PAPPE mobile – SIO SLAM

PharmaEquip

Application mobile de gestion des accès aux équipements de laboratoire

Emma Zhao - BTS SIO - 2026
Institution Des Chartreux, 38 rue Pierre Dupont - 69004 Lyon

Table des matières

2. Présentation du projet 2
Objectifs 2
Périmètre fonctionnel 2
3. Contraintes du projet 3
3.1 Ergonomie 3
3.2 Environnement de développement 3
3.3 Accès et rôles 3
3.4 Contrainte métier forte 3
4. Architecture technique 4
4.1 Architecture globale 4
4.2 Arborescence du projet 4
4.3 Endpoints de l'API REST 4
5. Base de données 6
5.1 Modèle relationnel 6
5.2 Description des tables 6
5.3 Logique des créneaux horaires 7
5.4 Soft delete et statuts 7
6. Sécurité et authentification 8
6.1 Authentification JWT 8
6.2 Middleware d'autorisation 8
6.3 Déploiement sécurisé 8
7. Réalisation du projet 9
7.1 Page de connexion 9
7.2 Espace employé - Catalogue des équipements 10
7.3 Espace employé - Réservation d'un créneau 11
7.5 Espace employé - Suivi des demandes 12
7.6 Espace technicien - Dashboard 13
7.7 Espace technicien - Gestion des équipements 14
7.8 Espace administrateur - Gestion des demandes 15
7.9 Espace administrateur - Gestion des équipements 16
7.10 Espace administrateur - Gestion des utilisateurs 17
7.11 Notifications in-app 17
7.12 Paramètres du compte 18
8. Versioning et accès a la production 19
9. Conclusion 19

1. Contexte

Le laboratoire Galaxy Swiss Bourdin (GSB) est issu de la fusion entre le géant américain Galaxy, spécialisé dans le secteur des maladies virales, et le conglomérat européen Swiss Bourdin, travaillant sur des médicaments plus conventionnels. Cette entité regroupe plusieurs centaines de collaborateurs répartis sur différents sites en France.

Dans ce contexte industriel pharmaceutique, GSB dispose d'équipements de laboratoire sensibles et coûteux : instruments analytiques (HPLC, spectrophotomètres), matériel de stérilisation, systèmes de centrifugation, cryoconservateurs. L'accès à ces équipements est réglementé : chaque utilisation doit être encadrée par un technicien qualifié et tracée dans le système d'information.

Jusqu'à présent, la gestion des demandes d'accès se faisait manuellement, ce qui rendait difficile le suivi des utilisations, l'assignation des techniciens et la consultation des disponibilités. L'entreprise souhaite moderniser ce processus en mettant à disposition de ses équipes une application mobile accessible depuis leur téléphone professionnel.

2. Présentation du projet

Le projet PharmaEquip consiste à développer une application mobile permettant à l'ensemble des collaborateurs de GSB de consulter les équipements disponibles, d'effectuer des demandes de réservation sur des créneaux horaires, et de recevoir des notifications sur l'état de leurs demandes.

Objectifs

- Permettre aux employés de consulter le catalogue des équipements disponibles
- Permettre la réservation d'équipements sur des créneaux horaires prédéfinis
- Permettre aux techniciens de gérer les équipements dont ils sont responsables
- Permettre aux administrateurs de valider ou refuser les demandes et d'assigner les techniciens
- Tracer toutes les utilisations d'équipements via un historique (access_logs)
- Envoyer des notifications in-app à chaque changement de statut d'une demande

Périmètre fonctionnel

L'application couvre trois rôles distincts :

Rôle	Fonctionnalités
Employé	Consulter les équipements, réserver un créneau, consulter et annuler ses demandes, recevoir des notifications
Technicien	Voir ses équipements assignés, consulter son planning d'interventions, se retirer d'un équipement
Administrateur	Gérer les utilisateurs et équipements, approuver/refuser les demandes, assigner les techniciens, consulter l'historique

