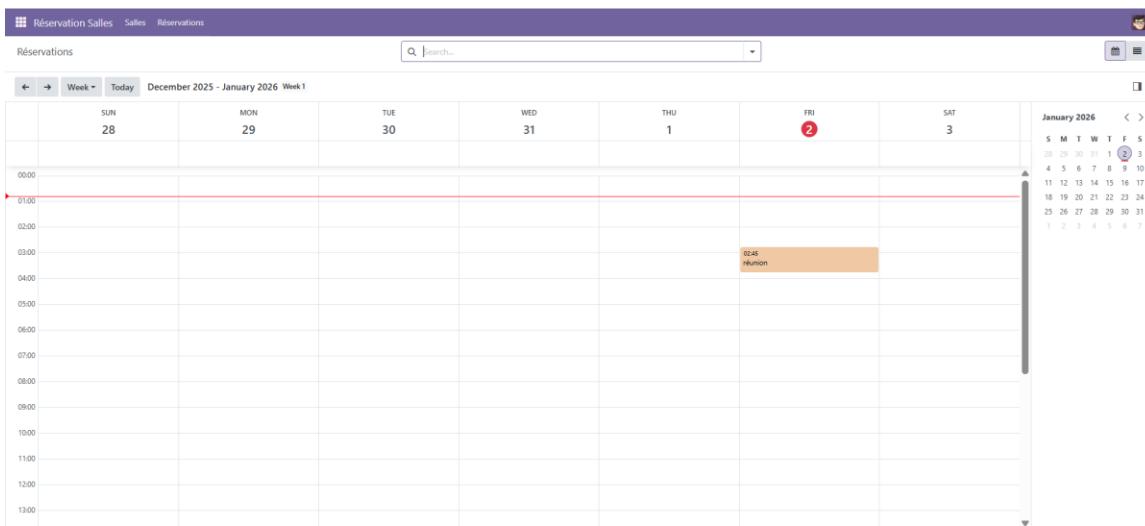
Une vue Calendrier a été intégrée afin de faciliter la visualisation et la planification des réservations.

Illustrations prévues :

- Crédit rapide d'une réservation depuis le calendrier.



- Vue calendrier hebdomadaire globale.

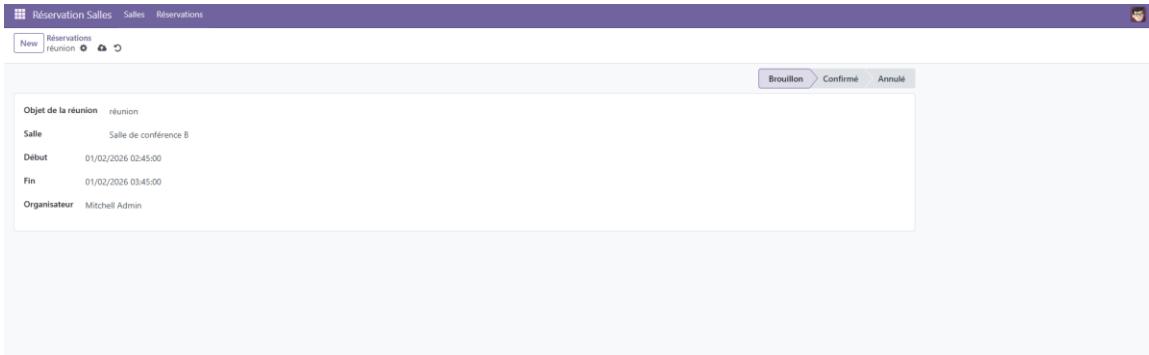


4.3 Suivi des Statuts (Workflow)

Un workflow simple a été mis en place avec une barre d'état permettant de suivre l'évolution des réservations (Brouillon, Confirmé, Annulé).

Illustrations prévues :

- Workflow de validation.



- Liste synthétique des réservations.

Réservation Salles				
New Réservations	Salle	Début	Fin	Statut
<input type="checkbox"/> Objet de la réunion	Salle de conférence B	01/02/2026 02:45:00	01/02/2026 03:45:00	Confirmé
<input type="checkbox"/> réunion				

5. Sécurité et Déploiement

Les droits d'accès ont été configurés dans le fichier `ir.model.access.csv` afin de définir les permissions CRUD (Create, Read, Update, Delete) aux utilisateurs.

Le déploiement a nécessité la configuration du `docker-compose.yml` pour monter correctement le volume des extra-addons afin qu'Odoo détecte le nouveau module personnalisé.

6. Conclusion

Ce projet a permis de mettre en pratique l'architecture modulaire d'Odoo. Le résultat est un module fonctionnel de gestion de réservation de salles, intégré nativement à l'ERP et offrant une interface fluide et adaptée aux besoins de l'organisation.