



Rapport de Stage

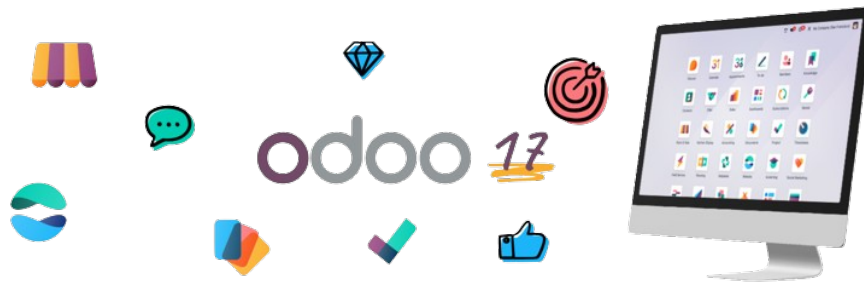
Maram Belhouchet

2ème année licence

Spécialité : Big Data

Institut Supérieur des Arts Multimédia de la Manouba

Encadrant : Achraf Alioui



1er juin 2024 - 30 juin 2024

Contents

1	Introduction	3
2	Présentation de l'entreprise	4
2.1	Historique et Fondation	4
2.2	Direction Générale	4
2.3	Services Offerts	4
2.4	Structure Organisationnelle	4
2.5	Implantation et Impact Régional	5
3	Objectifs du stage	5
4	Odoo	6
4.1	Avantages d'Odoo	6
4.2	CRM et ERP	6
4.3	Utilisation et Importance	7
4.4	Utilisation avec PyCharm et autres plateformes	7
5	Comment installer Odoo avec PyCharm	8
5.1	Étapes d'installation	8
6	Mes tâches et mes missions	10
6.1	Développement d'un module d'apprentissage pour Odoo	10
6.2	Développement d'un système de gestion d'agence automobile	16
6.3	Tests et validation des fonctionnalités	21
7	Détails du Projet Car Agency	22
7.1	Structure du Module	22
7.2	Fichier <code>__manifest__.py</code>	22
7.3	Fichier <code>__init__.py</code>	23
7.4	Modèles Python	23
7.5	Vues XML	28
7.6	Access rights et création de users admin et user normal et création d'un groupe :	39
7.6.1	Utilisateurs	41
7.6.2	Groupe	42
7.7	Publication du Module sur GitHub et la Plateforme Odoo	46
8	Conclusion	50

Remerciements

Je souhaite exprimer ma profonde gratitude à toute l'équipe de Softifi, spécialement à mon encadrant Achraf Alioui, pour leur soutien inestimable, leurs conseils éclairés et leur accueil chaleureux qui ont enrichi mon expérience tout au long de ce stage. Leur expertise dans le domaine de la programmation et du développement de modules Odoo a été d'une grande valeur pour moi. Leur disponibilité et leur encouragement m'ont permis de relever des défis techniques et de développer des compétences essentielles. Ce stage a été une occasion précieuse pour moi de non seulement acquérir des connaissances pratiques, mais aussi de renforcer ma capacité à travailler en équipe et à m'adapter aux exigences d'un environnement professionnel dynamique.

1 Introduction

Pendant mon stage chez Softifi, situé à Jerjis dans le Smart Center, j'ai eu l'opportunité enrichissante de plonger dans l'univers d'Odoo, un système ERP open-source largement utilisé pour sa flexibilité et sa capacité à transformer les opérations métier. Ce stage, qui s'est déroulé sur une période d'un mois en juin 2024, représente une étape clé dans mon parcours académique et professionnel, me permettant d'appliquer mes connaissances théoriques à des projets concrets au sein d'un environnement professionnel dynamique.

Odoo, avec sa capacité à intégrer et à automatiser une gamme étendue de processus d'entreprise, m'a fasciné dès le début. Mon objectif principal était de maîtriser cet outil puissant, non seulement pour comprendre ses fonctionnalités de base, mais aussi pour explorer ses capacités de personnalisation et d'extension à travers le développement de modules sur mesure. Travailler au sein de Softifi m'a offert bien plus que la simple opportunité de développer des compétences techniques ; j'ai eu la chance de collaborer avec une équipe expérimentée, d'échanger des idées et des connaissances, et de contribuer activement à des projets d'envergure.

Au fil des semaines, j'ai pu découvrir et mettre en pratique divers aspects d'Odoo, depuis la création de modules simples jusqu'à la gestion avancée de données et la personnalisation d'interfaces utilisateur. Chaque jour a été une nouvelle occasion d'apprentissage et de croissance, renforçant ma compréhension des défis et des exigences du développement logiciel en milieu professionnel.

Ce stage a été particulièrement enrichissant grâce aux projets stimulants auxquels j'ai contribué, notamment la création d'un module d'apprentissage intégrant des fonctionnalités telles que les boutons intelligents, la gestion de menus, et la gestion des images. Chaque étape de ce parcours m'a permis de voir concrètement comment Odoo peut être adapté et personnalisé pour répondre aux besoins spécifiques des entreprises.

En résumé, mon stage chez Softifi a été une expérience transformative qui a non seulement consolidé mes compétences techniques et professionnelles, mais aussi renforcé ma passion pour le développement logiciel et l'intégration de solutions ERP comme Odoo.

2 Présentation de l'entreprise

2.1 Historique et Fondation

Softifi a été fondée à Jerjis par Adel Ben Merteh, avec une vision centrée sur l'intégration de solutions numériques avancées pour répondre aux besoins croissants des entreprises dans le domaine des technologies de l'information.

2.2 Direction Générale

- **Fondateur** : Adel Ben Merteh
- **Directeur Général** : Rabia Radhouane

Rabia Radhouane assume la responsabilité de la direction générale de Softifi, guidant l'entreprise vers l'innovation continue et la croissance dans divers secteurs technologiques.

2.3 Services Offerts

Softifi propose une gamme complète de services numériques et technologiques, notamment :

- **Digital Marketing** : Stratégies numériques avancées pour optimiser la visibilité en ligne et la génération de leads.
- **E-commerce** : Développement de plateformes e-commerce robustes et évolutives pour faciliter le commerce en ligne.
- **Web Development** : Conception et développement de sites web personnalisés, adaptés aux besoins spécifiques des clients.
- **Mobile Development** : Création d'applications mobiles sur mesure pour iOS et Android, en utilisant les dernières technologies.
- **Outsourcing** : Externalisation des services technologiques pour optimiser les coûts et améliorer l'efficacité opérationnelle.
- **Odoo ERP** : Intégration et personnalisation du système ERP Odoo pour automatiser les processus métier et améliorer la gestion des entreprises.
- **PEGA** : Utilisation de la plateforme PEGA pour le développement d'applications d'entreprise basées sur la gestion des processus métier.

2.4 Structure Organisationnelle

Softifi opère avec une structure organisationnelle efficace, comprenant plusieurs départements clés :

- **Développement Mobile** : Ce département se concentre sur la création d'applications mobiles innovantes et performantes.

- **Développement Web** :Ce département est spécialisé dans la conception et le développement de solutions web avancées.
- **DevOps** :Ce département assure l'intégration continue, le déploiement et la gestion des infrastructures technologiques.
- **Développement Odoo** : Ce département se focalise sur l'intégration et la personnalisation du système ERP Odoo pour répondre aux besoins spécifiques des clients.

2.5 Implantation et Impact Régional

Softifi est installée au Parc d'Activités Économiques de Zarzis (PAEZ), un emplacement stratégique qui accueille régulièrement des visites officielles de haut niveau. Ces visites témoignent de l'engagement de Softifi envers l'excellence et illustrent son rôle crucial dans le développement économique local et régional.

3 Objectifs du stage

L'objectif principal de mon stage chez Softifi était d'acquérir une maîtrise approfondie des fonctionnalités d'Odoo, un ERP open-source réputé pour sa flexibilité et son potentiel de transformation des processus métier. À travers ce parcours de formation pratique d'un mois, réalisé en juin 2024, j'ai cherché non seulement à explorer les capacités intrinsèques d'Odoo, mais aussi à les appliquer dans des projets concrets au sein d'un environnement professionnel dynamique. Ce stage représentait pour moi une occasion unique de consolider mes connaissances théoriques acquises au cours de ma formation universitaire en Big Data, en les adaptant aux exigences spécifiques du développement logiciel dans une entreprise innovante comme Softifi. J'ai abordé ce stage avec un objectif clair : développer des compétences pratiques en programmation et en développement de modules sur mesure, tout en m'immergeant pleinement dans le processus d'intégration et de personnalisation d'Odoo. Chaque jour a été une opportunité de croissance personnelle et professionnelle, enrichie par les conseils avisés de mes collègues et l'encadrement précieux de mon superviseur, Achraf Alioui.

4 Odoo

Odoo est un système ERP (Enterprise Resource Planning) open-source largement utilisé par les entreprises pour automatiser et gérer leurs processus métier. Il offre une suite intégrée d'applications couvrant divers domaines fonctionnels tels que la gestion des ventes, des achats, des stocks, la comptabilité, les ressources humaines, le marketing, et plus encore. Conçu pour être flexible et modulaire, Odoo permet aux entreprises de personnaliser et d'étendre ses fonctionnalités selon leurs besoins spécifiques.

4.1 Avantages d'Odoo

Odoo présente plusieurs avantages significatifs :

- **Flexibilité et modularité** : Odoo est constitué de modules indépendants qui peuvent être activés ou désactivés selon les besoins de l'entreprise, offrant ainsi une solution flexible et adaptable.
- **Personnalisation** : Les entreprises peuvent personnaliser les fonctionnalités d'Odoo pour répondre précisément à leurs processus métier spécifiques, grâce à un large éventail d'options de configuration et de développement.
- **Intégration** : Odoo permet l'intégration transparente de différentes applications et modules au sein d'une même plateforme, favorisant ainsi une gestion centralisée et efficace des opérations.
- **Accessibilité et convivialité** : Avec une interface utilisateur intuitive et conviviale, Odoo facilite l'adoption par les utilisateurs finaux et réduit le temps nécessaire à la formation.
- **Coût** : En tant que solution open-source, Odoo offre une alternative économique par rapport aux ERP traditionnels, tout en fournissant des fonctionnalités puissantes et évolutives.

4.2 CRM et ERP

Le CRM (Customer Relationship Management) et l'ERP (Enterprise Resource Planning) sont deux systèmes clés pour la gestion efficace des entreprises. Le CRM se concentre sur la gestion des relations avec les clients, incluant la collecte et l'analyse des données clients, la gestion des interactions, et l'automatisation des processus de vente et de marketing. L'ERP, en revanche, englobe une gamme plus large de fonctions de gestion d'entreprise, incluant la finance, les ressources humaines, la gestion des stocks, la production et la chaîne d'approvisionnement. Ces systèmes permettent une intégration complète des différents processus métier, assurant une fluidité opérationnelle et une prise de décision basée sur des données précises et en temps réel.

