



ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES -
KHOUREBGA

DÉPARTEMENT INFORMATIQUE

Développement d'un Module de Gestion de Stock Sous Odoo

Réalisé par :
Anas Slimani
Badr Ezziyati
Filière : Gi3

Encadré par :
Prof. NIDAL LAMGHARI



Année Universitaire : 2024 - 2025

Table des matières

1	Introduction	2
2	Présentation d'Odoo	3
2.1	Architecture Technique	3
2.2	La modularité	3
3	Analyse des besoins	4
3.1	Besoins Fonctionnels	4
3.2	Besoins Non-Fonctionnels	4
4	Architecture du module	5
5	Modèles de données	8
5.1	Le modèle Produit (<code>simple.product</code>)	8
5.2	Le modèle Emplacement (<code>simple.location</code>)	8
5.3	Le modèle Mouvement (<code>simple.move</code>)	8
6	Vues et interfaces	9
6.1	Gestion des produits	9
6.2	Configuration des emplacements	10
7	Sécurité	11
8	Workflow des mouvements	12
8.1	État Brouillon	12
8.2	État Validé	12
9	Démonstration	13
9.1	Réalisation d'un inventaire	13
9.2	Traçabilité et Collaboration	13
10	Extraits de code (Python/XML)	14
10.1	Logique de calcul du stock	14
10.2	Définition des vues XML	14
11	Installation	16
12	Conclusion	17

Chapitre 1

Introduction

Dans le paysage technologique actuel, les entreprises cherchent sans cesse à optimiser leurs flux opérationnels. La gestion des stocks constitue le pivot central de la chaîne logistique (Supply Chain). Une mauvaise gestion peut entraîner soit des ruptures de stock coûteuses, soit des surplus immobilisant inutilement du capital.

L'objectif de ce projet est de concevoir et développer un module personnalisé au sein de l'ERP Odoo. Ce module, baptisé "Stock Simple", vise à offrir une alternative légère mais robuste au module de stock standard d'Odoo, en se concentrant sur les fonctionnalités essentielles : gestion des produits, emplacements, mouvements et inventaires.

Chapitre 2

Présentation d'Odoo

Odoo est une suite d'applications d'entreprise open-source couvrant tous les besoins de la gestion commerciale : de la comptabilité à la gestion de projet, en passant par le CRM et le stock.

2.1 Architecture Technique

Odoo repose sur une architecture multicouche :

- **Serveur** : Écrit en Python.
- **Base de données** : PostgreSQL.
- **Client** : Interface web moderne utilisant JavaScript (OWL Framework).

2.2 La modularité

Le point fort d'Odoo réside dans sa modularité. Chaque fonctionnalité est encapsulée dans un module qui peut être installé ou désinstallé sans compromettre l'intégrité du système global.

Chapitre 3

Analyse des besoins

3.1 Besoins Fonctionnels

Le système doit permettre :

- La création et la maintenance d'une base de données d'articles.
- Le suivi précis des quantités disponibles en temps réel.
- La gestion de différents types d'emplacements (Internes, Clients, Fournisseurs).
- La traçabilité complète des transferts de marchandises.
- La réalisation d'ajustements d'inventaire pour corriger les erreurs de comptage.

3.2 Besoins Non-Fonctionnels

- **Sécurité** : Seuls les responsables peuvent valider les inventaires.
- **Ergonomie** : Interface intuitive pour une saisie rapide.
- **Extensibilité** : Le code doit être structuré pour permettre l'ajout futur de fonctionnalités (ex : codes-barres).

Chapitre 4

Architecture du module

La structure du module respecte les standards de développement Odoo.

