

# ÉCOLE MAROCAINE DES SCIENCES DE L'INGÉNIEUR



## Rapport de TP : ERP Odoo

*Création du module "Gestion des Annonces"*

**Réalisé par :**

ISSAM MOUNSSIF

**Encadré par :**

MR. MOHAMMED AIT DAOUD

Année Universitaire 2025/2026

# Dédicaces

Je dédie ce travail à mes parents, pour leur soutien inconditionnel et leurs encouragements tout au long de mon parcours académique.

À mes amis et collègues de classe, avec qui j'ai partagé des moments d'étude inoubliables.

À tous ceux qui ont cru en moi.

# Remerciements

Je tiens tout d'abord à remercier mon professeur, MOHAMMED AIT DAOUD, pour la qualité de son enseignement, sa pédagogie et sa disponibilité durant ce module ERP. Ses conseils ont été précieux pour la réalisation de ce projet.

Je remercie également l'ensemble du corps professoral de l'EMSI pour la formation d'excellence qu'ils nous dispensent. Enfin, merci à tous ceux qui ont contribué de près ou de loin à la réussite de ce travail.

# Table des matières

<b>Dédicaces</b>	<b>1</b>
<b>Remerciements</b>	<b>2</b>
<b>1 Introduction Générale</b>	<b>6</b>
<b>2 Contexte et Outils Technologiques</b>	<b>7</b>
2.1 Présentation de l'ERP Odoo . . . . .	7
2.1.1 Architecture . . . . .	7
2.1.2 L'ORM Odoo . . . . .	7
2.2 La Technologie Docker . . . . .	8
2.2.1 Pourquoi Docker ? . . . . .	8
2.2.2 Configuration du Projet . . . . .	8
<b>3 Analyse et Conception</b>	<b>9</b>
3.1 Analyse des Besoins . . . . .	9
3.1.1 Besoins Fonctionnels . . . . .	9
3.1.2 Données à Gérer . . . . .	9
3.2 Conception Technique . . . . .	10
3.2.1 Modélisation . . . . .	10
3.2.2 Structure du Module . . . . .	10
<b>4 Réalisation et Développement</b>	<b>11</b>
4.1 Déclaration du Module ( <code>__manifest__.py</code> ) . . . . .	11
4.2 Implémentation du Modèle ( <code>models/annonce.py</code> ) . . . . .	12
4.3 Configuration de la Sécurité ( <code>ir.model.access.csv</code> ) . . . . .	12
4.4 Définition de l'Interface Utilisateur ( <code>views/annonce_views.xml</code> ) . . . . .	13
4.4.1 L'Action de Fenêtre . . . . .	13
4.4.2 Les Menus . . . . .	13
4.4.3 La Vue Liste (Tree) . . . . .	14

4.4.4	La Vue Formulaire (Form)	14
<b>5</b>	<b>Déploiement et Tests</b>	<b>16</b>
5.1	Procédure de Déploiement	16
5.1.1	Mise à jour du docker-compose	16
5.1.2	Démarrage	16
5.2	Tests Fonctionnels	16
5.2.1	Installation du module	16
5.2.2	Création d'une annonce	17
5.3	Problèmes Rencontrés et Solutions	17
<b>6</b>	<b>Conclusion et Perspectives</b>	<b>18</b>
6.1	Bilan	18
6.2	Perspectives d'Amélioration	18

# Table des figures

4.1	Interface de gestion des annonces . . . . .	15
-----	---	----

f	listings
---	----------

# Chapitre 1

## Introduction Générale

L'intégration des systèmes d'information est devenue un enjeu majeur pour les entreprises modernes. Les ERP (Enterprise Resource Planning) ou PGI (Progiciels de Gestion Intégrés) permettent de centraliser et d'optimiser la gestion de l'ensemble des processus opérationnels d'une organisation, allant de la gestion des ressources humaines à la comptabilité, en passant par les ventes et les stocks. Dans le cadre de notre formation en génie informatique, la maîtrise de ces outils est indispensable. Ce rapport présente le travail réalisé lors du laboratoire pratique consacré au développement de modules sous Odoo, l'un des ERP open-source les plus populaires au monde.