Odoo combine les fonctionnalités de CRM et d'ERP en une seule plateforme intégrée. Cette intégration permet aux entreprises de bénéficier d'une vue holistique de leurs opérations, allant de la gestion des clients à la planification des ressources. Par exemple, les données collectées par le CRM sur les comportements et les préférences des clients peuvent être utilisées par l'ERP pour optimiser la gestion des stocks et la production. De plus, les équipes de vente et de marketing peuvent collaborer plus efficacement avec

les départements financiers et opérationnels, grâce à la centralisation des informations et à l'automatisation des workflows. Ainsi, Odoo facilite une synergie entre CRM et ERP, permettant aux entreprises d'améliorer leur efficacité, leur réactivité et leur compétitivité sur le marché.

4.3 Utilisation et Importance

Odoo est largement utilisé dans divers secteurs industriels pour rationaliser les processus opérationnels, améliorer l'efficacité et optimiser la gestion des ressources. Son adoption aide les entreprises à gagner en compétitivité en automatisant les tâches répétitives, en réduisant les erreurs humaines et en fournissant des données précieuses pour la prise de décision stratégique. En tant que plateforme flexible et évolutive, Odoo accompagne la croissance des entreprises en s'adaptant aux changements et aux défis du marché, tout en soutenant l'innovation continue et la transformation numérique.

4.4 Utilisation avec PyCharm et autres plateformes

Pour développer et personnaliser des modules Odoo, les développeurs utilisent des outils de développement intégrés (IDE) tels que PyCharm. PyCharm offre un environnement robuste pour la programmation en Python, le langage de programmation utilisé pour développer des modules Odoo. Parmi ses fonctionnalités, on trouve la coloration syntaxique, l'auto-complétion, le débogage, et l'intégration avec des systèmes de gestion de versions comme Git, ce qui facilite le processus de développement.

En plus de PyCharm, d'autres plateformes et outils sont souvent utilisés en conjonction avec Odoo pour maximiser son efficacité et ses fonctionnalités. Cela peut inclure des environnements de développement collaboratif comme GitHub pour la gestion de projet et le partage de code, ainsi que des outils de déploiement comme Docker pour faciliter le déploiement et la gestion des instances Odoo dans différents environnements.

5 Comment installer Odoo avec PyCharm

Pour développer et personnaliser des modules Odoo, il est courant d'utiliser PyCharm comme environnement de développement intégré (IDE). Voici les étapes détaillées pour installer Odoo avec PyCharm :



5.1 Étapes d'installation

1. Prérequis :

- Assurez-vous d'avoir Python (version 3.7 ou supérieure) installé sur votre système.
- Installez PostgreSQL, un système de gestion de base de données relationnelles, nécessaire pour Odoo.
- Téléchargez et installez PyCharm depuis le site officiel de JetBrains.

2. Téléchargement d'Odoo :

```
git clone https://www.github.com/odoo/odoo --depth 1 --branch 17.0
```

Cette commande télécharge la dernière version stable d'Odoo depuis GitHub.

3. Configuration de l'environnement virtuel :

```
python3 -m venv odoo-venv  
source odoo-venv/bin/activate
```

Activez un environnement virtuel pour isoler les dépendances d'Odoo.

4. Installation des dépendances Python :

```
pip install -r odoo/requirements.txt
```

Installez les packages Python requis par Odoo.

5. Configuration de PostgreSQL :

```
sudo -u postgres createuser -s odoo
sudo -u postgres createdb odoo -O odoo
```

Créez un utilisateur et une base de données pour Odoo dans PostgreSQL.

6. Configuration d'Odoo :

```
cp odoo/debian/odoo.conf odoo.conf
```

Copiez le fichier de configuration d'Odoo et modifiez-le selon vos besoins.

7. Démarrage d'Odoo :

```
python3 odoo/odoo-bin -c odoo.conf
```

Lancez Odoo en spécifiant le fichier de configuration.

8. Configuration de PyCharm :

- Ouvrez PyCharm et créez un nouveau projet Python en sélectionnant l'interpréteur Python de votre environnement virtuel (`odoo-venv`).
- Configurez un nouvel exécutable Python dans PyCharm pour pointer vers `odoo/odoo-bin` dans votre projet.
- Configurez les paramètres d'exécution dans PyCharm pour spécifier le fichier de configuration `odoo.conf`.

9. Développement avec PyCharm :

- Utilisez les fonctionnalités de PyCharm telles que la coloration syntaxique avancée, l'auto-complétion, et le débogage pour développer des modules Odoo.
- Intégrez PyCharm avec des systèmes de gestion de versions comme Git pour gérer votre code efficacement.

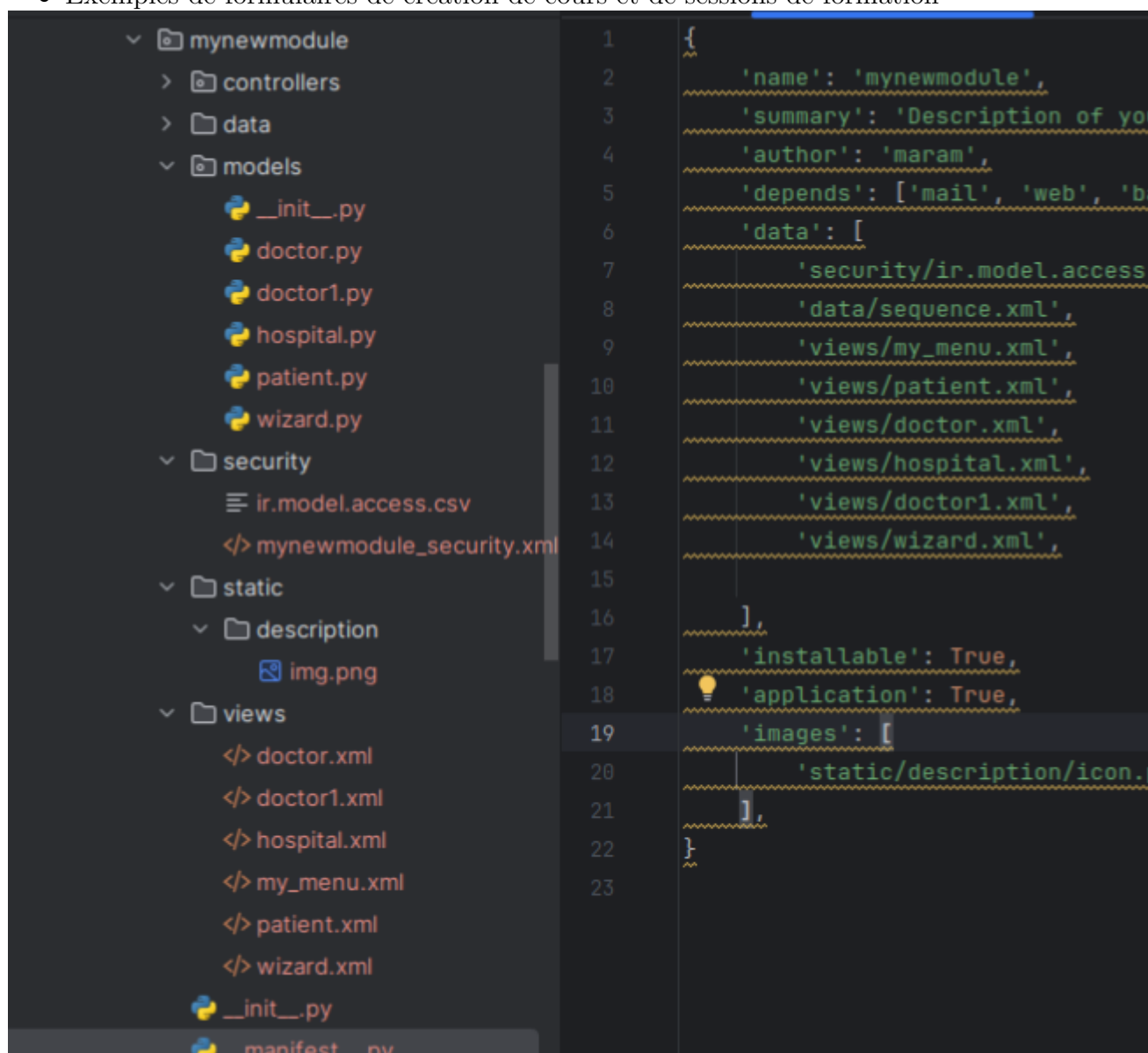
6 Mes tâches et mes missions

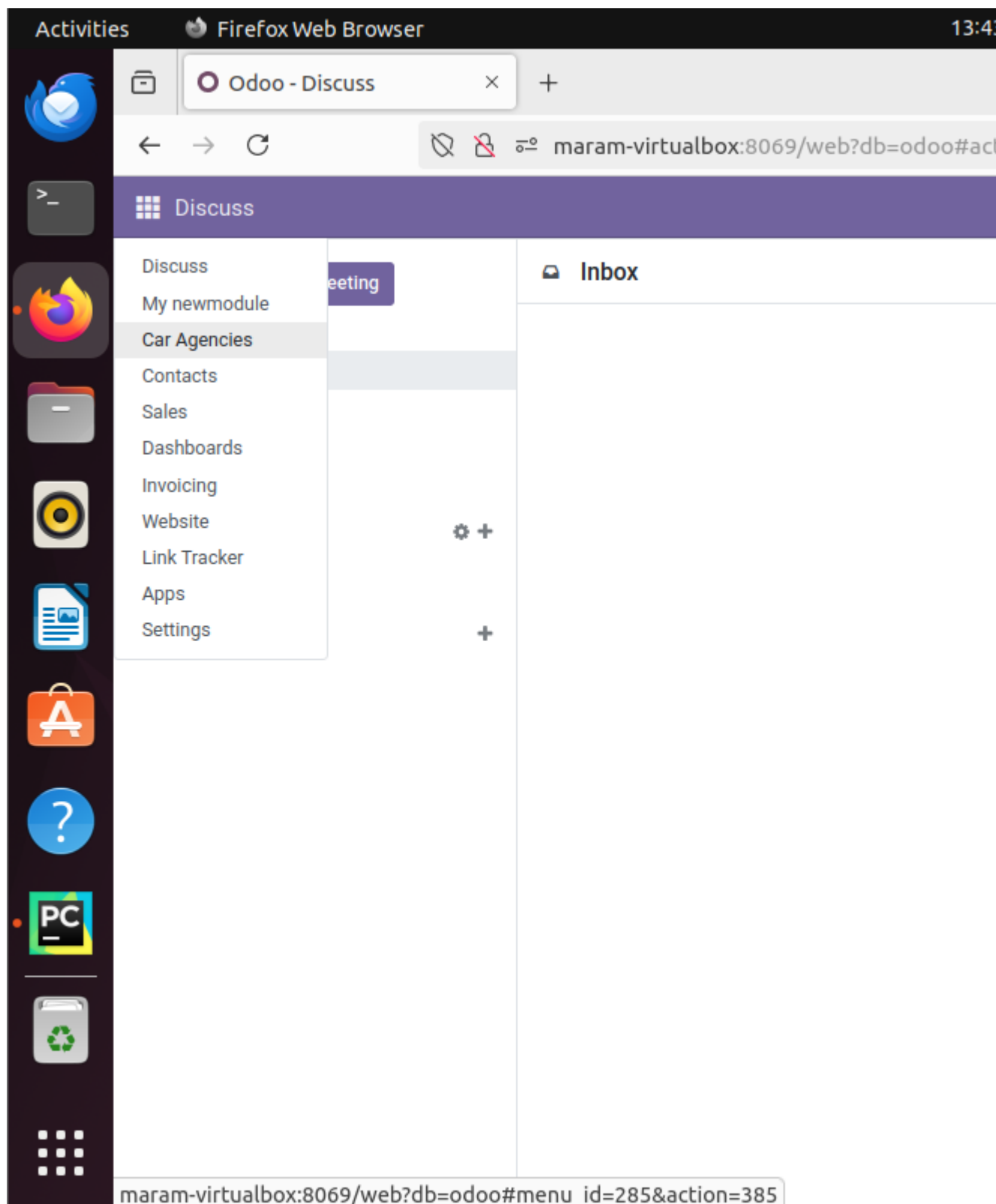
6.1 Développement d'un module d'apprentissage pour Odoo

