



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

Filière : 5ème année en Ingénierie Informatique et Réseaux

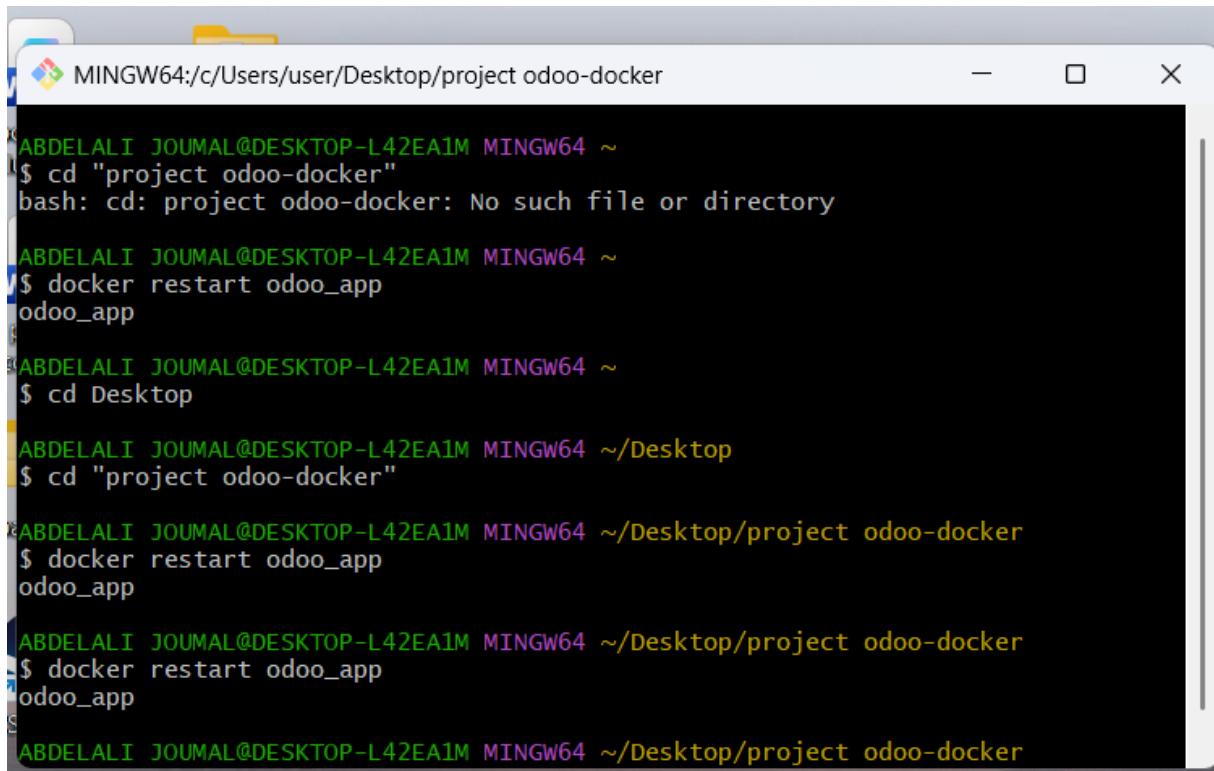
PROJET :

