

# 0. Synthèse de l'implémentation effective (Déc. 2025)

---

Cette section liste, point par point, ce qui a été effectivement implémenté et fonctionne déjà dans le backend et le frontend, conformément au MCD et aux exigences du module Utilisateurs & RH.

## 0.1 Gestion des utilisateurs (authentification, rôles, accès)

- Création, modification, désactivation (soft delete) et listing des utilisateurs (admin, opérateur, client)
- Authentification JWT opérationnelle (connexion, refresh, vérification)
- Les utilisateurs peuvent se connecter et accéder à leur espace respectif selon leur rôle :
  - **ADMIN** : accès complet à l'administration, gestion des équipes, RH, reporting, etc.
  - **OPÉRATEUR** : accès à l'espace opérateur, interventions, planning, inventaire, réclamations
  - **CLIENT** : accès au portail client, suivi des interventions, inventaire, réclamations
- Le mapping des rôles backend/frontend est strictement respecté (pas de mismatch)
- Les droits d'accès sont vérifiés côté backend et frontend (menu, endpoints, UI)

## 0.2 Gestion des rôles et permissions

- Modèle N-N utilisateur/rôle (un utilisateur peut avoir plusieurs rôles)
- Attribution automatique des rôles à la création (ex : opérateur + chef d'équipe)
- Endpoints de gestion des rôles exposés

## 0.3 Gestion des opérateurs

- CRUD complet opérateur (création, modification, désactivation, listing)
- Statut (actif, inactif, en congé) géré et visible
- Affectation à une ou plusieurs équipes (chef ou membre)
- Soft delete avec retrait automatique des équipes et conservation de l'historique

## 0.4 Gestion des clients

- CRUD complet client (création, modification, listing)
- Lien 1-1 avec utilisateur

## 0.5 Gestion des équipes

- CRUD complet équipe (création, modification, désactivation, listing)
- Chef d'équipe obligatoire (contrainte respectée)
- Un opérateur peut être chef de plusieurs équipes
- Affectation/désaffectation dynamique des membres

## 0.6 Gestion des compétences

- CRUD complet compétence (création, modification, listing)
- Attribution des compétences aux opérateurs avec niveau (débutant, intermédiaire, expert, etc.)
- Matrice de compétences visible et modifiable dans l'UI

## 0.7 Gestion des absences

- CRUD complet absence (création, modification, suppression, listing)
- Validation des statuts (demandée, validée, refusée)
- Impact automatique sur la disponibilité des équipes

## 0.8 Historique RH

- Historique des affectations équipes/opérateurs
- Historique des absences
- Historique des compétences (date d'acquisition)
- Endpoints de recherche par opérateur, équipe, période

## 0.9 Intégration frontend

- Sidebar, navigation et affichage des pages selon le rôle utilisateur (admin, opérateur, client)
- Les utilisateurs voient uniquement les pages et actions autorisées par leur rôle
- Portail client, espace opérateur, espace admin fonctionnels et différenciés
- Les tests de connexion et de navigation par rôle sont validés

---

## 7. Gestion des migrations & api\_users\_migration.sql

Après l'ajout du module `api_users`, des problèmes de migration peuvent survenir (conflits, dépendances, initialisation de la base). Pour garantir une base cohérente **sans formater toute la base de données** (et donc sans perdre les données d'inventaire ou de cartographie déjà présentes) :

- Le fichier `api_users_migration.sql` contient le script SQL de migration initiale pour toutes les tables du module utilisateurs et RH (conforme au MCD).
- Il permet de créer ou corriger uniquement la structure liée à `api_users` sans toucher aux autres tables existantes.
- Utilisation recommandée :
  1. Sauvegarder les données importantes
  2. Supprimer les anciennes migrations Django du module concerné si besoin (`find . -path "*/migrations/*.py" -not -name "__init__.py" -delete`)
  3. Appliquer le script SQL sur la base (via psql, pgAdmin, etc.)
  4. Recréer les migrations Django (`python manage.py makemigrations api_users` puis `python manage.py migrate`)
- Ce script doit toujours être synchronisé avec le modèle de données décrit dans cette documentation.

