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

Ecole Reconnue par l'Etat

ÉCOLE MAROCAINE DES SCIENCES DE L'INGÉNIEUR

Filière Ingénierie Informatique et Réseaux

RAPPORT DE PROJET

Développement d'un module ERP sous Odoo 17 :

Gestion Informatisée des Projets TP

Réalisé par :
Oussama KARBAL

Encadré par :
Pr. M. AIT DAOUD

Année Universitaire 2025 - 2026

Table des matières

Table des figures

Remerciements

L'aboutissement de ce projet de développement sous Odoo 17 est le fruit d'un apprentissage intensif sur les technologies ERP et la conteneurisation. Je tiens, en premier lieu, à exprimer ma profonde gratitude à mon encadrant, **Pr. M. AIT DAOUD**, pour son expertise technique, sa pédagogie et ses précieux conseils lors des séances de TP qui ont permis la réalisation de ce module. Je remercie également l'ensemble du corps professoral de l'EMSI pour la qualité de la formation en ingénierie. Enfin, je remercie ma famille pour leur soutien constant tout au long de mon cursus universitaire.

Résumé

Ce projet porte sur la conception et le développement d'un module personnalisé au sein de l'ERP Odoo 17, intitulé `tp_gestion_taches`. L'objectif principal est de fournir une solution numérique centralisée pour l'organisation et le suivi des activités quotidiennes au sein d'une structure. Le système permet l'enregistrement précis de chaque tâche, l'affectation d'un responsable dédié, la définition des échéances et le suivi en temps réel de l'état d'avancement (Brouillon, En cours, Terminé). L'architecture technique repose sur un environnement conteneurisé Docker, assurant une isolation parfaite des services et une portabilité maximale. Ce travail illustre la synergie entre la puissance de l'ORM Python pour la logique métier et la flexibilité du XML pour la création d'interfaces utilisateurs ergonomiques.

Abstract

This project involves the design and implementation of a custom module for Odoo 17, named `tp_gestion_taches`. The core objective is to deliver a centralized digital solution for organizing and monitoring daily tasks within an organization. The system enables precise task recording, assignment of responsible personnel, deadline management, and real-time progress tracking (Draft, In Progress, Completed). The technical architecture is built on a Docker containerized environment, ensuring service isolation and high portability. This work demonstrates the effective integration of Python ORM for business logic and XML for developing intuitive and responsive user interfaces

Introduction Générale

Dans une ère marquée par l'accélération de la transformation digitale, la gestion rigoureuse des flux de travail est devenue une priorité stratégique pour toute organisation. La gestion des tâches, souvent traitée de manière informelle par des notes éparses ou des outils non centralisés, représente un défi logistique majeur. Le manque de visibilité sur "qui fait quoi" et sur l'état d'avancement des dossiers peut entraîner une baisse de productivité significative.

C'est dans ce contexte que s'inscrit le présent projet, visant à concevoir un module de gestion des tâches sur mesure sous l'ERP Odoo 17. En exploitant ce framework, nous ambitionnons de transformer une gestion manuelle souvent désorganisée en un processus automatisé, transparent et sécurisé. Ce rapport détaille la mise en place d'une infrastructure moderne basée sur Docker Compose, la modélisation des données en Python et la création d'interfaces dynamiques en XML, offrant ainsi un outil robuste pour le pilotage opérationnel des activités.

Chapitre 1

Présentation du Projet

1.1 Contexte du projet

Le projet a été réalisé dans un cadre pédagogique à l'EMSI pour simuler la gestion des projets internes. Face à la multiplication des tâches et des intervenants, il est devenu indispensable de disposer d'un outil centralisé permettant de savoir "Qui fait quoi" et "Où en est le projet". Ce module est une réponse directe à ce besoin de visibilité administrative.

1.2 Problématique

La gestion manuelle ou via des tableurs classiques engendre souvent des erreurs de saisie, un manque de synchronisation entre les collaborateurs et une difficulté à suivre l'historique des modifications. Les questions clés sont : Comment structurer les données de projet de manière persistante ? Comment offrir une interface simple pour les responsables ? Et enfin, comment assurer que l'outil de développement soit identique à l'outil de production ?

1.3 Objectifs du projet

L'objectif est de livrer un module opérationnel répondant aux critères suivants :

- **Infrastructure stable :** Utilisation de Docker Compose pour lier Odoo et PostgreSQL.
- **Modélisation métier :** Création du modèle `tp.projet` avec des champs spécifiques (Nom, Responsable, Date, Statut).
- **Interface ergonomique :** Développement de vues Liste et Formulaire intuitives.
- **Sécurité :** Mise en place de droits d'accès via le fichier CSV de sécurité.

