📓 Documentation Technique - Système de Gestion des Stages TechPal

Version : 1.0.0
Framework : Odoo 17.0
Langage : Python 3.12
Date : Janvier 2025
Auteur : SILUE - TechPal Casablanca

📋 Table des Matières

1. Architecture Générale
2. Structure du Projet
3. Modèles de Données
4. Diagramme ERD
5. Sécurité
6. Vues et Interface
7. Workflows et Automatisations
8. Rapports PDF
9. Dashboard OWL
10. API et Méthodes
11. Best Practices
12. Tests et Débogage

---

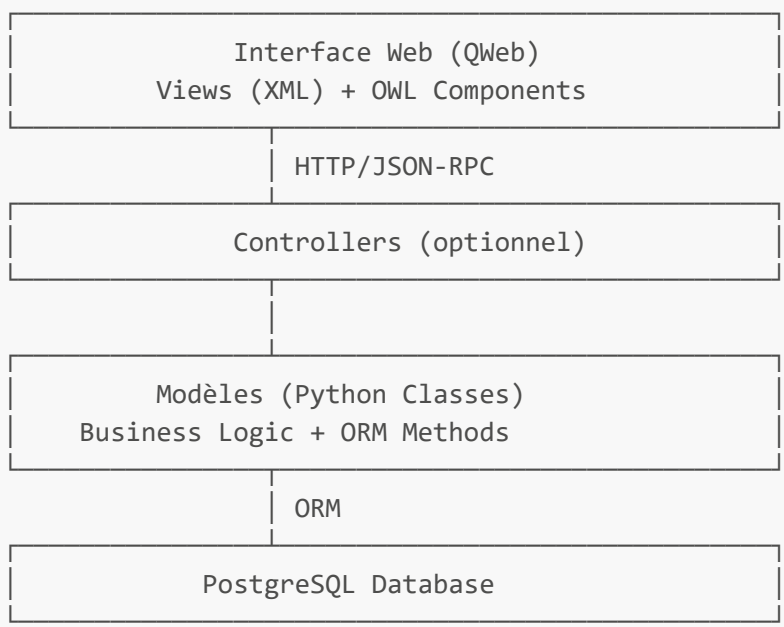
1. Architecture Générale

1.1 Stack Technique

```
Backend:
  Framework: Odoo 17.0
  Langage: Python 3.11
  ORM: Odoo ORM (PostgreSQL)

Frontend:
  Framework: OWL (Odoo Web Library) 17.0
  Templating: QWeb
  Styles: SCSS/CSS
  JS: ES6+

Base de données:
  SGBD: PostgreSQL 16.x
  Encoding: UTF-8

Serveur Web:
```

```
Development: Werkzeug (intégré Odoo)
Production: Nginx + Gunicorn (recommandé)
```

1.2 Architecture MVC Odoo



1.3 Modules du Projet

```
internship_management/      # Module principal
├── models/                 # Logique métier
├── views/                  # Interface utilisateur
├── reports/                # Rapports PDF
├── security/               # Droits d'accès
├── data/                   # Données initiales
└── static/src/            # Assets frontend
    ├── dashboard/         # Dashboard OWL
    └── scss/              # Styles

internship_theme/          # Module thème (login)
├── views/                 # Templates login
└── static/src/scss/       # Styles login
```

2. Structure du Projet

2.1 Arborescence Complète

```
internship_management/
├── __init__.py                          # Init principal
├── __manifest__.py                      # Déclaration module
```

```
│
├── models/                                  # Modèles métier
│   ├── __init__.py
│   ├── internship_stage.py                 # Gestion stages (CORE)
│   ├── internship_student.py               # Profils étudiants
│   ├── internship_supervisor.py            # Encadrants
│   ├── internship_document.py              # Documents
│   ├── internship_document_feedback.py     # Feedback docs
│   ├── internship_presentation.py          # Présentations soutenance
│   ├── internship_meeting.py               # Réunions
│   ├── internship_task.py                  # Tâches/To-Do
│   ├── internship_area.py                  # Domaines d'expertise
│   └── internship_skill.py                 # Compétences
│
├── views/                                   # Vues XML
│   ├── internship_stage_views.xml          # Vues stages
│   ├── internship_student_views.xml
│   ├── internship_supervisor_views.xml
│   ├── internship_document_views.xml
│   ├── internship_document_feedback_views.xml
│   ├── internship_presentation_views.xml
│   ├── internship_meeting_views.xml
│   ├── internship_todo_views.xml
│   ├── internship_area_views.xml
│   ├── internship_skill_views.xml
│   ├── internship_security_views.xml       # UI gestion sécurité
│   ├── internship_dashboard_action.xml     # Action dashboard OWL
│   └── internship_menus.xml                # Menus navigation
│
├── reports/                                 # Rapports PDF
│   ├── __init__.py
│   ├── internship_reports.py               # Logique génération PDF
│   ├── internship_reports.xml              # Déclaration rapports
│   └── internship_report_templates.xml     # Templates QWeb PDF
│
├── security/                                # Sécurité
│   ├── internship_security.xml             # Groupes utilisateurs
│   └── ir.model.access.csv                 # Droits CRUD
│
├── data/                                    # Données & automatisations
│   ├── sequences.xml                       # Séquences (ref stages)
│   ├── internship_cron.xml                 # Tâches automatisées
│   ├── mail_activity_type_data.xml         # Types d'activités
│   ├── internship_meeting_mail_templates.xml # Templates emails
│   └── internship_demo_data.xml            # Données démo
│
├── static/                                  # Assets frontend
│   ├── description/
│   │   └── icon.png                        # Icône module
│   └── src/
│       ├── dashboard/                      # Dashboard OWL
│       │   ├── dashboard.js                # Composant JS
│       │   └── dashboard.xml               # Template OWL
│       └── scss/
```

```
│           └── dashboard.scss              # Styles dashboard
│
└── controllers/                             # Controllers web (optionnel)
    └── __init__.py

internship_theme/                            # Module séparé - Login
├── __init__.py
├── __manifest__.py
├── views/
│   └── login_template.xml                   # Login customisé
└── static/src/scss/
    └── login.scss                           # Styles login
```

