Université Sultan Moulay Slimane
École Nationale des Sciences Appliquées (ENSA) - Khouribga

Tuto nouveau module Odoo

Filière : Génie Informatique (3ème année)

Professeur : LAMGHARI Nidal

Réalisé par :

MOUNA ETTALBI

ELMECHHOURI Hajar

Année universitaire : 2025

Sommaire

Introduction

Les systèmes de gestion intégrés (ERP - Enterprise Resource Planning) jouent un rôle crucial dans les organisations modernes en unifiant les processus métier essentiels tels que la gestion des finances, des stocks, des ventes et des ressources humaines. Parmi les solutions ERP disponibles, Odoo se distingue par sa flexibilité, sa modularité et son interface conviviale.

Odoo offre un environnement robuste permettant aux entreprises de personnaliser et d'étendre ses fonctionnalités via des modules. Ces modules permettent d'adapter le logiciel aux besoins spécifiques des utilisateurs tout en intégrant harmonieusement les nouvelles fonctionnalités aux composants existants.

Dans ce contexte, le présent travail pratique (TP) vise à initier les apprenants au développement de modules Odoo. L'objectif est de comprendre comment concevoir, implémenter et intégrer un module pour répondre à un besoin métier spécifique, tout en exploitant les capacités offertes par le framework Odoo.

À travers la création d'un module intitulé "Gestion des Commandes", ce TP explore les concepts fondamentaux tels que :

- La structure et l'organisation des modules dans Odoo.
- La définition et l'utilisation des modèles (ORM) pour gérer les données.
- La création de vues pour interagir avec les utilisateurs.
- L'intégration du module dans l'écosystème Odoo existant.

Création de la Structure du Module

La création d'un module Odoo repose sur une structure bien organisée de dossiers et de fichiers. Cette étape garantit la lisibilité, la maintenabilité, et le bon fonctionnement du module.

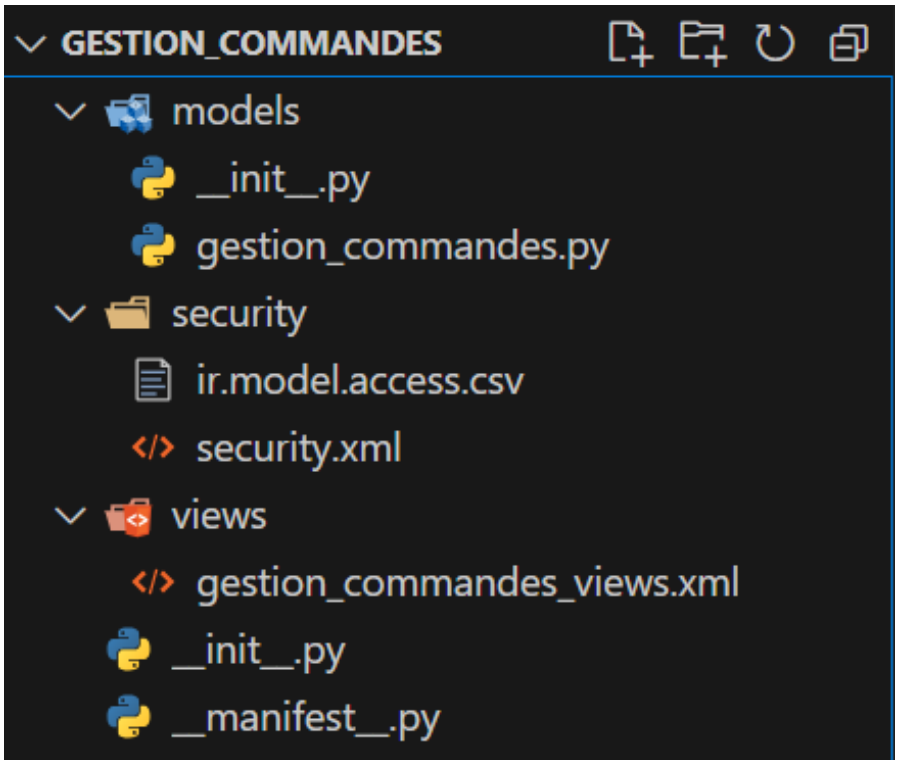
1. Organisation des Dossiers et Fichiers

