



UNIVERSITÉ SULTAN MOULAY SLIMANE  
ÉCOLE NATIONALE DES SCIENCES  
APPLIQUÉES DE K HOURIBGA



# Rapport du Projet

Filière : Génie Informatique

## Système de gestion complète pour établissements scolaires

**Réalisé par :**

Sawssan Aatiq  
Assia Amahouch

**Encadré par :**

Nidale Lamghari

Année académique 2025-2026

## Table des matières

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Contexte et Objectifs . . . . .	3
1.2	Portée du Projet . . . . .	3
1.3	Technologies Utilisées . . . . .	3
<b>2</b>	<b>Architecture du Système</b>	<b>4</b>
2.1	Structure du Module . . . . .	4
2.2	Modèle de Données . . . . .	4
2.3	Relations entre Modèles . . . . .	5
<b>3</b>	<b>Fonctionnalités Détaillées</b>	<b>6</b>
3.1	Gestion des Étudiants . . . . .	6
3.1.1	Champs Principaux . . . . .	6
3.1.2	Fonctionnalités Automatiques . . . . .	6
3.1.3	Validations . . . . .	6
3.2	Gestion des Enseignants . . . . .	6
3.2.1	Caractéristiques . . . . .	7
3.3	Gestion des Classes . . . . .	7
3.3.1	Organisation . . . . .	7
3.4	Gestion des Cours . . . . .	7
3.4.1	Informations . . . . .	7
3.5	Système de Notes . . . . .	7
3.5.1	Types d'Évaluations . . . . .	7
3.5.2	Calculs Automatiques . . . . .	8
3.5.3	Système de Lettres . . . . .	8
3.6	Emploi du Temps . . . . .	8
3.6.1	Caractéristiques . . . . .	8
3.6.2	Détection de Conflits . . . . .	9
3.7	Gestion de la Présence . . . . .	9
3.7.1	Statuts . . . . .	9
3.7.2	Fonctionnalités . . . . .	9
3.7.3	Calcul du Taux de Présence . . . . .	9
<b>4</b>	<b>Sécurité et Permissions</b>	<b>10</b>
4.1	Groupes d'Utilisateurs . . . . .	10
4.2	Droits d'Accès par Modèle . . . . .	10
<b>5</b>	<b>Interface Utilisateur</b>	<b>10</b>
5.1	Structure des Menus . . . . .	10
5.2	Vues Disponibles . . . . .	11
<b>6</b>	<b>Captures d'écran de l'Application</b>	<b>12</b>
6.1	Menu Principal . . . . .	12
6.2	Gestion des Étudiants . . . . .	13
6.3	Gestion des Enseignants . . . . .	13

6.4	Gestion des Classes et Cours . . . . .	14
6.5	Système de Notes . . . . .	15
6.6	Emploi du Temps . . . . .	16
6.7	Gestion de la Présence . . . . .	16
6.8	Rapports et Bulletins . . . . .	17
<b>7</b>	<b>Rapports</b>	<b>18</b>
7.1	Types de Rapports . . . . .	18
7.2	Template QWeb . . . . .	18
<b>8</b>	<b>Implémentation Technique</b>	<b>18</b>
8.1	Modèles Python . . . . .	18
8.2	Séquences Automatiques . . . . .	18
8.3	Validations et Contraintes . . . . .	18
8.4	Calculs Automatiques . . . . .	19
<b>9</b>	<b>Tests et Validation</b>	<b>19</b>
9.1	Scénarios de Test . . . . .	19
9.2	Contraintes Validées . . . . .	19
<b>10</b>	<b>Conclusion</b>	<b>20</b>
10.1	Réalisations . . . . .	20
10.2	Perspectives d'Amélioration . . . . .	20
10.3	Apports Pédagogiques . . . . .	20
<b>11</b>	<b>Annexes</b>	<b>21</b>
11.1	Annexe A : Dépendances . . . . .	21
11.2	Annexe B : Installation . . . . .	21
11.3	Annexe C : Licence . . . . .	21

# Introduction

---

## Contexte et Objectifs

Ce projet consiste en le développement d'un module complet de gestion scolaire pour la plateforme Odoo 18. Le système vise à digitaliser et automatiser les processus administratifs et académiques d'un établissement scolaire, offrant une solution intégrée pour la gestion des étudiants, enseignants, cours, notes, emplois du temps et présence.

Les objectifs principaux de ce projet sont :

- Développer un système centralisé de gestion des informations académiques
- Automatiser le calcul des moyennes et statistiques
- Faciliter le suivi de la présence des étudiants
- Générer des rapports et bulletins de notes automatiquement
- Offrir une interface intuitive pour tous les acteurs (administrateurs, enseignants, étudiants)
- Assurer la sécurité et la traçabilité des données

## Portée du Projet

Le module couvre les fonctionnalités essentielles suivantes :

1. Gestion des étudiants avec fiches complètes
2. Gestion des enseignants et de leurs affectations
3. Organisation des classes et niveaux
4. Gestion des cours et matières
5. Système de notation avec calculs automatiques
6. Emploi du temps avec détection de conflits
7. Suivi de la présence quotidienne
8. Génération de rapports PDF

## Technologies Utilisées

- **Plateforme** : Odoo 18 (ERP open-source)
- **Langage de programmation** : Python 3.8+
- **Base de données** : PostgreSQL 12+
- **Framework** : Odoo Framework (ORM, vues XML, sécurité)
- **Interface** : Odoo Web Client (JavaScript, XML)
- **Rapports** : QWeb Templates (PDF)