3. Contraintes du projet

3.1 Ergonomie

L'application doit être simple d'utilisation et accessible à des collaborateurs non techniques. L'interface adopte un design professionnel avec une palette de couleurs cohérente (bleu primaire #137FEC sur fond clair) et une typographie lisible. Chaque rôle dispose d'une navigation par onglets adaptée à ses besoins spécifiques

3.2 Environnement de développement

L'application est développée avec le framework React Native via Expo (managed workflow), permettant un code source unique déployable sur Android et iOS. Le backend est une API REST développée avec Node.js et Express. La base de données est MariaDB 11, conteneurisée via Docker.

Technologie	Usage
React Native + Expo	Application mobile iOS/Android
Expo Router	Navigation par fichiers avec protection par rôle
Node.js + Express	API REST backend
MariaDB 11 (Docker)	Base de données relationnelle conteneurisée
JWT (jsonwebtoken)	Authentification et autorisation
Zustand	Gestion d'état global (auth, notifications)
Expo SecureStore	Stockage sécurisé du token JWT
EAS Build	Compilation et génération de l'APK

3.3 Accès et rôles

L'application ne propose pas d'inscription autonome. Les comptes sont créés exclusivement par l'administrateur. Trois rôles sont définis dans la base de données : employee, technician, admin. Chaque rôle donne accès à une section différente de l'application, avec une redirection automatique après login basée sur le rôle détecté dans le token JWT.

3.4 Contrainte métier forte

Chaque équipement doit avoir au moins un technicien assigné pour apparaître dans le catalogue employé. Chaque demande de réservation est automatiquement associée au technicien référent de l'équipement. L'administrateur peut réassigner un technicien différent lors de l'approbation d'une demande d'utilisation d'équipement.

4. Architecture technique

4.1 Architecture globale

L'architecture suit un modèle client-serveur classique. L'application mobile communique avec le backend via des requêtes HTTP sécurisées. Le backend expose une API REST et interagit avec la base de données MariaDB via un pool de connexions mysql2.

4.2 Arborescence du projet

Dossier / Fichier	Description
backend/src/controllers/	Traitement des requêtes HTTP, validation des données, appel des modèles
backend/src/models/	Requêtes SQL, logique métier, interactions avec MariaDB
backend/src/routes/	Définition des endpoints REST et association middleware/contrôleur
backend/src/middleware/	Authentification JWT (protectRoute) et contrôle des rôles (authorize)
backend/src/lib/	Pool de connexion MySQL (db.js) et utilitaire JWT (jwt.js)
mobile/src/app/	Pages Expo Router organisées par groupe de rôle : (auth), (employee), (technician), (admin)
mobile/src/components/	Composants réutilisables : Badge, Button, Input, EmptyState, SlotPicker, EquipmentCard, RequestCard
mobile/src/services/	Couche d'appels API : api.js (instance Axios + intercepteurs), auth/equipment/request/notification.service.js
mobile/src/store/	Stores Zustand : authStore.js (session), notificationStore.js (badge et liste)
mobile/src/hooks/	Hooks personnalisés avec pagination : useEquipments.js, useRequests.js
mobile/src/constants/	Design tokens : colors.js, typography.js, layout.js