La structure de base d'un module Odoo suit une hiérarchie stricte. Voici les dossiers et fichiers nécessaires pour un module fonctionnel :

- Dossier principal du module :
- Ce dossier porte le nom du module, par exemple, **gestion_commande**s. Il contient tous les fichiers et sous-dossiers nécessaires au module.
- **Fichiers obligatoires :**
  - **__manifest__.py** : Contient les métadonnées du module (nom, version, dépendances, etc.).
  - **__init__.py** : Permet d'initialiser le module et d'importer les fichiers Python nécessaires.
- **Dossier models :**
- Ce dossier contient les fichiers Python définissant la logique métier et les modèles de données du module. Par exemple, un fichier models.py pour le modèle gestion.commandes.
- **Dossier views :**
- Ce dossier contient les fichiers XML définissant l'interface utilisateur. Par exemple, un fichier gestion_commandes_view.xml pour les vues liste et formulaire.
- Autres dossiers (optionnels) :
  - security : Pour définir les règles d'accès au module.
  - data : Pour charger des données initiales ou des paramètres par défaut.

2. Présentation de la Structure du Module

Voici une structure typique pour un module Odoo nommé gestion_commandes :



Explications des Composants

1. **__init__.py**
   - Indique à Odoo que ce répertoire est un package Python.
   - Importe les modèles définis dans le dossier models.
2. **__manifest__.py**
   - Fournit les informations essentielles pour enregistrer et afficher le module dans Odoo.
3. **Dossier models**
   - Contient les définitions des modèles, par exemple, le modèle GestionCommandes.
4. **Dossier views**
   - Contient les définitions des interfaces utilisateur, comme les vues liste et formulaire.
5. **Dossier security**
   - Définit les droits d'accès au module, garantissant que seules les bonnes personnes peuvent interagir avec les données.

Développement du Module

1. Configuration du Fichier __manifest__.py

1.1. Rôle du Fichier

Le fichier **__manifest__.py** joue un rôle crucial dans le développement d'un module Odoo. Il contient toutes les métadonnées nécessaires pour décrire le module, ses dépendances, et ses fichiers constitutifs. Ce fichier permet :

- D'identifier le module dans l'interface d'administration Odoo.
- De spécifier les modules dont il dépend pour fonctionner correctement.
- De lister les fichiers nécessaires à l'installation (vues, données, sécurité, etc.).

1.2. Exemple de Configuration

Voici un exemple de configuration typique pour un module gestion_commandes :

_manifest__.py

```python
{
    'name': 'Gestion Commandes',
    'version': '1.0',
    'author': 'Votre Nom',
    'category': 'Custom',
    'summary': 'Module pour gérer les commandes des clients',
    'description': """
    Gestion des commandes pour les clients, incluant :
- Création de commandes
- Suivi de l'état des commandes
    """,
    'depends': ['base'],
    'data': [
        'security/security.xml',
        'security/ir.model.access.csv',
        'views/gestion_commandes_views.xml',
    ],
    'installable': True,
    'application': True,
    'license': 'LGPL-3',
}
```

2. Définition des Modèles dans models.py

2.1. Introduction aux Modèles Odoo

Les modèles dans Odoo définissent la structure de la base de données et encapsulent la logique métier. Ils permettent :

- De représenter les données sous forme d'objets Python.
- D'interagir avec la base de données de manière simple et efficace.
- De relier les entités (ex. clients, produits, commandes) grâce à des relations comme Many2one ou One2many.