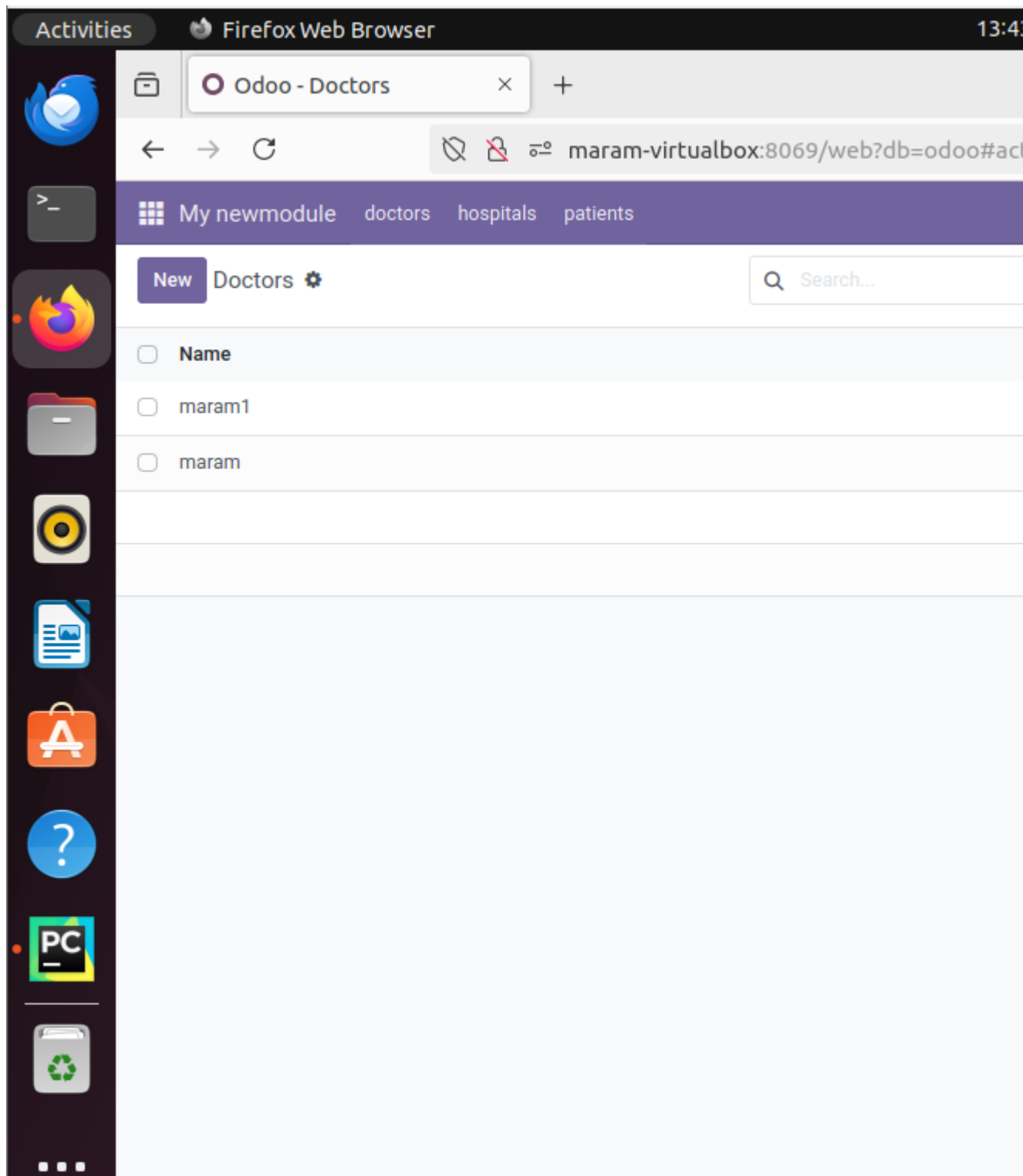
Pendant mon stage chez Softifi, j'ai été chargé de concevoir et développer un module d'apprentissage intégré à Odoo. Ce projet comprenait la création de modèles de données, l'implémentation de workflows personnalisés, et l'intégration d'interfaces utilisateur conviviales...

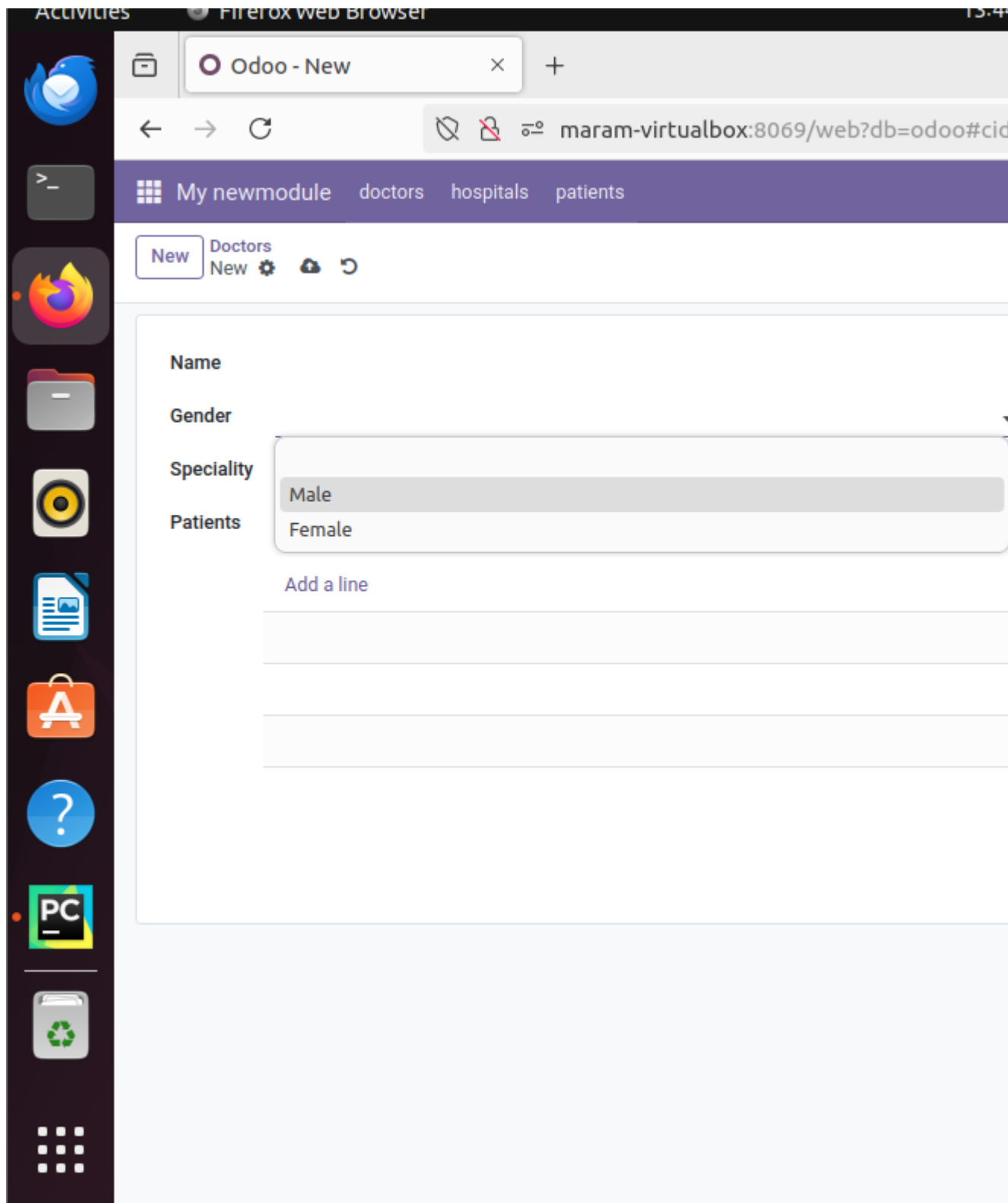
Captures d'écran :