Gestion des Animaux et des Vaccins

Réaliser par : AMINE ALLAMI

Année universitaire : 2025/2026

Git Bash



```
MINGW64:/c/Users/user/Desktop/project odoo-docker
$ cd "project odoo-docker"
bash: cd: project odoo-docker: No such file or directory

ABDELALI JOURNAL@DESKTOP-L42EA1M MINGW64 ~
$ docker restart odoo_app
odoo_app

ABDELALI JOURNAL@DESKTOP-L42EA1M MINGW64 ~
$ cd Desktop

ABDELALI JOURNAL@DESKTOP-L42EA1M MINGW64 ~/Desktop
$ cd "project odoo-docker"

ABDELALI JOURNAL@DESKTOP-L42EA1M MINGW64 ~/Desktop/project odoo-docker
$ docker restart odoo_app
odoo_app

ABDELALI JOURNAL@DESKTOP-L42EA1M MINGW64 ~/Desktop/project odoo-docker
$ docker restart odoo_app
odoo_app

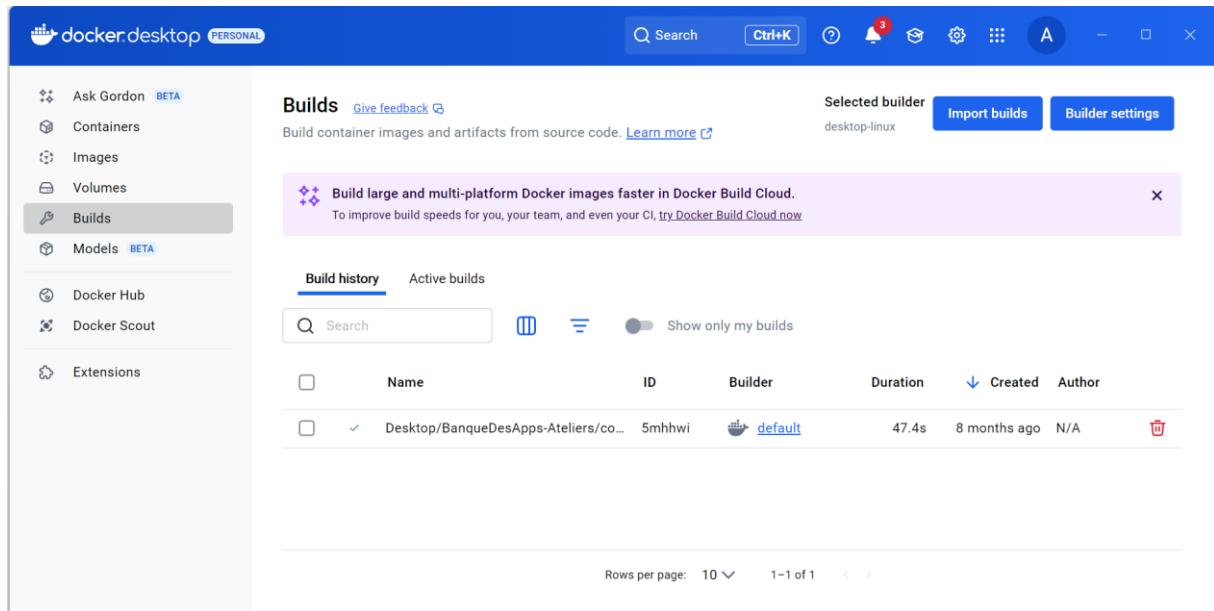
ABDELALI JOURNAL@DESKTOP-L42EA1M MINGW64 ~/Desktop/project odoo-docker
```

Redémarrage du conteneur Odoo via Git Bash

Cette capture d'écran illustre l'utilisation de **Git Bash sous Windows** pour gérer l'environnement Docker du projet Odoo dédié à la gestion des animaux et des vaccins. L'utilisateur commence par se positionner dans le répertoire du projet Odoo hébergé localement, après correction du chemin d'accès initialement incorrect. Une fois dans le bon dossier, la commande :

```
docker restart odoo_app
```

Cette opération est essentielle après toute modification apportée au module personnalisé, notamment au niveau des modèles Python ou des vues XML. Le message de confirmation retourné par Docker atteste du redémarrage réussi du conteneur **odoo_app**, garantissant ainsi la prise en compte des changements effectués.



Supervision des conteneurs Docker via Docker Desktop

l'environnement Docker du projet. La section *Containers* affiche les conteneurs actifs, notamment **odoo_app** et **odoo_db**, qui constituent respectivement l'application Odoo et la base de données PostgreSQL.

Les indicateurs de consommation des ressources système (CPU et mémoire) montrent une utilisation modérée, témoignant d'un fonctionnement stable et optimisé de l'application.

