

Rapport de Projet

Module Odoo 17

Gestion des Attestations

*Développement d'un Module Personnalisé
pour la Gestion Pédagogique*

Étudiant : Mounib Youssef

Version Odoo : 17.0

Date : 31 décembre 2025

Projet Pédagogique
Formation en Développement Odoo

Table des matières

Résumé Exécutif	4
1 Introduction	5
1.1 Contexte du Projet	5
1.2 Objectifs	5
1.3 Technologies Utilisées	5
2 Architecture du Projet	6
2.1 Architecture Globale	6
2.2 Structure du Projet	6
2.3 Architecture Docker	6
2.3.1 Configuration Docker Compose	6
2.3.2 Configuration Odoo	7
3 Développement du Module	8
3.1 Structure du Module	8
3.1.1 Fichier Manifeste	8
3.1.2 Initialisation du Module	8
3.2 Modèle de Données	9
3.2.1 Définition du Modèle	9
3.2.2 Champs du Modèle	9
3.2.3 Initialisation des Modèles	9
3.3 Vues et Interface Utilisateur	9
3.3.1 Action de Fenêtre	9
3.3.2 Structure des Menus	10
3.3.3 Vue Liste (Tree)	10
3.3.4 Vue Formulaire	10
3.4 Sécurité et Droits d'Accès	11
4 Déploiement et Configuration	12
4.1 Prérequis	12
4.2 Installation	12
4.2.1 Étape 1 : Cloner/Créer le Projet	12
4.2.2 Étape 2 : Configuration Docker Desktop	12
4.2.3 Étape 3 : Démarrage des Conteneurs	12
4.2.4 Étape 4 : Vérification du Montage des Volumes	13
4.3 Installation du Module dans Odoo	13
4.3.1 Accès à l'Interface	13
4.3.2 Activation du Mode Développeur	13
4.3.3 Mise à Jour de la Liste des Applications	13

4.3.4	Installation du Module	13
5	Résolution de Problèmes	14
5.1	Problème : Module Non Visible	14
5.1.1	Symptôme	14
5.1.2	Causes Possibles et Solutions	14
5.2	Problème : Erreurs au Démarrage	15
5.2.1	Consultation des Logs	15
5.2.2	Erreurs Courantes	15
5.3	Commandes Utiles	15
5.3.1	Gestion des Conteneurs	15
5.3.2	Inspection et Débogage	16
5.3.3	Gestion de la Base de Données	16
6	Tests et Validation	17
6.1	Tests Fonctionnels	17
6.1.1	Test 1 : Création d'une Attestation	17
6.1.2	Test 2 : Modification d'une Attestation	17
6.1.3	Test 3 : Suppression d'une Attestation	17
6.2	Tests Techniques	17
6.2.1	Vérification de l'Intégrité du Module	17
6.2.2	Vérification de la Base de Données	18
7	Améliorations Futures	19
7.1	Fonctionnalités Proposées	19
7.1.1	Génération de PDF	19
7.1.2	Workflow de Validation	19
7.1.3	Notifications par Email	19
7.1.4	Historique des Modifications	19
7.1.5	Recherche Avancée	19
7.2	Optimisations Techniques	19
7.2.1	Performance	19
7.2.2	Sécurité	20
7.2.3	Interface Utilisateur	20
8	Conclusion	21
8.1	Résultats Obtenus	21
8.2	Compétences Acquises	21
8.2.1	Techniques	21
8.2.2	Méthodologiques	21
8.3	Perspectives	21
8.4	Remerciements	22
A	Annexes	23
A.1	Code Complet du Module	23
A.1.1	Fichier __manifest__.py	23
A.1.2	Fichier models/attestation.py	23
A.1.3	Fichier views/attestation_views.xml	24
A.2	Commandes de Référence	25

A.2.1	Docker	25
A.2.2	Odoo	25
A.3	Ressources et Références	25
A.3.1	Documentation Officielle	25
A.3.2	Tutoriels et Guides	26
A.4	Glossaire	26

Résumé Exécutif

Ce rapport présente le développement complet d'un module personnalisé pour Odoo 17, destiné à la gestion des attestations dans un contexte pédagogique. Le projet a été réalisé en utilisant Docker pour l'environnement de développement, permettant une isolation complète et une portabilité maximale.