## Architecture du Système

### Structure du Module

Le module `school_management` suit l'architecture standard d'Odoo avec une organisation modulaire claire :

```

1 school_management/
2     __init__.py           # Point d'entr e du module
3     __manifest__.py       # Manifeste (m tadonn es)
4     models/               # Mod les de donn es
5         __init__.py
6         student.py        # Mod le tudiant
7         teacher.py        # Mod le Enseignant
8         class_level.py    # Mod le Classe
9         course.py         # Mod le Cours
10        grade.py          # Mod le Note
11        schedule.py       # Mod le Emploi du temps
12        attendance.py     # Mod le Pr sence
13    views/                 # Interfaces utilisateur
14        student_views.xml
15        teacher_views.xml
16        class_views.xml
17        course_views.xml
18        grade_views.xml
19        schedule_views.xml
20        attendance_views.xml
21        menu_views.xml
22    security/              # S curit et permissions
23        school_security.xml
24        ir.model.access.csv
25    data/                  # Donn es initiales
26        sequence_data.xml
27    reports/               # Templates de rapports
28        __init__.py
29        report_template.xml
30        student_report.xml
31        bulletin_report.xml
32    static/                # Ressources statiques
33        description/

```

Listing 1 – Structure du module

### Modèle de Données

Le système est basé sur huit modèles principaux interconnectés :

TABLE 1 – Modèles de données principaux

Modèle	Description	Clé primaire
<code>school.student</code>	Fiche étudiant complète	<code>id</code>
<code>school.teacher</code>	Profil enseignant	<code>id</code>
<code>school.class</code>	Classe/Niveau	<code>id</code>
<code>school.course</code>	Cours/Matière	<code>id</code>
<code>school.grade</code>	Note d'évaluation	<code>id</code>
<code>school.schedule</code>	Emploi du temps	<code>id</code>
<code>school.attendance</code>	Enregistrement présence	<code>id</code>
<code>school.attendance.report</code>	Vue rapport présence	<code>id</code>

## Relations entre Modèles

Le diagramme de relations montre les dépendances entre les entités :

- **Étudiant** → Classe (Many2one)
- **Étudiant** → Notes (One2many)
- **Étudiant** → Présences (One2many)
- **Enseignant** → Cours (One2many)
- **Enseignant** → Emploi du temps (One2many)
- **Classe** → Étudiants (One2many)
- **Classe** → Cours (One2many)
- **Cours** → Enseignant (Many2one)
- **Cours** → Classe (Many2one)
- **Cours** → Notes (One2many)
- **Note** → Étudiant (Many2one)
- **Note** → Cours (Many2one)
- **Emploi du temps** → Classe, Cours, Enseignant (Many2one)
- **Présence** → Étudiant, Cours (Many2one)

## Fonctionnalités Détaillées

---

### Gestion des Étudiants

Le modèle `school.student` gère toutes les informations relatives aux étudiants.

#### Champs Principaux

- **Informations personnelles** : Nom, date de naissance, genre, groupe sanguin, nationalité
- **Contact** : Email, téléphone, mobile, adresse complète
- **Informations académiques** : Matricule (auto-généré), classe, date d'admission, statut
- **Tuteur** : Nom, téléphone, email, relation
- **Statistiques calculées** : Moyenne générale, taux de présence

#### Fonctionnalités Automatiques

```
1 @api.depends('date_of_birth')
2 def _compute_age(self):
3     for record in self:
4         if record.date_of_birth:
5             today = date.today()
6             record.age = today.year - record.date_of_birth.year - (
7                 (today.month, today.day) <
8                 (record.date_of_birth.month, record.date_of_birth.day)
9             )
```

Listing 2 – Calcul automatique de l'âge

```
1 @api.depends('grade_ids.grade')
2 def _compute_average_grade(self):
3     for record in self:
4         if record.grade_ids:
5             total = sum(record.grade_ids.mapped('grade'))
6             record.average_grade = total / len(record.grade_ids)
```

Listing 3 – Calcul automatique de la moyenne

#### Validations

Le système inclut des validations pour :

- Format d'email valide
- Date de naissance non future
- Génération automatique de matricule unique via séquence

### Gestion des Enseignants

Le modèle `school.teacher` gère les profils des enseignants avec leurs informations professionnelles.

### Caractéristiques

- Matricule auto-généré
- Informations professionnelles : département, spécialisation, qualification
- Date d'embauche et statut (actif, en congé, terminé)
- Relations avec les cours enseignés
- Calcul automatique du nombre de cours

### Gestion des Classes

Le modèle `school.class` organise les étudiants par niveaux et sections.

### Organisation

- **Niveaux** : 6 niveaux (1 à 6)
- **Section** : Division au sein d'un niveau
- **Capacité** : Nombre maximum d'étudiants
- **Enseignant principal** : Responsable de la classe
- **Statistiques** : Nombre d'étudiants, moyenne de classe

### Gestion des Cours

Le modèle `school.course` définit les matières enseignées.

### Informations

- Code unique (contrainte SQL)
- Nom et description
- Crédits et heures par semaine
- Enseignant et classe associés
- Dates de début et fin
- Calcul automatique de la moyenne du cours

### Système de Notes

Le modèle `school.grade` gère toutes les évaluations.

