UNIVERSITE DU BURUNDI

FACULTE DES SCIENCES DE L'INGENIEUR

DEPARTEMENT DES TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

Filière : Génie Informatique & Réseaux et télécommunications

Cours : APPLICATION WEB

Enseignant : Msc Ir CIZA Innocent

UserFlow : Gestion des utilisateurs et accès.

D'EL SHADDAI First: 20/03612
IRAMBONA Dorine: 20/03956
KAZE Kelly Darlène: 20/
NTAHIMPERA Déo: 20/03782
SINDAYIDAYA Franck: 20/03801

Date de Soumission : Le 18/04/2025

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce projet. Chaque soutien, chaque conseil et chaque mot d'encouragement a été une source de motivation précieuse dans notre progression.

Nous adressons un remerciement tout particulier à notre Enseignant, **MscIr CIZA Innocent**, pour sa patience, sa disponibilité et la richesse de ses conseils. Son accompagnement constant tout au long de ce projet nous a permis d'avancer sereinement, de surmonter les difficultés rencontrées, et d'élever la qualité de notre travail. Ses remarques constructives et son regard critique ont largement contribué à la maturité de notre réflexion.

Nous remercions également notre Université, **Université du Burundi, Faculté des Sciences de l'Ingénieur, Département des TIC**, pour les ressources matérielles, pédagogiques et techniques mises à notre disposition. Le cadre d'apprentissage fourni a grandement facilité notre travail, et nous a permis de mettre en œuvre nos compétences dans des conditions optimales.

Enfin, nous exprimons notre reconnaissance envers nos camarades de promotion. Les discussions, les partages d'expériences et l'entraide entre pairs ont enrichi cette aventure humaine et professionnelle. Leur soutien moral et leur bienveillance ont apporté une atmosphère collaborative et motivante tout au long de cette expérience.

Résumé (Abstract)

Ce projet consiste à développer un système de gestion des utilisateurs, rôles et permissions basé sur Django. L'objectif principal est de fournir une interface d'administration sécurisée et ergonomique permettant de créer, modifier et attribuer des rôles et permissions aux utilisateurs via des APIs et une interface en JavaScript. Grâce à une structure modulaire, ce système permet une gestion granulaire des accès. Les résultats incluent un back-office complet avec formulaires dynamiques, modals de gestion, et API personnalisées. Le projet met l'accent sur la sécurité, l'extensibilité et l'expérience utilisateur. Il peut être intégré à des systèmes existants ou utilisé comme base pour des applications plus larges.

Il permet :

- La **création/gestion des rôles** (Admin, Éditeur, etc.)

- L'**attribution dynamique de permissions** (créer/supprimer du contenu, gérer les utilisateurs, etc.)
- Une **API REST sécurisée** avec authentification JWT
- Une **interface admin** et un **frontend en Tailwind CSS**

Méthodologie : Développement agile avec tests unitaires. Résultat : Une application modulaire, sécurisée et scalable pour contrôler l'accès aux ressources.

Table des matières

Introduction

Contexte du projet

Avec l'augmentation des systèmes web complexes, la gestion des utilisateurs et de leurs accès devient une nécessité cruciale pour la sécurité et l'organisation. Ce projet s'inscrit dans ce besoin en fournissant une solution complète et flexible pour l'administration des utilisateurs.

Justification ou problématique

Les outils standards sont souvent trop rigides ou insuffisants pour des besoins spécifiques. Le défi était de construire un système personnalisable et modulaire permettant une gestion détaillée des rôles et permissions avec une bonne ergonomie.

Objectifs

Objectif général : Développer un système web de gestion des utilisateurs, rôles et permissions basé sur Django.

Objectifs spécifiques :

- Implémenter une base de données relationnelle modélisant les utilisateurs, rôles et permissions.
- Créer une API REST sécurisée pour interagir avec les données.
- Concevoir une interface moderne en HTML/TailwindCSS/JS pour la gestion dynamique via modals.
- Permettre l'assignation facile de rôles aux utilisateurs et de permissions aux rôles.