Le module **Attestations** permet de créer, gérer et suivre des attestations pour des étudiants, avec un système de statuts (brouillon, validée, annulée) et une interface utilisateur intuitive basée sur les standards Odoo.

Objectifs atteints :

- Création d'un module Odoo fonctionnel et installable
- Mise en place d'un environnement Docker complet (Odoo 17 + PostgreSQL 16)
- Implémentation d'un modèle de données avec champs personnalisés
- Développement d'interfaces utilisateur (vues liste et formulaire)
- Configuration des droits d'accès et sécurité
- Résolution de problèmes de déploiement et configuration

Chapitre 1

Introduction

1.1 Contexte du Projet

Dans le cadre de la formation en développement Odoo, ce projet vise à créer un module personnalisé pour la gestion des attestations. Les attestations sont des documents administratifs importants dans le contexte éducatif, nécessitant un système de gestion structuré et fiable.

1.2 Objectifs

Les objectifs principaux de ce projet sont :

1. **Apprentissage** : Maîtriser les concepts fondamentaux du développement de modules Odoo
2. **Pratique** : Mettre en œuvre les meilleures pratiques de développement
3. **Déploiement** : Utiliser Docker pour créer un environnement de développement reproductible
4. **Fonctionnalité** : Créer un module opérationnel répondant à un besoin réel

1.3 Technologies Utilisées

- **Odoo 17.0** : Framework ERP open-source
- **Python 3** : Langage de programmation pour la logique métier
- **PostgreSQL 16** : Système de gestion de base de données
- **Docker & Docker Compose** : Conteneurisation et orchestration
- **XML** : Définition des vues et interfaces utilisateur
- **Windows 11** : Système d'exploitation de développement

Chapitre 2

Architecture du Projet

2.1 Architecture Globale

Le projet utilise une architecture conteneurisée basée sur Docker, comprenant deux services principaux :

- **Service Odoo** : Conteneur exécutant Odoo 17.0
- **Service PostgreSQL** : Conteneur de base de données PostgreSQL 16

2.2 Structure du Projet

Listing 2.1 – Arborescence du projet

```
C:\Users\OMEN\odoo\  
  docker-compose.yml          # Configuration Docker  
  config/  
    odoo.conf                 # Configuration Odoo  
  addons/  
    attestations/             # Module personnalis  
      __init__.py  
      __manifest__.py  
    models/  
      __init__.py  
      attestation.py  
    views/  
      attestation_views.xml  
    security/  
      ir.model.access.csv  
  DOCKER_VOLUME_FIX.md       # Documentation
```

2.3 Architecture Docker

2.3.1 Configuration Docker Compose

Le fichier `docker-compose.yml` définit l'infrastructure :

Listing 2.2 – `docker-compose.yml`

```
version: "3.8"

services:
  db:
    image: postgres:16
    container_name: odoo_db
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: odoo
      POSTGRES_PASSWORD: odoo
      PGDATA: /var/lib/postgresql/data/pgdata
    volumes:
      - odoo-db-data:/var/lib/postgresql/data

  odoo17:
    image: odoo:17.0
    container_name: odoo_app
    depends_on:
      - db
    ports:
      - "8069:8069"
    volumes:
      - odoo-web-data:/var/lib/odoo
      - ./config:/etc/odoo
      - ./addons:/mnt/extra-addons
    command: >
      odoo -d odoo_db
      -i base
      --db_user=odoo
      --db_password=odoo
      --db_host=db

volumes:
  odoo-web-data:
  odoo-db-data:
```

2.3.2 Configuration Odoo

Le fichier `odoo.conf` configure les chemins d'addons :

Listing 2.3 – config/odoo.conf

```
[options]
addons_path = /usr/lib/python3/dist-packages/odoo/addons,/mnt/extra
             -addons
data_dir = /var/lib/odoo
```

Note importante : La syntaxe correcte est `addons_path` (avec underscore) et non `addons path` (avec espace). Cette erreur de configuration était la cause principale du problème de non-détection du module.