### Types d'Évaluations

- Devoir (homework)
- Interrogation (quiz)
- Examen partiel (midterm)
- Examen final (final)
- Projet (project)
- Présentation (presentation)
- Participation (participation)



## Calculs Automatiques

```

1 @api.depends('grade', 'max_grade')
2 def _compute_percentage(self):
3     for record in self:
4         if record.max_grade > 0:
5             record.percentage = (record.grade / record.max_grade) * 100
6
7 @api.depends('percentage')
8 def _compute_grade_letter(self):
9     for record in self:
10        percentage = record.percentage
11        if percentage >= 95:
12            record.grade_letter = 'a+'
13        elif percentage >= 90:
14            record.grade_letter = 'a'
15        # ... autres seuils

```

Listing 4 – Calcul du pourcentage et de la lettre

## Système de Lettres

TABLE 2 – Échelle de notation

Pourcentage	Lettre
≥ 95%	A+ (Excellent)
≥ 90%	A (Très bien)
≥ 85%	B+ (Bien)
≥ 80%	B (Assez bien)
≥ 75%	C+ (Passable)
≥ 70%	C (Moyen)
≥ 60%	D (Insuffisant)
< 60%	F (Échec)

## Emploi du Temps

Le modèle `school.schedule` gère les horaires hebdomadaires.

### Caractéristiques

- **Jours** : Lundi à Dimanche
- **Heures** : Format décimal (ex : 9.5 = 09 :30)
- **Durée** : Calcul automatique
- **Type de session** : Cours magistral, TD, TP, Examen
- **Période** : Dates de début et fin
- **Salle** : Numéro de salle

## Détection de Conflits

Le système vérifie automatiquement les conflits d'horaire :

```

1 @api.constrains('class_id', 'teacher_id', 'day_of_week',
2                 'start_time', 'end_time')
3 def _check_conflicts(self):
4     # V r i f i e  l e s  c o n f l i t s  p o u r  l a  c l a s s e
5     conflict = self.search([
6         ('id', '!=', record.id),
7         ('class_id', '=', record.class_id.id),
8         ('day_of_week', '=', record.day_of_week),
9         ('active', '=', True),
10        # V r i f i c a t i o n  d e s  c h e v a u c h e m e n t s  t e m p o r e l s
11    ])
12    if conflict:
13        raise ValidationError("Conflit d'horaire d e t e c t é ")

```

Listing 5 – Vérification des conflits

## Gestion de la Présence

Le modèle `school.attendance` enregistre la présence quotidienne.

### Statuts

- **Présent** : Étudiant présent
- **Absent** : Absence non justifiée
- **En retard** : Arrivée tardive
- **Absent justifié** : Absence avec justification

### Fonctionnalités

- Enregistrement des heures d'arrivée et départ
- Calcul automatique du taux de présence
- Marquage en masse pour une classe
- Prévention des doublons (même étudiant, cours, date)
- Vue analytique pour rapports

### Calcul du Taux de Présence

```

1 @api.depends('attendance_ids')
2 def _compute_attendance_rate(self):
3     for record in self:
4         if record.attendance_ids:
5             present = len(record.attendance_ids.filtered(
6                 lambda a: a.status == 'present'))
7             total = len(record.attendance_ids)
8             record.attendance_rate = (present / total) * 100 if total >
0         else 0.0

```

Listing 6 – Calcul du taux de présence

## Sécurité et Permissions

---

### Groupes d'Utilisateurs

Le système définit quatre niveaux d'accès :

TABLE 3 – Groupes de sécurité

Groupe	Lecture	Écriture	Création	Suppression
Utilisateur				
Enseignant			*	
Gestionnaire				
Administrateur				

\*L'enseignant peut créer/modifier uniquement les notes et présences.

### Droits d'Accès par Modèle

- **Utilisateur** : Accès en lecture seule à tous les modèles
- **Enseignant** : Peut gérer les notes et présences de ses cours
- **Gestionnaire** : Peut gérer étudiants, classes, cours, emploi du temps
- **Administrateur** : Accès complet à tous les modèles

## Interface Utilisateur

---

### Structure des Menus

Le module propose une navigation organisée :

- **Gestion Scolaire** (Menu principal)
  - Étudiants
    - Tous les étudiants
  - Enseignants
    - Tous les enseignants
  - Académique
    - Classes
    - Cours
    - Emploi du temps
  - Notes
    - Toutes les notes
  - Présence

- Présence journalière
- Rapports de présence
- Rapports

## Vues Disponibles

Pour chaque modèle, plusieurs types de vues sont disponibles :

- **Vue arbre** : Liste des enregistrements
- **Vue formulaire** : Détails d'un enregistrement
- **Vue recherche** : Filtres et recherche
- **Vue pivot** : Analyse de données (pour notes)
- **Vue graphique** : Visualisations (pour notes)
- **Vue calendrier** : Emploi du temps (pour schedule)

## Captures d'écran de l'Application

---

Cette section présente les principales interfaces de l'application développée.