2.2. Exemple de Modèle pour la Gestion des Commandes

Un modèle typique pour gérer les commandes clients dans Odoo pourrait être défini comme suit :

gestion_commandes.py 1

models > gestion_commandes.py > ...

```python
from odoo import models, fields

class GestionCommandes(models.Model):
    _name = 'gestion.commandes'
    _description = 'Gestion des Commandes'

    name = fields.Char(string='Nom de la commande', required=True)
    date_commande = fields.Datetime(string='Date de la commande', default=fields.Datetime.now)
    client_id = fields.Many2one('res.partner', string='Client', required=True)
    total = fields.Float(string='Total', required=True)
    status = fields.Selection([
        ('draft', 'Brouillon'),
        ('confirmed', 'Confirmée'),
        ('done', 'Livrée')
    ], default='draft', string='Statut')

```

3. Création des Vues Utilisateur dans views.xml

3.1. Vue Liste (Tree)

La vue liste affiche les enregistrements sous forme de tableau. Elle est utile pour visualiser rapidement plusieurs commandes. Voici un exemple :

gestion_commandes_views.xml

views > gestion_commandes_views.xml

```xml
<odoo>
    <record id="view_gestion_commandes_tree" model="ir.ui.view">
        <field name="name">gestion.commandes.tree</field>
        <field name="model">gestion.commandes</field>
        <field name="arch" type="xml">
            <tree string="Commandes">
                <field name="name"/>
                <field name="date_commande"/>
                <field name="client_id"/>
                <field name="total"/>
                <field name="status"/>
            </tree>
        </field>
    </record>

    <record id="view_gestion_commandes_form" model="ir.ui.view">
        <field name="name">gestion.commandes.form</field>
        <field name="model">gestion.commandes</field>
        <field name="arch" type="xml">
            <form string="Commande">
                <sheet>
                    <group>
                        <field name="name"/>
                        <field name="date_commande"/>
                        <field name="total"/>
                        <field name="status"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <menuitem id="menu_gestion_commandes_root" name="Gestion Commandes" sequence="10"/>

    <menuitem id="menu_gestion_commandes" name="Commandes" parent="menu_gestion_commandes_root" actior

    <record id="action_gestion_commandes" model="ir.actions.act_window">
        <field name="name">Commandes</field>
        <field name="res_model">gestion.commandes</field>
        <field name="view_mode">tree,form</field>
    </record>
</odoo>
```

Avec ces configurations, les utilisateurs peuvent gérer les commandes directement depuis l'interface d'Odoo. Les vues permettent d'assurer une expérience fluide et intuitive, tandis que les modèles structurent les données et les interactions métier.

Installation du Module dans Odoo

1. Introduction

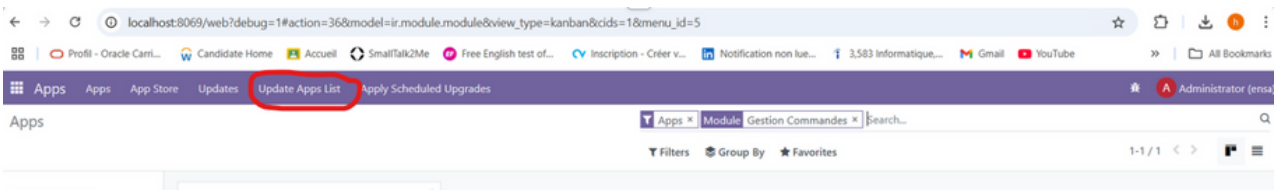
Une fois le développement du module terminé, la dernière étape consiste à l'installer dans l'environnement Odoo. Cela permet de rendre le module fonctionnel et accessible aux utilisateurs via l'interface.

2. Étapes pour Installer le Module

2.1. Recharger les Modules Odoo

Avant de pouvoir installer un nouveau module, il est nécessaire de mettre à jour la liste des modules disponibles. Voici comment procéder :

