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

Projet Odoo

Gestion des Notes Internes



Préparé par : EL MOUTAOUADI Ahmed Ali

Année universitaire : 2025–2026

Encadré par : Prof. Mohammed AIT DAOUD

Table des matières

1 Introduction

1.1 Contexte du Projet

Ce projet consiste en le développement d'un module Odoo complet pour la gestion des notes internes au sein d'une organisation. Le module a été conçu avec une attention particulière portée à l'expérience utilisateur, notamment avec l'implémentation d'un système de thème sombre/clair et d'une interface moderne et intuitive.

1.2 Objectifs du Projet

Les objectifs principaux de ce projet sont :

- Créer un système de gestion de notes interne complet et fonctionnel
- Implémenter un système de thème sombre/clair avec basculement dynamique
- Développer une interface utilisateur moderne et responsive
- Fournir des fonctionnalités avancées de gestion (priorités, favoris, échéances)
- Assurer une bonne expérience utilisateur avec des notifications et des retours visuels

1.3 Technologies Utilisées

- **Framework :** Odoo 15+
- **Langage de programmation :** Python 3
- **Base de données :** PostgreSQL
- **Interface :** XML (vues), CSS3, JavaScript
- **Architecture :** MVC (Model-View-Controller)

2 Architecture du Module

2.1 Structure du Projet

Le module suit la structure standard d'un module Odoo :

```
tp_gestion_notes/
          __init__.py
          __manifest__.py
          models/
                __init__.py
                note_interne.py
                user_preferences.py
          views/
                note_views.xml
                date_filter_wizard.xml
                wizard_view.xml
          wizards/
                __init__.py
                theme_config_wizard.py
                date_filter_wizard.py
          security/
                ir.model.access.csv
```

```
        static/
              src/
                  css/
                        dark_theme.css
                        components.css
                  js/
                      dark_theme.js
                      theme_config.js
        migrations/
            3.2.0/
                post-migration.py
```

2.2 Modèle de Données

2.2.1 Modèle Principal : tp.note.interne

Le modèle principal `tp.note.interne` hérite de `mail.thread` et `mail.activity.mixin` pour bénéficier des fonctionnalités de messagerie et d'activités d'Odoo.

Champs principaux :

- `titre` : Titre de la note (requis, indexé)
- `contenu` : Contenu HTML de la note
- `description` : Description courte de la note
- `auteur_id` : Auteur de la note (Many2one vers res.users)
- `date_note` : Date de création de la note
- `date_echeance` : Date d'échéance (nouveau)
- `statut` : Statut de la note (brouillon, publié, archivé)
- `priority` : Priorité (Basse, Normale, Haute, Urgente) (nouveau)
- `is_favorite` : Indicateur de favori (nouveau)
- `is_overdue` : Calculé automatiquement si la note est en retard
- `theme_couleur` : Couleur du thème
- `is_dark_mode` : Mode sombre activé

2.2.2 Méthodes Principales

```python
def action_publier(self):
    """Publie la note avec notification am lior e"""

def action_archiver(self):
    """Archive la note avec notification"""

def action_restaurer(self):
    """Restaure une note archiv e"""

def action_toggle_favorite(self):
    """Toggle le statut favori d'une note"""

def get_statistics(self):
    """Retourne les statistiques des notes pour le dashboard"""
```

3 Fonctionnalités Principales

3.1 Gestion des Notes

3.1.1 Création et Édition

Le module permet de créer et éditer des notes avec :

— Un éditeur HTML riche pour le contenu
— Un système de titre et description
— Attribution automatique de l'auteur
— Gestion de la date de création

3.1.2 Statuts des Notes

Le système de statuts permet de gérer le cycle de vie des notes :

1. **Brouillon** : Note en cours de rédaction
2. **Publié** : Note finalisée et visible
3. **Archivé** : Note archivée mais conservée

3.2 Système de Priorité

Un système de priorité à 4 niveaux a été implémenté :

Niveau	Description
0	Basse
1	Normale (par défaut)
2	Haute ()
3	Urgente ()