4.3 Endpoints de l'API REST

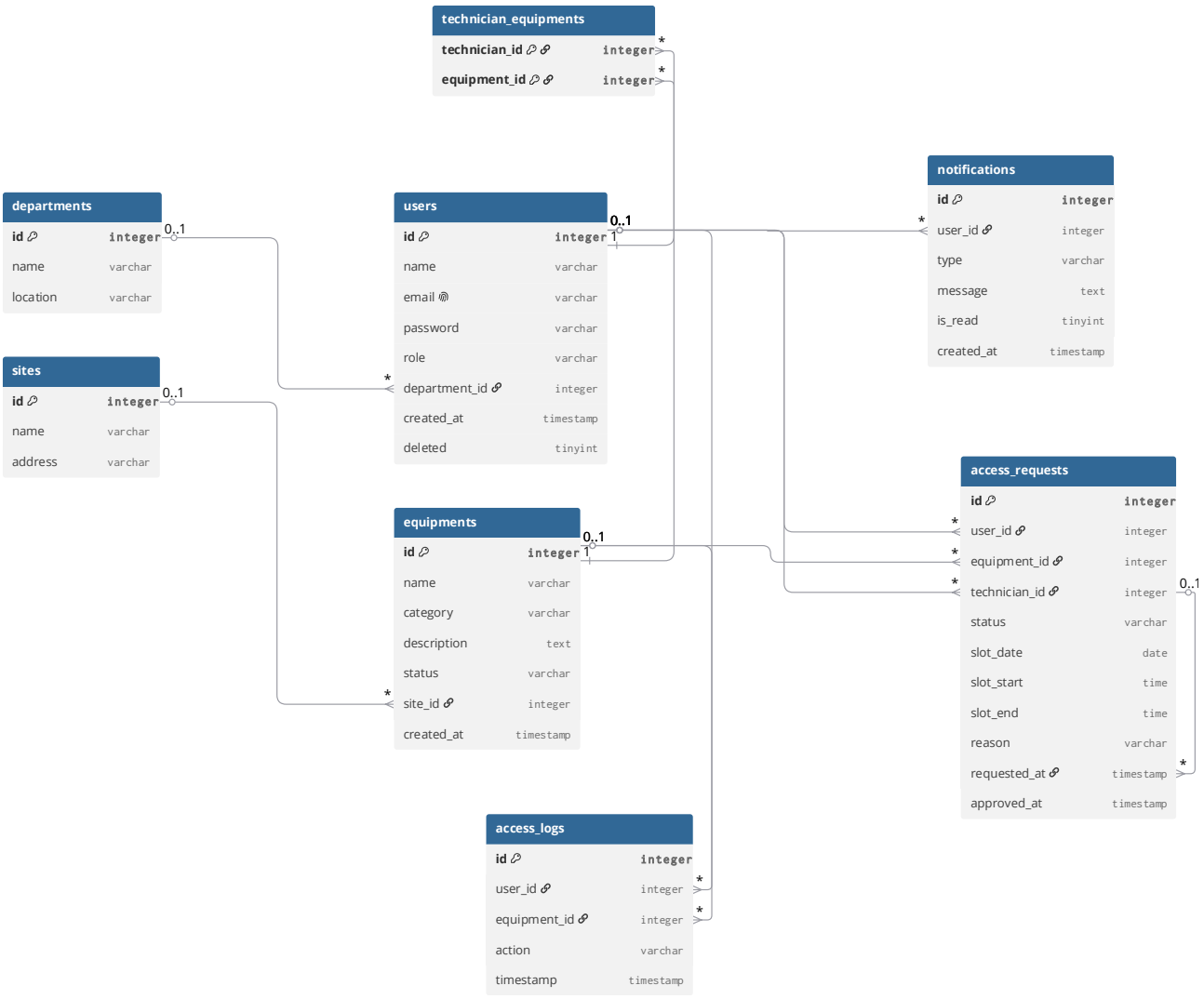
Méthode	Route	Description
POST	/api/auth/login	Authentification, retourne un token JWT
GET	/api/auth/me	Récupère le profil de l'utilisateur connecté
GET/POST	/api/users	Liste et création d'utilisateurs (admin)
PUT/DELETE	/api/users/:id	Modification et suppression (soft delete)
PATCH	/api/users/:id/password	Changement de mot de passe sécurisé
GET	/api/equipments	Liste filtrée selon le rôle connecté
GET	/api/equipments/:id/slots	Créneaux horaires disponibles pour une date
POST/PUT/DELETE	/api/equipments/:id	CRUD équipements (admin)
POST/DELETE	/api/equipments/:id/technicians	Assignation/retrait d'un technicien