2.2 Fichier manifest.py

```python
# -*- coding: utf-8 -*-
{
    'name': 'Gestion des Stages TechPal',
    'version': '17.0.1.0.0',
    'category': 'Ressources Humaines',
    'summary': 'Système de gestion des stages pour TechPal Casablanca',

    'author': 'SILUE Stagiaire - TechPal Casablanca',
    'website': 'https://www.techpalservices.com/',
    'license': 'LGPL-3',

    'depends': [
        'base',
        'mail',
        'contacts',
        'portal',
        'hr',
        'calendar',
    ],

    'data': [
        # Security (ORDRE CRITIQUE)
        'security/internship_security.xml',
        'security/ir.model.access.csv',

        # Data
        'data/sequences.xml',
        'data/internship_cron.xml',
        'data/internship_meeting_mail_templates.xml',
        'data/mail_activity_type_data.xml',

        # Reports
        'reports/internship_report_templates.xml',
        'reports/internship_reports.xml',

        # Views (ORDRE IMPORTANT)
```

```
            'views/internship_presentation_views.xml',
            'views/internship_meeting_views.xml',
            'views/internship_stage_views.xml',
            'views/internship_student_views.xml',
            'views/internship_supervisor_views.xml',
            'views/internship_area_views.xml',
            'views/internship_skill_views.xml',
            'views/internship_security_views.xml',
            'views/internship_document_views.xml',
            'views/internship_document_feedback_views.xml',
            'views/internship_todo_views.xml',
            'views/internship_dashboard_action.xml',

            # Menus (TOUJOURS EN DERNIER)
            'views/internship_menus.xml',
        ],

        'assets': {
            'web.assets_backend': [
                'internship_management/static/src/dashboard/dashboard.js',
                'internship_management/static/src/dashboard/dashboard.xml',
                'internship_management/static/src/scss/dashboard.scss',
            ],
        },

        'demo': [
            'data/internship_demo_data.xml',
        ],

        'installable': True,
        'application': True,
        'auto_install': False,
        'sequence': 15,
    }
```