### Remarque :

- Cette méthode permet de ne pas formater la base complète et de préserver toutes les autres données métiers (inventaire, cartographie, etc.)

### Attention :

- Toujours vérifier la cohérence entre le schéma SQL, les modèles Django et le MCD avant toute modification majeure.
- En cas de doute, consulter l'équipe technique ou la documentation du projet.

# Documentation du module Utilisateurs & RH (api\_users)

---

## 1. Modélisation conforme au MCD

### Entités principales

- **UTILISATEUR :**

- id\_utilisateur (PK)
- nom, prenom, email (unique), mot\_de\_passe\_hash
- type\_utilisateur : 'ADMIN', 'OPERATEUR', 'CLIENT'
- date\_creation, actif (bool), derniere\_connexion

- **CLIENT :**

- id\_client (PK, FK → UTILISATEUR)
- nom\_structure, adresse, telephone, contact\_principal, email\_facturation, logo

- **OPERATEUR :**

- id\_operateur (PK, FK → UTILISATEUR)
- nom, prenom, numero\_immatriculation, statut (Actif/Inactif/En congé)
- id\_equipe (FK → EQUIPE), date\_embauche, photo, telephone

- **ROLE :**

- id\_role (PK), nom\_role ('ADMIN', 'CLIENT', 'CHEF\_EQUIPE', ...), description

- **UTILISATEUR\_ROLE :**

- id\_utilisateur (FK), id\_role (FK)
- N-N, PK composée

- **EQUIPE :**

- id\_equipe (PK), nom\_equipe, id\_chef\_equipe (FK), specialite, statut, date\_creation

- **COMPETENCE :**

- id\_competence (PK), nom\_competence, categorie, description

- **COMPETENCE\_OPERATEUR :**

- id\_competence (FK), id\_operateur (FK), niveau\_competence

- **ABSENCE :**

- id\_absence (PK), id\_jardinier (FK), type\_absence, date\_debut, date\_fin, statut

- **HISTORIQUE\_EQUIPE\_OPERATEUR :**

- id, id\_operateur (FK), id\_equipe (FK), date\_debut, date\_fin, role\_dans\_equipe

## 2. Fonctionnalités implémentées

- CRUD complet sur Utilisateur, Opérateur, Client, Equipe, Compétence, Absence
- Gestion des rôles multiples par utilisateur (ADMIN, OPERATEUR, CHEF\_EQUIPE, CLIENT)
- Soft delete sur les opérateurs (désactivation, retrait des équipes, historique conservé)
- Attribution et gestion des compétences par opérateur (niveau, date)
- Gestion des absences (CRUD, validation, impact sur équipes)
- Historique RH : affectations équipes, absences, compétences
- Contraintes d'intégrité :
  - Un chef d'équipe doit avoir la compétence "Gestion d'équipe"
  - Un opérateur peut être chef de plusieurs équipes
  - Un utilisateur peut avoir plusieurs rôles

## 3. API & Endpoints principaux

### Utilisateurs

- GET /api/users/ : Liste des utilisateurs
- POST /api/users/ : Création utilisateur
- GET /api/users/<id>/ : Détail utilisateur
- PUT/PATCH /api/users/<id>/ : Modification
- DELETE /api/users/<id>/ : Désactivation (soft delete)
- GET /api/users/me/ : Profil connecté

### Opérateurs

- GET /api/operators/ : Liste des opérateurs
- POST /api/operators/ : Création opérateur
- GET /api/operators/<id>/ : Détail opérateur
- PUT/PATCH /api/operators/<id>/ : Modification
- DELETE /api/operators/<id>/ : Désactivation

### Clients

- GET /api/clients/ : Liste des clients
- POST /api/clients/ : Création client
- GET /api/clients/<id>/ : Détail client