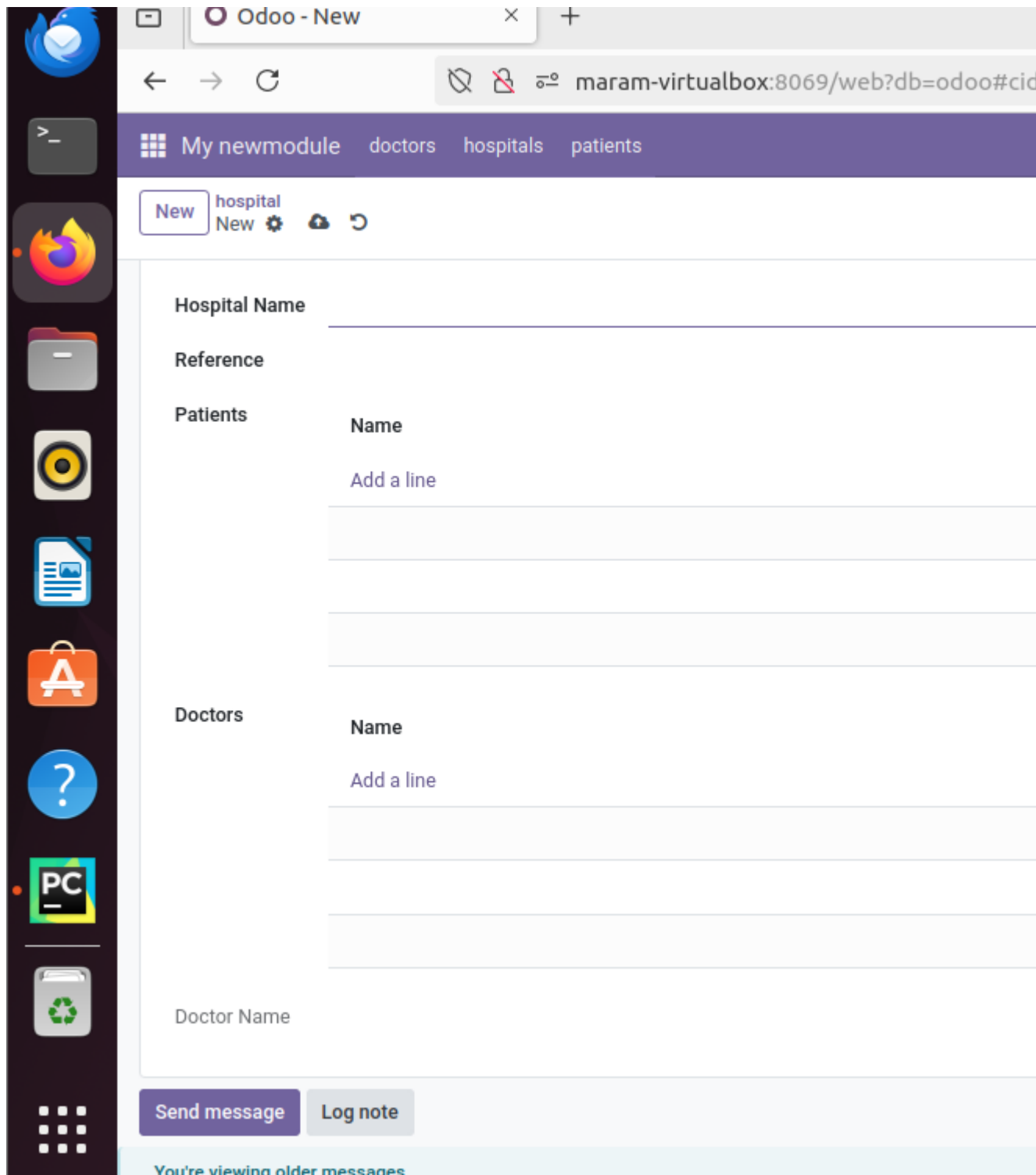
- Interface principale du module d'apprentissage
- Exemples de formulaires de création de cours et de sessions de formation














Activities


Firefox Web Browser


13:4





























Odoo - My New Module

← → ↻

🔒 🔒 🌐 maram-virtualbox:8069/web?db=odoo#ac

My newmodule

doctors

hospitals

patients

My New Module

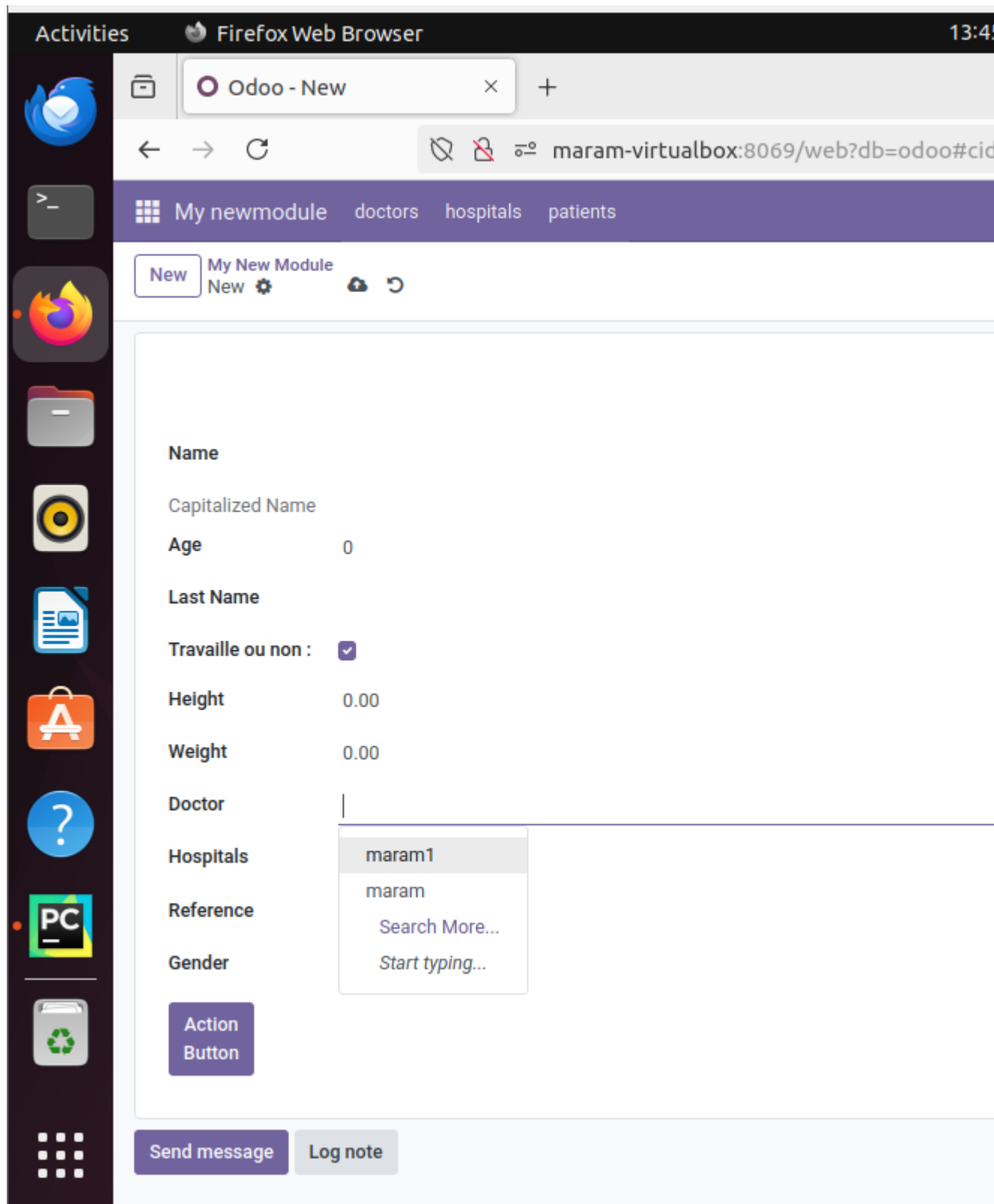
🔍 Search...

← → Week Today

June 2024

Week 26

	SUN 23	MON 24	TUE 25	WED 26
07:00				
08:00				
09:00				
10:00				
11:00				
12:00				
13:00				
14:00				
15:00				
16:00				
17:00				



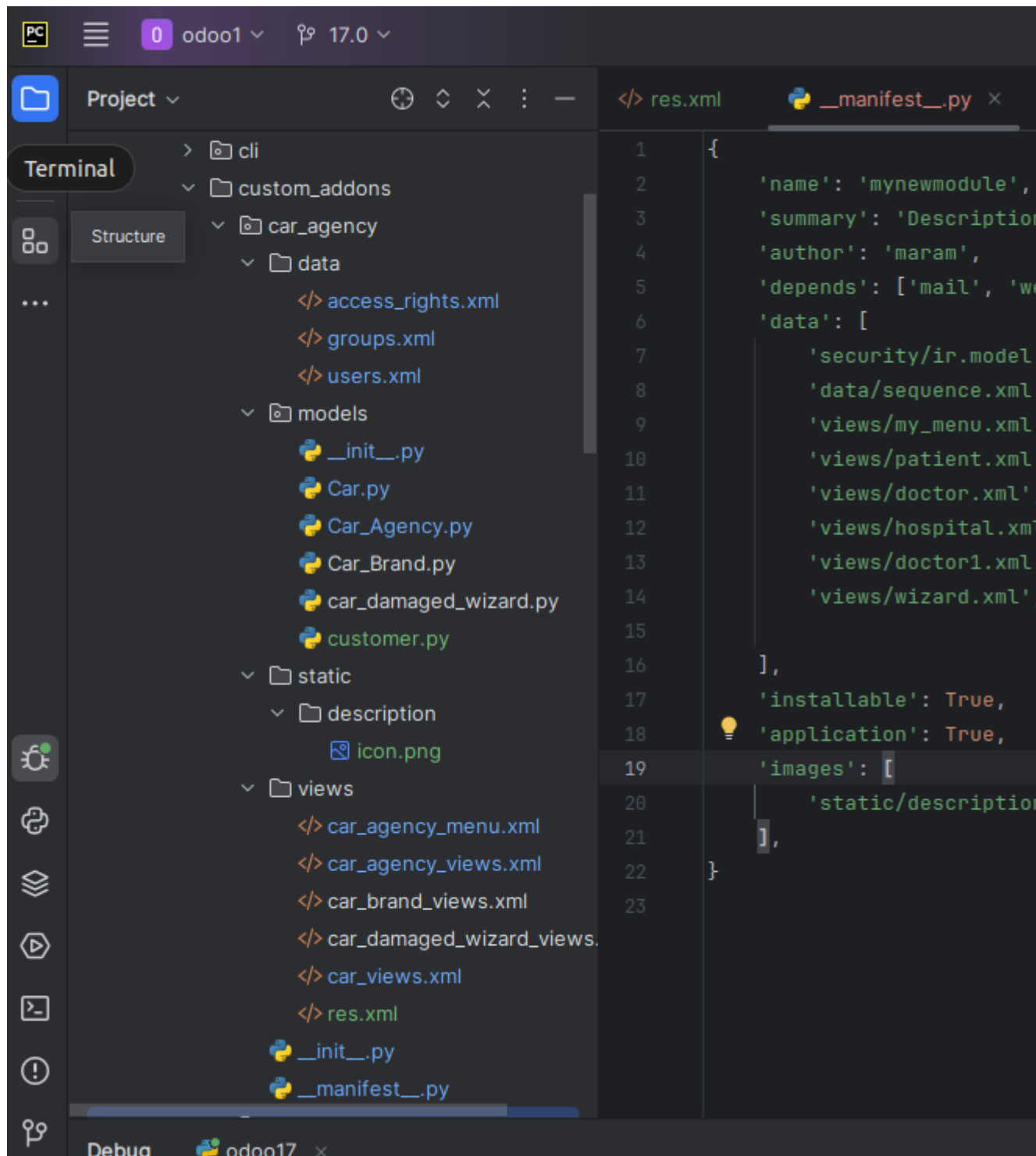
6.2 Développement d'un système de gestion d'agence automobile

Un autre projet important était la création d'un système complet de gestion d'agence automobile utilisant Odoo. Ce système permettait la gestion des stocks de véhicules, des

marques et modèles, ainsi que des ventes et transactions.

