GUIDE COMPLET

Structure des Utilisateurs

Architecture Base de Données — Plan de Classe

Application :	Plan de Classe
Version :	1.0.0
Dernière MAJ :	08/12/2025
Base de données :	Supabase (PostgreSQL)

1. Vue d'Ensemble

1.1 Architecture des tables

Le système de gestion des utilisateurs repose sur une architecture relationnelle avec une table centrale **profiles** et des tables d'extension pour les rôles spécifiques.

Table Principale	Tables d'Extension	Tables de Référence
`profiles` *(authentification)*	• `students` • `teachers`	• `establishments` • `classes`

1.2 Principe clé : Table profiles centrale

⚠ La table profiles est la TABLE PRINCIPALE pour l'authentification de TOUS les utilisateurs.

- Les tables `students` et `teachers` sont des extensions avec des informations spécifiques au rôle
- Le lien utilisateur → établissement se fait via `profiles.establishment_id`
- L'authentification utilise `username` + `password_hash` (bcrypt)

2. Structure des Tables

2.1 Table establishments

Contient les établissements scolaires. Chaque utilisateur appartient à un seul établissement.

Colonne	Type	Description
id	UUID	Clé primaire
name	VARCHAR	Nom complet (ex: "ST-MARIE 14000")
code	VARCHAR	Code unique (ex: "stm001", "vh001")
created_at	TIMESTAMP	Date de création

Données existantes :

```
-- Établissements actuels
INSERT INTO establishments (name, code) VALUES
  ('ST-MARIE 14000', 'stm001'),
  ('VICTOR-HUGO 18760', 'vh001');
```

2.2 Table profiles (TABLE PRINCIPALE)

✓ C'est ICI que sont stockés les identifiants de connexion pour TOUS les utilisateurs.

Colonne	Type	Description
id	UUID	Clé primaire
establishment_id	UUID (FK)	→ establishments.id
role	ENUM	'vie-scolaire' | 'professeur' | 'delegue'
username	VARCHAR	IDENTIFIANT DE CONNEXION (unique)
password_hash	VARCHAR	MOT DE PASSE HACHÉ (bcrypt)
first_name	VARCHAR	Prénom
last_name	VARCHAR	Nom de famille
email	VARCHAR	Email (optionnel)
phone	VARCHAR	Téléphone (optionnel)
can_create_subrooms	BOOLEAN	Autorisation créer des sous-salles

2.3 Table students

Extension pour les élèves. Liée à `profiles` via `profile_id`.

Colonne	Type	Description
`id`	UUID	Clé primaire (table students)
`profile_id`	UUID (FK)	→ profiles.id (LIEN PRINCIPAL)
`establishment_id`	UUID (FK)	→ establishments.id
`class_id`	UUID (FK)	→ classes.id
`first_name`	VARCHAR	Prénom (dupliqué depuis profiles)
`last_name`	VARCHAR	Nom (dupliqué depuis profiles)
`role`	ENUM	'delegue' | 'eco-delegue'
`username`	VARCHAR	Identifiant (dupliqué)
`password_hash`	VARCHAR	Mot de passe (dupliqué)

2.4 Table teachers

Extension pour les professeurs. Structure similaire à students.

Colonne	Type	Description
`id`	UUID	Clé primaire
`profile_id`	UUID (FK)	→ profiles.id
`establishment_id`	UUID (FK)	→ establishments.id
`subject`	VARCHAR	Matière enseignée
`classes`	TEXT[]	Tableau des noms de classes
`allow_delegate_subrooms`	BOOLEAN	Autorise les délégués à créer (défaut: true)

2.5 Table classes

Colonne	Type	Description
`id`	UUID	Clé primaire
`name`	VARCHAR	Nom (ex: "6ème A", "3ème B")
`level`	VARCHAR	Niveau (ex: "6ème", "5ème")
`establishment_id`	UUID (FK)	→ establishments.id
`created_by`	UUID (FK)	→ profiles.id (créateur)

3. Où Stocker les Informations ?

3.1 Récapitulatif

Information	Table principale	Exemple
Identifiant	`profiles.username`	"jean.dupont"
Mot de passe	`profiles.password_hash`	Hash bcrypt
Rôle	`profiles.role`	'vie-scolaire', 'professeur', 'delegue'
Prénom / Nom	`profiles.first_name / last_name`	"Jean", "Dupont"
Code établissement	`establishments.code`	"stm001", "vh001"
Lien établissement	`profiles.establishment_id`	UUID de l'établissement
Classe (élève)	`students.class_id`	UUID de la classe
Matière (prof)	`teachers.subject`	"Mathématiques"

ℹ Le code établissement n'est PAS stocké directement avec l'utilisateur. Il est récupéré via la relation profiles.establishment_id → establishments.id

4. Processus de Création d'Utilisateur

4.1 Vue d'ensemble

La création d'un utilisateur suit un processus en deux étapes :

1. **Créer le profil** dans la table `profiles` (avec identifiants)
2. **Créer l'extension** dans `students` ou `teachers` avec le `profile_id`