3. Modèles de Données

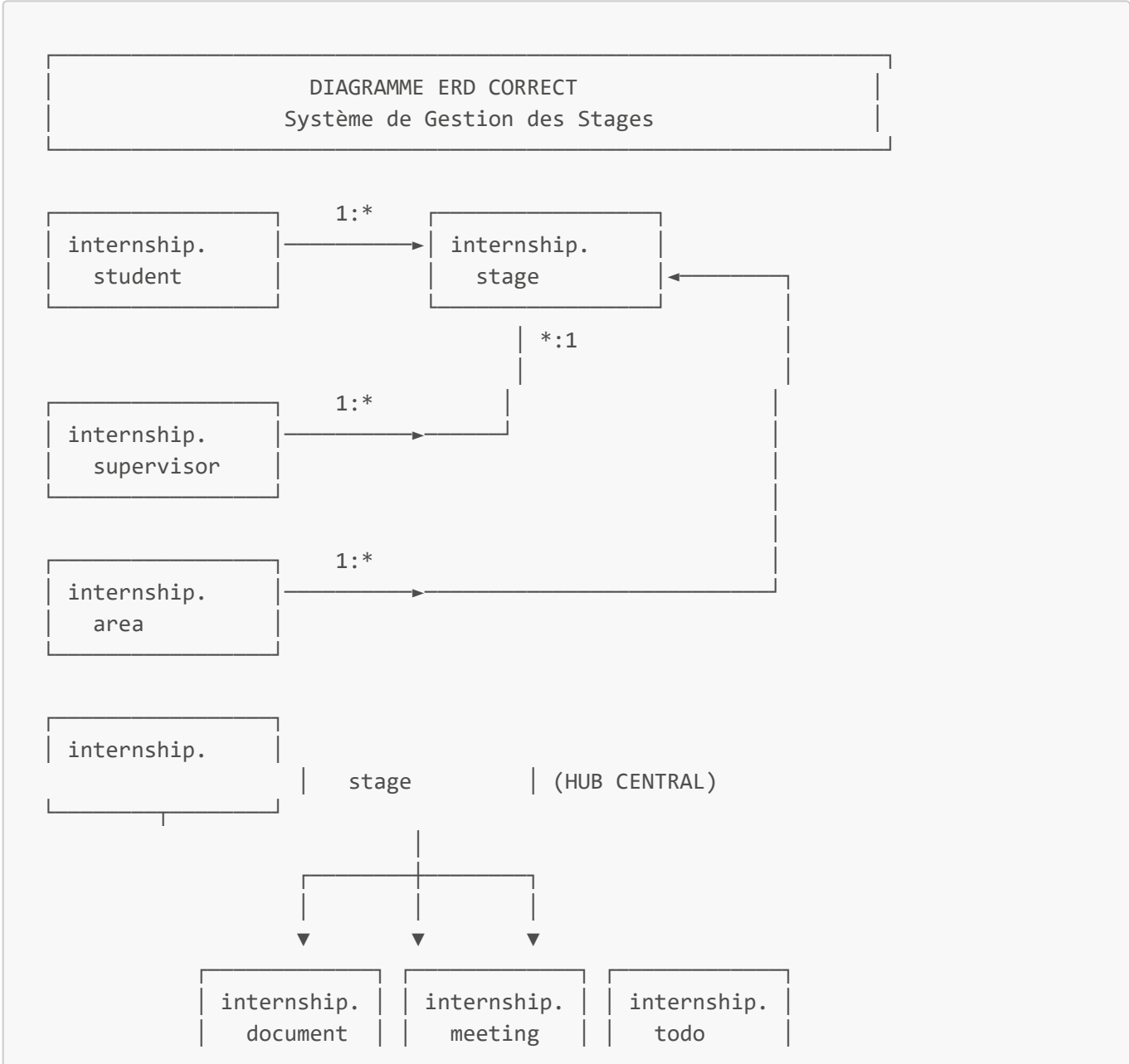
3.1 Modèles Principaux

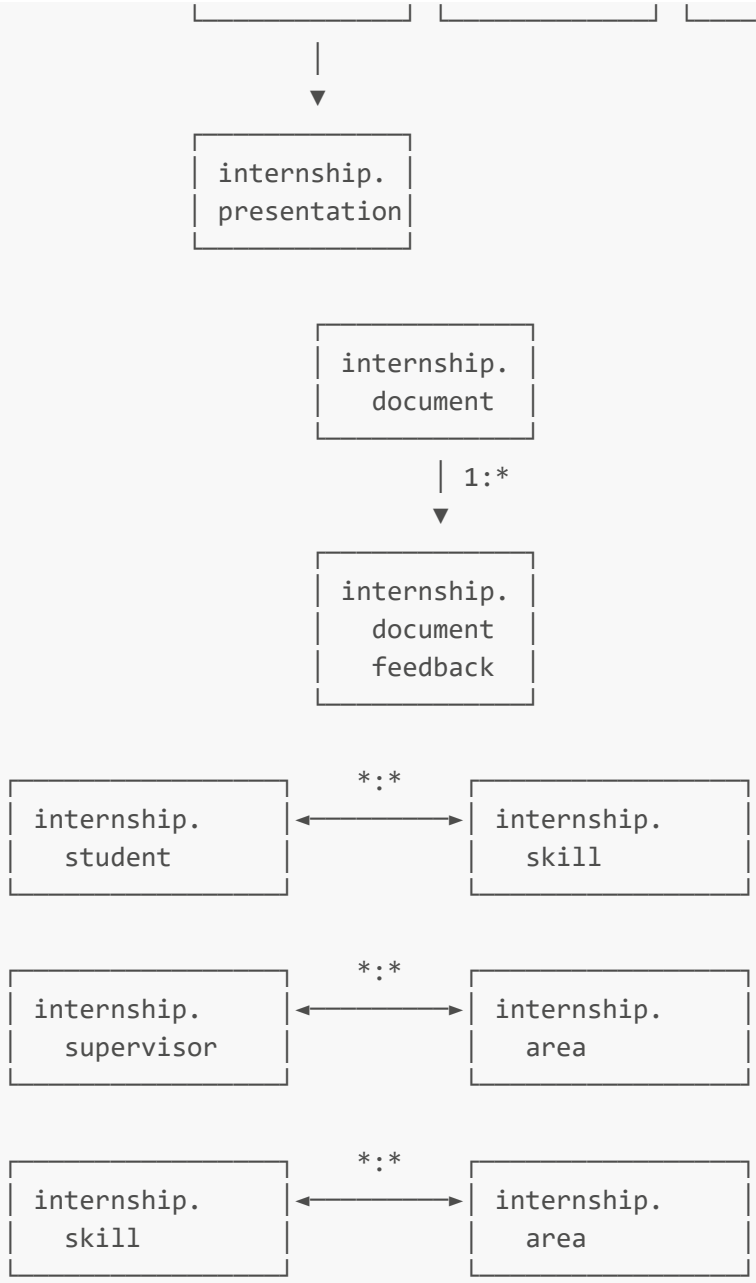
Le système comprend **10 modèles principaux** organisés autour d'un hub central `internship.stage` :

Modèle	Description	Relations Principales
`internship.stage`	**Hub central** - Gestion complète du stage	Relations 1:* vers tous les autres
`internship.student`	Profils étudiants avec compétences	Many2many avec skills, One2many vers stages
`internship.supervisor`	Encadrants avec domaines d'expertise	Many2many avec areas, One2many vers stages

Modèle	Description	Relations Principales
`internship.document`	Documents de stage avec workflow	Many2one vers stage, One2many vers feedback
`internship.document.feedback`	Commentaires sur documents	Many2one vers document
`internship.meeting`	Réunions planifiées	Many2one vers stage
`internship.presentation`	Présentations de soutenance	Many2one vers stage
`internship.todo`	Tâches et livrables	Many2one vers stage
`internship.area`	Domaines d'expertise hiérarchiques	Many2many avec supervisors/skills
`internship.skill`	Compétences techniques et générales	Many2many avec students/areas