GET/POST	/api/access-requests	Liste et création de demandes de réservation
PATCH	/api/access-requests/:id/approve	Approbation d'une demande (admin)
PATCH	/api/access-requests/:id/reject	Refus d'une demande (admin)
PATCH	/api/access-requests/:id/cancel	Annulation d'une demande pending (employé)
GET/PATCH	/api/notifications	Récupération et marquage lu des notifications
GET	/api/equipments/:id/logs	Historique d'utilisation d'un équipement (admin)

5. Base de données

5.1 Modèle relationnel

La base de données MariaDB est composée de huit tables. Le schéma a été conçu pour tracer toutes les interactions avec les équipements tout en gérant les relations entre utilisateurs, équipements, sites et départements.



5.2 Description des tables

Table	Description et colonnes principales
users	id, name, email (unique), password (bcrypt), role (employee/technician/admin), department_id, created_at, deleted (soft delete)
equipments	id, name, category, description, status (available/in_use/unavailable), site_id, created_at
technician_equipments	Table d'association technicien ↔ équipement (clé composite). Un technicien peut gérer plusieurs équipements.
access_requests	id, user_id, equipment_id, technician_id, status (pending/approved/rejected), slot_date, slot_start, slot_end, reason, requested_at, approved_at. Contrainte UNIQUE sur (equipment_id, slot_date, slot_start).

access_logs	Historique horodaté de chaque action sur un équipement : borrowed, returned, maintenance.
notifications	id, user_id, type, message, is_read, created_at. Générées automatiquement lors des approbations/refus.
sites	id, name, address. Localisation physique des équipements.
departments	id, name, location. Service d'appartenance des utilisateurs.

5.3 Logique des créneaux horaires

Les créneaux ne sont pas stockés en base de données. Ils sont générés dynamiquement par le backend à partir d'une liste fixe de huit plages horaires (08h-09h, 09h-10h, 10h-11h, 11h-12h, 13h-14h, 14h-15h, 15h-16h, 16h-17h). Pour une date et un équipement donnés, le backend interroge access_requests pour en déduire les créneaux déjà réservés (status pending ou approved), et retourne uniquement les créneaux libres.

Une vérification supplémentaire est effectuée sur la disponibilité du technicien : si le technicien assigné à l'équipement est déjà occupé sur un autre équipement au même créneau, ce créneau est également bloqué. Si l'équipement possède plusieurs techniciens, le créneau n'est bloqué que si tous les techniciens sont simultanément occupés ailleurs.

La contrainte UNIQUE (equipment_id, slot_date, slot_start) dans access_requests garantit l'unicité au niveau base de données en cas de requêtes simultanées.

5.4 Soft delete et statuts

Les suppressions sont logiques et non physiques : les utilisateurs sont marqués deleted = 1, les équipements passent en status = unavailable. Cela permet d’avoir de l'intégrité référentielle et l'historique des access_logs. Un utilisateur supprimé ne peut plus se connecter (filtrage WHERE deleted = 0 dans le middleware d'authentification).

6. Sécurité et authentification

6.1 Authentification JWT

L'authentification repose sur des JSON Web Tokens (JWT) signés avec une clé secrète stockée dans les variables d'environnement. À chaque login, le serveur génère un token avec une durée de validité de 7 jours. Ce token est transmis dans le header Authorization : Bearer <token> à chaque requête.

6.2 Middleware d'autorisation

Deux middlewares sont appliqués aux routes protégées :

- protectRoute : vérifie la présence et la validité du token JWT, récupère l'utilisateur en base (en excluant les supprimés), et attache req.user à la requête
- authorize(...roles) : vérifie que req.user.role appartient à la liste de rôles autorisés pour la route concernée. Retourne 403 si non autorisé.

Côté frontend, le fichier _layout.jsx racine d'Expo Router joue le rôle de guard : au démarrage de l'application, il appelle GET /api/auth/me pour restaurer la session depuis le token stocké. En cas d'échec (token expiré ou absent), l'utilisateur est redirigé vers la page de connexion. Après un login réussi, la redirection est automatique vers le dashboard correspondant au rôle.