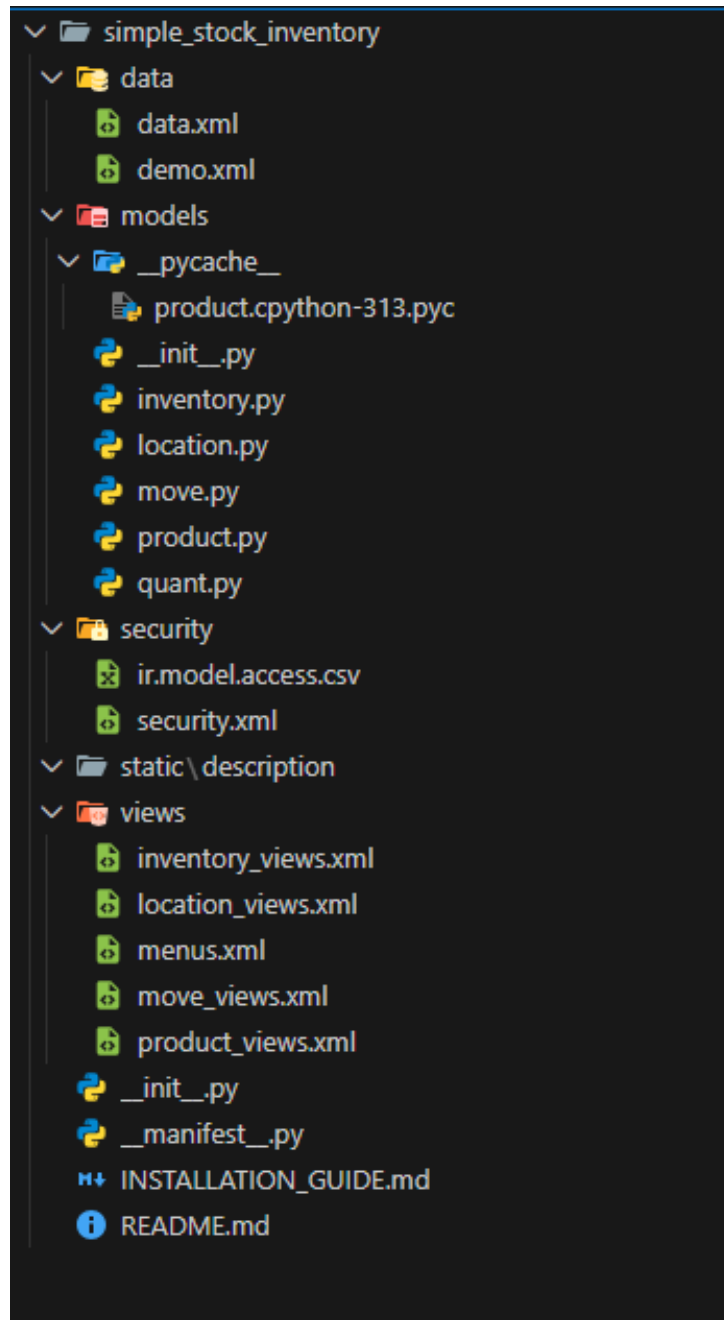


FIGURE 4.1 – Arborescence technique du module Stock Simple

L'image ci-dessus montre l'organisation typique :

- `__manifest__.py` : Déclare les dépendances et les fichiers à charger.
- `models/` : Contient la logique métier (Python).
- `views/` : Définit les interfaces XML.
- `security/` : Gère les droits d'accès.

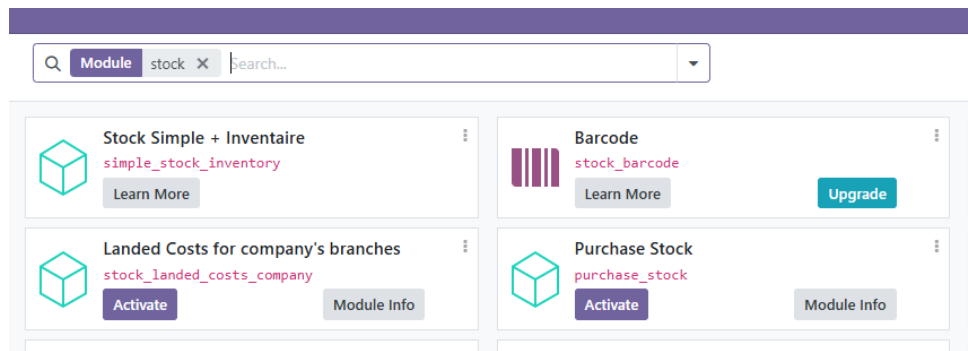


FIGURE 4.2 – Installation du module dans l'ERP

Chapitre 5

Modèles de données

Nous avons implémenté quatre modèles principaux :

5.1 Le modèle Produit (`simple.product`)

Ce modèle stocke les informations de base des articles. Le champ `qty_available` est un champ calculé (`compute`) qui agrège tous les mouvements validés.

5.2 Le modèle Emplacement (`simple.location`)

Définit où les produits sont stockés. On distingue :

- **Interne** : Stock réel appartenant à l'entreprise.
- **Client/Fournisseur** : Emplacements virtuels pour gérer les entrées/sorties.