Chapitre 3

Développement du Module

3.1 Structure du Module

3.1.1 Fichier Manifeste

Le fichier `__manifest__.py` définit les métadonnées du module :

```
1 {  
2     "name": "Attestations",  
3     "version": "1.0",  
4     "summary": "Module pédagogique pour gérer des attestations",  
5     "category": "Training",  
6     "author": "EMSI",  
7     "depends": ["base"],  
8     "data": [  
9         "security/ir.model.access.csv",  
10        "views/attestation_views.xml",  
11    ],  
12    "installable": True,  
13    "application": True,  
14 }
```

Listing 3.1 – addons/attestations/`__manifest__.py`

Éléments clés :

- `depends` : Liste des modules requis (ici uniquement `base`)
- `data` : Fichiers de données à charger (sécurité et vues)
- `installable` : Module installable
- `application` : Module affiché comme application principale

3.1.2 Initialisation du Module

```
1 from . import models
```

Listing 3.2 – addons/attestations/`__init__.py`

3.2 Modèle de Données

3.2.1 Définition du Modèle

Le modèle `tp.attestation` représente une attestation :

```

1 from odoo import models, fields
2
3 class TpAttestation(models.Model):
4     _name = "tp.attestation"
5     _description = "Attestation"
6
7     name = fields.Char(string="Nom de l'attestation", required=True
8                        )
9     etudiant = fields.Char(string="Nom de l'etudiant ")
10    date_creation = fields.Date(string="Date de creation")
11    statut = fields.Selection([
12        ("brouillon", "Brouillon"),
13        ("valide", "Valid e"),
14        ("annulee", "Annul e"),
15    ], string="Statut", default="brouillon")
16    description = fields.Text(string="Description")

```

Listing 3.3 – `addons/attestations/models/attestation.py`

3.2.2 Champs du Modèle

Champ	Type	Requis	Description
name	Char	Oui	Nom de l'attestation
etudiant	Char	Non	Nom de l'étudiant concerné
date_creation	Date	Non	Date de création de l'attestation
statut	Selection	Non	État de l'attestation (brouillon par défaut)
description	Text	Non	Description détaillée

TABLE 3.1 – Champs du modèle `tp.attestation`

3.2.3 Initialisation des Modèles

```

1 from . import attestation

```

Listing 3.4 – `addons/attestations/models/__init__.py`

3.3 Vues et Interface Utilisateur

3.3.1 Action de Fenêtre

```

1 <record id="action_tp_attestation" model="ir.actions.act_window">
2   <field name="name">Attestations</field>
3   <field name="res_model">tp.attestation</field>
4   <field name="view_mode">tree,form</field>
5 </record>

```

Listing 3.5 – Définition de l'action

3.3.2 Structure des Menus

```

1 <!-- Menu principal -->
2 <menuitem id="menu_tp_root" name="Attestations" sequence="10"/>
3
4 <!-- Sous-menu -->
5 <menuitem id="menu_tp_attestations"
6   name="Gestion des Attestations"
7   parent="menu_tp_root"
8   action="action_tp_attestation"
9   sequence="10"/>

```

Listing 3.6 – Hiérarchie des menus

3.3.3 Vue Liste (Tree)

```

1 <record id="view_tp_attestation_tree" model="ir.ui.view">
2   <field name="name">tp.attestation.tree</field>
3   <field name="model">tp.attestation</field>
4   <field name="arch" type="xml">
5     <tree>
6       <field name="name"/>
7       <field name="etudiant"/>
8       <field name="date_creation"/>
9       <field name="statut"/>
10    </tree>
11  </field>
12 </record>