TABLE 1 – Système de priorité

3.3 Système de Favoris

Les utilisateurs peuvent marquer des notes comme favorites pour un accès rapide :

— Bouton étoile dans toutes les vues
— Filtre dédié "Favoris"
— Indicateur visuel dans les cartes Kanban
— Tri automatique des favoris en premier

3.4 Gestion des Échéances

— Champ `date_echeance` pour définir une date limite
— Calcul automatique des notes en retard (`is_overdue`)
— Filtre "En retard" dans la recherche
— Indicateur visuel avec animation pour les notes en retard

3.5 Filtrage et Recherche

Le module offre plusieurs options de filtrage :

- **Mes Notes** : Notes de l'utilisateur connecté
- **Favoris** : Notes marquées comme favorites
- **Par statut** : Brouillons, Publiés, Archivés
- **Urgentes** : Notes avec priorité urgente
- **En retard** : Notes avec échéance dépassée
- **Par date** : Filtrage par date spécifique via wizard

3.6 Wizard de Filtrage par Date

Un wizard dédié permet de filtrer les notes par date :

- Sélection de date via un calendrier
- Affichage de toutes les notes de la date sélectionnée
- Vue filtrée avec titre dynamique

4 Interface Utilisateur

4.1 Vues Disponibles

4.1.1 Vue Liste (Tree View)

La vue liste offre :

- Édition inline des notes
- Colonnes : Favori, Priorité, Titre, Auteur, Date, Échéance, Statut
- Actions rapides : Publier, Archiver, Restaurer, Supprimer
- Décoration visuelle selon le statut
- Tri par défaut : Date décroissante

4.1.2 Vue Formulaire (Form View)

La vue formulaire comprend :

- En-tête avec actions principales
- Barre de statut interactive
- Section titre avec style amélioré
- Zone de description
- Métadonnées (Priorité, Auteur, Dates)
- Notebook avec onglets :
  - Contenu : Éditeur HTML
  - Informations : Suivi et historique

4.1.3 Vue Kanban

La vue Kanban offre :

- Cartes stylisées avec gradients
- Groupement par statut par défaut
- Indicateurs visuels :
  - Étoile pour les favoris
  - Icône feu pour les urgentes
  - Bordure colorée pour les priorités
- Métadonnées : Auteur avec avatar, Date relative
- Description tronquée automatiquement

4.2 Système de Thème

4.2.1 Thème Sombre

Un thème sombre complet a été implémenté avec :

- Variables CSS pour la cohérence des couleurs
- Palette de couleurs optimisée :
  - Primaire : #1a1d21
  - Secondaire : #2d3136
  - Tertiaire : #3d4249
  - Accent : #4a90e2
- Adaptation de tous les composants Odoo
- Scrollbar personnalisée

4.2.2 Bouton de Basculement

Un bouton flottant permet de basculer entre les thèmes :

- Position : Bas droite de l'écran
- Icône dynamique : Lune (mode clair) / Soleil (mode sombre)
- Animation au survol
- Persistance dans localStorage

4.3 Améliorations Visuelles

4.3.1 Design Moderne

- **Cards** : Bordures arrondies, ombres portées, effets de survol
- **Boutons** : Gradients, animations, icônes
- **Typographie** : Hiérarchie claire, espacements optimisés
- **Couleurs** : Palette cohérente avec indicateurs visuels

4.3.2 Animations et Transitions

— Transitions fluides (0.3s ease)
— Effets de survol sur les cartes
— Animation pulse pour les notes en retard
— Transformations au survol des boutons

4.3.3 État Vide Amélioré

L'état vide (quand il n'y a pas de notes) affiche :
— Grande icône stylisée
— Titre accrocheur
— Description informative
— Boîte d'astuce avec icône

5 Améliorations Techniques

5.1 Performance

5.1.1 Indexation

Des index ont été ajoutés sur les champs fréquemment recherchés :
— `titre` : Recherche textuelle
— `auteur_id` : Filtrage par auteur
— `date_note` : Tri et filtrage par date
— `statut` : Filtrage par statut
— `priority` : Tri par priorité
— `is_favorite` : Filtrage des favoris