Captures d'écran :

- Vue d'ensemble du tableau de bord du système de gestion d'agence
- Exemples de pages de détail des véhicules et de gestion des ventes



es
Firefox Web Browser
15:02 27 جوان

Odoo - agency
x
+

← → ↻
maram-virtualbox:8069/web?db=oc

Car Agencies
agency
brand
Cars
Customer

New
agency
⚙️

🔍 Search...

<input type="checkbox"/> Name	Responsible
<input type="checkbox"/>	
<input type="checkbox"/> maram	Azure Interior
<input type="checkbox"/> marwa	Deco Addict
<input type="checkbox"/> agence	Deco Addict
<input type="checkbox"/>	Deco Addict, Addison Olson
<input type="checkbox"/> nada	Azure Interior

New
agency
New
⚙️

Name
Responsible
Cars
Agency Image

View Brands

←

→

↺

Car Agencies

New

Cars

New

⚙

Car Agencies

agency

brand

Cars

Customer

New

My brand

New

⚙

🔒

↺

Brand Name

Brand Image

Upload your file

Description

Agency

Registration Number

Car Model

Mileage

Status

Start Date

End Date

Agency

Customer

19

Car Agencies

agencybrandCarsCustomer

NewCars⚙

Q Search...

<input type="checkbox"/>	Registration Number	Car Model	Status
<input type="checkbox"/>	12345678	toyota	Available
<input type="checkbox"/>	87654321	zd	Damaged
<input type="checkbox"/>	12345678	toyota	Available
<input type="checkbox"/>	65478935		
<input type="checkbox"/>	56625863		
<input type="checkbox"/>	12345678		
<input type="checkbox"/>	15972574		
<input type="checkbox"/>	45652626		

Mark Car as Damaged

Damage Note

Save

Cancel

Car Agencies

agencybrandCarsCustomer

New

Customer
New

Name

CIN Number

Email

Phone

admin1

Azure Interior

Azure Interior, Brandon Freeman

Azure Interior, Colleen Diaz

Azure Interior, Nicole Ford

Deco Addict

Deco Addict, Addison Olson

Deco Addict, Douglas Fletcher

Search More...

Start typing...

6.3 Tests et validation des fonctionnalités

En plus du développement, j'ai également été impliqué dans les phases de test et de validation des fonctionnalités développées, en m'assurant que chaque module répondait aux spécifications et aux normes de qualité définies par l'équipe.

7 Détails du Projet Car Agency

Le projet Car Agency a consisté à développer un module Odoo pour la gestion des agences automobiles. Ce module inclut plusieurs fonctionnalités telles que la gestion des marques de voitures, des véhicules, et des agences, ainsi que l'association des clients aux véhicules loués.

7.1 Structure du Module

Le module est composé de plusieurs fichiers principaux :

- __manifest__.py** : Ce fichier contient les informations de base sur le module, telles que le nom, la version, les dépendances, et les données à charger.

- __init__.py** : Ce fichier initialise le module et charge les différentes parties du code.

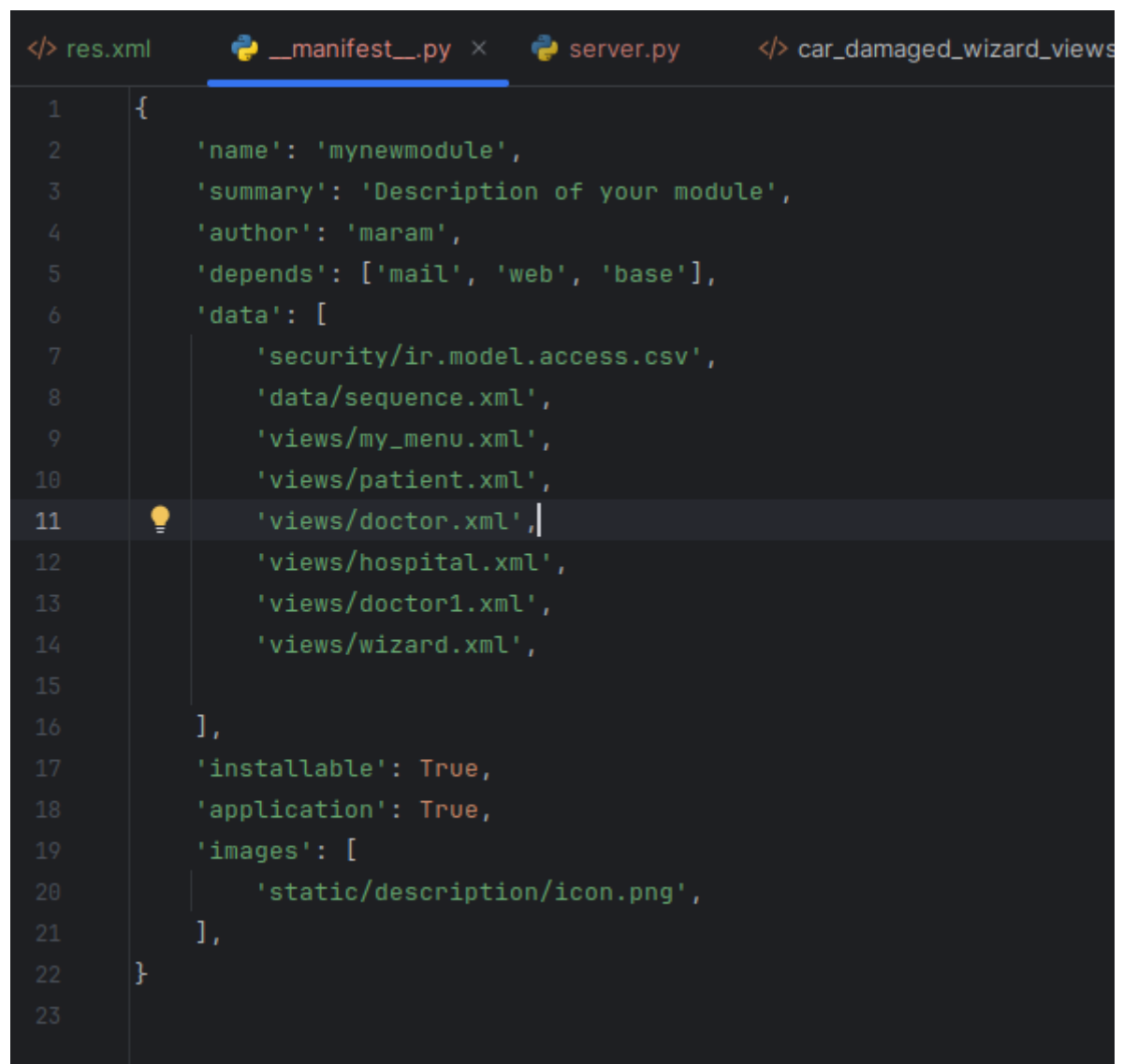
- models/** : Ce dossier contient les fichiers Python qui définissent les modèles de données utilisés par le module.

- views/** : Ce dossier contient les fichiers XML qui définissent les vues de l'interface utilisateur pour le module.

- security/** : Ce dossier contient les fichiers de sécurité, tels que les règles d'accès pour les différents modèles.

7.2 Fichier __manifest__.py

Le fichier `__manifest__.py` est essentiel car il décrit le module et ses dépendances.



```
</> res.xml  __manifest__.py ×  server.py  </> car_damaged_wizard_views

1      {
2          'name': 'mynewmodule',
3          'summary': 'Description of your module',
4          'author': 'maram',
5          'depends': ['mail', 'web', 'base'],
6          'data': [
7              'security/ir.model.access.csv',
8              'data/sequence.xml',
9              'views/my_menu.xml',
10             'views/patient.xml',
11             'views/doctor.xml',
12             'views/hospital.xml',
13             'views/doctor1.xml',
14             'views/wizard.xml',
15
16         ],
17         'installable': True,
18         'application': True,
19         'images': [
20             'static/description/icon.png',
21         ],
22     }
23
```

7.3 Fichier `__init__.py`

Le fichier `__init__.py` initialise les modèles et autres composants du module :

```
from . import models
```

7.4 Modèles Python

Les modèles sont définis dans le dossier **models/**.

```

class Car(models.Model):
    """
    registration_number = fields.CharField(required=True, size=8)
    model = fields.CharField(required=True)
    price = fields.FloatField()
    status = fields.Selection([
        'available', 'Available'),
        'rented', 'Rented'),
        'damaged', 'Damaged')
    is_available = fields.BooleanField(required=True, default='available')
    purchase_date = fields.DateField()
    rental_date = fields.DateField()
    agency_id = fields.Many2One('car_agency', string='Agency', required=True)
    customer_id = fields.Many2One('res_partner', string='Customer')
    damage_note = fields.Text(string='Damage Note')
    brand_id = fields.Many2One('car_brand', string='Brand', required=True)

    @staticmethod
    def constraints('registration_number')
    def check_registration_number(self):
    for record in self:
        if len(record.registration_number) != 8 or not record.registration_number.isdigit():
            raise ValidationError('The registration number must be a positive integer with e

```



```

class Car(models.Model):
    registration_number = fields.Char(required=True, size=8)
    car_model = fields.Char(required=True)
    mileage = fields.Float()
    status = fields.Selection([
        ('available', 'Available'),
        ('rented', 'Rented'),
        ('damaged', 'Damaged')
    ], required=True, default='available')
    start_date = fields.Date()
    end_date = fields.Date()
    agency_id = fields.Many2one('car_agency', string='Agency', required=True)
    customer_id = fields.Many2one('res', string='Customer')
    note = fields.Text(string='Damage Note')
    brand_id = fields.Many2one('car_brand', string='Brand', required=True)

    @odoo *
    @api.constrains('registration_number')
    def _check_registration_number(self):
        for record in self:
            if len(record.registration_number) != 8 or not record.registration_number.isdigit():
                raise ValidationError('The registration number must be a positive integer')

```

```

from odoo import models, fields, api
from odoo.exceptions import ValidationError

# odoo *
class CarAgency(models.Model):
    _name = 'car_agency'
    _description = 'Car Agency'

    name = fields.Char(required=True)
    responsible_id = fields.Many2one('res.partner', string='Responsible', required=True)
    car_ids = fields.One2many('car', 'agency_id', string='Cars')
    image = fields.Binary(string='Agency Image')
    brand_count = fields.Integer(string='Number of Brands', compute='_compute_brand_count')

    new *
    @api.depends('car_ids')
    def _compute_brand_count(self):
        for agency in self:
            agency.brand_count = self.env['car_brand'].search_count([('agency_id', '=', agency.id)])

# odoo *
def action_view_brands(self):
    self.ensure_one()
    return {
        'type': 'ir.actions.act_window',
        'name': f'Car Brands ({self.brand_count} Brands)',
        'view_mode': 'tree,form',
        'res_model': 'car_brand',
        'domain': [('agency_id', '=', self.id)]
    }