Méthodologie générale

- Utilisation du framework Django pour le backend.
- Développement d'APIViews personnalisées avec Django Rest Framework.
- Interface frontend avec TailwindCSS et vanilla JavaScript.
- Approche modulaire et centrée sur l'utilisateur.

Structure du rapport

Le rapport décrit d'abord le projet, son déroulement, les résultats obtenus, une discussion critique, des recommandations et se termine par une conclusion générale.

Développement du projet

a. Présentation du projet

Thème

Développement d'un système de gestion des utilisateurs, rôles et permissions pour une application web sécurisée.

Objectifs visés

Faciliter l'administration des accès utilisateurs dans un environnement multi-rôles.

Outils utilisés

- **Backend** :
  - **Python** : Langage de programmation utilisé pour écrire la logique du serveur.
  - **Django** : Framework web Python pour créer rapidement des applications sécurisées et maintenables.
  - **Django Rest Framework (DRF)** : Extension de Django pour créer facilement des API RESTful.
  - **MySQL Client**: facilitates database connectivity and translates ORM commands into SQL for MySQL.
- **Frontend** :
  - **HTML** : Langage de base pour structurer les pages web.
  - **TailwindCSS** : Framework CSS utilitaire pour créer rapidement des interfaces responsives et modernes.
  - **JavaScript** : Langage pour rendre les pages interactives (modals, fetch, actions dynamiques).
- **Base de données** :
  - **MySQL** : Système de gestion de base de données relationnelle, utilisé pour stocker les utilisateurs, rôles, permissions, etc.
- **Environnement** :
  - **VS Code** : Éditeur de code utilisé pour écrire et organiser le code du projet.
  - **Git** : Outil de versionnement pour suivre les changements et collaborer en équipe.
  - **Postman** : Outil pour tester les API, simuler les requêtes et vérifier les réponses du backend.

b. Déroulement du projet

Étapes

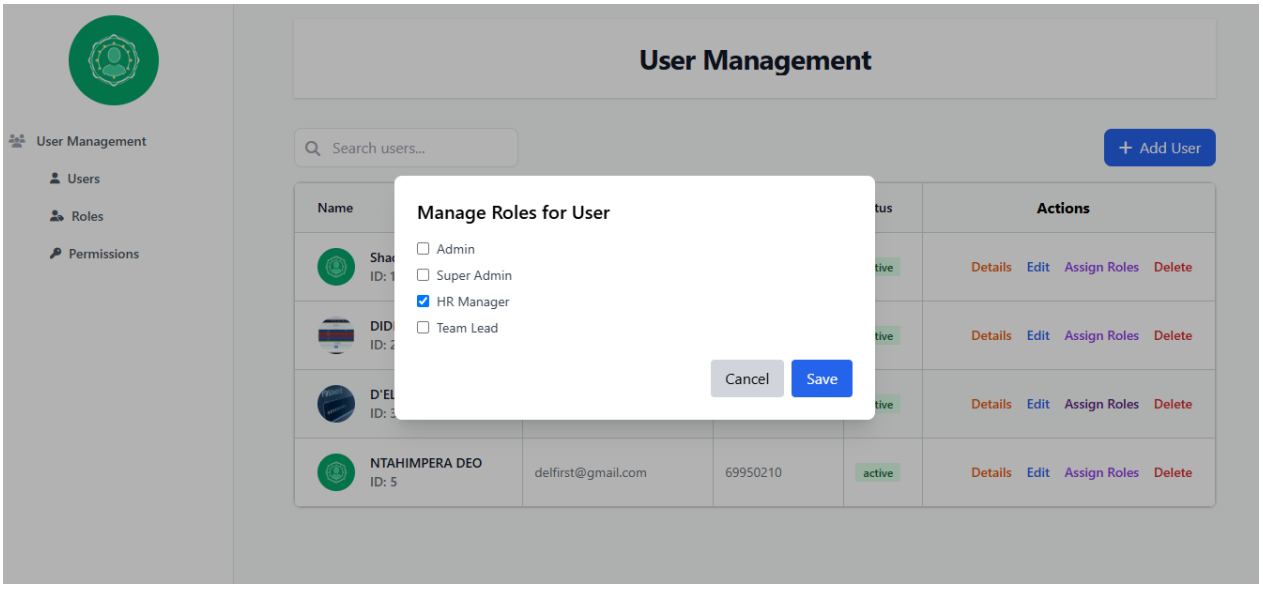
1. Conception des modèles : User, Role, Permission, UserRole, RolePermission
2. Création des APIViews : CRUD utilisateurs, assignation rôles/permissions
3. Développement de l'interface : affichage dynamique, modals, formulaires
4. Liaison entre frontend et backend avec JavaScript
5. Tests fonctionnels et ajustements UI/UX