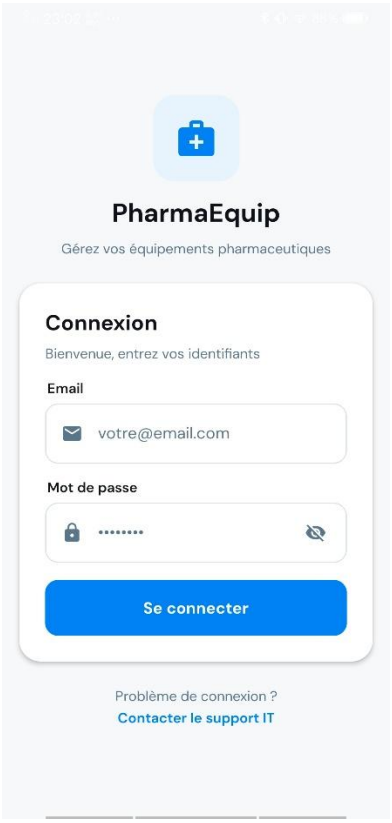
6.3 Déploiement sécurisé

En production, l'API est exposée derrière Cloudflare. Les variables sensibles (JWT_SECRET, credentials BDD) sont injectées via des variables d'environnement dans Docker Compose et ne sont jamais versionnées dans le dépôt Git. Le fichier .env est listé dans .gitignore à tous les niveaux du projet.

7. Réalisation du projet

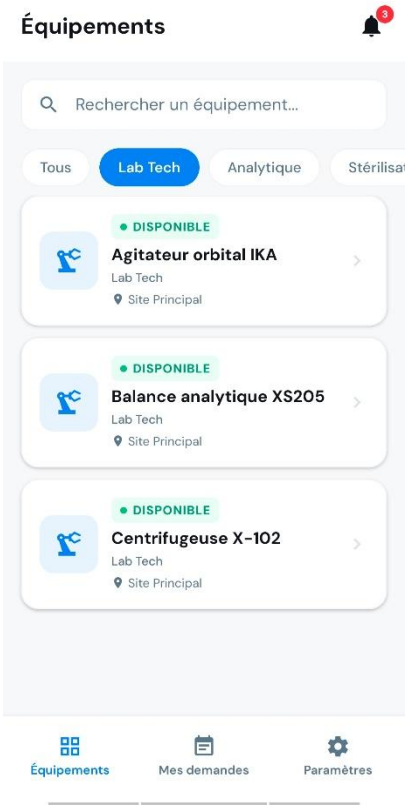
7.1 Page de connexion

La page de connexion est commune à tous les rôles. Après validation des identifiants, le token JWT est stocké et l'utilisateur est redirigé automatiquement selon son rôle. Une validation locale vérifie que les champs sont remplis avant l'appel API.



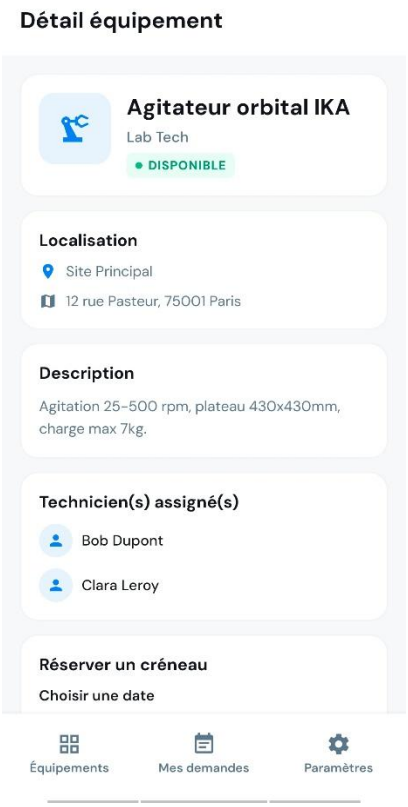
7.2 Espace employé - Catalogue des équipements

L'employé accède à la liste des équipements disponibles. Seuls les équipements ayant au moins un technicien assigné et dont le statut n'est pas unavailable sont affichés. Un champ de recherche et des filtres par catégorie permettent de retrouver rapidement un équipement. La liste supporte le pull-to-refresh et la pagination infinie.