Docker Desktop offre également des fonctionnalités de gestion centralisée telles que le démarrage, l'arrêt et le redémarrage des conteneurs, facilitant ainsi l'administration et la maintenance de l'environnement Odoo utilisé pour le module *Gestion des Animaux et des Vacci*.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** view: Shows the project structure under "ODOO-DOCKER". The "addons\tp_animaux_vaccins" folder contains subfolders like "__pycache__", "models", "security", and "views", along with XML files "ir.model.access.csv", "animal_views.xml", and "vaccin_views.xml", and Python files "__init__.py" and "_manifest.py".
- EDITOR** view: Displays the content of the "_manifest.py" file. The code is as follows:

```
You, 1 hour ago | 1 author (You)
You, 1 hour ago * Initial commit: Module Odoo Gestion Animaux et ...
1 {
2     'name': 'Gestion Animaux et Vaccins',
3     'version': '1.0',
4     'summary': 'Enregistrer les animaux et leurs vaccins',
5     'category': 'Education',
6     'author': 'Nom Etudiant',
7     'depends': ['base'],
8     'data': [
9         'security/ir.model.access.csv',
10        'views/animal_views.xml',
11        'views/vaccin_views.xml',
12    ],
13    'installable': True,
14    'application': True,
15 }
16
```

- TERMINAL** view: Shows the command "PS C:\Users\AMINE ALLAMI\Downloads\odoo-docker>" indicating the current working directory.

Structure du module Odoo et exécution des conteneurs Docker dans Visual Studio Code

Code, combiné à Docker. Dans l'explorateur de fichiers, on observe la structure du module personnalisé **tp_gestion_animaux_vaccins**, organisée conformément aux bonnes pratiques de développement Odoo.

Le module comprend notamment les répertoires *models*, *views* et *security*, garantissant une séparation claire des responsabilités.

La partie inférieure de l'interface affiche le terminal intégré confirmant l'exécution des conteneurs Docker à l'aide de la commande docker ps. Les ports standards sont exposés, notamment le port **8069** pour l'accès à l'interface web d'Odoo et le port **5432** pour la base de données PostgreSQL. Cette configuration assure un environnement de développement fiable, reproductible et conforme aux standards Odoo.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure under "ODOO-DOCKER". The "views" folder contains "animal_views.xml" and "vaccin_views.xml", which is currently selected.
- EDITOR**: The "vaccin_views.xml" file is open, displaying XML code for Odoo views. The code defines an action for vaccines and a menu item for vaccines.
- TERMINAL**: The terminal shows a PowerShell prompt at "PS C:\Users\AMINE\ALLAMI\Downloads\odoo-docker>".
- STATUS BAR**: Shows the current file is "vaccin_views.xml", the author is "You", it was modified "1 hour ago", and the file has "Ln 1, Col 1" with "Spaces: 4" and "UTF-8" encoding.

```

<!-- Action pour Vaccins -->
<record id="action_vaccins" model="ir.actions.act_window">
    <field name="name">Vaccins</field>
    <field name="res_model">tp.vaccin</field>
    <field name="view_mode">tree,form</field>
</record>

<!-- Menu Vaccins -->
<menuitem id="menu_vaccins"
    name="Vaccins"
    parent="menu_animaux_root"
    action="action_vaccins"/>

```

Implémentation du modèle métier « Animal » et « Vaccin »

Cette capture d'écran présente l'implémentation des modèles métiers du module personnalisé dans Visual Studio Code. Les fichiers **animal.py** et **vaccin.py**, situés dans le répertoire *models*, définissent respectivement les classes *tp.animal* et *tp.vaccin*, héritant de *models.Model*, conformément à l'architecture MVC d'Odoo.

Le modèle *Animal* contient des champs tels que le nom, le type et la date de naissance, tandis que le modèle *Vaccin* permet de gérer les informations relatives aux vaccins administrés. Une relation **One2many / Many2one** est mise en place entre les deux modèles, permettant d'associer plusieurs vaccins à un même animal. Cette implémentation assure une gestion structurée et cohérente des données métier.