### Menu Principal

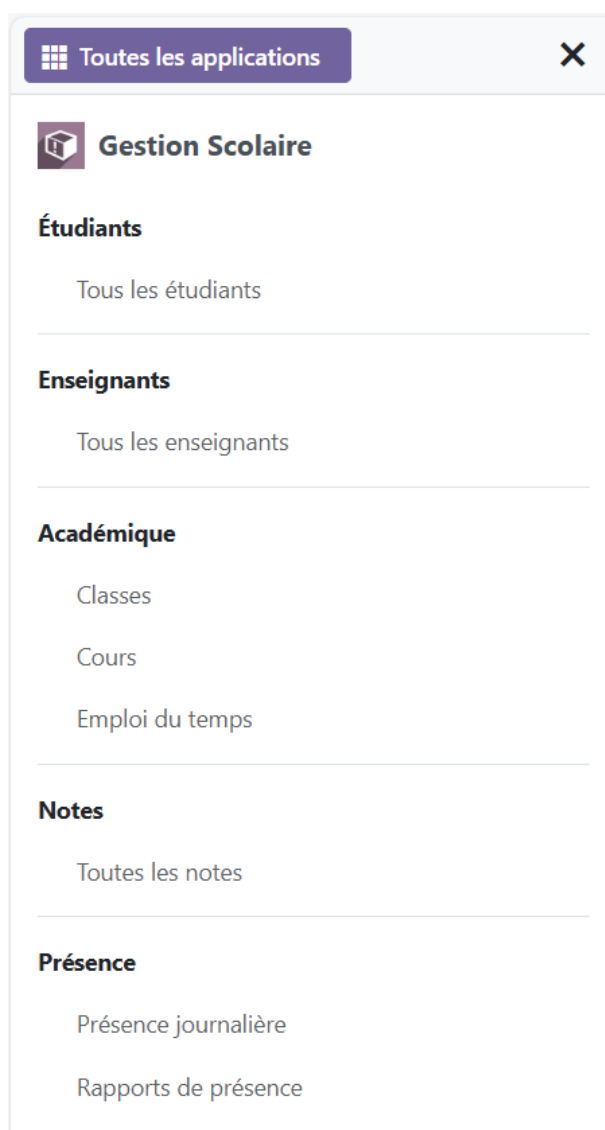
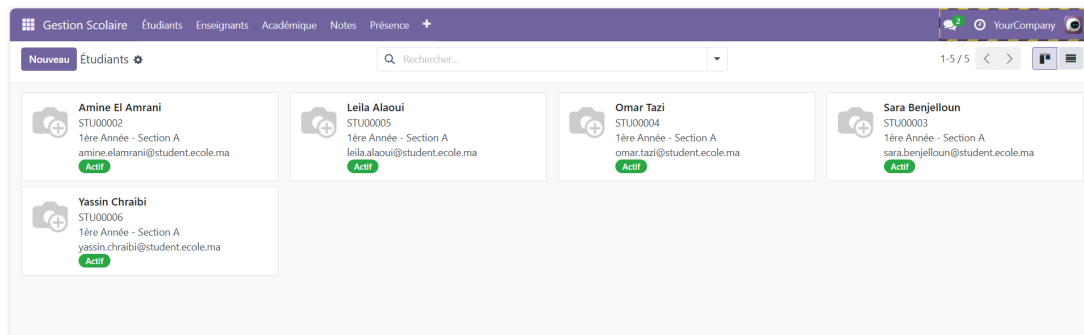


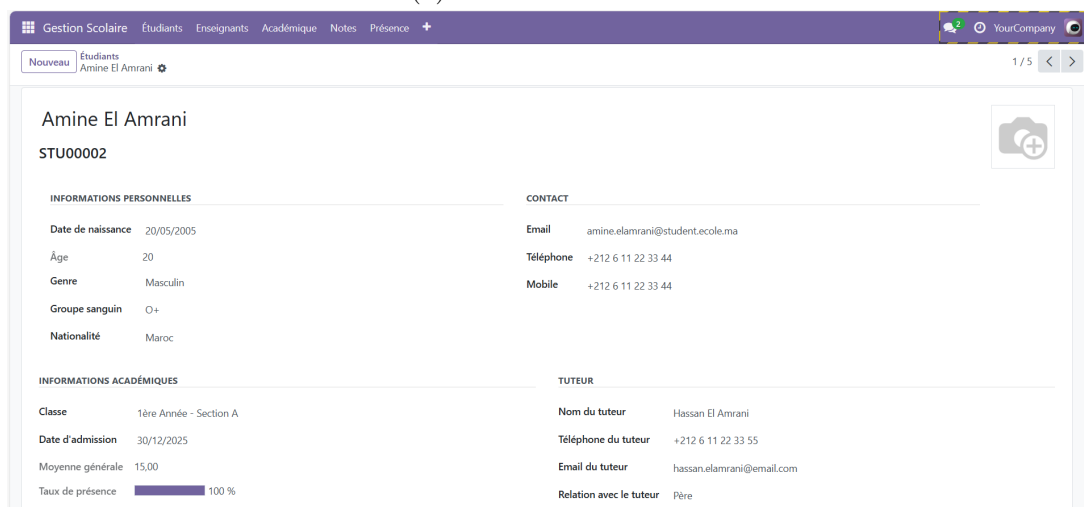
FIGURE 1 – Menu principal du module de gestion scolaire

Le menu principal offre un accès organisé à toutes les fonctionnalités du système.

## Gestion des Étudiants



(a) Liste des étudiants



(b) Fiche détaillée d'un étudiant

FIGURE 2 – Interface de gestion des étudiants

L'interface permet de visualiser la liste complète des étudiants et d'accéder à leurs fiches détaillées avec toutes les informations académiques et personnelles.