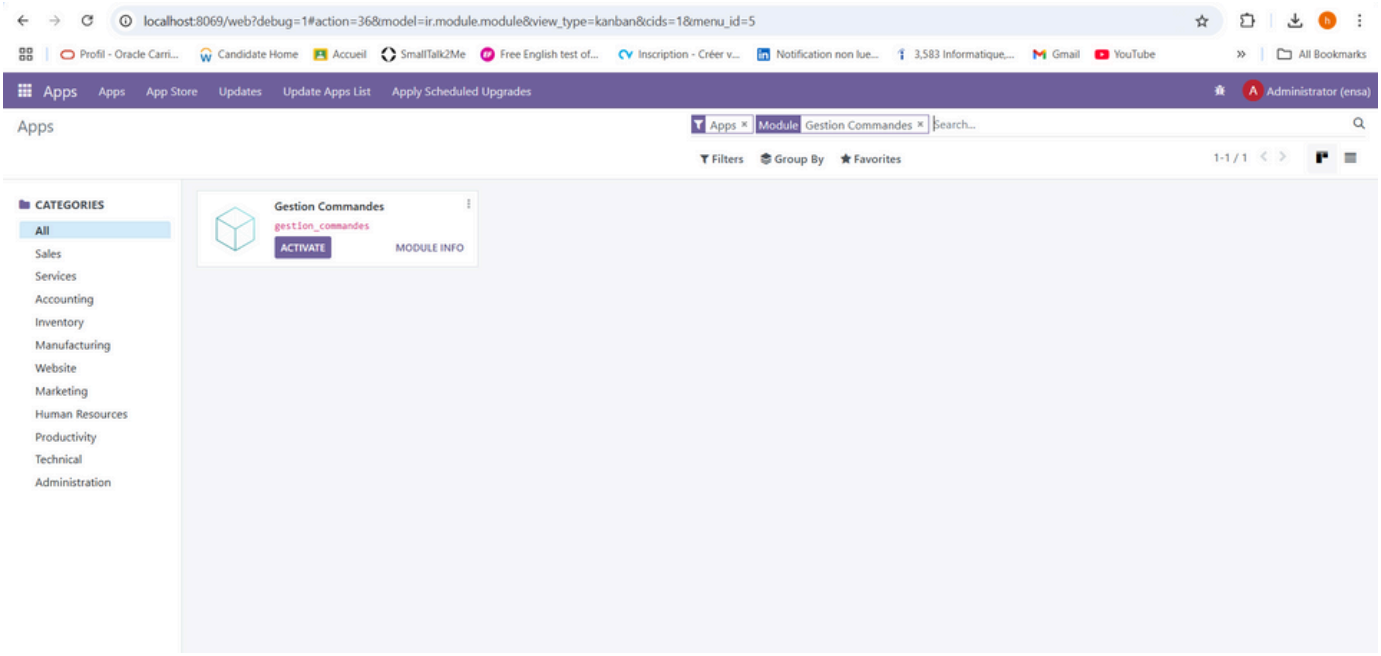
1. Connectez-vous à l'interface d'administration d'Odoo.
2. Accédez au menu Applications dans la barre de navigation principale.
3. Cliquez sur le bouton Mettre à jour la liste des modules situé en haut à droite ou dans le menu déroulant si disponible.