4.1 Diagramme ERD





4.2 Modèle Principal : internship.stage

Fichier : `models/internship_stage.py`

```
class InternshipStage(models.Model):
    """
    Modèle central du système.
    Gère le cycle de vie complet d'un stage.
    """
    _name = 'internship.stage'
    _description = 'Gestion de Stage'
    _inherit = ['mail.thread', 'mail.activity.mixin']
    _order = 'start_date desc, title'
    _rec_name = 'title'

    # ==============================
```

```
    # CHAMPS D'IDENTIFICATION
    # ==============================

    title = fields.Char(
        string='Titre du Stage',
        required=True,
        tracking=True,
        help="Sujet principal ou titre du projet de stage."
    )

    reference_number = fields.Char(
        string='Numéro de Référence',
        readonly=True,
        copy=False,
        default='Nouveau',
        help="Numéro de référence unique pour ce stage."
    )

    internship_type = fields.Selection([
        ('final_project', 'Projet de Fin d\'Études (PFE)'),
        ('summer_internship', 'Stage d\'été'),
        ('observation_internship', 'Stage d\'observation'),
        ('professional_internship', 'Stage professionnel')
    ], string='Type de Stage', tracking=True)

    # ==============================
    # CHAMPS RELATIONNELS
    # ==============================

    student_id = fields.Many2one(
        'internship.student',
        string='Étudiant(e)',
        tracking=True,
        ondelete='restrict',
        help="Étudiant(e) assigné(e) à ce stage."
    )

    supervisor_id = fields.Many2one(
        'internship.supervisor',
        string='Encadrant(e)',
        tracking=True,
        ondelete='restrict',
        help="Encadrant(e) académique ou professionnel."
    )

    area_id = fields.Many2one(
        'internship.area',
        string='Domaine d\'Expertise',
        help="Domaine d'expertise de ce stage."
    )

    company_id = fields.Many2one(
        'res.company',
        string='Entreprise',
```

```
        default=lambda self: self.env.company,
        readonly=True,
        help="Entreprise où se déroule le stage."
    )

    # ================================
    # GESTION DU TEMPS
    # ================================

    start_date = fields.Date(
        string='Date de Début',
        tracking=True,
        help="Date de début officielle du stage."
    )

    end_date = fields.Date(
        string='Date de Fin',
        tracking=True,
        help="Date de fin officielle du stage."
    )

    @api.depends('start_date', 'end_date')
    def _compute_duration_days(self):
        """Calcule la durée du stage en jours."""
        for stage in self:
            if stage.start_date and stage.end_date:
                if stage.end_date >= stage.start_date:
                delta = stage.end_date - stage.start_date
                stage.duration_days = delta.days + 1
                else:
                    stage.duration_days = 0
            else:
                stage.duration_days = 0

    duration_days = fields.Integer(
        string='Durée (Jours)',
        compute='_compute_duration_days',
        store=True,
        help="Durée totale du stage en jours."
    )

    # ================================
    # GESTION D'ÉTAT (WORKFLOW)
    # ================================

    state = fields.Selection([
        ('draft', 'Brouillon'),
        ('submitted', 'Soumis'),
        ('approved', 'Approuvé'),
        ('in_progress', 'En Cours'),
        ('completed', 'Terminé'),
        ('evaluated', 'Évalué'),
        ('cancelled', 'Annulé')
    ], string='Statut', default='draft', tracking=True)
```

```
    # ==============================
    # CHAMPS RELATIONNELS (One2many)
    # ==============================

    document_ids = fields.One2many(
        'internship.document', 'stage_id', string='Documents',
        help="Tous les documents liés à ce stage."
    )

    meeting_ids = fields.One2many(
        'internship.meeting', 'stage_id', string='Réunions',
        help="Réunions planifiées pour ce stage."
    )

    task_ids = fields.One2many(
        'internship.todo', 'stage_id', string='Tâches',
        help="Tâches et livrables pour ce stage."
    )

    presentation_ids = fields.One2many(
        'internship.presentation', 'stage_id', string='Présentations',
        help="Présentations de l'étudiant pour la soutenance."
    )

    # ==============================
    # CHAMPS CALCULÉS
    # ==============================

    @api.depends('task_ids.state', 'start_date', 'end_date', 'state')
    def _compute_completion_percentage(self):
        """
        Calcule le pourcentage d'achèvement.
        - Si le stage est terminé ou évalué, le pourcentage est de 100%.
        - Si des tâches existent, le calcul se base sur le ratio de tâches
terminées.
        - Sinon, il se base sur le temps écoulé par rapport à la durée totale.
        """
        for stage in self:
            if stage.state in ('completed', 'evaluated'):
                stage.completion_percentage = 100.0
                continue
            elif stage.state == 'cancelled':
                stage.completion_percentage = 0.0
                continue

            progress_value = 0.0
            total_tasks = len(stage.task_ids)
            if total_tasks > 0:
                completed_tasks = len(stage.task_ids.filtered(lambda t: t.state ==
'done'))
                progress_value = (completed_tasks / total_tasks) * 100.0
            else:
                if stage.start_date and stage.end_date and stage.end_date >=
```

```
stage.start_date:
                    total_duration = (stage.end_date - stage.start_date).days + 1
                    if total_duration > 0:
                        today = fields.Date.context_today(stage)
                        if today <= stage.start_date:
                            elapsed_days = 0
                        elif today >= stage.end_date:
                            elapsed_days = total_duration
                else:
                            elapsed_days = (today - stage.start_date).days
                        progress_value = (elapsed_days / total_duration) * 100.0

            stage.completion_percentage = max(0.0, min(100.0,
round(progress_value, 2)))

    completion_percentage = fields.Float(
        string='Achèvement %',
        compute='_compute_completion_percentage',
        store=True,
        tracking=True,
        help="Pourcentage global d'achèvement du stage."
    )

    # ================================
    # CONTRAINTES DE VALIDATION
    # ================================

    @api.constrains('start_date', 'end_date')
    def _check_date_consistency(self):
        """Vérifie que la date de fin est postérieure à la date de début."""
        for stage in self:
            if stage.start_date and stage.end_date and stage.start_date >
stage.end_date:
                raise ValidationError(_("La date de fin doit être après la date de
début."))

    @api.constrains('final_grade', 'defense_grade')
    def _check_grade_range(self):
        """Vérifie que les notes sont dans une plage valide (0-20)."""
        for stage in self:
            if stage.final_grade and not (0 <= stage.final_grade <= 20):
                raise ValidationError(_("La note finale doit être entre 0 et
20."))
            if stage.defense_grade and not (0 <= stage.defense_grade <= 20):
                raise ValidationError(_("La note de soutenance doit être entre 0
et 20."))

    # ================================
    # MÉTHODES CRUD
    # ================================

    @api.model_create_multi
    def create(self, vals_list):
        """Surcharge de la méthode create pour générer les numéros de
```

```
référence."""
        for vals in vals_list:
            if vals.get('reference_number', 'Nouveau') == 'Nouveau':
                vals['reference_number'] =
self.env['ir.sequence'].next_by_code('internship.stage') or 'STG-N/A'

        stages = super().create(vals_list)

        for stage in stages:
            _logger.info(f"Stage créé : {stage.reference_number} - {stage.title}")
            # Notifier l'étudiant via le Chatter
            if stage.student_id.user_id:
                stage.message_post(
                    body=_(
                        "Bienvenue ! Vous avez été assigné(e) au stage "
                        "\"<strong>%s</strong>\". Veuillez consulter les
détails.",
                        stage.title
                    ),
                    partner_ids=[stage.student_id.user_id.partner_id.id],
                    message_type='comment',
                    subtype_xmlid='mail.mt_comment',
                )
        return stages

    # ================================
    # MÉTHODES MÉTIER (ACTIONS DES BOUTONS)
    # ================================

    def action_submit(self):
        """Soumet le stage pour approbation."""
        self.write({'state': 'submitted'})

    def action_approve(self):
        """Approuve le stage."""
        self.write({'state': 'approved'})

    def action_start(self):
        """Démarre le stage."""
        self.write({'state': 'in_progress'})

    def action_complete(self):
        """Marque le stage comme terminé."""
        self.write({'state': 'completed'})

    def action_schedule_defense(self):
        """
        Crée une activité pour l'encadrant afin qu'il planifie la soutenance.
        """
        self.ensure_one()
        if self.state != 'completed':
            raise ValidationError(_("Seuls les stages terminés peuvent avoir une
soutenance planifiée."))
```

```
        if self.supervisor_id.user_id:
            self.activity_schedule(
                'mail.mail_activity_data_todo',
                summary=_("Planifier la soutenance pour %s", self.title),
                note=_(
                    "Le stage est terminé. Veuillez configurer la date, "
                    "le lieu et les membres du jury dans l'onglet 'Soutenance &
Évaluation'."
                ),
                user_id=self.supervisor_id.user_id.id,
            )

        return {
            'type': 'ir.actions.client',
            'tag': 'display_notification',
            'params': {
                'title': _('Planification de la Soutenance'),
                'message': _('Une activité a été créée pour que l\'encadrant(e)
configure la soutenance.'),
                'type': 'info',
            }
        }

    def action_evaluate(self):
        """Marque le stage comme évalué après vérifications."""
        self.ensure_one()
        if self.state != 'completed':
            raise ValidationError(_("Seuls les stages terminés peuvent être
évalués."))

        if not all([self.defense_date, self.jury_member_ids, self.defense_grade,
self.final_grade]):
            raise ValidationError(
                _("Date de soutenance, membres du jury et notes doivent être
renseignés avant d'évaluer."))

        self.write({'state': 'evaluated', 'defense_status': 'completed'})

        # Notifier les parties prenantes via le Chatter
        partner_ids = []
        if self.student_id.user_id:
            partner_ids.append(self.student_id.user_id.partner_id.id)
        if self.supervisor_id.user_id:
            partner_ids.append(self.supervisor_id.user_id.partner_id.id)

        self.message_post(
            body=_(
                "<strong>Évaluation du Stage Terminée</strong><br/>"
                "Note de Soutenance: <strong>%s/20</strong><br/>"
                "Note Finale: <strong>%s/20</strong>",
                self.defense_grade, self.final_grade
            ),
            partner_ids=partner_ids,
            message_type='comment',
```

```
                subtype_xmlid='mail.mt_comment',
            )

    # ==============================
    # TÂCHE AUTOMATISÉE (CRON)
    # ==============================

    @api.model
    def _cron_internship_monitoring(self):
        """
        Méthode principale appelée par le Cron pour lancer toutes les détections
        et transformer les problèmes trouvés en Activités.
        """
        _logger.info("CRON: Démarrage du suivi des stages...")

        # 1. Détection des tâches en retard (appel de la méthode dédiée)
        self.env['internship.todo']._cron_detect_overdue_tasks()

        # 2. Détection des stages inactifs
        fourteen_days_ago = fields.Datetime.now() - timedelta(days=14)
        inactive_stages = self.search([
            ('state', '=', 'in_progress'),
            ('write_date', '<', fourteen_days_ago),
            ('activity_ids', '=', False)
        ])

        for stage in inactive_stages:
            if stage.supervisor_id.user_id:
                stage.activity_schedule(
                    'internship_management.activity_type_internship_alert',
                    summary=_("Stage potentiellement inactif : %s", stage.title),
                    note=_(
                        "Aucune modification sur ce stage depuis plus de 14 jours.
"
                        "Un suivi est peut-être nécessaire."
                    ),
                    user_id=stage.supervisor_id.user_id.id
                )

        _logger.info("CRON: Suivi des stages terminé.")
```