1.4 Méthodologie

Nous avons suivi une approche itérative : 1. Installation de l'environnement Docker. 2. Configuration du fichier `odoo.conf`. 3. Développement du code Python pour la structure des données. 4. Création des vues XML pour l'interface utilisateur. 5. Tests de déploiement et mise à jour via les options de ligne de commande d'Odoo.

Chapitre 2

Conception

2.1 Diagramme de séquence

Le diagramme de séquence illustre la dynamique du système en montrant comment les messages circulent entre l'utilisateur, le serveur Odoo et la base de données. Il permet de comprendre le processus de validation interne lors de l'enregistrement d'un objet ou du passage d'un état à un autre au sein du workflow. C'est un outil indispensable pour déboguer la logique de contrôle et s'assurer que le flux d'informations respecte scrupuleusement les règles métier établies.

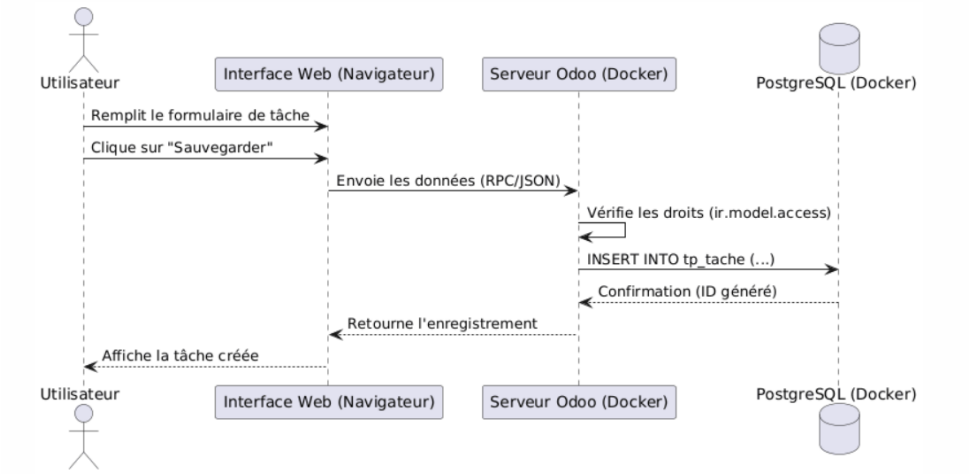


FIGURE 2.1 – Diagramme de séquence

2.2 Diagramme de classes

Le diagramme de classes représente la structure statique du module en détaillant les entités et les attributs stockés en base de données PostgreSQL. Il définit comment l'ORM d'Odoo va traduire nos objets Python en tables relationnelles, incluant les champs de

texte, de date et les listes de sélection. Cette modélisation est cruciale pour assurer la cohérence des données au sein de l'ERP et faciliter les futures extensions du schéma de données métier.

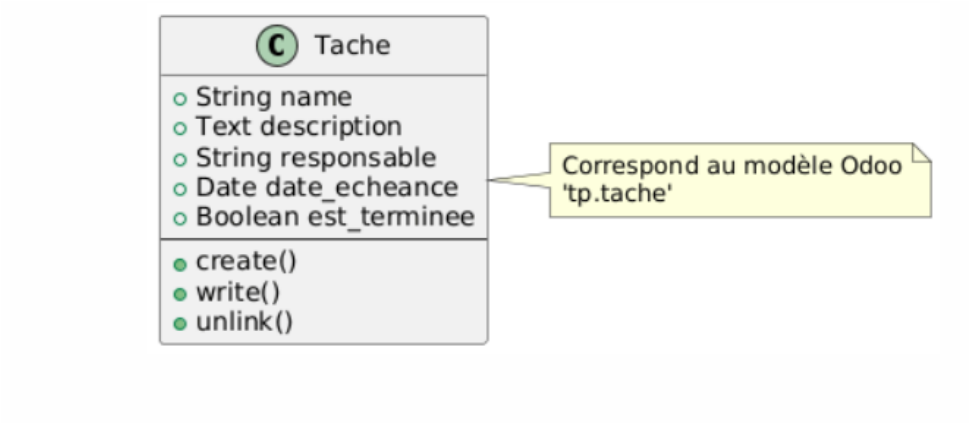


FIGURE 2.2 – Diagramme de classes

2.3 Schéma de données (Modèle)

Le modèle de données définit comment les projets sont stockés. Chaque projet possède un nom unique, un responsable (chaîne de caractères), une date de début et un statut parmi trois choix : Brouillon, En cours ou Terminé. Cette structure simple garantit une performance optimale de la base de données.