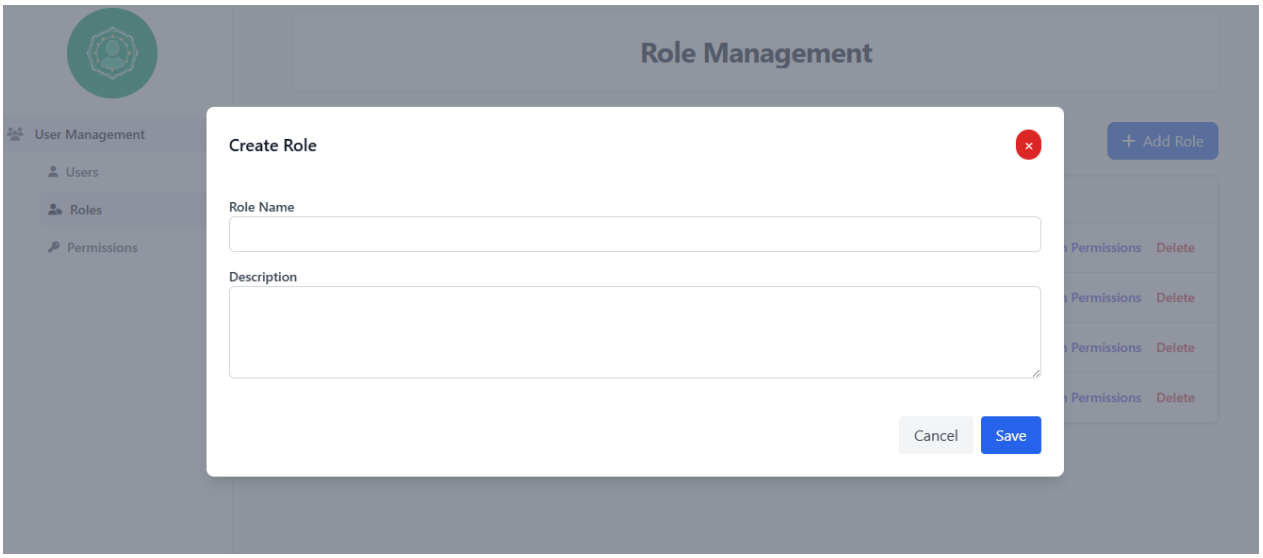
Répartition des tâches (exemple à adapter)