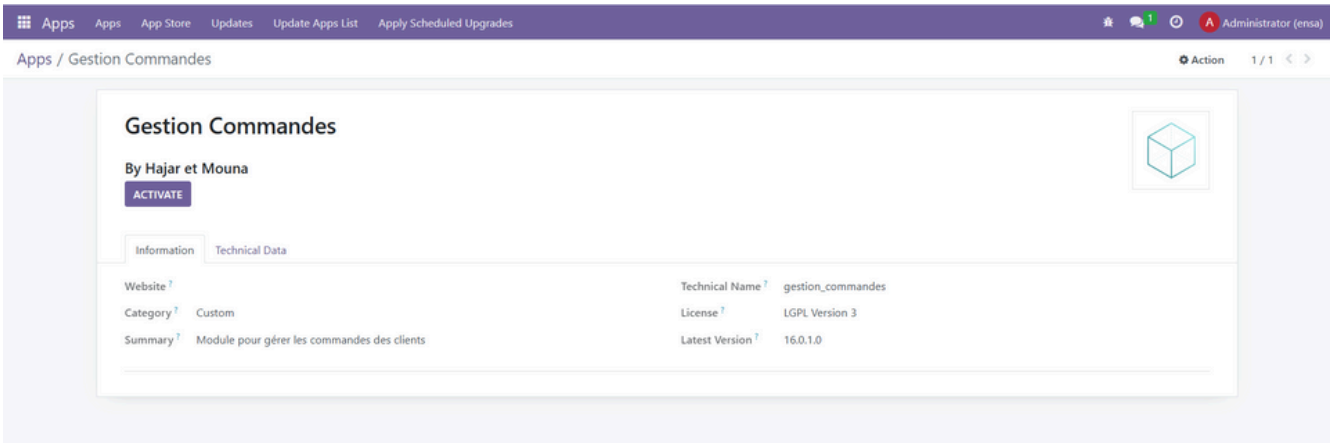


2.2. Installer le Module

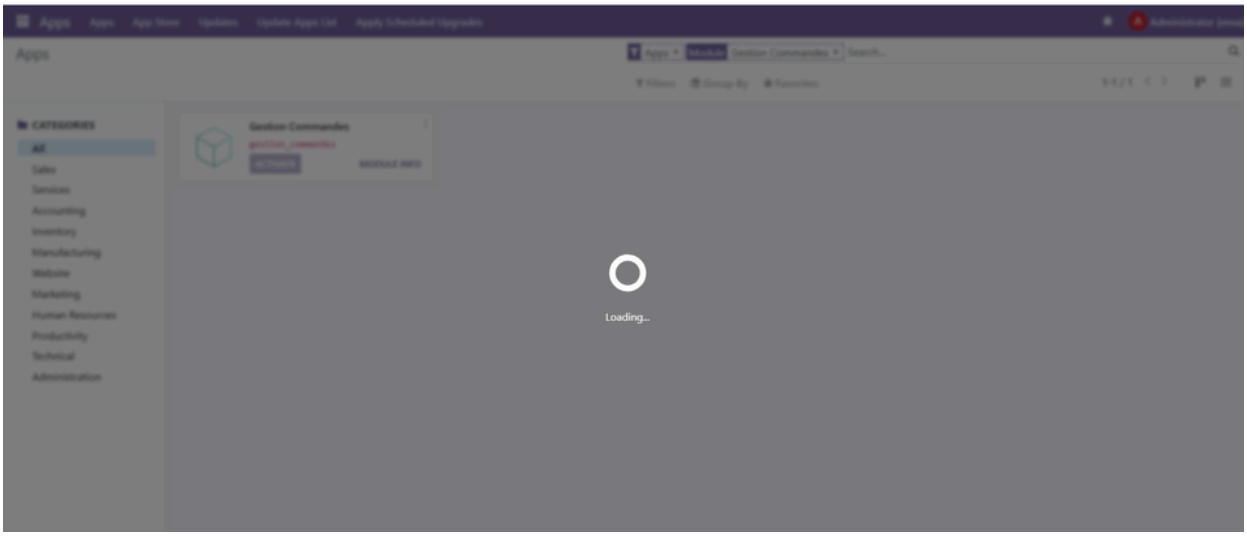
Une fois la liste des modules mise à jour :