L'objectif principal de ce projet est de concevoir et développer un module personnalisé nommé "Gestion des Annonces". Ce module a pour vocation de permettre la diffusion d'informations, d'alertes ou d'événements au sein d'une organisation fictive.

Ce rapport s'articule autour de quatre grands axes :

- Le premier chapitre présente le cadre général du projet, incluant une présentation de l'ERP Odoo et de la technologie Docker.
- Le deuxième chapitre est consacré à l'analyse et à la conception du module.
- Le troisième chapitre détaille la réalisation technique et le code source.
- Enfin, le quatrième chapitre illustre le fonctionnement du module à travers des tests et un manuel d'utilisation.

# Chapitre 2

## Contexte et Outils Technologiques

### 2.1 Présentation de l'ERP Odoo

Odoo est une suite d'applications de gestion d'entreprise open-source. Anciennement connu sous le nom d'OpenERP (et TinyERP avant cela), il a su s'imposer comme une référence grâce à sa modularité et sa flexibilité.

#### 2.1.1 Architecture

Odoo repose sur une architecture trois tiers :

- **Base de données (PostgreSQL)** : C'est là que sont stockées toutes les données de l'entreprise ainsi que la configuration des modules.
- **Serveur d'application (Python)** : Le cœur d'Odoo, écrit en Python, gère la logique métier, les appels API et la communication avec la base de données via un ORM (Object-Relational Mapping).
- **Client Web (JavaScript/XML)** : L'interface utilisateur est une application web moderne (Single Page Application) qui communique avec le serveur via JSON-RPC.

#### 2.1.2 L'ORM Odoo

L'une des grandes forces d'Odoo est son ORM puissant. Il permet aux développeurs de manipuler des objets Python au lieu d'écrire des requêtes SQL complexes. Par exemple, créer une nouvelle table dans la base de données se fait simplement en définissant une classe Python héritant de `models.Model`.

## 2.2 La Technologie Docker

Pour ce projet, nous avons utilisé Docker afin de faciliter le déploiement et garantir un environnement de développement iso-fonctionnel pour tous les étudiants.

### 2.2.1 Pourquoi Docker ?

Docker est une plateforme permettant de lancer des applications dans des conteneurs logiciels. Contrairement aux machines virtuelles, les conteneurs partagent le noyau du système d'exploitation hôte mais isolent les processus de l'application. Les avantages pour notre projet Odoo sont multiples :

- **Installation rapide** : Pas besoin d'installer Python, PostgreSQL et toutes les dépendances manuellement. Un simple fichier `docker-compose.yml` suffit.
- **Isolation** : On peut faire tourner plusieurs versions d'Odoo sur la même machine sans conflit.
- **Reproductibilité** : Le code fonctionne exactement de la même manière sur la machine de l'étudiant et chez le professeur.

### 2.2.2 Configuration du Projet

Notre environnement se compose de deux services principaux définis dans le fichier `docker-compose.yml` :

1. **db** : Le conteneur de base de données utilisant l'image officielle `postgres:16`.
2. **odoo17** : Le conteneur application utilisant l'image `odoo:17.0`.

Nous avons configuré des volumes pour persister les données de la base PostgreSQL (`odoo-db-data`) et les fichiers de configuration web (`odoo-web-data`). De plus, un volume "bind mount" a été créé pour lier notre dossier local `addons/` au répertoire `/mnt/extra-addons` du conteneur, permettant ainsi de développer notre code localement et de le voir exécuté dans le conteneur.

# Chapitre 3

## Analyse et Conception

### 3.1 Analyse des Besoins

Le besoin exprimé est de disposer d'un outil simple intégré à Odoo pour gérer des annonces internes.

#### 3.1.1 Besoins Fonctionnels

Le système doit permettre de :

1. **Créer une annonce** : Un utilisateur doit pouvoir saisir une nouvelle annonce.
2. **Lister les annonces** : Une vue globale doit présenter l'ensemble des annonces existantes.
3. **Modifier/Supprimer** : La gestion complète du cycle de vie de l'annonce (CRUD).
4. **Catégoriser** : Distinguer les annonces par type (Information, Alerte, Événement).