5.1.2 Optimisation des Requêtes

— Utilisation de `search_fetch` pour les grandes listes
— Calculs optimisés pour les statistiques
— Mise en cache des préférences utilisateur

5.2 Sécurité

5.2.1 Droits d'Accès

Le fichier `ir.model.access.csv` définit :
— Accès complet pour le modèle `tp.note.interne`
— Accès au wizard de filtrage par date
— Permissions : Lecture, Écriture, Création, Suppression

5.2.2 Validation des Données

Des contraintes ont été ajoutées :

— Validation de la date de note

— Validation de la date d'échéance

— Vérification de cohérence entre les dates

5.3 Migrations

Un script de migration a été créé pour :

— Ajouter les nouvelles colonnes (`priority`, `is_favorite`, `date_echeance`)

— Créer les index nécessaires

— Préserver les données existantes

— Appliquer les valeurs par défaut

6 Notifications et Retours Utilisateur

6.1 Système de Notifications

Toutes les actions importantes génèrent des notifications :

— **Publication** : "X note(s) publiée(s) avec succès"

— **Archivage** : "X note(s) archivée(s) avec succès"

— **Restauration** : "X note(s) restaurée(s) avec succès"

— **Suppression** : "X note(s) supprimée(s) avec succès"

— **Favoris** : "Note ajoutée aux/retirée des favoris"

6.2 Messages de Suivi

Le système de messagerie Odoo est utilisé pour :

— Enregistrer les changements de statut

— Historique des actions

— Traçabilité complète

7 Code Source

7.1 Exemple de Modèle

```
from odoo import models, fields, api

class TpNoteInterne(models.Model):
    _name = "tp.note.interne"
    _description = "Note Interne"
    _inherit = ['mail.thread', 'mail.activity.mixin']
    _order = 'date_note desc, create_date desc'

```

```
    titre = fields.Char(string="Titre", required=True,
                        tracking=True, index=True)
    contenu = fields.Html(string="Contenu")
    auteur_id = fields.Many2one("res.users",
                                default=lambda self: self.env.user,
                                tracking=True, index=True)
    date_note = fields.Date(string="Date",
                            default=fields.Date.today,
                            tracking=True, index=True)
    statut = fields.Selection([
        ("brouillon", "Brouillon"),
        ("publie", "Publi "),
        ("archive", "Archiv "),
    ], string="Statut", default="brouillon",
       tracking=True, index=True)

    priority = fields.Selection([
        ('0', 'Basse'),
        ('1', 'Normale'),
        ('2', 'Haute'),
        ('3', 'Urgente'),
    ], string="Priorit ", default='1',
       tracking=True, index=True)

    is_favorite = fields.Boolean(string="Favori",
                                default=False,
                                tracking=True, index=True)
    date_echeance = fields.Date(string="Date d'   chance   ",
                                tracking=True)

    @api.depends('date_echeance', 'statut')
    def _compute_is_overdue(self):
        today = fields.Date.today()
        for record in self:
            record.is_overdue = (
                record.date_echeance
                and record.date_echeance < today
                and record.statut != 'archive'
            )
```

Listing 1 – Modèle tp.note.interne (extrait)

7.2 Exemple de Vue

```
<tree string="Notes Internes"
      editable="bottom"
      class="tp_note_dark tp_list_view"
      decoration-success="statut == 'publie'"
      decoration-warning="statut == 'brouillon'"
      decoration-muted="statut == 'archive'"
      default_order="date_note desc">

    <field name="is_favorite" widget="boolean_toggle"/>
    <field name="priority" widget="priority"/>
    <field name="titre" string="Titre"/>
    <field name="auteur_id" widget="many2one_avatar_user"/>
    <field name="date_note" widget="date"/>
```

```
14      <field name="statut" widget="badge"/>
15
16      <button name="action_toggle_favorite"
17              type="object"
18              icon="fa-star"
19              invisible="not is_favorite"/>
20      <button name="action_publier"
21              type="object"
22              string="Publier"
23              icon="fa-check"
24              invisible="statut != 'brouillon'"/>
25  </tree>
```