4.3 Autres Modèles Clés

internship.student

```
class InternshipStudent(models.Model):
    _name = 'internship.student'
    _inherit = ['mail.thread', 'mail.activity.mixin']

    full_name = fields.Char(required=True, tracking=True)
    user_id = fields.Many2one('res.users', ondelete='restrict')
```

```
    email_from_user = fields.Char(related='user_id.login', store=True)
    phone = fields.Char(related='user_id.phone', readonly=False)
    institution = fields.Char()
    education_level = fields.Selection([...])

    internship_ids = fields.One2many('internship.stage', 'student_id')
    skill_ids = fields.Many2many('internship.skill')
```

internship.document

```
class InternshipDocument(models.Model):
    _name = 'internship.document'
    _inherit = ['mail.thread', 'mail.activity.mixin']

    name = fields.Char(required=True)
    document_type = fields.Selection([...])
    stage_id = fields.Many2one('internship.stage', required=True)
    file = fields.Binary(required=True, attachment=True)

    state = fields.Selection([
        ('draft', 'Brouillon'),
        ('submitted', 'Soumis'),
        ('under_review', 'En Révision'),
        ('approved', 'Approuvé'),
        ('rejected', 'Rejeté')
    ])

    def action_submit_for_review(self):
        """Soumet pour révision + notification encadrant."""
        self.state = 'submitted'
        if self.supervisor_id.user_id:
            self.activity_schedule(
                'mail.activity_data_todo',
                user_id=self.supervisor_id.user_id.id,
                summary=_("Réviser : %s", self.name)
            )
```

5. Sécurité

5.1 Groupes Utilisateurs

Fichier : `security/internship_security.xml`

```
<odoo>
    <data noupdate="1">

        <!-- Catégorie -->
        <record id="module_category_internship" model="ir.module.category">
            <field name="name">Gestion des Stages</field>
```

```
            <field name="sequence">20</field>
        </record>

        <!-- GROUPE 1 : Stagiaire -->
        <record id="group_internship_student" model="res.groups">
            <field name="name">Stagiaire</field>
            <field name="category_id" ref="module_category_internship"/>
            <field name="implied_ids" eval="[(4, ref('base.group_user'))]"/>
        </record>

        <!-- GROUPE 2 : Encadrant -->
        <record id="group_internship_supervisor" model="res.groups">
            <field name="name">Encadrant</field>
            <field name="category_id" ref="module_category_internship"/>
            <field name="implied_ids" eval="[(4,
ref('group_internship_student'))]"/>
        </record>

        <!-- GROUPE 3 : Coordinateur -->
        <record id="group_internship_coordinator" model="res.groups">
            <field name="name">Coordinateur</field>
            <field name="category_id" ref="module_category_internship"/>
            <field name="implied_ids" eval="[(4,
ref('group_internship_supervisor'))]"/>
        </record>

        <!-- GROUPE 4 : Administrateur -->
        <record id="group_internship_admin" model="res.groups">
            <field name="name">Administrateur</field>
            <field name="category_id" ref="module_category_internship"/>
            <field name="implied_ids" eval="[(4,
ref('group_internship_coordinator'))]"/>
        </record>

    </data>
</odoo>
```