7.3 Espace employé - Réservation d'un créneau

En cliquant sur un équipement, l'employé accède à sa fiche détaillée : description, localisation, techniciens assignés et calendrier de réservation. Le composant SlotPicker interroge l'endpoint GET /api/equipments/:id/slots?date= pour afficher les créneaux libres du jour sélectionné. Les créneaux déjà pris (status pending ou approved) ne sont pas retournés et donc pas affichés. L'employé sélectionne un créneau, saisit optionnellement un motif, puis soumet la demande.

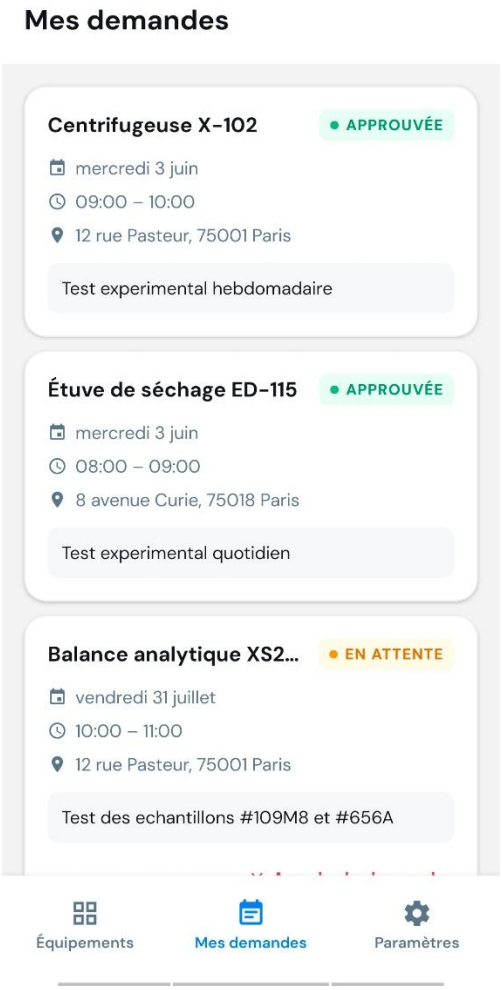


7.4 - Réservation d'un créneau

En cliquant sur un équipement, l'employé accède à sa fiche détaillée : description, localisation, techniciens assignés et calendrier de réservation. Le composant SlotPicker intègre un calendrier complet via la librairie react-native-calendars. Les weekends et les dates passées sont désactivés visuellement et non sélectionnables. En sélectionnant un jour, le composant interroge l'endpoint GET /api/equipments/:id/slots?date= pour afficher les créneaux libres. Les créneaux déjà pris ou dont le technicien est indisponible ne sont pas retournés. L'employé sélectionne un créneau, saisit optionnellement un motif, puis soumet la demande.

7.5 Espace employé - Suivi des demandes

L'employé consulte l'historique de toutes ses demandes, triées par date décroissante. Chaque carte affiche le statut (en attente, approuvée, refusée), la date et le créneau. Les demandes en statut pending peuvent être annulées directement depuis cette page, déclenchant une alerte de confirmation.