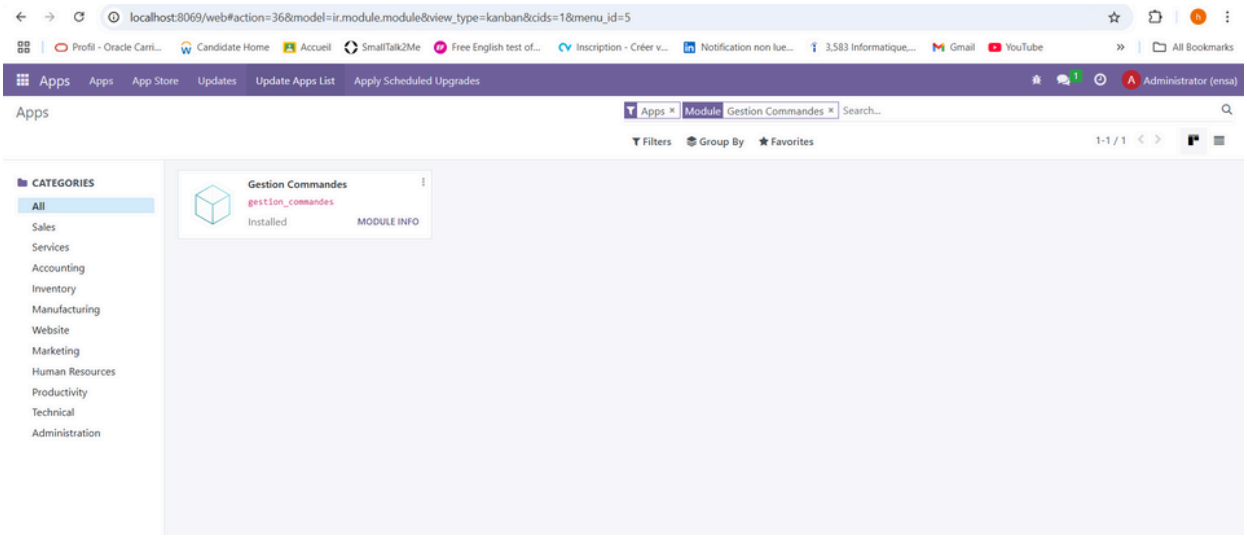
- Recherchez le nom de votre module, par exemple, Gestion des Commandes, dans la barre de recherche.





- Une fois que le module apparaît dans les résultats, cliquez sur le bouton Installer associé.