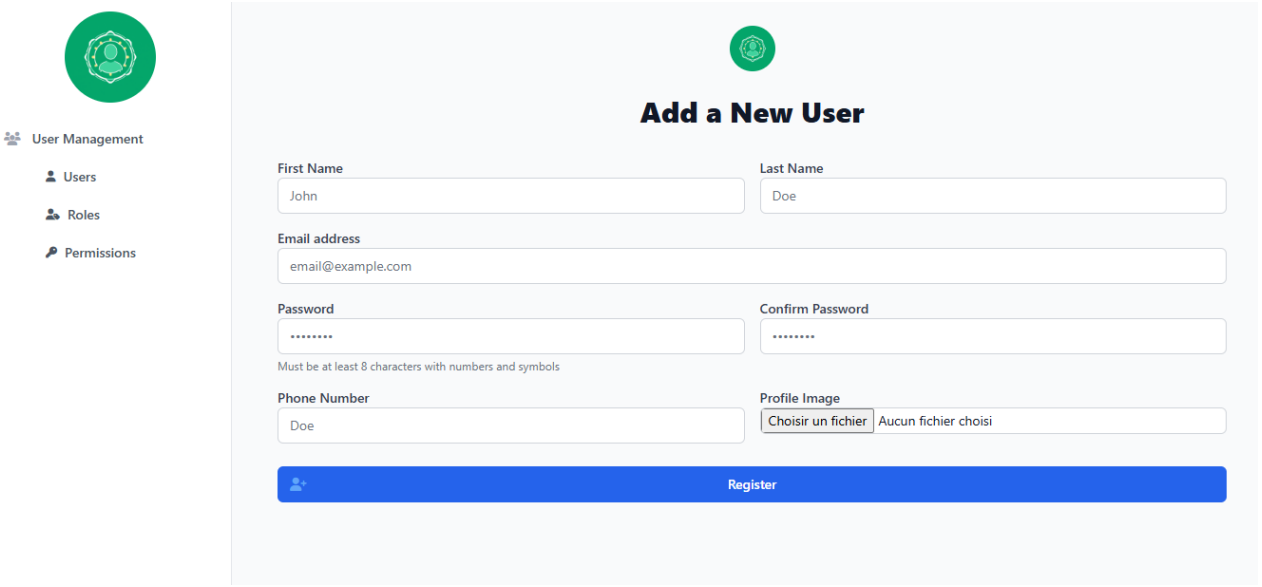
- Développement backend : Membre 1 (D'EL SHADDAÏ First)
- Intégration frontend : Membre 2 (NTAHIMPERA Déo)
- Connexion API/JS : Membre 3 (SINDAYIGAYA Franck)
- Tests : Membre 4 (KAZE Kelly Darlène)
- Documentation : Membre 5 (IRAMBONA Dorine)

Illustrations

(Captures des écrans des pages : liste des utilisateurs, formulaire de rôle, modal d'assignation, etc.)