```

```
from odoo import models, fields
```

```
o odoo
```

```
class CarBrand(models.Model):
```

```
    _name = 'car_brand'
```

```
    _description = 'Car Brand'
```



```
    name = fields.Char(string='Brand Name', required=True)
```

```
    image = fields.Binary(string='Brand Image')
```

```
    description = fields.Text(string='Description')
```

```
    agency_id = fields.Many2one('car_agency', string='Agency', required=True)
```

```
from odoo import models, fields, api
```

```
from odoo.exceptions import ValidationError
```

```
new *
```

```
class ResPartner(models.Model):
```

```
    _inherit = 'res.partner'
```

```
    cin_number = fields.Char(string='CIN Number')
```

```
    phone = fields.Char(string='Phone')
```

```
    email = fields.Char(string='Email')
```

```
o odoo *
```

```
class Car(models.Model):
```

```
    _name = 'car'
```

```
    _description = 'Car'
```

```
    _sql_constraints = [
```

```
        ('unique_registration_number', 'unique(registration_number)', 'The registration number must be unique'),
```

```
    ]
```

```
    registration_number = fields.Char(required=True, size=8)
```

```
    car_model = fields.Char(required=True)
```

```
    mileage = fields.Float()
```

```
    status = fields.Selection([
```

```
        ('available', 'Available'),
```

```
        ('rented', 'Rented'),
```

```
        ('damaged', 'Damaged')
```

```
    ], required=True, default='available')
```

```
    start_date = fields.Date()
```

```

from odoo import models, fields

new *
class Res(models.Model):
    _name = 'res'

    cin_number = fields.Char(string='CIN Number')
    partner_id = fields.Many2one('res.partner', string='Name')
    email = fields.Char(related='partner_id.email', string='Email', readonly=True)
    phone = fields.Char(related='partner_id.phone', string='Phone', readonly=True)

```

```

from odoo import models, fields, api

o odoo
class CarDamageWizard(models.TransientModel):
    _name = 'car_damaged_wizard'
    _description = 'Car Damage Wizard'

    note = fields.Text(string='Damage Note')

o odoo
def save_damage_note_and_close(self):
    car_id = self.env.context.get('active_id')
    car = self.env['car'].browse(car_id)
    car.note = self.note
    car.status = 'damaged'
    return {'type': 'ir.actions.act_window_close'}

```

7.5 Vues

XML

Les vues définissent l'interface utilisateur et sont stockées dans le dossier.

```
__manifest__.py  Car.py  groups.xml  customer.py  car_agency_menu.  
<?xml version="1.0" encoding="utf-8"?>  
<odoo>  
  <data>  
    <menuitem  
      id="menu_car_agency_root"  
      name="Car Agencies"  
      web_icon="car_agency,static/description/img.png"  
      sequence="10"/>  
    <menuitem  
      id="agency"  
      name="agency"  
      parent="menu_car_agency_root"  
      sequence="10"/>  
    <menuitem  
      id="brand"  
      name="brandcar"  
      parent="menu_car_agency_root"  
      sequence="10"/>  
    <menuitem  
      id="menu_car"  
      name="Cars"  
      parent="menu_car_agency_root"  
      action="action_car"/>  
  </data>  
</odoo>
```

```
<odoo>
```

```
  <record id="view_car_agency_tree" model="ir.ui.view">
```

```
    <field name="name">car_agency.tree</field>
```

```
    <field name="model">car_agency</field>
```

```
    <field name="arch" type="xml">
```

```
      <tree>
```

```
        <field name="name"/>
```

```
        <field name="responsible_id"/>
```

```
      </tree>
```

```
    </field>
```

```
</record>
```

```
<record id="view_car_agency_form" model="ir.ui.view">
```

```
  <field name="name">car_agency.form</field>
```

```
  <field name="model">car_agency</field>
```

```
  <field name="arch" type="xml">
```

```
    <form>
```

```
      <sheet>
```

```
        <group>
```

```
          <field name="name"/>
```

```
          <field name="responsible_id"/>
```

```
          <field name="car_ids"/>
```

```
          <field name="image" widget="image"/>
```

```
        </group>
```

```
        <button name="action_view_brands" type="object" string="View Brands">
```

```
      </sheet>
```

```
    </form>
```

```
vizard.py  __manifest__.py  Car.py  customer.py  </> car_agency_menu.xml

<odoo>
    <record id="view_car_agency_form" model="ir.ui.view">
        <field name="arch" type="xml">
            <form>
            </form>
        </field>
    </record>

    <record id="action_agency" model="ir.actions.act_window">
        <field name="name">agency</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">car_agency</field>
        <field name="view_mode">tree,form,pivot,graph</field>
    </record>

    <menuitem
        id="caragency"
        name="agency"
        parent="menu_car_agency_root"
        action="action_agency"
        sequence="10"/>
</odoo>
```

```

<?xml version="1.0" encoding="utf-8"?>
<odoo>

  <record id="view_car_brand_tree" model="ir.ui.view">
    <field name="name">car_brand.tree</field>
    <field name="model">car_brand</field>
    <field name="arch" type="xml">
      <tree>
        <field name="name"/>
        <field name="agency_id"/>
      </tree>
    </field>
  </record>

  <record id="view_car_brand_form" model="ir.ui.view">
    <field name="name">car_brand.form</field>
    <field name="model">car_brand</field>
    <field name="arch" type="xml">
      <form>
        <group>
          <field name="name"/>
          <field name="image"/>
          <field name="description"/>
          <field name="agency_id"/>
        </group>
      </form>
    </field>
  </record>

  <record id="action_brand" model="ir.actions.act_window">

```




```
customer.py  car_agency_menu.xml  car_agency_views.xml

<odoo>
    <record id="view_car_brand_form" model="ir.ui.view">
        <field name="arch" type="xml">
            <form>
                </group>
            </form>
        </field>
    </record>
    <record id="action_brand" model="ir.actions.act_window">
        <field name="name">My brand</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">car_brand</field>
        <field name="view_mode">tree,form</field>
    </record>
    <menuitem id="b"
        name="brand"
        parent="menu_car_agency_root"
        action="action_brand"
        sequence="10"/>

</odoo>
```

```

agency_menu.xml    </> car_agency_views.xml    </> car_brand_views.xml    </> car_damaged_wizard_vie
1    <odoo>
2     <record id="view_car_damage_wizard" model="ir.ui.view">
3        <field name="name">car_damaged_wizard</field>
4        <field name="model">car_damaged_wizard</field>
5        <field name="arch" type="xml">
6            <form string="Car Damage">
7                <group>
8                    <field name="note"/>
9                </group>
10               <footer>
11                   <button string="Save" type="object" name="save_damage_note_and
12                   <button string="Cancel" class="btn btn-secondary" special="car
13               </footer>
14           </form>
15       </field>
16   </record>
17
18   <record id="action_car_damaged_wizard" model="ir.actions.act_window">
19       <field name="name">Mark Car as Damaged</field>
20       <field name="res_model">car_damaged_wizard</field>
21       <field name="view_mode">form</field>
22       <field name="target">new</field>
23   </record>
24 </odoo>

```

```

<odoo>
  <record id="view_car_tree" model="ir.ui.view">
    <field name="name">car.tree</field>
    <field name="model">car</field>
    <field name="arch" type="xml">
      <tree>
        <field name="registration_number"/>
        <field name="car_model"/>
        <field name="status"/>
        <field name="agency_id"/>
        <button name="%(car_agency.action_car_damaged_wizard)d" type="button" value="Car Damaged Wizard" />
      </tree>
    </field>
  </record>

  <record id="view_car_form" model="ir.ui.view">
    <field name="name">car.form</field>
    <field name="model">car</field>
    <field name="arch" type="xml">
      <form>
        <group>
          <field name="registration_number"/>
          <field name="car_model"/>
          <field name="mileage"/>
        </group>
      </form>
    </field>
  </record>

```

```

<odoo>
  <record id="view_car_tree" model="ir.ui.view">
    <field name="name">car.tree</field>
    <field name="model">car</field>
    <field name="arch" type="xml">
      <tree>
        <field name="registration_number"/>
        <field name="car_model"/>
        <field name="status"/>
        <field name="agency_id"/>
        <button name="%(car_agency.action_car_damaged_wizard)d" type="action" s
      </tree>
    </field>
  </record>

  <record id="view_car_form" model="ir.ui.view">
    <field name="name">car.form</field>
    <field name="model">car</field>
    <field name="arch" type="xml">
      <form>
        <group>
          <field name="registration_number"/>
          <field name="car_model"/>
          <field name="mileage"/>
          <field name="status" invisible="status == 'damaged'"/>
          <field name="start_date"/>
          <field name="end_date"/>
        </group>
      </form>
    </field>
  </record>

```

```

<odoo>
  <record id="view_car_form" model="ir.ui.view">
    <field name="arch" type="xml">
      <form>
        <group>
          <field name="mileage"/>
          <field name="status" invisible="status == 'damaged'"/>
          <field name="start_date"/>
          <field name="end_date"/>
          <field name="agency_id"/>
          <field name="customer_id"/>
          <field name="note" invisible="status != 'damaged'"/>
        </group>
      </form>
    </field>
  </record>

  <record id="action_car" model="ir.actions.act_window">
    <field name="name">Cars</field>
    <field name="res_model">car</field>
    <field name="view_mode">tree,form</field>
  </record>
</odoo>

```

```
</> res.xml x Car_Agency.py Car_Brand.py car_damaged_wizard.py cus
1 <odoo>
2   <record id="view_res_tree" model="ir.ui.view">
3     <field name="name">res.tree</field>
4     <field name="model">res</field>
5     <field name="arch" type="xml">
6       <tree>
7         <field name="partner_id"/>
8         <field name="cin_number"/>
9         <field name="email"/>
10        <field name="phone"/>
11      </tree>
12    </field>
13  </record>
14
15  <record id="view_res_form" model="ir.ui.view">
16    <field name="name">res.form</field>
17    <field name="model">res</field>
18    <field name="arch" type="xml">
19      <form>
20        <group>
21          <field name="partner_id"/>
22          <field name="cin_number"/>
23          <field name="email"/>
24          <field name="phone"/>
25        </group>
26      </form>
27    </field>
28  </record>
```

```
</> res.xml × Car_Agency.py Car_Brand.py car_damaged_wizard.py
1      <odoo>
15          <record id="view_res_form" model="ir.ui.view">
18              <field name="arch" type="xml">
19                  <form>
20                      </group>
26                  </form>
27              </field>
28          </record>
29
30          <record id="action_res" model="ir.actions.act_window">
31              <field name="name">Customer</field>
32              <field name="type">ir.actions.act_window</field>
33              <field name="res_model">res</field>
34              <field name="view_mode">tree,form</field>
35          </record>
36
37          <menuitem id="res"
38              name="Customer"
39              parent="menu_car_agency_root"
40              action="action_res"
41              sequence="10"/>
42      </odoo>
```