5.2 Droits d'Accès (CRUD)

Fichier : `security/ir.model.access.csv`

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_stage_admin,stage.admin,model_internship_stage,group_internship_admin,1,1,1
,1
access_stage_coordinator,stage.coordinator,model_internship_stage,group_internship
_coordinator,1,1,1,0
access_stage_supervisor,stage.supervisor,model_internship_stage,group_internship_s
upervisor,1,1,1,0
access_stage_student,stage.student,model_internship_stage,group_internship_student
,1,0,0,0
access_document_student,document.student,model_internship_document,group_internshi
```

```
p_student,1,1,1,0
access_document_supervisor,document.supervisor,model_internship_document,group_int
ernship_supervisor,1,1,1,0
```

Légende :

- `perm_read` : Lecture
- `perm_write` : Modification
- `perm_create` : Création
- `perm_unlink` : Suppression

5.3 Règles d'Enregistrement

Les stagiaires ne voient QUE leurs propres stages :

```
<record id="internship_stage_student_rule" model="ir.rule">
    <field name="name">Stagiaire : Voir uniquement ses stages</field>
    <field name="model_id" ref="model_internship_stage"/>
    <field name="groups" eval="[(4, ref('group_internship_student'))]"/>
    <field name="domain_force">
        [('student_id.user_id', '=', user.id)]
    </field>
</record>
```

Les encadrants voient leurs stages supervisés :

```
<record id="internship_stage_supervisor_rule" model="ir.rule">
    <field name="name">Encadrant : Voir ses stages encadrés</field>
    <field name="model_id" ref="model_internship_stage"/>
    <field name="groups" eval="[(4, ref('group_internship_supervisor'))]"/>
    <field name="domain_force">
        [('supervisor_id.user_id', '=', user.id)]
    </field>
</record>
```

6. Vues et Interface

6.1 Types de Vues

Vue	Usage	Fichier exemple
`tree`	Liste	`view_internship_stage_tree`
`form`	Formulaire	`view_internship_stage_form`
`kanban`	Kanban	`view_internship_stage_kanban`
`calendar`	Calendrier	`view_internship_meeting_calendar`

Vue	Usage	Fichier exemple
search	Filtres/recherche	view_internship_stage_search

6.2 Structure d'une Vue Form

Fichier : views/internship_stage_views.xml

```
<record id="view_internship_stage_form" model="ir.ui.view">
    <field name="name">internship.stage.form</field>
    <field name="model">internship.stage</field>
    <field name="arch" type="xml">
        <form string="Stage">

            <!-- HEADER : Boutons d'action + statusbar -->
            <header>
                <button name="action_complete"
                        string="Terminer"
                        type="object"
                        class="oe_highlight"
                        invisible="state != 'in_progress'"/>

                <field name="state" widget="statusbar"

statusbar_visible="draft,submitted,approved,in_progress,completed"/>
            </header>

            <!-- SHEET : Corps principal -->
            <sheet>
                <div class="oe_title">
                    <h1>
                        <field name="title" placeholder="Titre du stage..."/>
                    </h1>
                </div>

                <group>
                    <group name="basic" string="Informations">
                        <field name="internship_type"/>
                        <field name="student_id"/>
                        <field name="supervisor_id"/>
                    </group>
                    <group name="dates" string="Dates">
                        <field name="start_date"/>
                        <field name="end_date"/>
                        <field name="duration_days"/>
                    </group>
                </group>

                <!-- NOTEBOOK : Onglets -->
                <notebook>
                    <page name="tasks" string="Tâches">
                        <field name="task_ids">
```

```
                            <tree>
                                <field name="name"/>
                                <field name="deadline"/>
                                <field name="state"/>
                            </tree>
                        </field>
                    </page>

                    <page name="documents" string="Documents">
                        <field name="document_ids"/>
                    </page>
                </notebook>
            </sheet>

            <!-- CHATTER : Messagerie + Activités -->
            <div class="oe_chatter">
                <field name="message_follower_ids"/>
                <field name="activity_ids"/>
                <field name="message_ids"/>
            </div>
        </form>
    </field>
</record>
```

6.3 Attributs invisible vs readonly

```
<!-- ❌ AVANT (Odoo < 17) : attrs déprécié -->
<field name="email"
       attrs="{'invisible': [('user_id', '!=', False)]}"/>

<!-- ☑ APRÈS (Odoo 17) : invisible direct -->
<field name="email"
       invisible="user_id != False"/>
```

6. Workflows et Automatisations

6.1 Workflow des Stages

```
[Brouillon]
    | action_submit()
    ▼
[Soumis]
    | action_approve() (coordinateur)
    ▼
[Approuvé]
    | action_start() (encadrant)
    ▼
[En Cours]
    | action_complete() (encadrant)
```

```
    ▼
[Terminé]
    │ action_evaluate() (encadrant + jury)
    ▼
[Évalué] ☑
```

6.2 Tâche CRON

Fichier : `data/internship_cron.xml`

```xml
<record id="ir_cron_internship_monitoring" model="ir.cron">
    <field name="name">Suivi et Détection d'Anomalies</field>
    <field name="model_id" ref="model_internship_stage"/>
    <field name="state">code</field>
    <field name="code">model._cron_internship_monitoring()</field>

    <!-- S'exécute tous les jours à 2h du matin -->
    <field name="interval_number">1</field>
    <field name="interval_type">days</field>
    <field name="numbercall">-1</field>
    <field name="doall" eval="False"/>
    <field name="active" eval="True"/>
</record>
```

Logique (dans internship_stage.py) :

```python
@api.model
def _cron_internship_monitoring(self):
    """Détecte stages inactifs, tâches en retard, etc."""

    # 1. Détecter stages inactifs (14j sans modification)
    fourteen_days_ago = fields.Datetime.now() - timedelta(days=14)
    inactive = self.search([
        ('state', '=', 'in_progress'),
        ('write_date', '<', fourteen_days_ago)
    ])

    for stage in inactive:
        # Créer activité pour encadrant
        stage.activity_schedule(...)