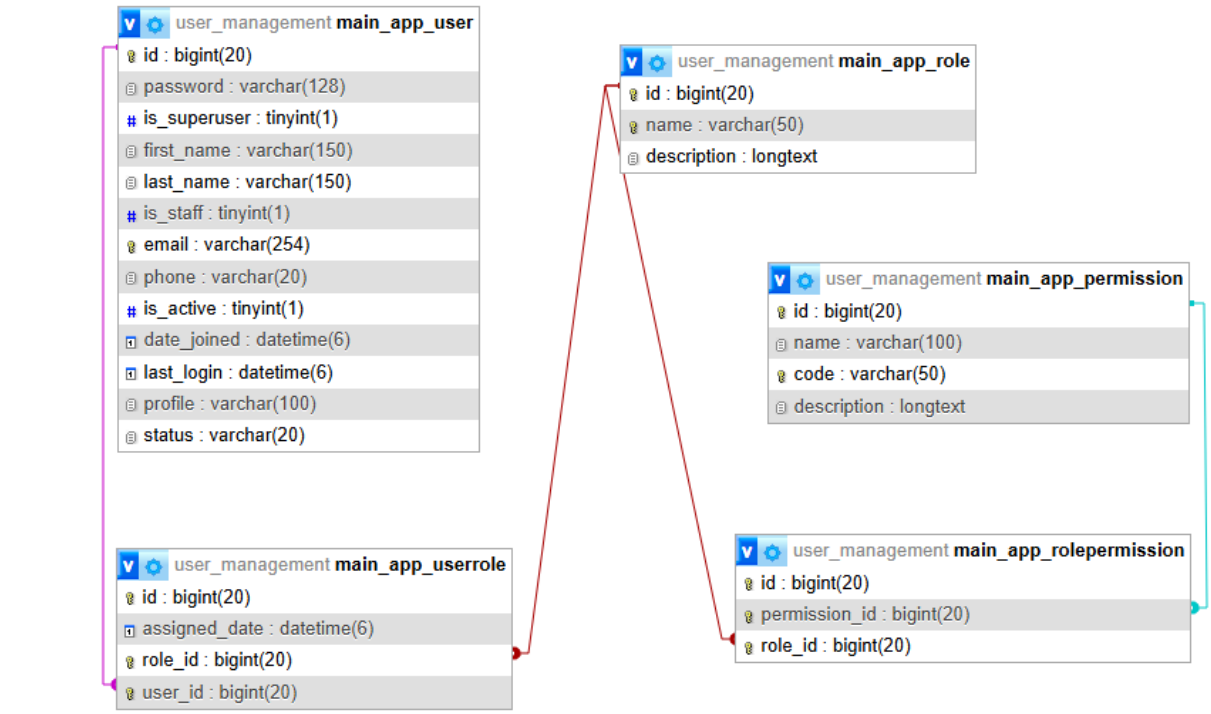




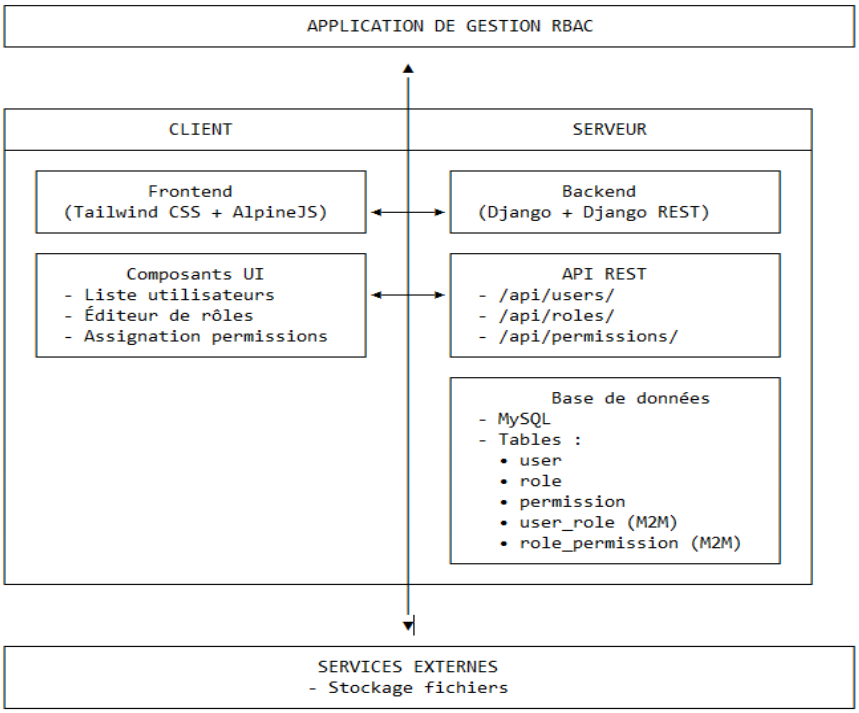


Liste des Figures :

- **Diagramme UML des modèles**



- **Schéma d'architecture ((Format UML Simplifié)**



c. Résultats obtenus

Produit livré

Un tableau de bord fonctionnel permettant :

- De créer/éditer des utilisateurs, rôles, permissions
- D'assigner dynamiquement des rôles aux utilisateurs
- D'assigner des permissions aux rôles via modals interactifs
- De consulter les détails d'un utilisateur ou d'un rôle

Fonctionnalités

- API sécurisée et personnalisée
- Formulaires validés dynamiquement
- Gestion asynchrone via JavaScript

d. Discussion

Analyse critique

- Projet fonctionnel et ergonomique
- Bonne modularité
- Interface intuitive

Difficultés rencontrées

- Gestion des IDs entre API/JS (problème de Set corrigé)
- Synchronisation entre frontend et backend (notamment les checkboxes)
- Styling conditionnel en JavaScript

Solutions apportées

- Logs pour debugging
- Refactorisation des fonctions JS
- Nettoyage du code CSS/Tailwind

e. Recommandations (facultatif)

- Ajouter une recherche côté API
- Ajouter un système de journalisation (logs) d'actions
- Intégration à des systèmes externes via JWT