## Gestion des Enseignants

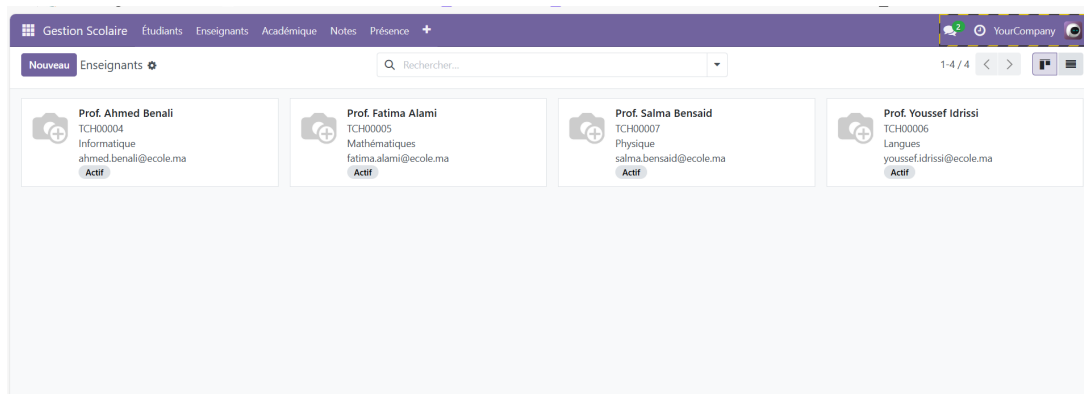
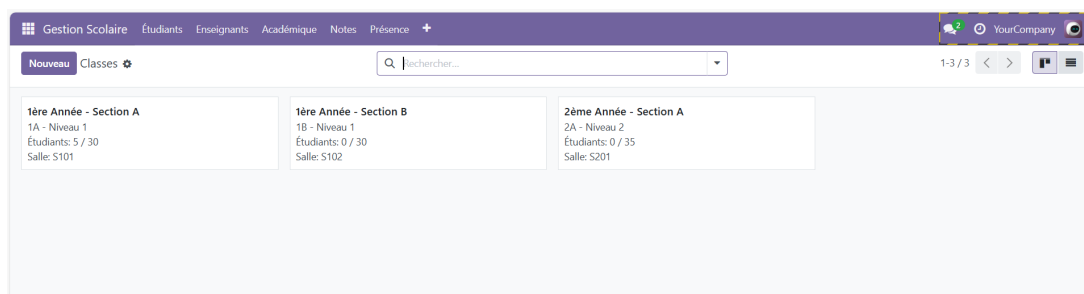
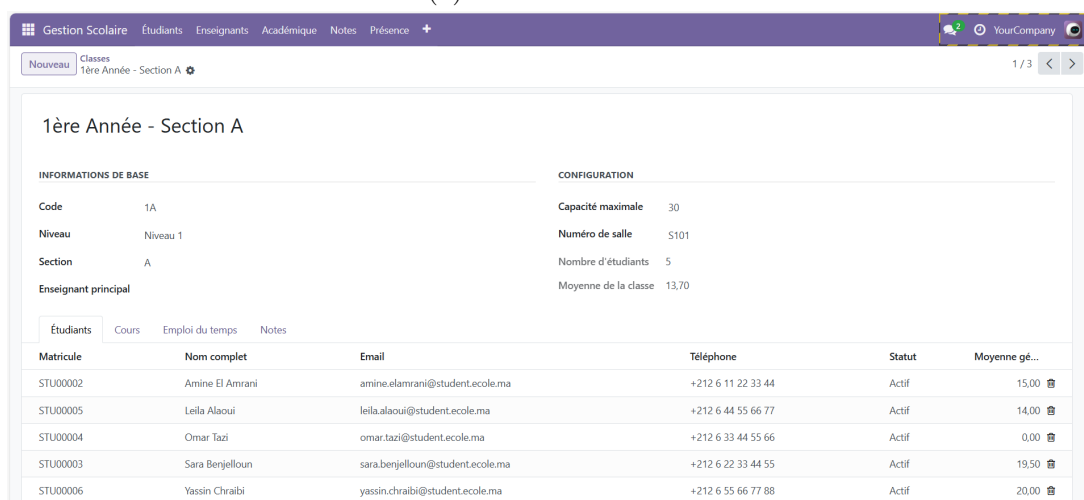