4.2 Exemple complet : Créer un élève délégué

Objectif : Créer "Jean Dupont", délégué de 6ème A à ST-MARIE

Étape 1 : Récupérer l'ID de l'établissement

```
SELECT id FROM establishments WHERE code = 'stm001';
-- Résultat : [uuid-establishment]
```

Étape 2 : Récupérer l'ID de la classe

```
SELECT id FROM classes
WHERE name = '6ème A'
  AND establishment_id = '[uuid-establishment]';
-- Résultat : [uuid-class]
```

Étape 3 : Créer le profil

```
INSERT INTO profiles (
  establishment_id, role, username, password_hash,
  first_name, last_name, email, can_create_subrooms
) VALUES (
  '[uuid-establishment]',
  'delegue',
  'jean.dupont',
  '$2a$10$...[hash bcrypt du mot de passe]',
  'Jean',
  'Dupont',
  'jean.dupont@email.com',
  false
) RETURNING id;
-- Résultat : [uuid-profile]
```

Étape 4 : Créer l'élève

```
INSERT INTO students (
  profile_id, establishment_id, class_id,
  first_name, last_name, email, role,
  username, password_hash
) VALUES (
  '[uuid-profile]',
  '[uuid-establishment]',
  '[uuid-class]',
  'Jean',
  'Dupont',
  'jean.dupont@email.com',
  'delegue',
  'jean.dupont',
  '$2a$10$...[hash bcrypt]'
);
```

4.3 Exemple : Créer un utilisateur Vie Scolaire

Pour la Vie Scolaire, **seule la table profiles est nécessaire** (pas de table d'extension).

```
INSERT INTO profiles (
  establishment_id,
  role,
  username,
  password_hash,
  first_name,
  last_name,
  email,
  can_create_subrooms
) VALUES (
  (SELECT id FROM establishments WHERE code = 'stm001'),
  'vie-scolaire',
  'marie.martin',
  crypt('MotDePasse123!', gen_salt('bf')),
  'Marie',
  'Martin',
  'marie.martin@stmarie.fr',
  true
);
```

ℹ Les utilisateurs Vie Scolaire n'ont PAS besoin d'entrée dans students ou teachers.

5. Utilisation dans le Code

5.1 Module user-management.ts

Le module `lib/user-management.ts` automatise la création d'utilisateurs.

Fonctionnalités automatiques :

- Génération d'identifiant unique (prénom.nom + suffixe si besoin)
- Génération de mot de passe sécurisé
- Hachage bcrypt automatique
- Création dans `profiles` ET `students/teachers`
- Enregistrement de l'action dans `action_logs`

Exemple d'utilisation :

```
import { createUser } from '@/lib/user-management';

// Créer un élève délégué
const result = await createUser({
  establishmentId: 'uuid-establishment',
  role: 'delegue',
  firstName: 'Jean',
  lastName: 'Dupont',
  email: 'jean.dupont@email.com',
  classId: 'uuid-class'  // requis pour élèves
});

// Résultat
console.log(result.username);  // 'jean.dupont'
console.log(result.password);  // mot de passe généré
```

✓ Vous n'avez PAS besoin de créer manuellement les utilisateurs en SQL ! Utilisez l'interface de gestion qui appelle createUser().

6. Bonnes Pratiques

6.1 Règles de base

✓	Toujours créer le profil AVANT l'extension (students/teachers)
✓	Utiliser le module user-management.ts pour la création via l'interface
✓	Hacher les mots de passe avec bcrypt (jamais en clair)
✓	Conserver le profile_id pour la liaison entre tables
✗	NE PAS stocker le code établissement directement (utiliser la FK)
✗	NE PAS créer de vie-scolaire dans students ou teachers

6.2 Duplication intentionnelle

Certaines colonnes sont volontairement dupliquées entre `profiles` et `students/teachers` :

- `username, password_hash` — Facilite les requêtes d'authentification
- `first_name, last_name` — Évite les JOIN systématiques
- `email, phone` — Accès rapide aux infos de contact

ℹ Cette duplication est un compromis performance vs normalisation. La source de vérité reste la table profiles.

7. Annexe : Requêtes Utiles

Lister tous les utilisateurs d'un établissement

```
SELECT p.username, p.role, p.first_name, p.last_name, e.name as etablissement
FROM profiles p
JOIN establishments e ON p.establishment_id = e.id
WHERE e.code = 'stm001'
ORDER BY p.role, p.last_name;
```

Trouver un élève avec sa classe

```
SELECT s.first_name, s.last_name, s.role, c.name as classe
FROM students s
JOIN classes c ON s.class_id = c.id
WHERE s.username = 'jean.dupont';
```

Vérifier les credentials (connexion)

```
SELECT id, role, first_name, last_name, establishment_id
FROM profiles
WHERE username = 'jean.dupont'
  AND password_hash = crypt('MotDePasse', password_hash);
```

— Fin du guide —

Plan de Classe — v1.0.0 — 08/12/2025