Conclusion

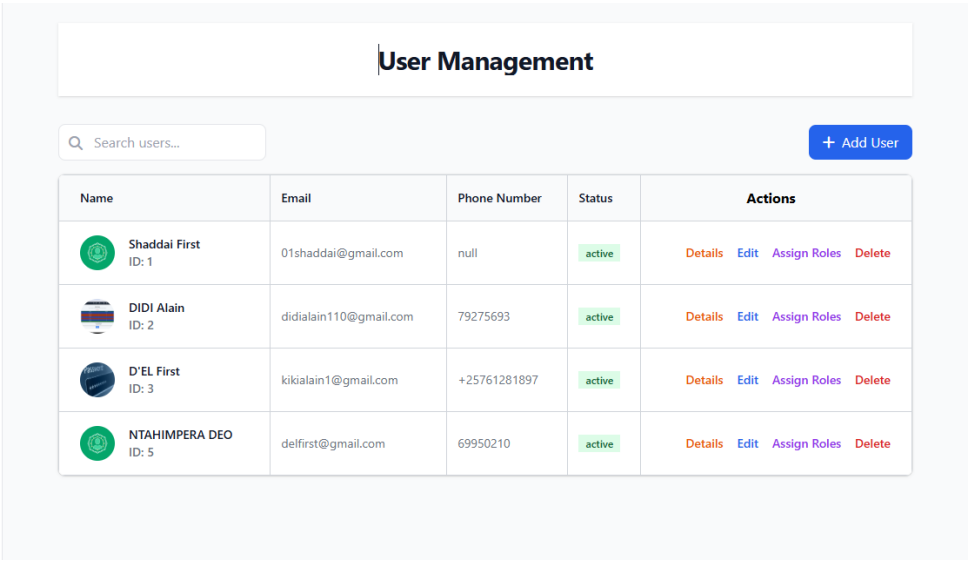
Ce projet nous a permis de renforcer nos compétences en développement web fullstack, en particulier sur le framework Django et l'intégration frontend avec JavaScript. Nous avons appris à structurer un projet de A à Z, à collaborer efficacement et à résoudre des problèmes techniques concrets. Le système est prêt à être étendu ou intégré dans des projets plus ambitieux.