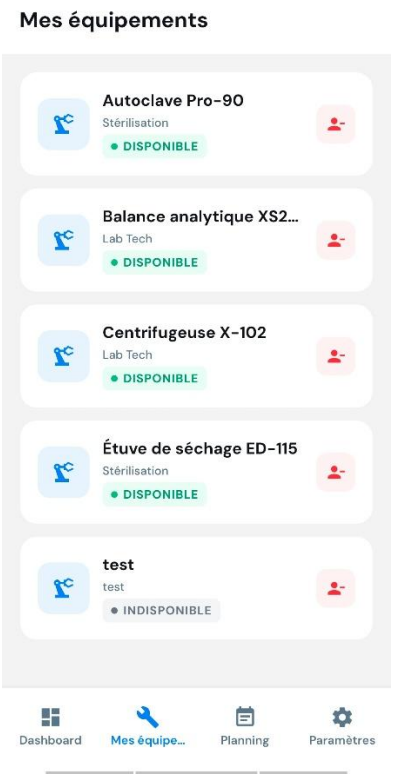
7.6 Espace technicien - Dashboard

Le technicien dispose d'un dashboard affichant en temps réel le nombre de ses équipements et d'interventions à venir. Les prochaines interventions (demandes approved avec slot_date >= aujourd'hui) sont listées avec leur équipement, leur horaire et le nom du demandeur.



7.7 Espace technicien - Gestion des équipements

Le technicien consulte la liste des équipements dont il est responsable. Il peut se retirer d'un équipement en cas d'indisponibilité ou de changement d'affectation, après confirmation par une alerte.



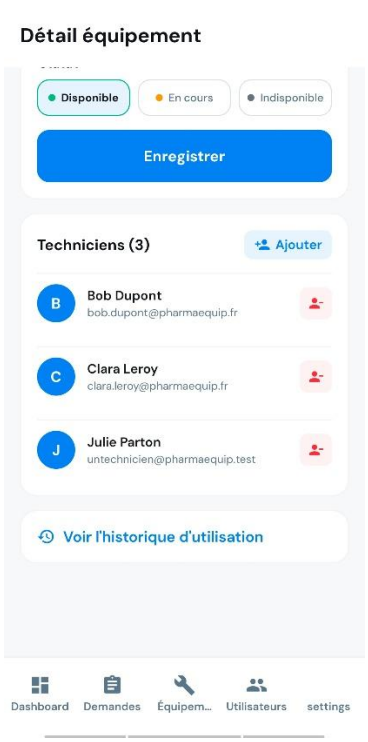
7.8 Espace administrateur - Gestion des demandes

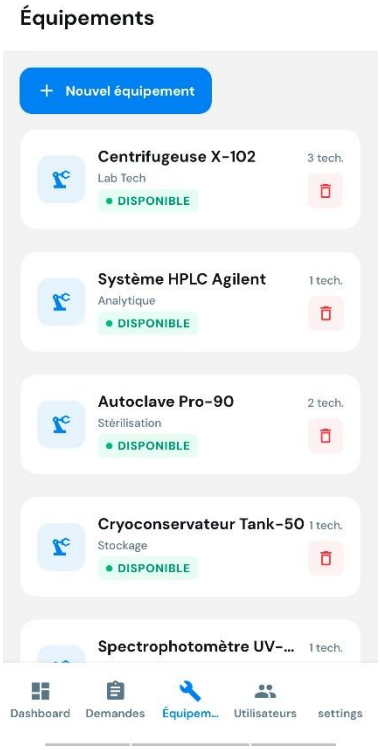
L'administrateur accède à toutes les demandes avec un système de filtres (Toutes, En attente, Approuvées, Refusées). En cliquant sur une demande en statut pending, un modal de détail s'ouvre avec les informations complètes. L'admin sélectionne le technicien à assigner via un Picker (parmi les techniciens de l'équipement) puis peut approuver ou refuser. Chaque action déclenche une notification in-app vers l'employé concerné.