    # 2. Détecter tâches en retard
    self.env['internship.todo']._cron_detect_overdue_tasks()
```

6.3 Notifications Email

Fichier : `data/internship_meeting_mail_templates.xml`

```
<record id="mail_template_meeting_invitation" model="mail.template">
    <field name="name">Réunion : Invitation</field>
    <field name="model_id" ref="model_internship_meeting"/>
    <field name="subject">
        Invitation : {{ object.name }} le {{ object.date.strftime('%d/%m/%Y') }}
    </field>
    <field name="partner_to">{{ ','.join(map(str, object.partner_ids.ids)) }}
</field>
    <field name="body_html" type="html">
        <div>
            <p>Bonjour,</p>
            <p>Vous êtes invité(e) à la réunion suivante :</p>
            <h2>{{ object.name }}</h2>
            <ul>
                <li><strong>Date :</strong> {{ object.date.strftime('%d/%m/%Y à
%H:%M') }}</li>
                <li><strong>Durée :</strong> {{ object.duration }} heures</li>
                <li t-if="object.location"><strong>Lieu :</strong> {{
object.location }}</li>
            </ul>
        </div>
    </field>
</record>
```

Envoi programmatique :

```
def action_schedule(self):
    """Planifie la réunion et envoie les invitations."""
    self.write({'state': 'scheduled'})

    template =
self.env.ref('internship_management.mail_template_meeting_invitation')
    if template:
        template.send_mail(self.id, force_send=True)
```

7. Rapports PDF

7.1 Architecture des Rapports

```
reports/
├── internship_reports.py              # Logique Python
├── internship_reports.xml             # Déclaration rapports
└── internship_report_templates.xml    # Templates QWeb
```

7.2 Déclaration d'un Rapport

Fichier : `reports/internship_reports.xml`

```
<record id="action_convention_report" model="ir.actions.report">
    <field name="name">Convention de Stage</field>
    <field name="model">internship.stage</field>
    <field name="report_type">qweb-pdf</field>
    <field name="report_name">
        internship_management.convention_report_document
    </field>
    <field name="print_report_name">
        'Convention - %s' % (object.title)
    </field>
    <field name="binding_model_id" ref="model_internship_stage"/>
    <field name="binding_type">report</field>
</record>
```

7.3 Template QWeb

Fichier : `reports/internship_report_templates.xml`

```
<template id="convention_report_document">
    <t t-call="web.html_container">
        <t t-foreach="docs" t-as="stage">
            <t t-call="web.external_layout">
                <div class="page">
                    <h2 class="text-center">CONVENTION DE STAGE</h2>
                    <p>Référence : <span t-field="stage.reference_number"/></p>

                    <h4>Article 1 : Les Parties</h4>
                    <ul>
                        <li>
                            <strong>Entreprise :</strong>
                            <span t-field="stage.company_id.name"/>
                        </li>
                        <li>
                            <strong>Stagiaire :</strong>
                            <span t-field="stage.student_id.full_name"/>
                        </li>
                        <li>
                            <strong>Encadrant :</strong>
                            <span t-field="stage.supervisor_id.name"/>
                        </li>
                    </ul>

                    <h4>Article 2 : Durée</h4>
                    <p>
                        Du <span t-field="stage.start_date" t-options="{'widget':
'date'}"/>
                        au <span t-field="stage.end_date" t-options="{'widget':
'date'}"/>
                    </p>
                </div>
```

```
            </t>
        </t>
    </t>
</template>
```

7.4 Logique Python (optionnel)

Fichier : `reports/internship_reports.py`

```python
class InternshipConventionReport(models.AbstractModel):
    _name = 'report.internship_management.convention_report_document'

    @api.model
    def _get_report_values(self, docids, data=None):
        """Prépare les données pour le PDF."""
        docs = self.env['internship.stage'].browse(docids)
        return {
            'doc_ids': docids,
            'doc_model': 'internship.stage',
            'docs': docs,
            'company': self.env.company,
            'custom_data': self._prepare_custom_data(docs),
        }

    def _prepare_custom_data(self, stages):
        """Logique supplémentaire."""
        data = {}
        for stage in stages:
            data[stage.id] = {
                'performance': self._get_performance(stage.final_grade),
            }
        return data

    def _get_performance(self, grade):
        if grade >= 16:
            return "Excellent"
        elif grade >= 14:
            return "Très Bien"
        # ...
```

8. Dashboard OWL

8.1 Architecture

```
static/src/dashboard/
├── dashboard.js            # Composant OWL
├── dashboard.xml           # Template OWL
└── dashboard.scss          # Styles
```

8.2 Action XML

Fichier : views/internship_dashboard_action.xml

```xml
<record id="action_internship_dashboard_owl" model="ir.actions.client">
    <field name="name">Tableau de Bord</field>
    <field name="tag">internship_dashboard</field>
    <field name="target">current</field>
</record>
```

8.3 Composant JavaScript

Fichier : static/src/dashboard/dashboard.js

```javascript
/** @odoo-module **/

import { Component, useState, onWillStart } from "@odoo/owl";
import { registry } from "@web/core/registry";
import { useService } from "@web/core/utils/hooks";

export class InternshipDashboard extends Component {
    setup() {
        this.orm = useService("orm");
        this.action = useService("action");

        this.state = useState({
            totalInternships: 0,
            activeInternships: 0,
            loading: true,
        });

        onWillStart(async () => {
            await this.loadDashboardData();
        });
    }

    async loadDashboardData() {
        try {
            // Charger stats depuis ORM
            this.state.totalInternships = await this.orm.call(
                "internship.stage",
                "search_count",
                [[]]  // domain vide = tous
            );

            this.state.activeInternships = await this.orm.call(
                "internship.stage",
                "search_count",
                [["state", "=", "in_progress"]]
            );
```

```
            this.state.loading = false;
        } catch (error) {
            console.error("Erreur chargement dashboard:", error);
            this.state.loading = false;
        }
    }

    openInternships() {
        this.action.doAction({
            type: "ir.actions.act_window",
            res_model: "internship.stage",
            views: [[false, "kanban"], [false, "list"]],
            target: "current",
        });
    }
}

InternshipDashboard.template = "internship_management.Dashboard";

registry.category("actions").add("internship_dashboard", InternshipDashboard);
```

8.4 Template OWL

Fichier : `static/src/dashboard/dashboard.xml`