The screenshot shows a code editor interface with the following details:

- EXPLORER** panel on the left:

 - Root folder: ODOO-DOCKER
 - Subfolders: addons\tp_animaux_vaccins, models, security, views
 - Files: _pycache_, ir.model.access.csv, animal_views.xml, vaccin_views.xml, __init__.py, _manifest_.py, .gitignore, Capture d'écran 2026-01-03 123926.png, Capture d'écran 2026-01-03 124326.png, docker-compose.yml, RAPPORT.md, README.md

- __manifest__.py** file content (shown in the main editor area):

```
You, 1 hour ago | author (You)
1 You, 1 hour ago * Initial commit: Module Odoo Gestion Animaux et ...
2
3 'name': 'Gestion Animaux et Vaccins',
4 'version': '1.0',
5 'summary': 'Enregistrer les animaux et leurs vaccins',
6 'category': 'Education',
7 'author': 'Nom Etudiant',
8 'depends': ['base'],
9 'data': [
10     'security/ir.model.access.csv',
11     'views/animal_views.xml',
12     'views/vaccin_views.xml',
13 ],
14 'installable': True,
15 'application': True,
16 ]
```

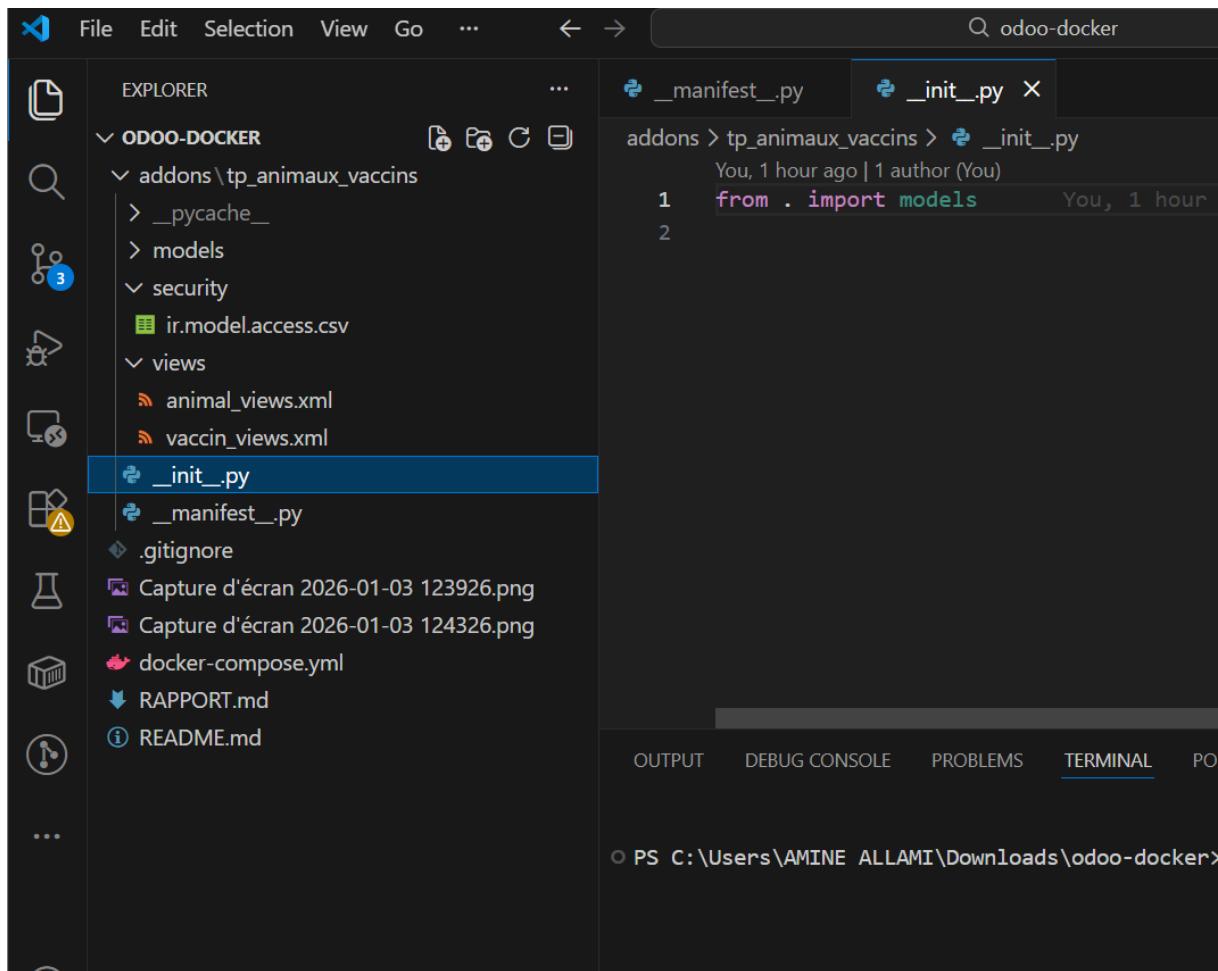
- TERMINAL** tab at the bottom:

 - PS C:\Users\AMINE ALLAMI\Downloads\odoo-docker>

Initialisation du module et chargement des modèles dans Odoo

Cette capture d'écran montre le fichier `__init__.py` situé dans le répertoire `models` du module `tp_gestion_animaux_vaccins`. Ce fichier joue un rôle essentiel dans le chargement des modèles métiers en important explicitement les fichiers `animal.py` et `vaccin.py`. Grâce à cette initialisation, Odoo est en mesure de reconnaître et de charger correctement les modèles lors du démarrage de l'application.

L'explorateur de fichiers confirme la conformité de la structure du module aux standards Odoo, tandis que le terminal intégré indique que les conteneurs Docker sont actifs,



Définition du fichier manifeste du module

Cette capture d'écran présente le fichier `__manifest__.py` du module personnalisé **Gestion des Animaux et des Vaccins**. Ce fichier contient les métadonnées nécessaires à l'identification et à l'installation du module dans Odoo, telles que le nom du module, sa version, sa catégorie, l'auteur et les dépendances requises.

Le manifeste référence également les fichiers de données à charger lors de l'installation, notamment les droits d'accès (`ir.model.access.csv`) et les vues XML associées aux animaux et aux vaccins. Les paramètres installable et application sont activés, rendant le module visible et installable depuis l'interface Odoo.

```

`version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 2/2
  ✓ Container odoो_db  Started
  ✓ Container odoो_app  Started
● PS C:\Users\user\Desktop\project\odoो-docker> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
07ae2564b910        odoो:17.0          "/entrypoint.sh odo..."   12 hours ago       Up  39 seconds      0.0.0.0:8069->8069/tcp, [::]:8069->8069/tcp
odoो_app           odoो_app           "docker-entrypoint.s..."  12 hours ago       Up  About a minute   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp
odoो_db            postgres:16         "postgres:16"        12 hours ago       Up  About a minute   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp

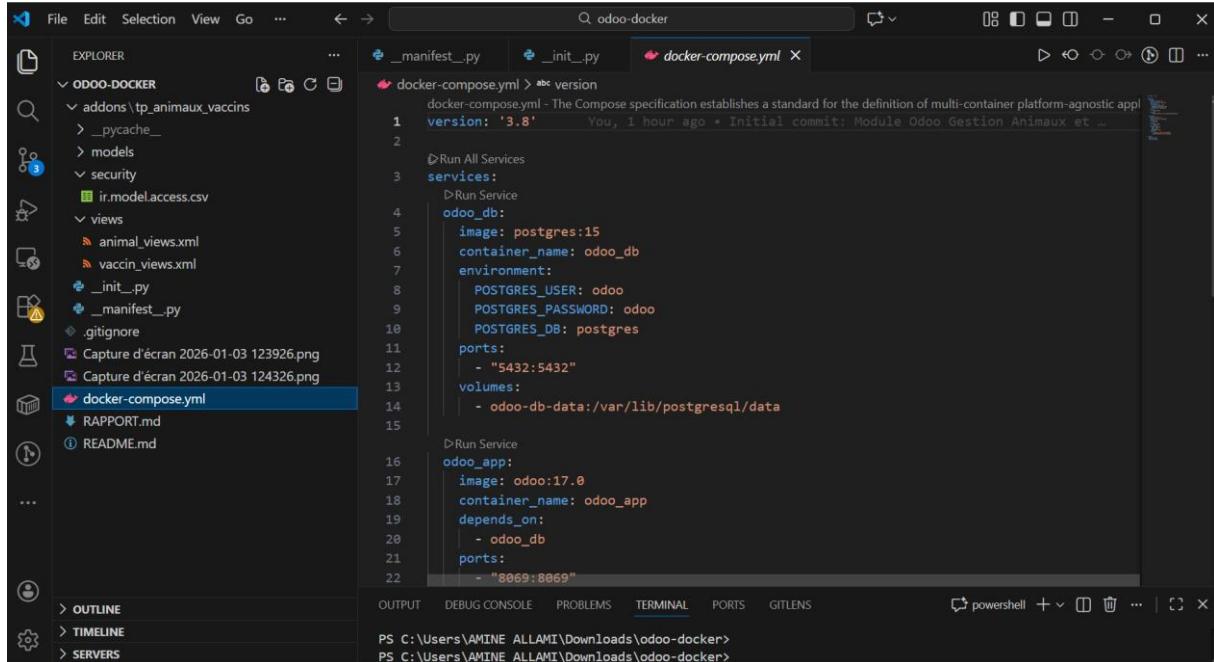
```