7.6 Access rights et création de users admin et user normal et création d'un groupe :

La gestion des droits d'accès et la création d'utilisateurs avec différents niveaux de privilèges sont des aspects cruciaux de la personnalisation d'Odoo. Voici comment procéder pour créer des utilisateurs administrateurs et normaux, ainsi qu'un groupe personnalisé.

```
</> res.xml  Car.py  </> access_rights.xml  car_damaged_wizard.py

1  <odoo>
2  <data>
3
4      <!-- Access rights for Car model -->
5      <record id="access_car" model="ir.model.access">
6          <field name="name">Access Car</field>
7          <field name="model_id" ref="model_car"/>
8          <field name="group_id" ref="custom_group_car_access"/>
9          <field name="perm_read" eval="1"/>
10         <field name="perm_write" eval="1"/>
11         <field name="perm_create" eval="1"/>
12         <field name="perm_unlink" eval="1"/>
13     </record>
14
15     <!-- Access rights for Car Brand model -->
16     <record id="access_car_brand" model="ir.model.access">
17         <field name="name">Access Car Brand</field>
18         <field name="model_id" ref="model_car_brand"/>
19         <field name="group_id" ref="custom_group_car_access"/>
20         <field name="perm_read" eval="1"/>
21         <field name="perm_write" eval="1"/>
22         <field name="perm_create" eval="1"/>
23         <field name="perm_unlink" eval="1"/>
24     </record>
25
26     <!-- Access rights for Car Agency model -->
27     <record id="access_car_agency" model="ir.model.access">
28         <field name="name">Access Car Agency</field>
```



```

<!-- Access rights for Car Damaged Wizard model -->
<record id="access_car_damaged_wizard" model="ir.model.access">
    <field name="name">Access Car Damaged Wizard</field>
    <field name="model_id" ref="model_car_damaged_wizard"/>
    <field name="group_id" ref="custom_group_car_access"/>
    <field name="perm_read" eval="1"/>
    <field name="perm_write" eval="1"/>
    <field name="perm_create" eval="1"/>
    <field name="perm_unlink" eval="1"/>
</record>

<!-- Access rights for Res Partner model -->
<record id="access_res_partner" model="ir.model.access">
    <field name="name">Access Res Partner</field>
    <field name="model_id" ref="model_res"/>
    <field name="group_id" ref="custom_group_car_access"/>
    <field name="perm_read" eval="1"/>
    <field name="perm_write" eval="1"/>
    <field name="perm_create" eval="1"/>
    <field name="perm_unlink" eval="1"/>
</record>

```

Ce fichier définit les permissions pour deux groupes d'utilisateurs : les utilisateurs normaux et les administrateurs (managers).

7.6.1 Utilisateurs

Pour créer des utilisateurs, allez dans l'interface Odoo sous **Paramètres** , **Utilisateurs** , **Entreprises** , **Utilisateurs** et cliquez sur **Créer**. Remplissez les informations nécessaires pour chaque utilisateur et affectez les utilisateurs au groupe.ou bien avec le code :

<odoo>

<data>

<record id="custom_user_1" model="res.users">

<field name="name">admin1</field>

<field name="login">admin1</field>

<field name="email">admin1@example.com</field>

<field name="groups_id" eval="[(6, 0, [ref('custom_group_car_access')])]"></field>

<field name="password">admin1</field>

<field name="groups_id" eval="[(4, ref('base.group_erp_manager'))]"></field>

</record>

<record id="custom_user_2" model="res.users">

<field name="name">user</field>

<field name="login">user</field>

<field name="email">user@example.com</field>

<field name="groups_id" eval="[(6, 0, [ref('custom_group_car_access')])]"></field>

<field name="password">user</field>

</record>

7.6.2 Groupe

Pour créer un groupe personnalisé, voici les étapes :

1. Allez dans **Paramètres , Utilisateurs , Entreprises , Groupes**.
2. Cliquez sur **Créer**.
3. Remplissez le formulaire pour le groupe personnalisé, par exemple "Car Agency Manager".
4. Ajoutez les règles d'accès et les utilisateurs au groupe. Ou bien avec le code :

```

<odoo>
  <data noupdate="1">
    <!-- Custom Group -->
    <record id="custom_group_car_access" model="res.groups">
      <field name="name">Custom Group Car Access</field>
      <field name="category_id" ref="base.module_category_hidden"/>
      <field name="implied_ids" eval="[(4, ref('base.group_user')))]"/>
    </record>
    <record id="access_custom_group_car_access_ir_module_module" model="ir.model.access" >
      <field name="name">Access Custom Group Car Access to ir.module.module</field>
      <field name="model_id" ref="base.model_ir_module_module"/>
      <field name="group_id" ref="custom_group_car_access"/>
      <field name="perm_read" eval="True"/>
      <field name="perm_write" eval="True"/>
      <field name="perm_create" eval="True"/>
      <field name="perm_unlink" eval="True"/>
    </record>
    <!-- Règles d'accès -->
    <record id="access_custom_group_car_access_ir_module_module" model="ir.model.access" >
      <field name="name">access_custom_group_car_access_ir_module_module</field>
      <field name="model_id" ref="base.model_ir_module_module"/>
      <field name="group_id" ref="custom_group_car_access"/>
      <field name="perm_read" eval="True"/>
      <field name="perm_write" eval="True"/>
      <field name="perm_create" eval="True"/>
      <field name="perm_unlink" eval="True"/>
    </record>
  </data>
</odoo>

```

```

<data nouupdate="1">
  <record id="access_custom_group_car_access_ir_module_module_exclusion" model="ir.mo
    <field name="name">access_custom_group_car_access_ir_module_module_exclusion</f
    <field name="model_id" ref="base.model_ir_module_module_exclusion"/>
    <field name="group_id" ref="custom_group_car_access"/>
    <field name="perm_read" eval="True"/>
    <field name="perm_write" eval="True"/>
    <field name="perm_create" eval="True"/>
    <field name="perm_unlink" eval="True"/>
  </record>
</data>
<data>
<record id="access_custom_group_car_access_ir_ui_view" model="ir.model.access">
  <field name="name">Access Custom Group Car Access to ir.ui.view</field>
  <field name="model_id" ref="base.model_ir_ui_view"/>
  <field name="group_id" ref="custom_group_car_access"/>
  <field name="perm_read" eval="True"/>
  <field name="perm_write" eval="True"/>
  <field name="perm_create" eval="True"/>
  <field name="perm_unlink" eval="True"/>
</record>
</data>
<record id="access_custom_group_car_access_website_page" model="ir.model.access">
  <field name="name">Access Custom Group Car Access to website.page</field>
  <field name="model_id" ref="website.model_website_page"/>
  <field name="group_id" ref="custom_group_car_access"/>
  <field name="perm_read" eval="True"/>

```

```

<record id="access_custom_group_car_access_ir_ui_view" model="ir.model.access">
  </record>
</data>
<record id="access_custom_group_car_access_website_page" model="ir.model.access">
  <field name="name">Access Custom Group Car Access to website.page</field>
  <field name="model_id" ref="website.model_website_page"/>
  <field name="group_id" ref="custom_group_car_access"/>
  <field name="perm_read" eval="True"/>
  <field name="perm_write" eval="True"/>
  <field name="perm_create" eval="True"/>
  <field name="perm_unlink" eval="True"/>
</record>
<record id="access_custom_group_car_access_website_page" model="ir.model.access">
<field name="name">Access Custom Group Car Access to website.page</field>
<field name="model_id" ref="website.model_website_page"/>
<field name="group_id" ref="custom_group_car_access"/>
<field name="perm_read" eval="True"/>
<field name="perm_write" eval="True"/>
<field name="perm_create" eval="True"/>
<field name="perm_unlink" eval="True"/>
</record>
<record id="access_custom_group_car_access_res_config_settings" model="ir.model.acce
  <field name="name">Access Custom Group Car Access to res.config.settings</fi
  <field name="model_id" ref="base.model_res_config_settings"/>
  <field name="group_id" ref="custom_group_car_access"/>
  <field name="perm_read" eval="True"/>

```

```

<record id="access_custom_group_car_access_website_page" model="ir.model.access">
  <field name="perm_create" eval="True"/>
  <field name="perm_unlink" eval="True"/>
</record>

<record id="access_custom_group_car_access_res_config_settings" model="ir.model.access">
  <field name="name">Access Custom Group Car Access to res.config.settings</field>
  <field name="model_id" ref="base.model_res_config_settings"/>
  <field name="group_id" ref="custom_group_car_access"/>
  <field name="perm_read" eval="True"/>
  <field name="perm_write" eval="True"/>
  <field name="perm_create" eval="True"/>
  <field name="perm_unlink" eval="True"/>
</record>

<record id="access_custom_group_car_access_all_base_models" model="ir.model.access">
  <field name="name">Access Custom Group Car Access to All Base Models</field>
  <field name="model_id" ref="base.model_res_groups"/>
  <field name="group_id" ref="custom_group_car_access"/>
  <field name="perm_read" eval="True"/>
  <field name="perm_write" eval="True"/>
  <field name="perm_create" eval="True"/>
  <field name="perm_unlink" eval="True"/>
</record>
</odoo>

```

7.7 Publication du Module sur GitHub et la Plateforme Odoo

Après avoir développé et testé mon module Odoo, j'ai décidé de le publier sur GitHub pour le rendre accessible à tous et contribuer à la communauté open-source. J'ai également utilisé le lien GitHub de mon dépôt pour le publier gratuitement sur la plateforme Odoo, permettant ainsi à toute personne utilisant Odoo d'accéder à mon module de gestion d'agence automobile.