#### 3.1.2 Données à Gérer

Chaque annonce sera caractérisée par les informations suivantes :

- **Titre** : Un libellé court résumant l'annonce (Obligatoire).
- **Auteur** : Le nom de la personne publiant l'annonce.
- **Date** : La date de publication (par défaut, la date du jour).
- **Type** : Une classification (Info, Alerte, Événement).
- **Description** : Le contenu détaillé de l'annonce.

## 3.2 Conception Technique

### 3.2.1 Modélisation

Dans l'univers Odoo, la conception commence par la définition du Modèle. Nous allons créer un nouveau modèle technique nommé `gestion.annonce`.

**Attributs du modèle :**

Champ	Type Odoo	Détails
name	Char	Le titre, requis.
auteur	Char	Nom de l'auteur.
date__annonce	Date	Date de publication.
type	Selection	Liste déroulante (Info, Alerte, Event).
description	Text	Zone de texte multiligne.

TABLE 3.1 – Dictionnaire de données du modèle `gestion.annonce`

### 3.2.2 Structure du Module

Le module respectera la structure standard d'un add-on Odoo 17 :

```
gestion_annonces/
|-- __init__.py          # Initialiseur du package Python
|-- __manifest__.py      # Métadonnées du module
|-- models/              # Dossier des modèles
|   |-- __init__.py
|   |-- annonce.py       # Définition de la classe Python
|-- security/            # Dossier de sécurité
|   |-- ir.model.access.csv # Droits d'accès (ACL)
|-- views/               # Dossier des vues XML
|   |-- annonce_views.xml # Définition de l'interface
```

# Chapitre 4

## Réalisation et Développement

Dans ce chapitre, nous détaillons le code source produit pour répondre aux besoins.

### 4.1 Déclaration du Module (`__manifest__.py`)

Le fichier manifeste est la carte d'identité du module. Il indique à Odoo le nom, la version, et surtout les fichiers de données (XML, CSV) à charger.

```
1 {
2     "name": "Gestion des Annonces",
3     "version": "1.0",
4     "summary": "Module de gestion des annonces internes",
5     "category": "Generic Modules/Others",
6     "author": "EMSI - Issam MOUNSSIF",
7     "depends": ["base"],
8     "data": [
9         "security/ir.model.access.csv",
10        "views/annonce_views.xml",
11    ],
12    "installable": True,
13    "application": True,
14 }
```

Listing 4.1 – Code source du fichier manifest

## 4.2 Implémentation du Modèle (models/annonce.py)

C'est ici que nous définissons la structure de la base de données.

```
1 from odoo import models, fields
2
3 class GestionAnnonce(models.Model):
4     _name = "gestion.annonce"
5     _description = "Gestion des Annonces"
6
7     # Le champ 'name' est sp cial dans Odoo, il sert de
8     # titre par d faut
9     name = fields.Char(string="Titre", required=True)
10
11     auteur = fields.Char(string="Auteur")
12
13     # On met la date d'aujourd'hui par d faut
14     date_annonce = fields.Date(string="Date de l'annonce",
15                                default=fields.Date.today)
16
17     # Champ de s lection pour le type
18     type = fields.Selection([
19         ("info", "Information"),
20         ("alerte", "Alerte"),
21         ("event", "vnement"),
22     ], string="Type", default="info")
23
24     description = fields.Text(string="Contenu")
```

Listing 4.2 – Définition du modèle GestionAnnonce

Nous importons `models` et `fields` depuis le framework Odoo. La classe hérite de `models.Model`, ce qui signifie qu'elle sera persistée en base de données. Odoo créera automatiquement une table nommée `gestion_annonce`.