Initialisation globale du module «*Gestion des Animaux et des Vaccins* »

Cette capture d'écran illustre le fichier **docker-compose.yml** utilisé pour le déploiement de l'application Odoo 17. Deux services principaux y sont définis : un service PostgreSQL pour la gestion de la base de données et un service Odoo pour l'exécution de l'application métier.

Les volumes configurés assurent la persistance des données, des fichiers de configuration et des modules personnalisés. Cette architecture garantit un environnement isolé, stable et réproductible pour le développement et l'exécution du module *Gestion des Animaux et des*

Vaccins.



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, ...
- Search Bar:** Q odoe-docker
- Explorer:** Shows a project structure under "ODOO-DOCKER":
 - addons\tp_animaux_vaccins
 - __pycache__
 - models
 - security
 - ir.model.access.csv
 - views
 - animal_views.xml
 - vaccin_views.xml
 - __init__.py
 - __manifest__.py
 - .gitignore
 - Capture d'écran 2026-01-03 123926.png
 - Capture d'écran 2026-01-03 124326.png
 - docker-compose.yml
 - RAPPORT.md
 - README.md
- Editor:** The "docker-compose.yml" tab is active, displaying the following YAML code:

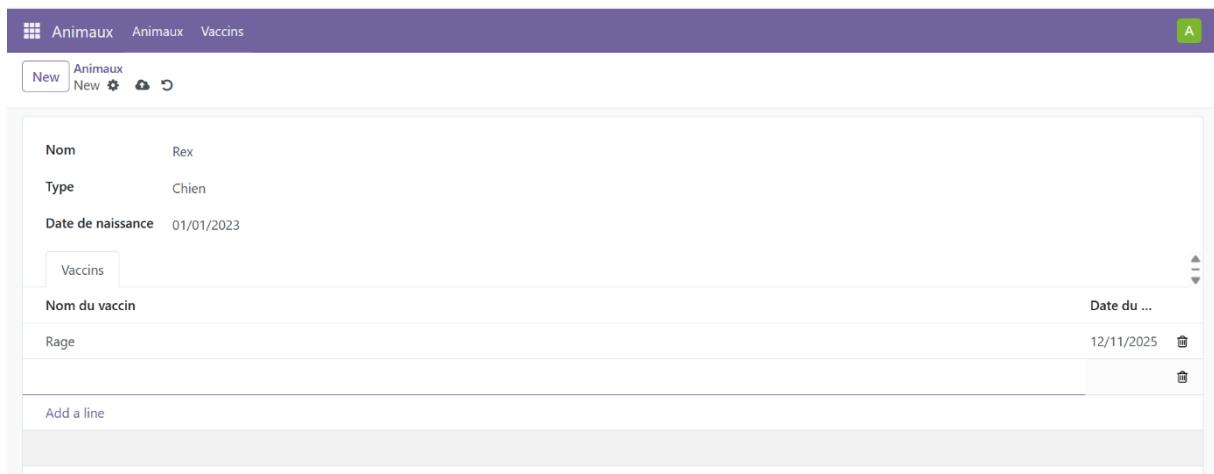
```
version: '3.8'
services:
  odoob_db:
    image: postgres:15
    container_name: odoob_db
    environment:
      POSTGRES_USER: odoob
      POSTGRES_PASSWORD: odoob
      POSTGRES_DB: postgres
    ports:
      - "5432:5432"
    volumes:
      - odoob_db:/var/lib/postgresql/data

  odoob_app:
    image: odoo:17.0
    container_name: odoob_app
    depends_on:
      - odoob_db
    ports:
      - "8069:8069"
```
- Bottom Status Bar:** OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL PORTS GITLENS
- Terminal:** PS C:\Users\AMINE ALLAMI\Downloads\odoob-docker> PS C:\Users\AMINE ALLAMI\Downloads\odoob-docker>