Voici les étapes de publication:

1. **Création d'un dépôt GitHub** : J'ai créé un nouveau dépôt sur GitHub en fournissant un nom de dépôt.
2. **Initialisation du dépôt local** : Sur mon ordinateur local, j'ai initialisé un dépôt Git dans le répertoire de mon module en utilisant les commandes suivantes :

```
git init
```

3. **Ajout des fichiers au dépôt** : J'ai ajouté tous les fichiers de mon module au dépôt Git local :

```
git add .
```

4. **Commit des modifications** : J'ai ensuite commit les fichiers ajoutés avec un message descriptif :

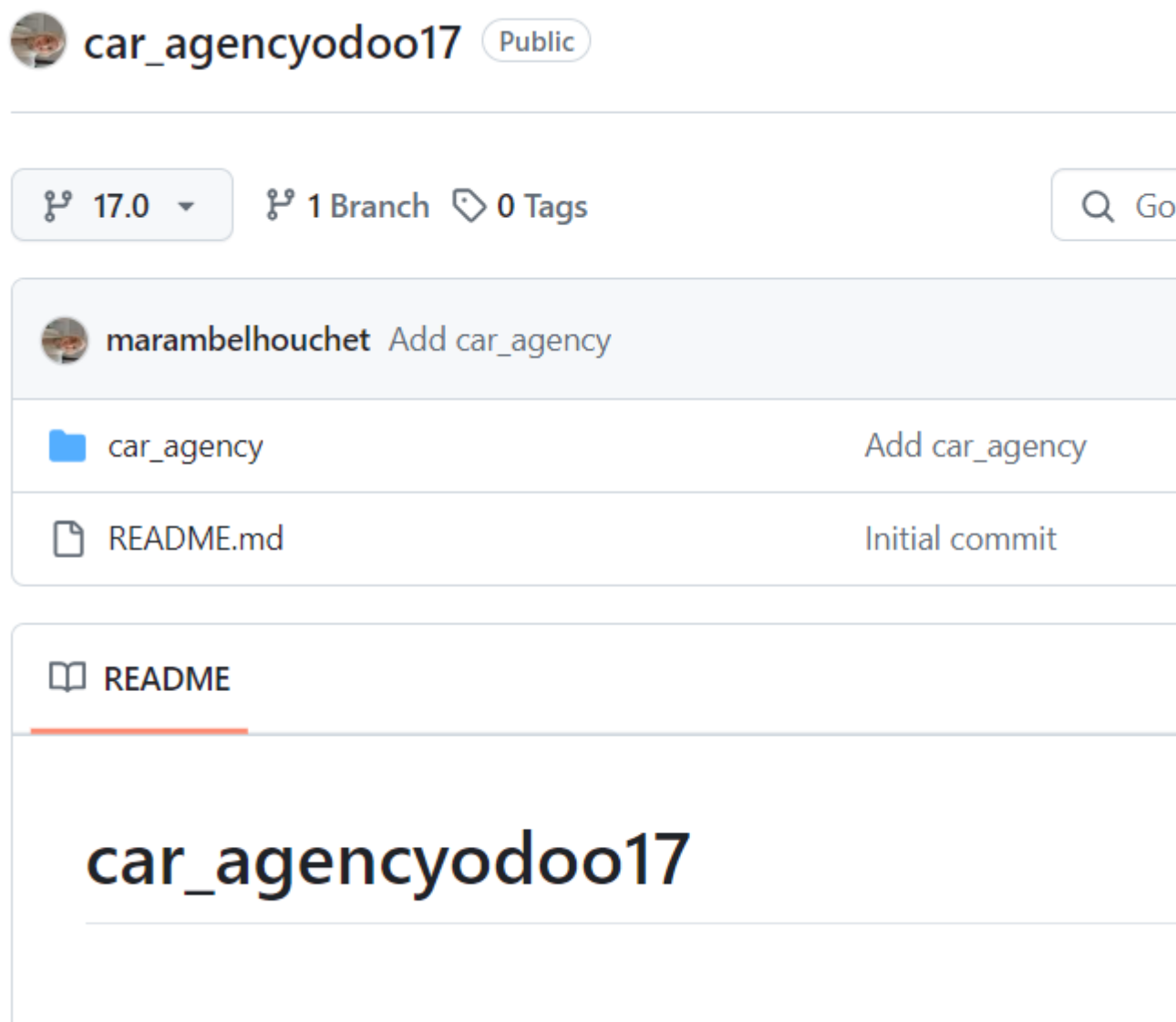
```
git commit -m "Initial commit of my Odoo module"
```

5. **Connexion à GitHub** : J'ai lié mon dépôt local au dépôt GitHub en utilisant l'URL du dépôt distant :

```
git remote add origin https://github.com/marambelhouchet/car_agencyodoo17.git
```

6. **Push des fichiers sur GitHub** : Enfin, j'ai poussé les fichiers de mon dépôt local vers GitHub :

```
git push -u origin 17.0
```



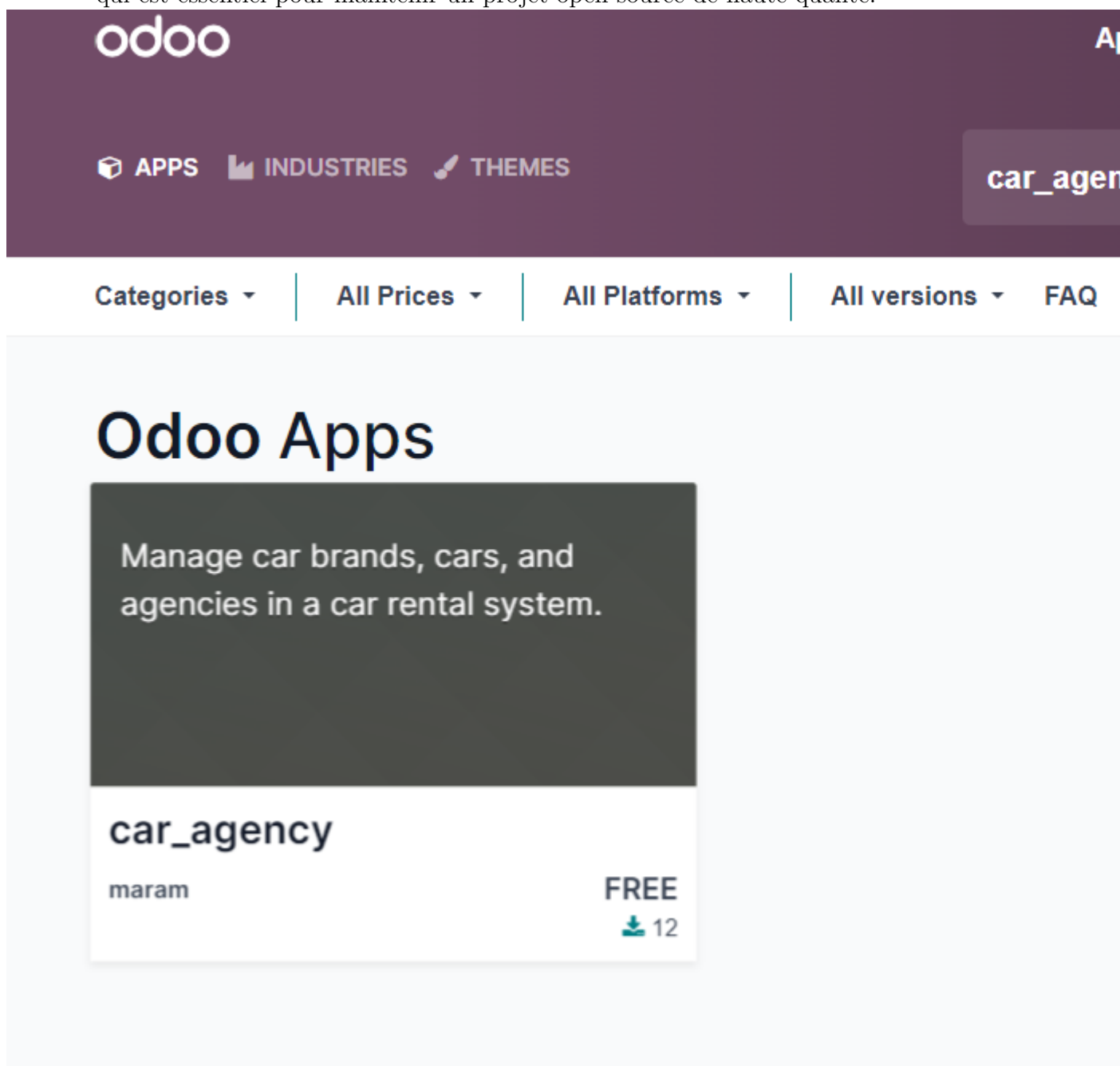
The screenshot shows the GitHub interface for a repository named 'car_agencyodoo17', which is marked as 'Public'. The repository is owned by 'marambelhouchet'. It shows 1 branch (17.0) and 0 tags. The commit history is displayed, showing a single commit 'Add car_agency' by 'marambelhouchet' with a commit message 'Add car_agency'. The commit includes two files: 'car_agency' (a folder) and 'README.md' (a file). The 'README.md' file is selected, and its content is displayed below, showing the repository name 'car_agencyodoo17' in a large font.

7. **Publication sur la plateforme Odoo** : J'ai utilisé le lien GitHub de mon dépôt pour publier le module sur la plateforme Odoo, le rendant ainsi disponible gratuitement pour tous les utilisateurs d'Odoo.

Accessibilité et Utilisation:

En publiant mon module sur GitHub et en le rendant accessible via la plateforme Odoo, j'ai élargi la portée de mon travail à un large public. Les développeurs et les utilisateurs de la plateforme Odoo peuvent désormais cloner ou forker le dépôt, et intégrer le module dans leurs propres installations Odoo. Cela facilite la collaboration, l'amélioration continue et l'adoption de nouvelles fonctionnalités par la communauté.

De plus, la publication sur GitHub permet de suivre les modifications, de gérer les versions et de recevoir des retours et contributions de la part d'autres développeurs, ce qui est essentiel pour maintenir un projet open-source de haute qualité.



The screenshot displays the Odoo Apps marketplace interface. At the top, the Odoo logo is visible on the left, and navigation links for 'APPS', 'INDUSTRIES', and 'THEMES' are in the center. A search bar on the right contains the text 'car_agen'. Below the navigation bar, there are filters for 'Categories', 'All Prices', 'All Platforms', 'All versions', and a link to 'FAQ'. The main section is titled 'Odoo Apps' and features a card for the 'car_agency' module. The card has a dark header with the text 'Manage car brands, cars, and agencies in a car rental system.' Below this, the module name 'car_agency' is shown, followed by the author 'maram'. The price is listed as 'FREE', and the download count is '12' with a download icon.



car_agency

by [maram](#)

v 17.0



Third Party



12

Download for v 17.0

Deploy on Odoo.sh

Availability

✗ Odoo Online

✓ Odoo.sh

✓ O

Odoo Apps Dependencies ⓘ

Discuss (mail)

Lines of code ⓘ

451

Technical Name

car_agency

You bought this module and need **support**? [Click here!](#)

8 Conclusion

Ce stage chez Softifi a été une expérience enrichissante et formatrice qui m'a permis d'approfondir mes connaissances en développement logiciel et en intégration de solutions ERP, notamment avec Odoo. Travailler au sein d'une équipe dynamique et passionnée m'a non seulement permis de mettre en pratique mes compétences théoriques, mais aussi d'acquérir de nouvelles perspectives sur les défis et les exigences du monde professionnel. La création de modules pour l'agence automobile et l'exploration des fonctionnalités avancées d'Odoo ont été des moments forts de mon apprentissage. Ce stage, bien que non obligatoire, a été une opportunité précieuse pour consolider mes compétences techniques et développer des compétences interpersonnelles essentielles telles que la collaboration d'équipe et la gestion de projet. J'ai particulièrement apprécié la liberté créative et la responsabilité qui m'ont été confiées dans la réalisation des projets, ce qui m'a permis de voir concrètement l'impact de mon travail sur les opérations de l'entreprise.

Je suis reconnaissant envers toute l'équipe de Softifi, en particulier mon encadrant Achraf Alioui, pour leur soutien continu, leurs conseils précieux et leur ambiance de travail stimulante. Ce stage a renforcé ma motivation à poursuivre une carrière dans le domaine du développement logiciel et m'a donné les outils nécessaires pour relever de nouveaux défis avec confiance à l'avenir.