```

Listing 3.7 – Vue liste des attestations

3.3.4 Vue Formulaire

```

1 <record id="view_tp_attestation_form" model="ir.ui.view">
2   <field name="name">tp.attestation.form</field>
3   <field name="model">tp.attestation</field>
4   <field name="arch" type="xml">
5     <form>
6       <sheet>
7         <group>

```

```
8         <field name="name"/>
9         <field name="etudiant"/>
10        <field name="date_creation"/>
11        <field name="statut"/>
12    </group>
13    <group>
14        <field name="description"/>
15    </group>
16 </sheet>
17 </form>
18 </field>
19 </record>
```

Listing 3.8 – Vue formulaire d’attestation

3.4 Sécurité et Droits d’Accès

Le fichier `ir.model.access.csv` définit les permissions :

Listing 3.9 – `addons/attestations/security/ir.model.access.csv`

```
id ,name ,model_id:id ,group_id:id ,perm_read ,perm_write ,perm_create ,perm_unlink
access_tp_attestation ,tp . attestation , model_tp_attestation , , 1 , 1 , 1 , 1
```

Permissions accordées :

- `perm_read` : Lecture (1 = autorisé)
- `perm_write` : Modification (1 = autorisé)
- `perm_create` : Création (1 = autorisé)
- `perm_unlink` : Suppression (1 = autorisé)

Chapitre 4

Déploiement et Configuration

4.1 Prérequis

- Docker Desktop installé et configuré
- WSL2 activé (pour Windows)
- Au moins 4 GB de RAM disponible
- Ports 8069 et 5432 disponibles

4.2 Installation

4.2.1 Étape 1 : Cloner/Créer le Projet

```
# Créer le répertoire du projet
mkdir C:\Users\OMEN\odoo
cd C:\Users\OMEN\odoo

# Créer la structure
mkdir config addons
```

4.2.2 Étape 2 : Configuration Docker Desktop

Important pour Windows : Docker Desktop nécessite un partage de fichiers explicite.

1. Ouvrir Docker Desktop
2. Aller dans Settings → Resources → File Sharing
3. Ajouter le chemin : C:\Users\OMEN\odoo
4. Cliquer sur "Apply & Restart"

4.2.3 Étape 3 : Démarrage des Conteneurs

```
# Démarrer les services
docker-compose up -d

# Vérifier l'état des conteneurs
```

```
docker ps

# Vérifier les logs
docker logs odoo_app --tail 50
```

4.2.4 Étape 4 : Vérification du Montage des Volumes

```
# Vérifier que le module est visible dans le conteneur
docker exec odoo_app ls -la /mnt/extra-addons

# Devrait afficher le dossier "attestations"
```

4.3 Installation du Module dans Odoo

4.3.1 Accès à l'Interface

1. Ouvrir un navigateur
2. Accéder à : <http://localhost:8069>
3. Se connecter avec les identifiants par défaut

4.3.2 Activation du Mode Développeur

1. Aller dans **Paramètres**
2. Descendre en bas de la page
3. Cliquer sur **Activer le mode développeur**
4. Ou ajouter `?debug=1` à l'URL

4.3.3 Mise à Jour de la Liste des Applications

1. Aller dans le menu **Applications**
2. Cliquer sur le menu (trois points)
3. Sélectionner **Mettre à jour la liste des applications**
4. Cliquer sur **Mettre à jour**

4.3.4 Installation du Module

1. Dans la barre de recherche, supprimer le filtre "Applications"
2. Rechercher "attestation" ou "TP"
3. Le module "Attestations" devrait apparaître
4. Cliquer sur **Installer**

Chapitre 5

Résolution de Problèmes

5.1 Problème : Module Non Visible

5.1.1 Symptôme

Le module n'apparaît pas dans la liste des applications Odoo.