Listing 2 – Vue Liste (extrait)

8 Conclusion

8.1 Résultats Obtenus

Le module développé répond aux objectifs initiaux :

- Système complet de gestion de notes internes
- Interface moderne et intuitive
- Thème sombre/clair fonctionnel
- Fonctionnalités avancées (priorités, favoris, échéances)
- Bonne expérience utilisateur
- Code maintenable et extensible

8.2 Perspectives d'Amélioration

Plusieurs améliorations pourraient être apportées :

- **Catégories et Tags** : Système de catégorisation avancé
- **Recherche Full-Text** : Recherche dans le contenu HTML
- **Export PDF** : Génération de PDF pour les notes
- **Partage** : Partage de notes entre utilisateurs
- **Commentaires** : Système de commentaires collaboratif
- **Dashboard** : Tableau de bord avec statistiques visuelles
- **Notifications Email** : Alertes par email pour les échéances
- **API REST** : API pour intégration externe

8.3 Apprentissages

Ce projet a permis de :

- Maîtriser le framework Odoo et son architecture MVC
- Développer des interfaces utilisateur modernes avec CSS3
- Implémenter des fonctionnalités complexes avec Python

— Gérer les migrations de base de données
— Optimiser les performances avec l'indexation
— Améliorer l'expérience utilisateur avec des animations et notifications

9 Annexes

9.1 Structure Complète des Fichiers

```
tp_gestion_notes/
          __init__.py
          __manifest__.py
          models/
                __init__.py
                note_interne.py (188 lignes)
                user_preferences.py
          views/
                note_views.xml (306 lignes)
                date_filter_wizard.xml (49 lignes)
                wizard_view.xml (75 lignes)
                categorie_views.xml
                tag_views.xml
                user_preferences.xml
          wizards/
                __init__.py
                theme_config_wizard.py (51 lignes)
                date_filter_wizard.py (31 lignes)
          security/
                ir.model.access.csv
          static/
                description/
                      icon.png
                src/
                    css/
                          dark_theme.css (835 lignes)
                          components.css (1013 lignes)
                    js/
                        dark_theme.js (113 lignes)
                        theme_config.js
          migrations/
              __init__.py
              3.2.0/
                  __init__.py
                  post-migration.py
```

Listing 3 – Arborescence complète

9.2 Statistiques du Projet

9.3 Champs du Modèle

Champ	Type	Description

titre	Char	Titre de la note (requis, indexé)
contenu	Html	Contenu HTML de la note
description	Text	Description courte
auteur_id	Many2one	Auteur (res.users, indexé)
date_note	Date	Date de création (indexé)
date_echeance	Date	Date d'échéance
statut	Selection	Brouillon/Publié/Archivé (indexé)
priority	Selection	0-3 : Basse/Normale/Haute/Urgente (indexé)
is_favorite	Boolean	Favori (indexé)
is_overdue	Boolean	En retard (calculé)
theme_couleur	Selection	Couleur du thème
is_dark_mode	Boolean	Mode sombre activé
display_name	Char	Nom d'affichage (calculé)

9.4 Méthodes du Modèle

Méthode	**Description**
_compute_display_name	Calcule le nom d'affichage avec icônes de priorité
_compute_is_overdue	Calcule si la note est en retard
_check_dates	Valide les dates (note et échéance)
action_toggle_favorite	Bascule le statut favori avec notification
action_publier	Publie la note avec notification
action_archiver	Archive la note avec notification
action_restaurer	Restaure une note archivée
action_supprimer	Supprime la note avec confirmation
get_statistics	Retourne les statistiques pour le dashboard

Fin du rapport

Élément	**Nombre**
Fichiers Python	5
Fichiers XML	6
Fichiers CSS	2
Fichiers JavaScript	2
Lignes de code Python	300
Lignes de code XML	500
Lignes de code CSS	1800
Lignes de code JavaScript	150
Total	**2750 lignes**

TABLE 2 – Statistiques du code source