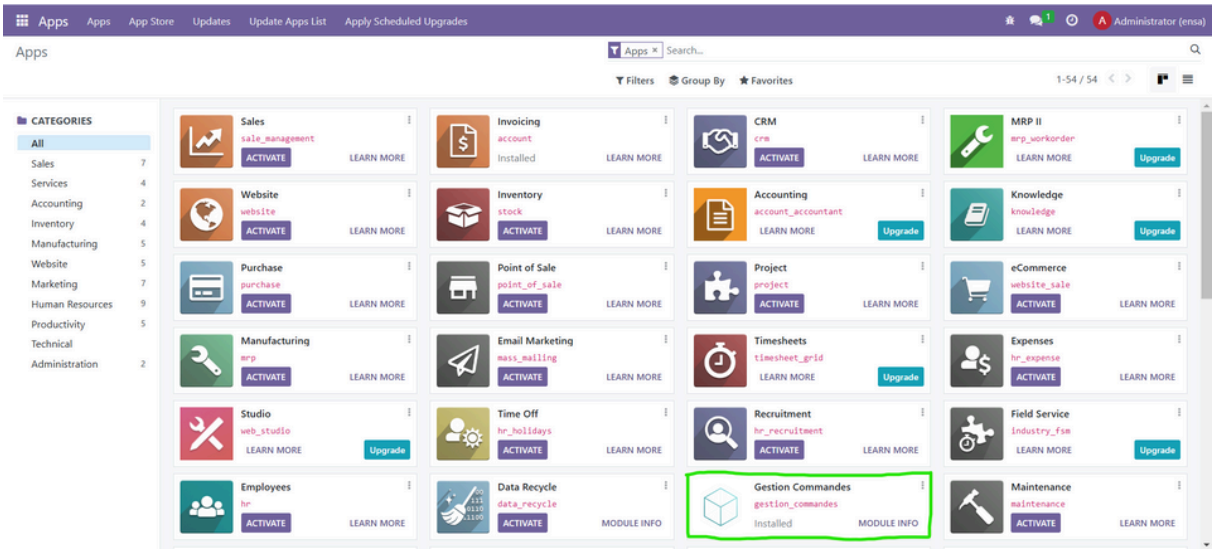


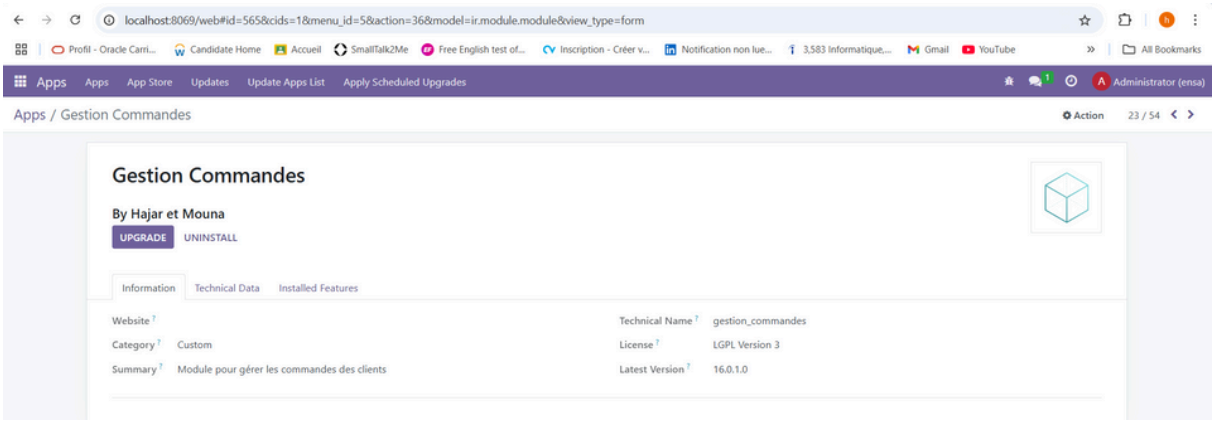


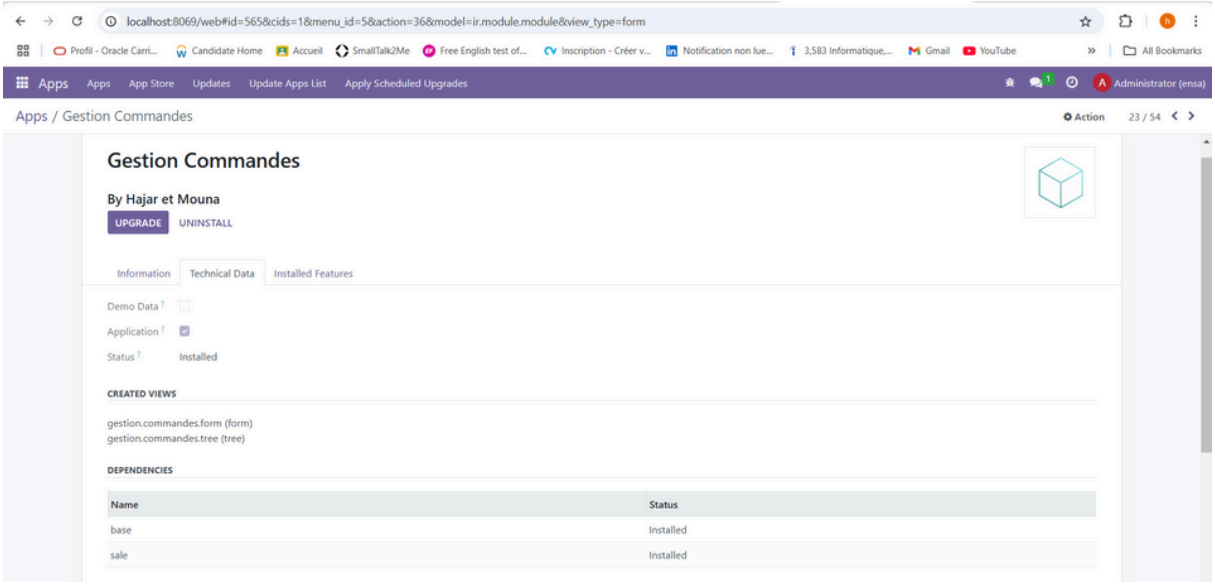
3. Vérification de l'Installation

Après l'installation, le module sera actif dans Odoo et pourra être utilisé. Vous pouvez vérifier cela en :

- Accédant au menu ou tableau de bord généré par le module dans l'interface.
- Testant les fonctionnalités principales, comme l'ajout d'un enregistrement ou l'affichage des vues.