2.4 Workflow du module

Le flux de travail permet à un utilisateur de créer un projet à l'état "Brouillon", puis de le faire progresser. L'interface XML traduit ces états visuellement dans Odoo, permettant un suivi en temps réel de l'avancement global.

Chapitre 3

Outils Utilisés

Odoo 17 : Framework utilisé pour le développement de la logique métier et de l'interface ERP.

Docker : Outil de conteneurisation utilisé pour isoler l'environnement et faciliter le déploiement.

PostgreSQL : SGBD relationnel gérant la persistance des données du module.

VS Code : Éditeur principal pour l'écriture des fichiers Python, XML et de configuration.

Chapitre 4

Partie Réalisation

4.1 Structure du module dans l'environnement Docker

La structure du module est organisée selon les standards stricts d'Odoo pour permettre une reconnaissance automatique par le serveur lors du démarrage. Nous y retrouvons le dossier models pour la logique Python, views pour l'interface XML et security pour les fichiers CSV de droits d'accès. Cette organisation modulaire facilite la maintenance à long terme et permet d'isoler chaque aspect technique du projet pour une meilleure lisibilité globale du code source.

4.2 Développement du Modèle Python

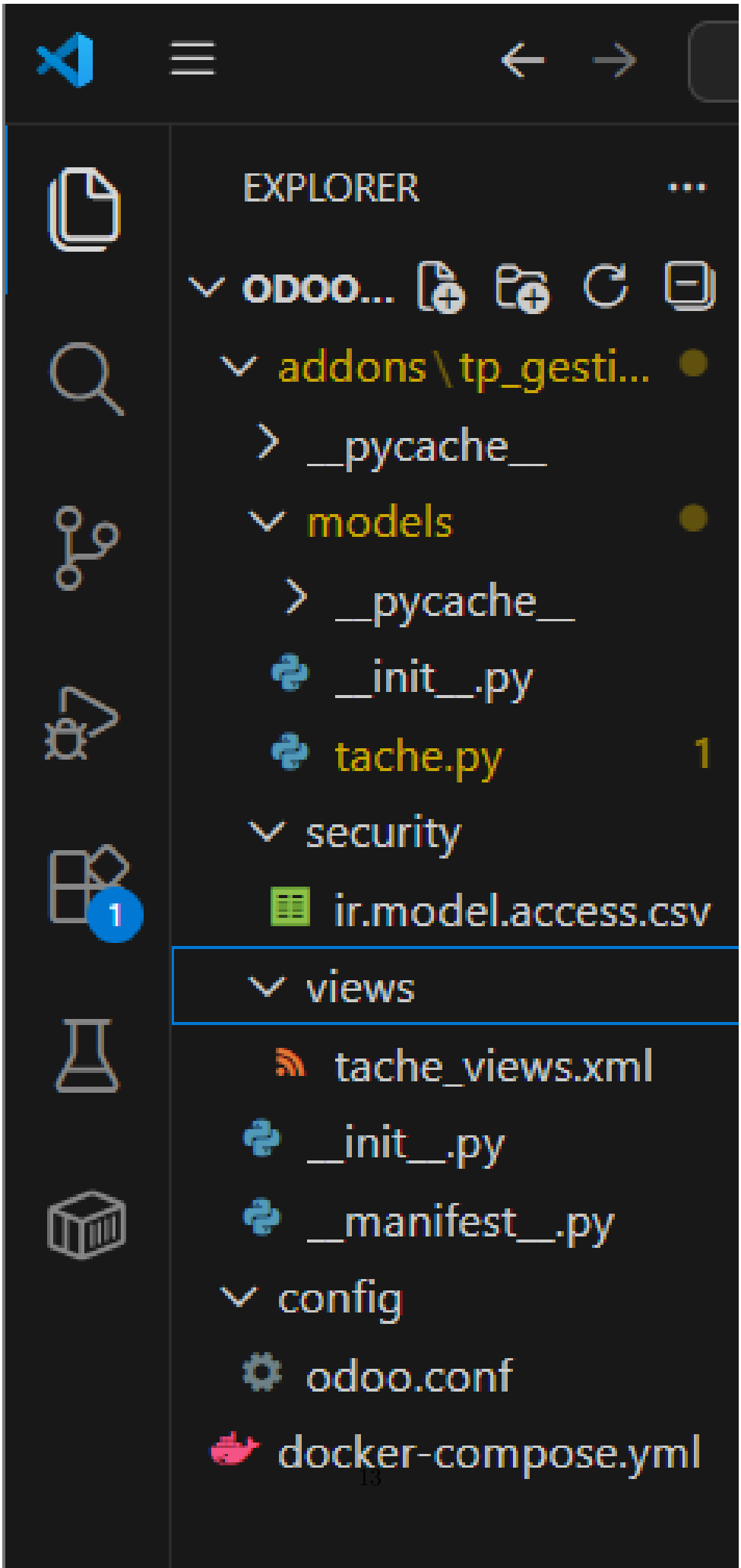
Le fichier `projet.py` définit la classe `TpProjet`. Nous utilisons l'ORM d'Odoo pour déclarer nos champs.