## 4.3 Configuration de la Sécurité (ir.model.access.csv)

Odoo possède un système de sécurité strict. Par défaut, personne n'a accès à un nouveau modèle. Nous devons définir des règles d'accès (Access Control List - ACL).

```

1 id,name,model_id:id,group_id:id,perm_read,perm_write,
  perm_create,perm_unlink
2 access_gestion_annonce,gestion.annonce,model_gestion_annonce
  ,,1,1,1,1

```

Listing 4.3 – Fichier CSV de sécurité

Dans cet exemple pédagogique, nous donnons tous les droits (lecture, écriture, création, suppression - les "1") à tout le monde (le champ `group_id` est vide).

## 4.4 Définition de l'Interface Utilisateur (views/annonce\_vie

L'interface d'Odoo est générée dynamiquement à partir de fichiers XML.

### 4.4.1 L'Action de Fenêtre

L'action définit ce qui se passe quand on clique sur un menu. Ici, elle ouvre la vue du modèle `gestion.annonce`.

```

1 <record id="action_gestion_annonce" model="ir.actions.
  act_window">
2   <field name="name">Annonces</field>
3   <field name="res_model">gestion.annonce</field>
4   <field name="view_mode">tree,form</field>
5 </record>

```

### 4.4.2 Les Menus

Nous créons une arborescence de menu pour accéder à notre application.

```

1 <!-- Menu racine -->
2 <menuitem id="menu_gestion_annonce_root" name="Gestion
  Annonces" sequence="10"/>
3
4 <!-- Sous-menu li l'action -->
5 <menuitem id="menu_gestion_annonce"
6   name="Annonces"
7   parent="menu_gestion_annonce_root"
8   action="action_gestion_annonce"
9   sequence="10"/>

```

### 4.4.3 La Vue Liste (Tree)

La vue liste définit les colonnes visibles dans le tableau principal.

```

1 <record id="view_gestion_annonce_tree" model="ir.ui.view">
2   <field name="name">gestion.annonce.tree</field>
3   <field name="model">gestion.annonce</field>
4   <field name="arch" type="xml">
5     <tree>
6       <field name="name"/>
7       <field name="auteur"/>
8       <field name="date_annonce"/>
9       <field name="type"/>
10    </tree>
11  </field>
12 </record>

```

### 4.4.4 La Vue Formulaire (Form)

La vue formulaire définit l'agencement des champs pour la création et l'édition. Nous avons utilisé des balises `<sheet>` et `<group>` pour organiser la mise en page en deux colonnes.

```

1 <record id="view_gestion_annonce_form" model="ir.ui.view">
2   <field name="name">gestion.annonce.form</field>
3   <field name="model">gestion.annonce</field>
4   <field name="arch" type="xml">
5     <form>
6       <sheet>
7         <group>
8           <field name="name"/>
9           <field name="auteur"/>
10          <field name="date_annonce"/>
11          <field name="type"/>
12        </group>
13        <group>
14          <field name="description"/>
15        </group>
16      </sheet>
17    </form>
18  </field>

```

```
19 </record>
```

### Aperçu de l'interface :

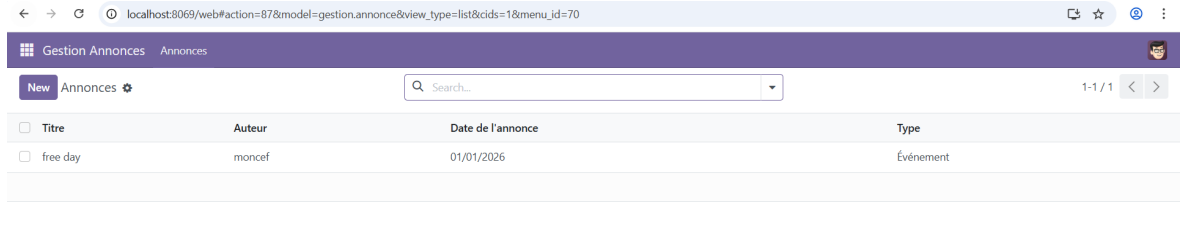


FIGURE 4.1 – Interface de gestion des annonces

*Description : La figure ci-dessus illustre la vue liste finale du module. Elle permet aux utilisateurs de consulter l'ensemble des annonces déjà saisies, avec leurs informations principales (Titre, Auteur, Date, Type). C'est depuis cette vue que l'on peut accéder au détail d'une annonce ou en créer une nouvelle.*

# Chapitre 5

## Déploiement et Tests

### 5.1 Procédure de Déploiement

Le déploiement du module se fait via Docker.