5.1.2 Causes Possibles et Solutions

Cause 1 : Erreur de Configuration addons_path

Problème : Le fichier `odoo.conf` contient `addons path` au lieu de `addons_path`.

Solution :

```
# Corriger dans config/odoo.conf
[options]
addons_path = /usr/lib/python3/dist-packages/odoo/addons,/mnt/extra
             -addons
data_dir = /var/lib/odoo
```

Cause 2 : Volume Docker Non Monté

Problème : Le dossier `addons` n'est pas accessible dans le conteneur.

Diagnostic :

```
docker exec odoo_app ls -la /mnt/extra-addons
# Si vide ou erreur, le volume n'est pas mont
```

Solution :

1. Vérifier le partage de fichiers dans Docker Desktop
2. Utiliser des chemins relatifs dans `docker-compose.yml`
3. Redémarrer les conteneurs : `docker-compose down && docker-compose up -d`

Cause 3 : Erreur dans le Manifeste

Problème : Le fichier `__manifest__.py` contient des erreurs de syntaxe.

Solution : Vérifier la syntaxe Python et la structure du dictionnaire.

Cause 4 : Permissions de Fichiers

Problème : Les fichiers ne sont pas lisibles par le conteneur.

Solution :

```
# Vérifier les permissions dans le conteneur
docker exec odoo_app ls -la /mnt/extra-addons/attestations
```

5.2 Problème : Erreurs au Démarrage

5.2.1 Consultation des Logs

```
# Logs complets
docker logs odoo_app

# Dernières 100 lignes
docker logs odoo_app --tail 100

# Suivre les logs en temps réel
docker logs odoo_app -f
```

5.2.2 Erreurs Courantes

Erreur	Solution
Missing 'license' key	Ajouter "license": "LGPL-3" dans __manifest__.py
Module not found	Vérifier addons_path dans odoo.conf
Database connection error	Vérifier que le conteneur PostgreSQL est démarré
Permission denied	Vérifier les droits d'accès aux fichiers

TABLE 5.1 – Erreurs courantes et solutions

5.3 Commandes Utiles

5.3.1 Gestion des Conteneurs

```
# Démarrer les services
docker-compose up -d

# Arrêter les services
docker-compose down

# Redémarrer un conteneur spécifique
docker restart odoo_app
```



```
# Voir l'état des conteneurs
docker ps

# Voir tous les conteneurs (même arrêtés)
docker ps -a
```

5.3.2 Inspection et Débogage

```
# Accéder au shell du conteneur Odoo
docker exec -it odoo_app /bin/bash

# Vérifier la configuration Odoo
docker exec odoo_app cat /etc/odoo/odoo.conf

# Lister les modules disponibles
docker exec odoo_app ls -la /mnt/extra-addons

# Vérifier les processus en cours
docker exec odoo_app ps aux
```

5.3.3 Gestion de la Base de Données

```
# Accéder à PostgreSQL
docker exec -it odoo_db psql -U odoo

# Lister les bases de données
docker exec odoo_db psql -U odoo -c "\l"

# Sauvegarder la base de données
docker exec odoo_db pg_dump -U odoo odoo_db > backup.sql
```

Chapitre 6

Tests et Validation

6.1 Tests Fonctionnels

6.1.1 Test 1 : Création d'une Attestation

1. Accéder au menu **Attestations** → **Gestion des Attestations**
2. Cliquer sur **Créer**
3. Remplir les champs :
 - Nom de l'attestation : "Attestation de Présence"
 - Nom de l'étudiant : "Jean Dupont"
 - Date de création : Date du jour
 - Statut : "Brouillon"
 - Description : "Attestation de présence au cours"
4. Cliquer sur **Enregistrer**
5. Vérifier que l'attestation apparaît dans la liste

6.1.2 Test 2 : Modification d'une Attestation

1. Ouvrir une attestation existante
2. Modifier le statut en "Validée"
3. Enregistrer
4. Vérifier que le changement est persisté