4.3 Création des Vues XML

Les vues définissent l'affichage. Nous avons créé une vue `tree` pour la liste et une vue `form` pour l'édition des projets.

4.4 Sécurité et Accès

Le fichier `ir.model.access.csv` a été configuré pour donner les droits de lecture, écriture et création aux utilisateurs du groupe de base, assurant ainsi que le module est utilisable dès son installation.


EXPLORER
ODOO...
addons \ tp_gesti...
__pycache__
models
__pycache__
__init__.py
tache.py
1
security
ir.model.access.csv
views
tache_views.xml
__init__.py
__manifest__.py
config
odoo.conf
docker-compose.yml

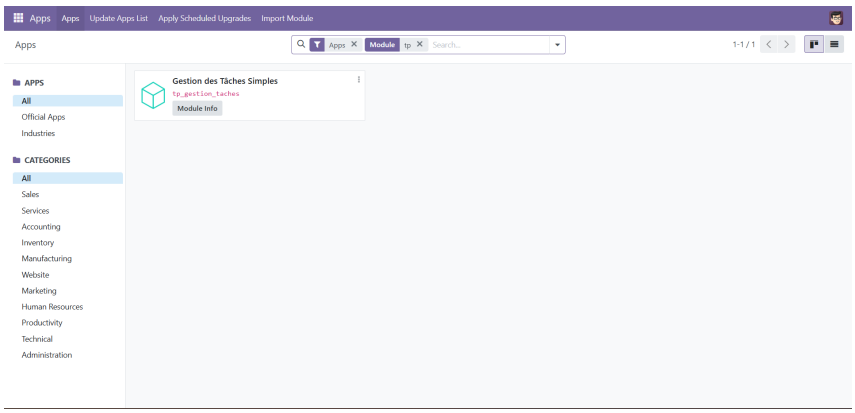


FIGURE 4.2 – Activation du module dans Odoo

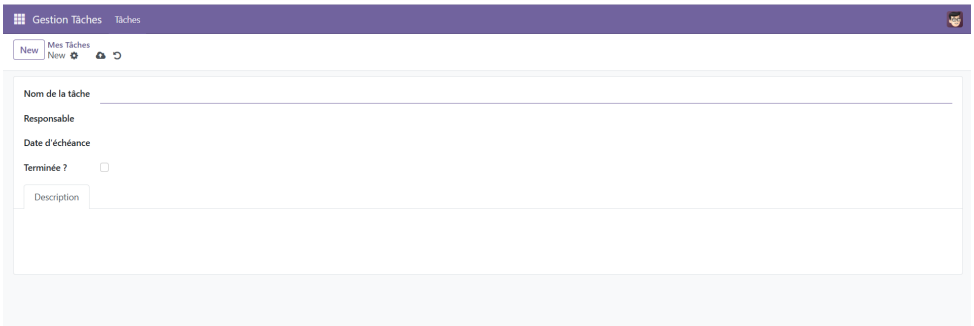


FIGURE 4.3 – Formulaire de déclaration des taches

Chapitre 5

Conclusion Générale

La réalisation du module `tp_gestion_projets` a permis de valider la maîtrise du cycle complet de développement sur Odoo 17. L'utilisation combinée de Docker pour l'infrastructure et de l'ORM Odoo pour la logique métier offre une solution robuste, évolutive et facile à maintenir. Ce projet démontre qu'avec une architecture bien pensée, il est possible de répondre rapidement aux besoins de gestion d'une organisation. Les perspectives d'évolution incluent l'ajout de rapports PDF et l'intégration de tableaux de bord statistiques pour une analyse plus fine des projets.