#### 5.1.1 Mise à jour du docker-compose

Une étape cruciale a été de modifier le fichier de composition pour que le conteneur prenne en compte notre nouveau module au démarrage et recharge les fichiers XML à la volée. Commande utilisée :

```
odoo -d odoo_db -u gestion_annonces --dev=xml
```

- `-u gestion_annonces` : Force la mise à jour (installation) du module.
- `-dev=xml` : Permet de modifier les vues XML sans redémarrer le serveur à chaque fois.

#### 5.1.2 Démarrage

La commande `docker compose restart odoo_app` permet de relancer le service et d'appliquer les changements.

### 5.2 Tests Fonctionnels

#### 5.2.1 Installation du module

Lors de la première connexion à l'interface Odoo (localhost :8069), nous nous sommes rendus dans le menu "Apps". Après avoir supprimé le filtre par défaut,

nous avons recherché "Gestion des Annonces". Le module est apparu et a été marqué comme installé.

### 5.2.2 Création d'une annonce

Nous avons cliqué sur le nouveau menu "Gestion Annonces". Le système affiche une liste vide. En cliquant sur "Nouveau", le formulaire de création s'ouvre.

1. Nous avons saisi "Réunion Générale" dans le titre.
2. Auteur : "Issam".
3. Type : "Événement".
4. Date : La date du jour s'est affichée automatiquement.

Après avoir cliqué sur le nuage (Sauvegarde manuelle) ou simplement en revenant à la liste, l'enregistrement a bien été sauvegardé en base de données.

## 5.3 Problèmes Rencontrés et Solutions

Durant ce TP, nous avons pu rencontrer quelques difficultés mineures :

- **Erreur de syntaxe XML** : Odoo est très strict sur le XML. Une balise mal fermée empêche le serveur de démarrer. La lecture des logs (`docker logs`) a permis d'identifier rapidement la ligne fautive.
- **Problème de droits** : Au début, le module ne s'affichait pas car nous avons oublié d'ajouter le fichier `ir.model.access.csv` dans la liste `data` du manifeste.

# Chapitre 6

## Conclusion et Perspectives

Ce travail pratique a été une excellente opportunité de plonger dans le monde du développement ERP avec Odoo.

### 6.1 Bilan

Nous avons réussi à :

- Mettre en place un environnement de développement professionnel avec Docker.
- Comprendre l'architecture modulaire d'Odoo.
- Créer un module complet de bout en bout (Back-end Python et Front-end XML).
- Appréhender le mécanisme de l'ORM et des vues.

Ce projet, bien que simple, pose les fondations nécessaires pour des développements plus complexes. Odoo s'avère être un framework extrêmement productif une fois la courbe d'apprentissage initiale passée.

### 6.2 Perspectives d'Amélioration

Le module "Gestion des Annonces" pourrait être amélioré de nombreuses façons pour devenir un véritable outil de communication interne :

1. **Workflow de validation** : Ajouter un statut "Brouillon", "À valider", "Publié" avec une barre d'état (`statusbar`).
2. **Notifications** : Envoyer un email automatique ou une notification interne Odoo aux employés lorsqu'une annonce de type "Alerte" est publiée.

3. **Droits d'accès avancés :** Restreindre la création d'annonces aux seuls gestionnaires RH, les autres employés n'ayant qu'un accès en lecture seule.
4. **Vue Calendrier :** Pour les annonces de type "Événement", une vue calendrier serait très pertinente.
5. **Tableau de bord :** Des graphiques statistiques sur le nombre d'annonces par auteur ou par type.

# Bibliographie

- [1] Documentation Officielle Odoo 17,  
<https://www.odoo.com/documentation/17.0/developer.html>
- [2] Documentation Docker,  
<https://docs.docker.com/>
- [3] Documentation Python 3,  
<https://docs.python.org/3/>
- [4] Support de cours ERP Odoo, M. AIT DAOUD, EMSI, 2025.