FIGURE 3 – Interface de gestion des enseignants

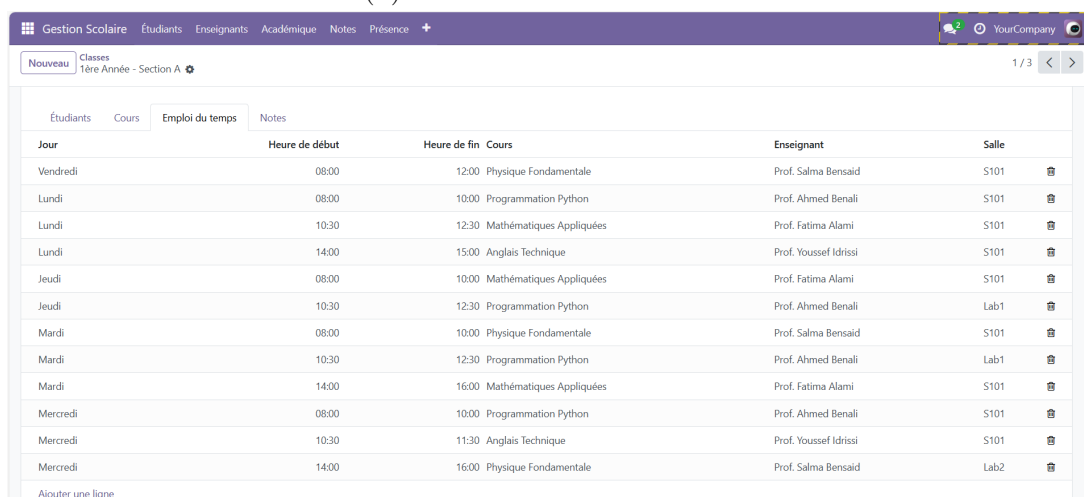
## Gestion des Classes et Cours



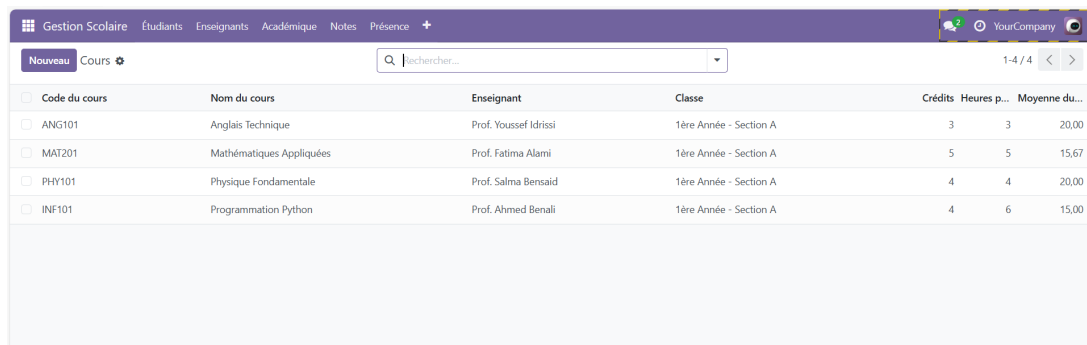
(a) Liste des classes



(b) Fiche détaillée d'une classe



(c) Emploi du temps d'une classe

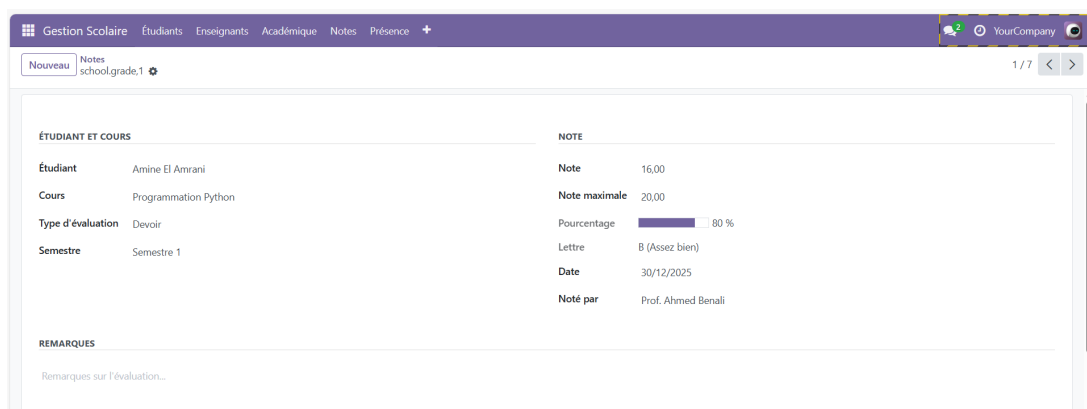


Code du cours	Nom du cours	Enseignant	Classe	Crédits	Heures p...	Moyenne du...
ANG101	Anglais Technique	Prof. Youssef Idrissi	1ère Année - Section A	3	3	20,00
IMAT201	Mathématiques Appliquées	Prof. Fatima Alami	1ère Année - Section A	5	5	15,67
PHY101	Physique Fondamentale	Prof. Salma Bensaid	1ère Année - Section A	4	4	20,00
INF101	Programmation Python	Prof. Ahmed Benali	1ère Année - Section A	4	6	15,00

(a) Liste des cours

FIGURE 5 – Interfaces de gestion académique

## Système de Notes

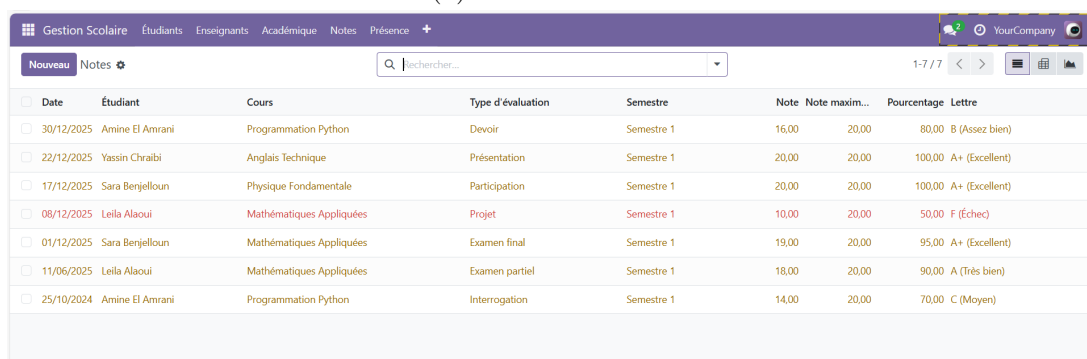


ÉTUDIANT ET COURS		NOTE	
Étudiant	Amine El Amrani	Note	16,00
Cours	Programmation Python	Note maximale	20,00
Type d'évaluation	Devoir	Pourcentage	80 %
Semestre	Semestre 1	Lettre	B (Assez bien)
		Date	30/12/2025
		Noté par	Prof. Ahmed Benali