### Rôles

- GET /api/roles/ : Liste des rôles
- POST /api/roles/ : Création rôle

## Compétences

- `GET /api/competences/` : Liste des compétences
- `POST /api/competences/` : Création compétence
- `GET /api/competences/<id>/` : Détail compétence

## Compétences Opérateur

- `GET /api/operators/<id>/competences/` : Compétences d'un opérateur
- `POST /api/operators/<id>/competences/` : Affecter/mettre à jour une compétence

## Équipes

- `GET /api/teams/` : Liste des équipes
- `POST /api/teams/` : Création équipe
- `GET /api/teams/<id>/` : Détail équipe
- `PUT/PATCH /api/teams/<id>/` : Modification
- `DELETE /api/teams/<id>/` : Désactivation

## Absences

- `GET /api/absences/` : Liste des absences
- `POST /api/absences/` : Création absence
- `GET /api/absences/<id>/` : Détail absence
- `PUT/PATCH /api/absences/<id>/` : Modification
- `DELETE /api/absences/<id>/` : Suppression

## Historique RH

- `GET /api/operators/<id>/history/` : Historique RH opérateur
- `GET /api/teams/<id>/history/` : Historique équipe

## 6. Convention de nommage (Backend/Frontend)

Pour garantir une intégration fluide entre le backend Django et le frontend React, les conventions suivantes sont appliquées :

- Les noms de rôles sont strictement identiques côté backend et frontend :
  - 'ADMIN', 'OPERATEUR', 'CLIENT', 'CHEF\_EQUIPE' (pas de traduction ni d'alias)
- Les noms des champs exposés par l'API REST sont en snake\_case (ex : `type_utilisateur`, `date_creation`)
- Les valeurs d'énumération (statut, type, rôle, etc.) sont transmises en majuscules, sans accent ni espace
- Les endpoints REST suivent la convention `/api/<ressource>/` (pluriel)
- Les objets JSON échangés utilisent les mêmes clés que les modèles Django (sauf cas d'alias documenté)
- Les statuts et types sont documentés dans le frontend pour éviter tout mismatch
- Les modifications de convention sont synchronisées entre les deux équipes (doc commune)

## Exemple :

```
{  
    "id": 12,  
    "nom": "Idrissi",  
    "prenom": "Hassan",  
    "email": "hassan.idrissi@greensig.ma",  
    "type_utilisateur": "OPERATEUR",  
    "roles": ["OPERATEUR", "CHEF_EQUIPE"],  
    "actif": true  
}
```

Le module utilise l'authentification JWT (JSON Web Token) pour sécuriser l'accès à l'API.

## Endpoints d'authentification

- [POST /api/token/](#) : Obtention d'un access token et refresh token
  - Body : { "email": "...", "password": "..." }
  - Réponse : { "access": "...", "refresh": "..." }
- [POST /api/token/refresh/](#) : Rafraîchir un access token expiré
  - Body : { "refresh": "..." }
  - Réponse : { "access": "..." }
- [POST /api/token/verify/](#) : Vérifier la validité d'un token
  - Body : { "token": "..." }

## Utilisation

- Le token d'accès doit être envoyé dans l'en-tête HTTP :
  - [Authorization: Bearer <access\\_token>](#)
- Le refresh token permet d'obtenir un nouveau access token sans se reconnecter.
- Les endpoints protégés nécessitent un access token valide.

## Sécurité

- Les tokens sont signés et expirent automatiquement (durée configurable)
- Les permissions sont vérifiées selon le rôle de l'utilisateur
- Toutes les entités sont soft-deletées sauf indication contraire
- Les endpoints respectent la sécurité (permissions, droits par rôle)
- Les validations métier sont appliquées côté backend (ex : chef d'équipe doit avoir la compétence adéquate)
- Les données sont exploitables pour la planification et le reporting RH

Pour toute extension ou modification, se référer au MCD et à cette documentation pour garantir la cohérence du modèle et des API.

---

**greenSIGteam**

Contact : greensig7@gmail.com

© 2025 GreenSIG. Tous droits réservés.