6.1.3 Test 3 : Suppression d'une Attestation

1. Sélectionner une attestation
2. Cliquer sur **Action** → **Supprimer**
3. Confirmer la suppression
4. Vérifier que l'attestation n'apparaît plus dans la liste

6.2 Tests Techniques

6.2.1 Vérification de l'Intégrité du Module

```
# Vérifier que le module est chargé
docker logs odoo_app | grep attestations

# Devrait afficher une ligne mentionnant le module
```

6.2.2 Vérification de la Base de Données

```
# Vérifier que la table est créée
docker exec odoo_db psql -U odoo -d odoo_db -c "\dt tp_attestation"
```

Chapitre 7

Améliorations Futures

7.1 Fonctionnalités Proposées

7.1.1 Génération de PDF

Ajouter la possibilité de générer des attestations au format PDF avec un template personnalisé.

7.1.2 Workflow de Validation

Implémenter un système de validation multi-niveaux :

- Soumission par l'étudiant
- Validation par le professeur
- Approbation finale par l'administration

7.1.3 Notifications par Email

Envoyer des notifications automatiques lors des changements de statut.

7.1.4 Historique des Modifications

Tracer toutes les modifications apportées aux attestations.

7.1.5 Recherche Avancée

Ajouter des filtres et des recherches avancées :

- Par période
- Par étudiant
- Par statut
- Par type d'attestation

7.2 Optimisations Techniques

7.2.1 Performance

- Ajouter des index sur les champs fréquemment recherchés

- Implémenter la pagination pour les grandes listes
- Optimiser les requêtes SQL

7.2.2 Sécurité

- Implémenter des groupes d'utilisateurs spécifiques
- Ajouter des règles d'enregistrement (record rules)
- Mettre en place un système d'audit

7.2.3 Interface Utilisateur

- Ajouter une vue Kanban
- Créer un tableau de bord avec statistiques
- Améliorer le design des formulaires
- Ajouter des graphiques et rapports

Chapitre 8

Conclusion

8.1 Résultats Obtenus

Ce projet a permis de développer avec succès un module Odoo fonctionnel pour la gestion des attestations. Les objectifs principaux ont été atteints :

- Module installable et opérationnel
- Interface utilisateur intuitive
- Gestion complète CRUD (Create, Read, Update, Delete)
- Environnement Docker configuré et fonctionnel
- Documentation complète du processus

8.2 Compétences Acquises

8.2.1 Techniques

- Développement de modules Odoo (architecture MVC)
- Programmation Python orientée objet avec l'ORM Odoo
- Création de vues XML pour interfaces utilisateur
- Configuration et gestion de Docker
- Administration PostgreSQL
- Résolution de problèmes de déploiement

8.2.2 Méthodologiques

- Analyse et conception de solutions
- Débogage et résolution de problèmes
- Documentation technique
- Gestion de projet de développement

8.3 Perspectives

Ce module constitue une base solide pour des développements futurs. Les améliorations proposées permettraient de transformer ce prototype pédagogique en une solution de production complète.

Le projet démontre également l'importance d'une bonne configuration de l'environnement de développement et de la documentation des problèmes rencontrés et de leurs solutions.

8.4 Remerciements

Ce projet a été réalisé dans le cadre de la formation EMSI en développement Odoo. Il représente une application pratique des concepts théoriques enseignés et une préparation aux défis réels du développement d'applications d'entreprise.