**REMARQUES**

Remarques sur l'évaluation...

(a) Saisie des notes



Date	Étudiant	Cours	Type d'évaluation	Semestre	Note	Note maxim...	Pourcentage	Lettre
30/12/2025	Amine El Amrani	Programmation Python	Devoir	Semestre 1	16,00	20,00	80,00	B (Assez bien)
22/12/2025	Yassin Chaib	Anglais Technique	Présentation	Semestre 1	20,00	20,00	100,00	A+ (Excellent)
17/12/2025	Sara Benjelloun	Physique Fondamentale	Participation	Semestre 1	20,00	20,00	100,00	A+ (Excellent)
08/12/2025	Leila Alaoui	Mathématiques Appliquées	Projet	Semestre 1	10,00	20,00	50,00	F (Échec)
01/12/2025	Sara Benjelloun	Mathématiques Appliquées	Examen final	Semestre 1	19,00	20,00	95,00	A+ (Excellent)
11/06/2025	Leila Alaoui	Mathématiques Appliquées	Examen partiel	Semestre 1	18,00	20,00	90,00	A (Très bien)
25/10/2024	Amine El Amrani	Programmation Python	Interrogation	Semestre 1	14,00	20,00	70,00	C (Moyen)

(b) Listes des notes

FIGURE 6 – Interface de gestion des notes

Le système permet la saisie des notes avec calcul automatique des pourcentages et lettres, ainsi que la visualisation graphique des performances.



## Emploi du Temps

Jour	Heure de début	Heure de fin	Durée (heur...	Classe	Cours	Enseignant	Salle	Type de session
Vendredi	08:00	12:00	4,00	1ère Année - Section A	Physique Fondamentale	Prof. Salma Bensaid	S101	TD
Lundi	08:00	10:00	2,00	1ère Année - Section A	Programmation Python	Prof. Ahmed Benali	S101	Cours magistral
Lundi	10:30	12:30	2,00	1ère Année - Section A	Mathématiques Appliquées	Prof. Fatima Alami	S101	Cours magistral
Lundi	14:00	15:00	1,00	1ère Année - Section A	Anglais Technique	Prof. Youssef Idrissi	S101	Cours magistral
Jeudi	08:00	10:00	2,00	1ère Année - Section A	Mathématiques Appliquées	Prof. Fatima Alami	S101	Cours magistral
Jeudi	10:30	12:30	2,00	1ère Année - Section A	Programmation Python	Prof. Ahmed Benali	Lab1	TP
Mardi	08:00	10:00	2,00	1ère Année - Section A	Physique Fondamentale	Prof. Salma Bensaid	S101	Cours magistral
Mardi	10:30	12:30	2,00	1ère Année - Section A	Programmation Python	Prof. Ahmed Benali	Lab1	TP
Mardi	14:00	16:00	2,00	1ère Année - Section A	Mathématiques Appliquées	Prof. Fatima Alami	S101	Cours magistral
Mercredi	08:00	10:00	2,00	1ère Année - Section A	Programmation Python	Prof. Ahmed Benali	S101	Cours magistral
Mercredi	10:30	11:30	1,00	1ère Année - Section A	Anglais Technique	Prof. Youssef Idrissi	S101	Cours magistral
Mercredi	14:00	16:00	2,00	1ère Année - Section A	Physique Fondamentale	Prof. Salma Bensaid	Lab2	TP

(a) Vue calendrier

FIGURE 7 – Vues de l'emploi du temps

L'emploi du temps peut être visualisé sous différentes formes : calendrier hebdomadaire ou diagramme de Gantt pour une vue d'ensemble.

## Gestion de la Présence

Date	Étudiant	Classe	Cours	Statut	Heure d'arrivée	Heure de départ
30/12/2025	Amine El Amrani	1ère Année - Section A	Mathématiques Appliquées	Absent		
30/12/2025	Amine El Amrani	1ère Année - Section A	Physique Fondamentale	Présent		
30/12/2025	Leila Alaoui	1ère Année - Section A	Anglais Technique	En retard		
30/12/2025	Sara Benjelloun	1ère Année - Section A	Mathématiques Appliquées	Absent justifié		
30/12/2025	Yassin Chaïbi	1ère Année - Section A	Anglais Technique	Absent justifié		

FIGURE 8 – Interface de gestion de la présence

L'interface permet de marquer rapidement la présence des étudiants avec différents statuts (présent, absent, en retard, absent justifié).