5.3 Le modèle Mouvement (`simple.move`)

C'est le journal des transactions. Chaque mouvement a une source, une destination et une quantité.

Chapitre 6

Vues et interfaces

L'interface utilisateur a été conçue pour être minimaliste.

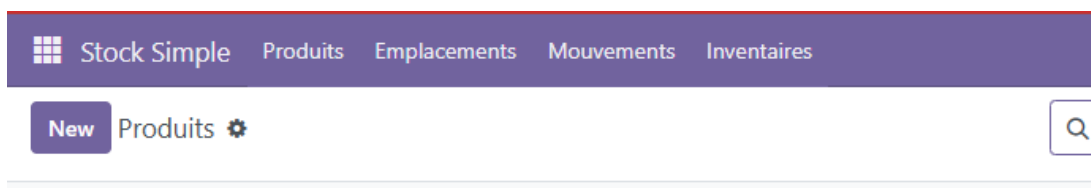


FIGURE 6.1 – Menu racine de l'application

6.1 Gestion des produits

La vue en liste permet de voir immédiatement l'état global du stock.

<input type="checkbox"/> Nom	Référence	Code-barres	Stock total	Prix de revient	Prix de vente
<input type="checkbox"/> Laptop HP ProBook	LAPTOP-001	5901234123457	25.00	450.00	699.00
<input type="checkbox"/> Souris sans fil Logitech	MOUSE-001	5901234123458	150.00	15.00	29.99
<input type="checkbox"/> Clavier mécanique	KEYB-001	5901234123459	80.00	35.00	79.99
<input type="checkbox"/> Écran Dell 27 pouces	MON-001	5901234123460	40.00	180.00	349.00
<input type="checkbox"/> Casque audio Bluetooth	HEAD-001	5901234123461	60.00	25.00	59.99
<input type="checkbox"/> Câble USB-C 2m	CABLE-001	5901234123462	500.00	3.00	12.99
<input type="checkbox"/> Souris			0.00	100.00	120.00

FIGURE 6.2 – Vue en liste des produits et quantités

La vue formulaire détaille les caractéristiques techniques de l'article.

The screenshot shows the 'New' form for a product in the 'Stock Simple' module. The form is divided into two main sections: 'Référence' (Reference) and 'Code-barres' (Barcode). The 'Référence' section contains fields for 'Prix de revient' (Cost Price) and 'Prix de vente' (Selling Price), both set to 0.00. The 'Code-barres' section contains a 'Stock total' field set to 0.00. Below these sections are two tabs: 'Stock par emplacement' (Stock by location) and 'Mouvements' (Movements). The 'Stock par emplacement' tab is active, showing a table with columns 'Emplacement' (Location) and 'Quantité' (Quantity). The table is currently empty, with a 'Add a line' button at the top.

FIGURE 6.3 – Détail d'un produit spécifique

6.2 Configuration des emplacements

The screenshot shows the 'Emplacements' (Locations) configuration page in the 'Stock Simple' module. The page has a search bar and a 'New' button. Below the search bar is a table with columns 'Nom' (Name), 'Usage' (Usage), and 'Actif' (Active). The table lists several locations, each with a checkbox in the 'Actif' column.

Nom	Usage	Actif
<input type="checkbox"/> Stock	Interne	<input checked="" type="checkbox"/>
<input type="checkbox"/> Fournisseurs	Fournisseur	<input checked="" type="checkbox"/>
<input type="checkbox"/> Clients	Client	<input checked="" type="checkbox"/>
<input type="checkbox"/> Ajustement Inventaire	Ajustement Inventaire	<input checked="" type="checkbox"/>
<input type="checkbox"/> Entrepôt A	Interne	<input checked="" type="checkbox"/>
<input type="checkbox"/> Entrepôt B	Interne	<input checked="" type="checkbox"/>

FIGURE 6.4 – Hiérarchie des emplacements de stockage

Chapitre 7

Sécurité

La sécurité est gérée via des groupes d'utilisateurs définis en XML et des fichiers de règles CSV.