Annexe A

Annexes

A.1 Code Complet du Module

A.1.1 Fichier `__manifest__.py`

```
1 {
2     "name": "Attestations",
3     "version": "1.0",
4     "summary": "Module pédagogique pour gérer des attestations",
5     "category": "Training",
6     "author": "EMSI",
7     "depends": ["base"],
8     "data": [
9         "security/ir.model.access.csv",
10        "views/attestation_views.xml",
11    ],
12    "installable": True,
13    "application": True,
14    "license": "LGPL-3",
15 }
```

A.1.2 Fichier `models/attestation.py`

```
1 from odoo import models, fields
2
3 class TpAttestation(models.Model):
4     _name = "tp.attestation"
5     _description = "Attestation"
6
7     name = fields.Char(string="Nom de l'attestation", required=True)
8
9     etudiant = fields.Char(string="Nom de l'étudiant")
10    date_creation = fields.Date(string="Date de création")
11    statut = fields.Selection([
12        ("brouillon", "Brouillon"),
13        ("valide", "Validée"),
14        ("annulee", "Annulée"),
15    ],
```



```

14 ], string="Statut", default="brouillon")
15 description = fields.Text(string="Description")

```

A.1.3 Fichier views/attestation_views.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <odoo>
3     <!-- Action -->
4     <record id="action_tp_attestation" model="ir.actions.act_window"
5         ">
6         <field name="name">Attestations</field>
7         <field name="res_model">tp.attestation</field>
8         <field name="view_mode">tree,form</field>
9     </record>
10
11     <!-- Menu principal -->
12     <menuitem id="menu_tp_root" name="Attestations" sequence="10"/>
13
14     <!-- Sous-menu -->
15     <menuitem id="menu_tp_attestations"
16         name="Gestion des Attestations"
17         parent="menu_tp_root"
18         action="action_tp_attestation"
19         sequence="10"/>
20
21     <!-- Vue liste -->
22     <record id="view_tp_attestation_tree" model="ir.ui.view">
23         <field name="name">tp.attestation.tree</field>
24         <field name="model">tp.attestation</field>
25         <field name="arch" type="xml">
26             <tree>
27                 <field name="name"/>
28                 <field name="etudiant"/>
29                 <field name="date_creation"/>
30                 <field name="statut"/>
31             </tree>
32         </field>
33     </record>
34
35     <!-- Vue formulaire -->
36     <record id="view_tp_attestation_form" model="ir.ui.view">
37         <field name="name">tp.attestation.form</field>
38         <field name="model">tp.attestation</field>
39         <field name="arch" type="xml">
40             <form>
41                 <sheet>
42                     <group>
43                         <field name="name"/>
44                         <field name="etudiant"/>
45                         <field name="date_creation"/>

```

```
45         <field name="statut"/>
46     </group>
47     <group>
48         <field name="description"/>
49     </group>
50 </sheet>
51 </form>
52 </field>
53 </record>
54 </odoo>
```

A.2 Commandes de Référence

A.2.1 Docker

```
# D marrer
docker-compose up -d

# Arr ter
docker-compose down

# Red marrer
docker-compose restart

# Logs
docker logs odoo_app -f

# Shell
docker exec -it odoo_app /bin/bash
```

A.2.2 Odoo

```
# Mettre jour un module
docker exec odoo_app odoo -u attestations -d odoo_db

# Installer un module
docker exec odoo_app odoo -i attestations -d odoo_db

# Mode shell Odoo
docker exec -it odoo_app odoo shell -d odoo_db
```

A.3 Ressources et Références

A.3.1 Documentation Officielle

— Odoo Documentation : <https://www.odoo.com/documentation/17.0/>

- Docker Documentation : <https://docs.docker.com/>
- PostgreSQL Documentation : <https://www.postgresql.org/docs/>

A.3.2 Tutoriels et Guides

- Odoo Development Tutorials
- Docker Compose Guide
- Python ORM Best Practices

A.4 Glossaire

Odoo Framework ERP (Enterprise Resource Planning) open-source

Module Extension fonctionnelle d'Odoo

ORM Object-Relational Mapping - Mapping objet-relationnel

Docker Plateforme de conteneurisation

PostgreSQL Système de gestion de base de données relationnelle

Manifest Fichier de déclaration d'un module Odoo

View Interface utilisateur dans Odoo

Model Représentation d'une table de base de données

Record Rule Règle de sécurité au niveau des enregistrements

ACL Access Control List - Liste de contrôle d'accès