Apps
Apps
App Store
Updates
Update Apps List
Apply Scheduled Upgrades
Administrator (ensa)
Apps
Search...
Filters
Group By
Favorites
1-54 / 54
CATEGORIES
All
Sales 7
Services 4
Accounting 2
Inventory 4
Manufacturing 5
Website 5
Marketing 7
Human Resources 9
Productivity 5
Technical
Administration 2
Sales
sale_management
ACTIVATE
LEARN MORE
Invoicing
account
Installed
LEARN MORE
CRM
crm
ACTIVATE
LEARN MORE
MRP II
mrp_workorder
LEARN MORE
Upgrade
Website
website
ACTIVATE
LEARN MORE
Inventory
stock
ACTIVATE
LEARN MORE
Accounting
account_accountant
LEARN MORE
Upgrade
Knowledge
knowledge
LEARN MORE
Upgrade
Purchase
purchase
ACTIVATE
LEARN MORE
Point of Sale
point_of_sale
ACTIVATE
LEARN MORE
Project
project
ACTIVATE
LEARN MORE
eCommerce
website_sale
ACTIVATE
LEARN MORE
Manufacturing
mrp
ACTIVATE
LEARN MORE
Email Marketing
mass_mailing
ACTIVATE
LEARN MORE
Timesheets
timesheet_grid
LEARN MORE
Upgrade
Expenses
hr_expense
ACTIVATE
LEARN MORE
Studio
web_studio
LEARN MORE
Upgrade
Time Off
hr_holidays
ACTIVATE
LEARN MORE
Recruitment
hr_recruitment
ACTIVATE
LEARN MORE
Field Service
industry_fsm
LEARN MORE
Upgrade
Employees
hr
ACTIVATE
LEARN MORE
Data Recycle
data_recycle
ACTIVATE
MODULE INFO
Gestion Commandes
gestion_commandes
Installed
MODULE INFO
Maintenance
maintenance
ACTIVATE
LEARN MORE


localhost:8069/web#id=565&cids=1&menu_id=5&action=36&model=ir.module.module&view_type=form
Apps / Gestion Commandes
Action
23 / 54
Gestion Commandes
By Hajar et Mouna
UPGRADE
UNINSTALL
Information
Technical Data
Installed Features
Website
Category Custom
Summary Module pour gérer les commandes des clients
Technical Name gestion_commandes
License LGPL Version 3
Latest Version 16.0.1.0


localhost:8069/web#id=565&cids=1&menu_id=5&action=36&model=ir.module.module&view_type=form
Apps / Gestion Commandes
Action
23 / 54
Gestion Commandes
By Hajar et Mouna
UPGRADE
UNINSTALL
Information
Technical Data
Installed Features
Demo Data
Application
Status Installed
CREATED VIEWS
gestion.commandes.form (form)
gestion.commandes.tree (tree)
DEPENDENCIES
Name
Status
base
Installed
sale
Installed

Conclusion

La réalisation d'un module personnalisé dans Odoo est un processus structuré et méthodique qui permet d'adapter l'application aux besoins spécifiques d'une organisation. Ce tutoriel a guidé pas à pas la création d'un module, depuis la structuration initiale des fichiers jusqu'à son installation et sa mise en service dans Odoo.

Nous avons vu comment configurer le fichier __manifest__.py pour définir les informations essentielles du module, comment implémenter des modèles pour gérer les données, et comment concevoir des vues utilisateur pour offrir une interface ergonomique. Enfin, nous avons détaillé les étapes nécessaires pour installer et tester le module dans Odoo.

Ce travail constitue une base solide pour aller plus loin, en ajoutant des fonctionnalités supplémentaires, en optimisant l'interface utilisateur ou en intégrant d'autres modules. Il démontre également les capacités de personnalisation d'Odoo pour répondre aux besoins spécifiques des entreprises, ce qui en fait une solution ERP de choix dans divers secteurs.

En conclusion, ce projet reflète l'importance de la maîtrise des concepts clés du développement dans Odoo, combinée à une bonne organisation, pour réussir l'intégration de solutions sur mesure et améliorer la productivité au sein des entreprises.