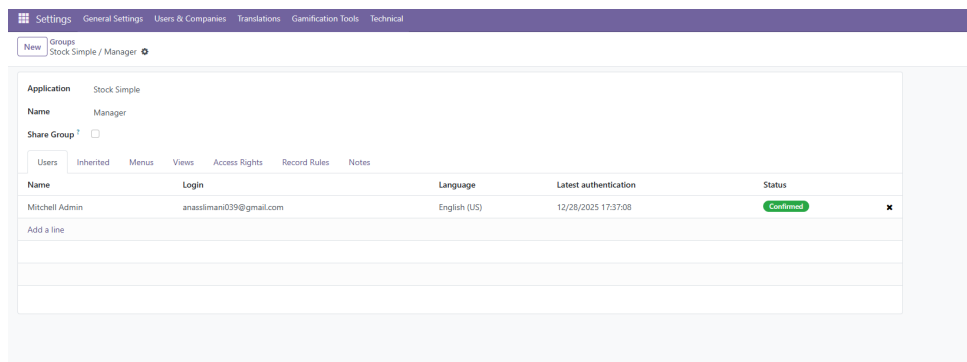


FIGURE 7.1 – Configuration des profils de sécurité

Nous avons deux niveaux :

1. **Utilisateur** : Peut créer des mouvements mais pas valider des inventaires.
2. **Manager** : Accès total, incluant la correction des erreurs d'inventaire.

Chapitre 8

Workflow des mouvements

Chaque transfert passe par un cycle de vie précis pour assurer l'intégrité des données.

8.1 État Brouillon

Lorsqu'un mouvement est créé, il est en état "Draft". Aucune quantité n'est encore soustraite ou ajoutée.

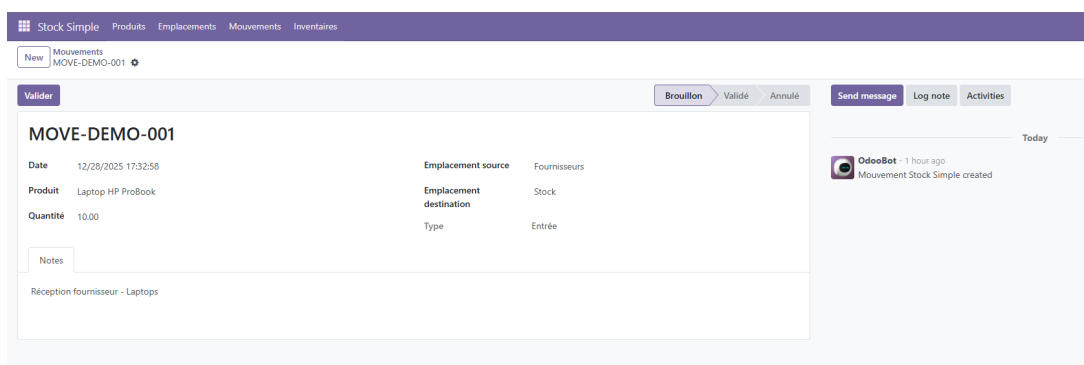


FIGURE 8.1 – Préparation d'un mouvement de stock

8.2 État Validé

Lors de la validation, le système vérifie la disponibilité (si configuré) et met à jour les stocks.

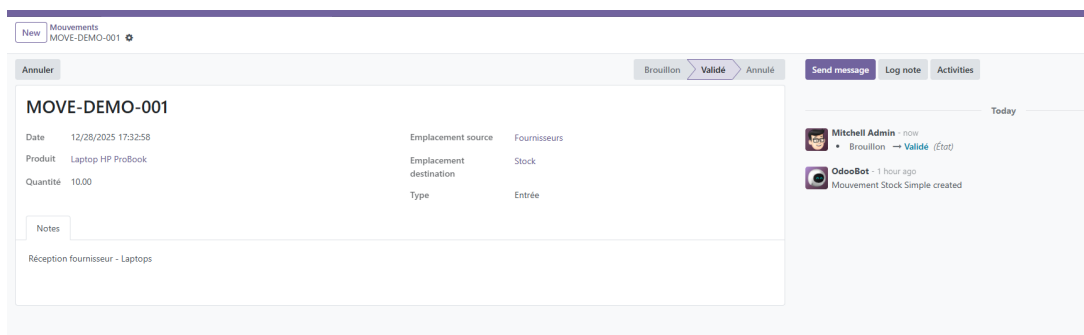


FIGURE 8.2 – Mouvement validé et stock mis à jour

Chapitre 9

Démonstration

9.1 Réalisation d'un inventaire

L'inventaire permet de synchroniser la réalité physique avec les données numériques.