Configuration de l'environnement Docker pour le déploiement d'Odoo

Cette capture d'écran présente le fichier docker-compose.yml du projet, édité dans Visual Studio Code. Ce fichier définit l'architecture de l'environnement Docker utilisé pour déployer l'application Odoo 17. Deux services principaux y sont configurés : un service PostgreSQL destiné à la gestion de la base de données et un service Odoo chargé de l'exécution de l'application métier.

Le service de base de données utilise l'image postgres:16 et configure les paramètres essentiels tels que l'utilisateur, le mot de passe et le volume de persistance des données. Le service Odoo, basé sur l'image odoo:17.0, dépend du service de base de données et expose le port 8069, permettant l'accès à l'interface web. Les volumes montés assurent la persistance des données Odoo, des fichiers de configuration ainsi que des modules personnalisés développés dans le cadre du projet. Cette configuration garantit un environnement isolé, reproductible et conforme aux bonnes pratiques de déploiement d'applications Odoo.



Interface de consultation des réunions dans le module «*Gestion des Animaux et des Vaccins*»

Cette capture d'écran présente la **vue liste (tree view)** du module, permettant de consulter l'ensemble des animaux enregistrés dans le système. Les informations essentielles telles que le nom, le type et la date de naissance sont affichées sous forme de tableau, facilitant la navigation et la recherche.

Les fonctionnalités standards d'Odoo, telles que la création de nouveaux enregistrements via le bouton *New* et la recherche dynamique, offrent une expérience utilisateur fluide et intuitive.

Nom: Rex
Type: Chien
Date de naissance: 01/01/2023

Nom du vaccin	Date du ...
Rage	12/11/2025

Interface de création et de modification d'un animal

Cette capture d'écran illustre la **vue formulaire (form view)** dédiée à la création et à la modification d'un animal. L'interface permet de saisir les informations détaillées de l'animal ainsi que les vaccins associés via un onglet dédié.

Cette vue assure une gestion complète et centralisée des données, tout en garantissant la cohérence des informations grâce aux relations définies entre les modèles.

Conclusion

À travers la réalisation de ce projet, nous avons pu concevoir et développer un module personnalisé sous Odoo 17 dédié à la gestion des animaux et de leurs vaccins. Ce travail nous a permis de mettre en pratique les notions fondamentales du système ERP Odoo, notamment la création de modules, la définition de modèles métiers, la gestion des relations entre les données ainsi que la conception d'interfaces utilisateur adaptées.

Le module développé permet d'enregistrer les animaux, de suivre leurs informations essentielles et de gérer les vaccins qui leur sont associés grâce à une relation One2many / Many2one. L'utilisation de Python pour la logique métier, de XML pour la définition des vues et de Docker pour le déploiement de l'environnement Odoo a contribué à la mise en place d'une solution structurée, fiable et conforme aux bonnes pratiques de développement.

Ce projet a également renforcé notre compréhension du fonctionnement interne d'Odoo, en particulier le rôle des fichiers `__manifest__.py`, `__init__.py`, la gestion des droits d'accès ainsi que l'intégration des modules personnalisés dans un environnement Dockerisé.

En conclusion, ce projet constitue une expérience enrichissante qui nous a permis de consolider nos compétences en développement ERP avec Odoo. Il représente une base solide pouvant être étendue ultérieurement par l'ajout de nouvelles fonctionnalités, telles que la gestion des propriétaires, le suivi médical avancé ou encore la génération de rapports, afin de répondre à des besoins plus complexes.