## Rapports et Bulletins

Total		Anglais Technique		Mathématiques Appliquées		Physique Fondamentale		Programmation Python	
Note	Pourcentage	Note	Pourcentage	Note	Pourcentage	Note	Pourcentage	Note	Pourcentage
20,00	100,00	47,00	235,00	20,00	100,00	30,00	150,00	117,00	585,00
Amine El Amrani						30,00	150,00	30,00	150,00
Leila Alaoui		28,00	140,00					28,00	140,00
Sara Benjelloun		19,00	95,00	20,00	100,00			39,00	195,00
Yassin Chraïbi	20,00	100,00						20,00	100,00

(a) Bulletin de notes

Gestion Scolaire

Étudiants

Enseignants

Académique

Notes

Présence

+

Rapport de présence ⚙

🔍

Rechercher...

Mesures ▾

↔

+

📄

	☰ Total								
	+	Amine El Amrani	+	Leila Alaoui	+	Sara Benjelloun	+	Yassin Chraïbi	
		Comptage		Comptage		Comptage		Comptage	Comptage
☰ Total		2		1		1		1	5
+	Absent	1							1
+	Absent justifié					1		1	2
+	En retard			1					1
+	Présent	1							1

(b) Rapport de présence

FIGURE 9 – Rapports générés par le système

Le système génère automatiquement des rapports PDF professionnels incluant les bulletins de notes et les statistiques de présence.

## Rapports

---

### Types de Rapports

Le module génère plusieurs types de rapports en PDF :

1. **Fiche Étudiant** : Informations complètes d'un étudiant
2. **Bulletin de Notes** : Notes par semestre avec moyennes
3. **Rapport de Présence** : Statistiques de présence
4. **Emploi du Temps** : Planning hebdomadaire

### Template QWeb

Les rapports utilisent des templates QWeb pour la génération PDF, permettant :

- Mise en page personnalisée
- Inclusion de logos et images
- Calculs dynamiques
- Formatage conditionnel

## Implémentation Technique

---

### Modèles Python

Tous les modèles héritent de `models.Model` et utilisent :

- `mail.thread` : Suivi des modifications
- `mail.activity.mixin` : Activités et tâches
- Champs calculés avec `@api.depends`
- Contraintes avec `@api.constrains`
- Méthodes de workflow (`action_set_active`, etc.)

### Séquences Automatiques

Le système utilise des séquences Odoo pour générer automatiquement :

- Matricules étudiants
- Matricules enseignants

### Validations et Contraintes

- **SQL Constraints** : Unicité des codes (cours, classes)
- **Python Constraints** : Validation des dates, emails, notes
- **ORM Constraints** : Prévention des doublons (présence)

## Calculs Automatiques

Le système calcule automatiquement :

- Âge à partir de la date de naissance
- Pourcentage des notes
- Lettres de notation
- Moyennes (étudiant, cours, classe)
- Taux de présence
- Durée des sessions
- Nombre d'étudiants par classe
- Nombre de cours par enseignant

## Tests et Validation

---

### Scénarios de Test

Les fonctionnalités suivantes ont été testées :

1. Création d'un étudiant avec génération automatique de matricule
2. Calcul correct de l'âge et de la moyenne
3. Validation des emails et dates
4. Détection des conflits d'horaire
5. Prévention des doublons de présence
6. Calcul correct des pourcentages et lettres
7. Génération des rapports PDF
8. Respect des permissions par groupe

### Contraintes Validées

- Unicité des codes de cours et classes
- Notes entre 0 et note maximale
- Dates cohérentes (pas de dates futures pour naissance)
- Heures d'emploi du temps valides (0-24h)
- Pas de chevauchement d'horaires

## Conclusion

---

### Réalisations

#### Principales réalisations

Ce projet a permis de développer un module complet de gestion scolaire intégré à Odoo 18, offrant :

- Une solution centralisée pour la gestion académique
- Des calculs automatiques fiables
- Une interface intuitive et organisée
- Un système de sécurité robuste
- Des rapports professionnels
- Une architecture extensible

### Perspectives d'Amélioration

Des améliorations futures pourraient inclure :

- Intégration avec un système de paiement des frais
- Module de communication (messagerie parents-enseignants)
- Application mobile pour étudiants et parents
- Intégration avec systèmes d'examens externes
- Analytics avancés et tableaux de bord
- Export de données vers formats standards
- Système de notifications automatiques
- Gestion des bibliothèques et ressources

### Apports Pédagogiques

Ce projet a permis de :

- Maîtriser le framework Odoo et son ORM
- Comprendre l'architecture MVC dans Odoo
- Implémenter des règles métier complexes
- Gérer la sécurité et les permissions
- Développer des interfaces utilisateur
- Générer des rapports professionnels
- Appliquer les bonnes pratiques de développement

## Annexes

---

### Annexe A : Dépendances

Le module dépend des modules Odoo suivants :

- **base** : Fonctionnalités de base
- **mail** : Système de messagerie et suivi
- **web** : Interface web

### Annexe B : Installation

1. Copier le module dans le répertoire **addons** d'Odoo
2. Redémarrer le serveur Odoo
3. Activer le mode développeur
4. Mettre à jour la liste des applications
5. Installer le module "Gestion Scolaire"
6. Configurer les groupes d'utilisateurs
7. Créer les classes, enseignants et cours
8. Ajouter les étudiants

### Annexe C : Licence

Ce module est distribué sous licence **LGPL-3**, permettant :

- Utilisation commerciale
- Modification
- Distribution
- Utilisation privée

*Fin du rapport*