The screenshot shows the 'Ajouter ou modifier un produit' (Add or edit product) form for inventory adjustment INV-00006. The 'Date' is 12/08/2025 18:44:02. The 'Emplacement' (Location) is 'Stock'. The 'Produit' (Product) is 'Laptop HP ProBook'. The 'Quantité théorique' (Theoretical quantity) is 0.00, 'Quantité comptée' (Counted quantity) is 5.00, and 'Écart' (Difference) is 5.00. There are buttons for 'Ajouter à la liste' (Add to list) and 'Annuler' (Cancel).

(a) Saisie des comptages

The screenshot shows the 'Validation' screen for inventory adjustment INV-00003. The 'Date' is 12/08/2025 18:45:47. The 'Emplacement' (Location) is 'Stock'. The 'Produit' (Product) is 'Laptop HP ProBook'. The 'Quantité théorique' (Theoretical quantity) is 0.00, 'Quantité comptée' (Counted quantity) is 5.00, and 'Écart' (Difference) is 5.00. There are buttons for 'Annuler' (Cancel) and 'Valider' (Validate).

(b) Validation et écarts

FIGURE 9.1 – Processus d'ajustement de stock

9.2 Traçabilité et Collaboration

Odoo propose nativement le "Chatter", qui permet de suivre qui a modifié quoi et quand.

The screenshot shows the 'Laptop HP ProBook' product form. The 'Référence' (Reference) is LAPTOP-001, 'Prix de revient' (Cost price) is 450.00, 'Code-barres' (Barcode) is 5501234123457, and 'Prix de vente' (Selling price) is 699.00. The 'Stock total' is 40.00. The 'Emplacement' (Location) is 'Stock' and the 'Quantité' (Quantity) is 40.00. The 'Chatter' (Audit Trail) shows a message from 'OdooBot' 1 hour ago: 'Produit Stock Simple created'.

FIGURE 9.2 – Historique des modifications (Audit Trail)

Chapitre 10

Extraits de code (Python/XML)

Cette section présente la logique technique fondamentale du module.

10.1 Logique de calcul du stock

Le code suivant montre comment nous calculons la quantité disponible en filtrant les mouvements validés.

```
1 @api.depends('move_ids.state')
2 def _compute_qty_available(self):
3     for product in self:
4         moves = self.env['simple.move'].search([
5             ('product_id', '=', product.id),
6             ('state', '=', 'done')
7         ])
8         in_qty = sum(moves.filtered(lambda m: m.location_dest_id.usage
9 == 'internal').mapped('qty'))
10        out_qty = sum(moves.filtered(lambda m: m.location_id.usage == '
internal').mapped('qty'))
11        product.qty_available = in_qty - out_qty
```

Listing 10.1 – Calcul de la quantité en stock

10.2 Définition des vues XML

```
1 <record id="view_simple_product_form" model="ir.ui.view">
2     <field name="name">simple.product.form</field>
3     <field name="model">simple.product</field>
4     <field name="arch" type="xml">
5         <form>
6             <sheet>
7                 <group>
8                     <field name="name"/>
9                     <field name="list_price"/>
10                    <field name="qty_available" readonly="1"/>
11                </group>
12            </sheet>
13        </form>
14    </field>
```

```
15 </record>
```

Listing 10.2 – Définition de la vue formulaire produit

Chapitre 11

Installation

Pour déployer ce module :

1. Installer Odoo (v16 ou supérieur) et PostgreSQL.
2. Copier le dossier `simple_stock` dans le répertoire `addons`.
3. Activer le mode développeur dans Odoo.
4. Aller dans le menu "Applications" et cliquer sur "Mettre à jour la liste des applications".
5. Rechercher "Stock Simple" et cliquer sur "Installer".

Chapitre 12

Conclusion

Le développement de ce module "Stock Simple" a permis de comprendre en profondeur le fonctionnement interne d'Odoo. Nous avons réussi à créer une solution capable de gérer les flux de base d'un entrepôt tout en respectant les bonnes pratiques de développement ERP.

Ce projet m'a également permis de renforcer mes compétences en Python, en modélisation de bases de données et en conception d'interfaces utilisateur centrées sur le métier. Les perspectives d'évolution sont nombreuses, notamment l'intégration d'un tableau de bord analytique pour visualiser les rotations de stock.