```
<templates xml:space="preserve">
    <t t-name="internship_management.Dashboard" owl="1">
        <div class="o_internship_dashboard">

            <!-- Loader -->
            <div t-if="state.loading" class="loader">
                <i class="fa fa-spin fa-circle-o-notch fa-3x"/>
                <p>Chargement...</p>
            </div>

            <!-- Content -->
            <div t-else="">
                <h2>Tableau de Bord</h2>

                <div class="row">
                    <!-- Card 1 -->
                    <div class="col-lg-3" t-on-click="openInternships">
                        <div class="card bg-primary">
                            <h3 t-esc="state.totalInternships"/>
                            <p>Stages Total</p>
                        </div>
                    </div>

                    <!-- Card 2 -->
                    <div class="col-lg-3">
```

```
                        <div class="card bg-success">
                            <h3 t-esc="state.activeInternships"/>
                            <p>En Cours</p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </t>
</templates>
```

8.5 Déclaration dans manifest.py

```
'assets': {
    'web.assets_backend': [
        'internship_management/static/src/dashboard/dashboard.js',
        'internship_management/static/src/dashboard/dashboard.xml',
        'internship_management/static/src/scss/dashboard.scss',
    ],
},
```

9. API et Méthodes

9.1 Méthodes ORM Courantes

```
# CREATE
stage = self.env['internship.stage'].create({
    'title': 'Mon stage',
    'student_id': 1,
})

# READ
stages = self.env['internship.stage'].search([
    ('state', '=', 'in_progress')
])

# SEARCH + READ
stages = self.env['internship.stage'].search_read(
    [('state', '=', 'in_progress')],
    ['title', 'student_id', 'start_date']
)

# UPDATE
stage.write({'state': 'completed'})

# DELETE (à éviter)
stage.unlink()
```

```
# BROWSE
stage = self.env['internship.stage'].browse(1)
```

9.2 Appels depuis OWL

```
// SEARCH COUNT
const count = await this.orm.call(
    "internship.stage",
    "search_count",
    [[["state", "=", "in_progress"]]]
);

// SEARCH READ
const stages = await this.orm.call(
    "internship.stage",
    "search_read",
    [[["state", "=", "in_progress"]]],
    ["title", "student_id"]
);

// APPELER MÉTHODE CUSTOM
await this.orm.call(
    "internship.stage",
    "action_complete",
    [[stageId]]
);
```

10. Best Practices

10.1 Nommage

```
# ☑ BIEN
class InternshipStage(models.Model):
    _name = 'internship.stage'
    student_id = fields.Many2one('internship.student')

# ❌ MAL
class Stage(models.Model):
    _name = 'stage'
    student = fields.Many2one('student')
```

10.2 Tracking

```
# Activer tracking pour historique
title = fields.Char(tracking=True)
state = fields.Selection([...], tracking=True)
```

10.3 Computed Fields

```
@api.depends('start_date', 'end_date')
def _compute_duration(self):
    for record in self:
        if record.start_date and record.end_date:
            delta = record.end_date - record.start_date
            record.duration = delta.days
        else:
            record.duration = 0

duration = fields.Integer(compute='_compute_duration', store=True)
```

10.4 Constraints

```
@api.constrains('start_date', 'end_date')
def _check_dates(self):
    for record in self:
        if record.start_date > record.end_date:
            raise ValidationError("Date de fin invalide")
```

###10.5 Transactions

```
# Toujours utiliser ensure_one() dans actions
def action_complete(self):
    self.ensure_one()  # ☑ Vérifie qu'il y a 1 seul enregistrement
    self.write({'state': 'completed'})
```

11. Tests et Débogage

11.1 Mode Debug Odoo

URL : http://localhost:8069/web?debug=1
Outils disponibles :
Vue des metadata
Éditer vue
Gérer filtres
Voir champs techniques

11.2 Logs Python

```
import logging
_logger = logging.getLogger(__name__)
```

```
# Dans une méthode
_logger.info("Stage créé : %s", stage.reference_number)
_logger.warning("Attention : stage sans encadrant")
_logger.error("Erreur lors de l'évaluation : %s", error)
```

Fichier log : `C:\Dev\odoo17-internship\logs\odoo.log`

11.3 Tests Unitaires (optionnel)

```
from odoo.tests import common

class TestInternshipStage(common.TransactionCase):

    def setUp(self):
        super().setUp()
        self.Stage = self.env['internship.stage']

    def test_create_stage(self):
        stage = self.Stage.create({
            'title': 'Test Stage',
        })
        self.assertEqual(stage.state, 'draft')

    def test_workflow(self):
        stage = self.Stage.create({'title': 'Test'})
        stage.action_submit()
        self.assertEqual(stage.state, 'submitted'
```

11.4 Outils de débogage

```
# Breakpoint Python
import pdb; pdb.set_trace()

# Print SQL généré
self.env.cr.execute("SELECT * FROM internship_stage WHERE id = %s", (1,))
print(self.env.cr.query)

# Inspecter ORM
print(stage._fields)
print(stage._name)
```

12. Déploiement Production

12.1 Configuration Production

```
[options]
# IMPORTANT : Changer le mot de passe admin
admin_passwd = MOT_DE_PASSE_FORT_ICI

# Workers (processus)
workers = 4  # Nombre de CPU x 2 + 1

# Timeouts
limit_time_cpu = 300
limit_time_real = 600

# Mémoire
limit_memory_hard = 2684354560
limit_memory_soft = 2147483648

# Logs
log_level = warn  # En prod : warn ou error
logfile = /var/log/odoo/odoo.log

# Proxy (si Nginx devant)
proxy_mode = True
```

12.2 Checklist Déploiement

Changer admin_passwd dans odoo.conf Activer workers = 4+ Configurer log_level = warn Désactiver mode debug (?debug=0) Backup automatique de la base de données Certificat SSL (HTTPS) Firewall configuré Surveillance (monitoring)

📞 Support Technique Développeur : SILUE Email : silue@techpal.ma GitHub : [https://github.com/Gninho-silue/internship_management]

Documentation Odoo officielle : https://www.odoo.com/documentation/17.0/