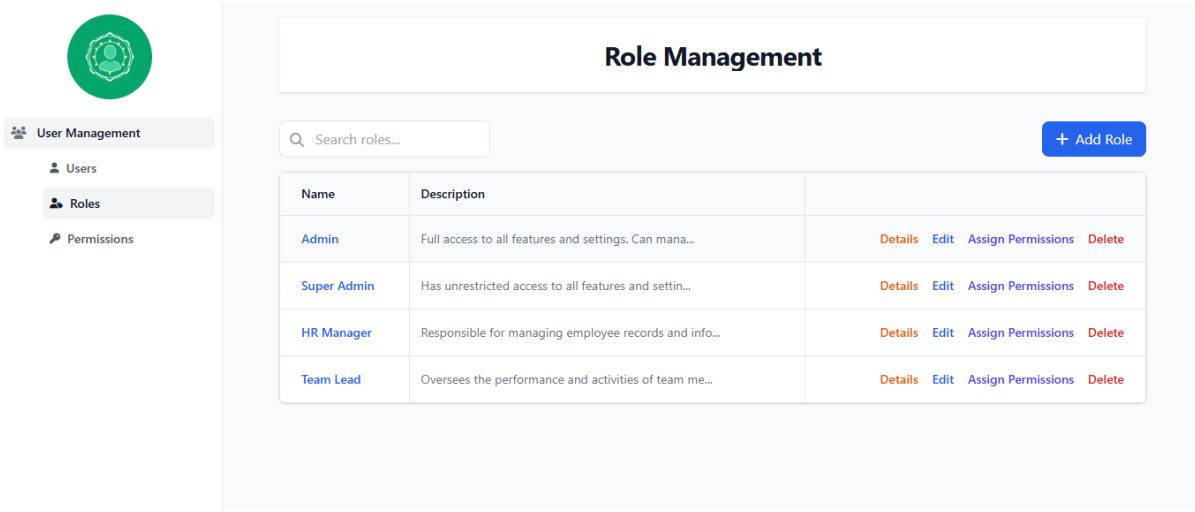
Références

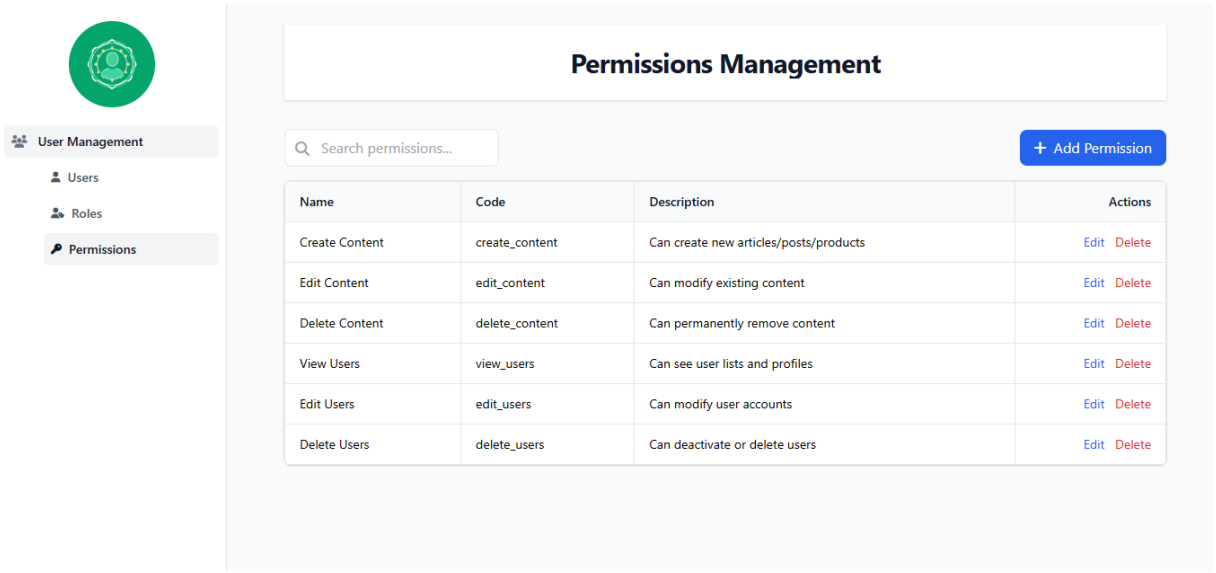
- Documentation Django
- Django Rest Framework
- Tailwind CSS
- Tutoriels YouTube, StackOverflow, ChatGPT

Annexes

- Captures d'écrans supplémentaires







- Code source sur GitHub :
    - Extraits de code :

```
class User(AbstractUser):
    """Modèle utilisateur personnalisé"""

    username = None
    email = models.EmailField(_('email address'), unique=True)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []

    objects = UserManager()
```

```python
    # Champs supplémentaires
    phone = models.CharField(max_length=20, blank=True, null=True)
    profile = models.FileField(upload_to='media/', default=None, blank=True, null=True)
    is_active = models.BooleanField(default=True)
    date_joined = models.DateTimeField(auto_now_add=True)
    last_login = models.DateTimeField(auto_now=True)
    status = models.CharField(
        max_length=20,
        choices=[
            ('active', 'Active'),
            ('not active', 'Not Active'),
            ('disabled', 'Disabled')
        ],
        default='active'
    )

    def __str__(self):
        return self.email

    def clean(self):
        super().clean()
        try:
            validate_email(self.email)
        except ValidationError:
            raise ValidationError({'email': 'Enter a valid email address.'})


class Role(models.Model):
    """Modèle pour les rôles utilisateurs"""
    name = models.CharField(max_length=50, unique=True)
    description = models.TextField(blank=True)

    def __str__(self):
        return self.name


class Permission(models.Model):
    """Modèle pour les permissions"""
```

```
name = models.CharField(max_length=100)
code = models.CharField(max_length=50, unique=True)
description = models.TextField(blank=True)

def __str__(self):
    return f"{self.name} ({self.code})"
```

https://github.com/FirDel67/Gestion-des-Utilisateurs.git