7.9 Espace administrateur - Gestion des équipements

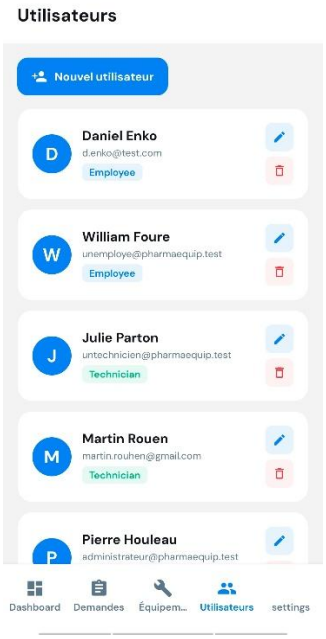
L'administrateur dispose d'un CRUD complet sur les équipements : création avec sélection du site via liste déroulante, modification du statut et des informations, désactivation (soft delete). Sur la fiche d'un équipement, il peut assigner ou retirer des techniciens via un modal de recherche, et consulter l'historique des accès (access_logs) via un modal dédié.





7.10 Espace administrateur - Gestion des utilisateurs

L'administrateur crée, modifie et supprime (soft delete) les utilisateurs. Le formulaire de création/édition intègre un sélecteur de rôle (employee/technician/admin) et un Picker pour le département. La création définit automatiquement le mot de passe transmis à l'utilisateur.



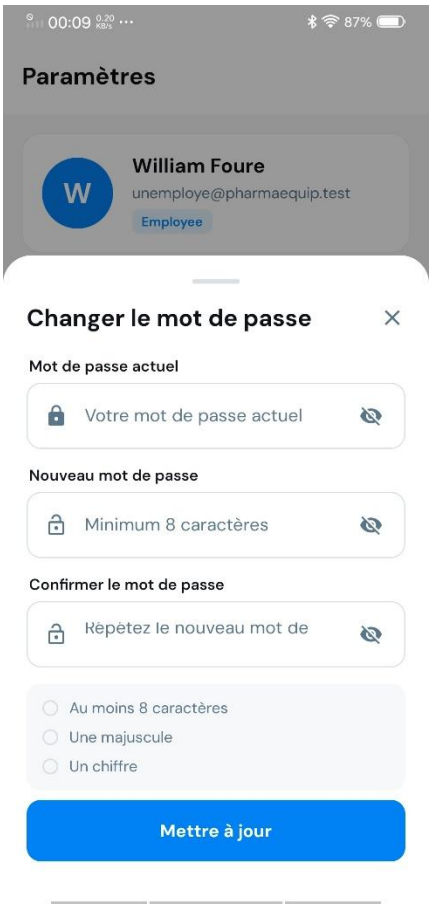
7.11 Notifications in-app

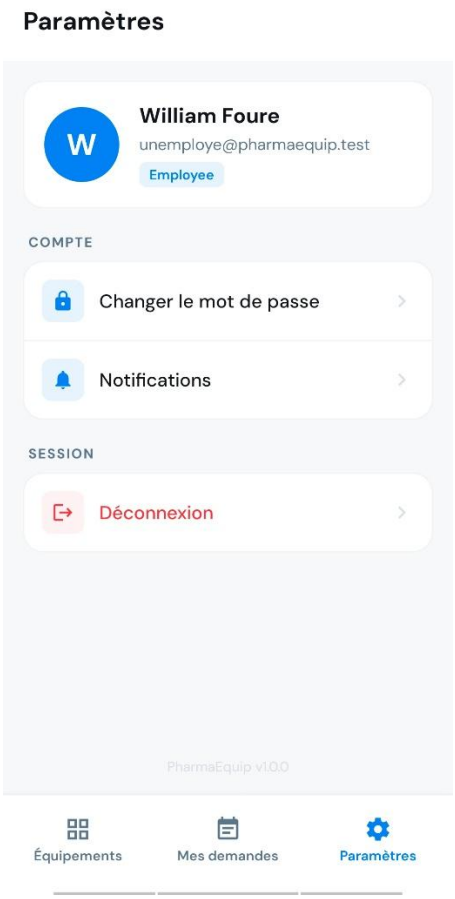
Un badge rouge sur l'icône de notification indique le nombre de notifications non lues. La page de notifications liste toutes les notifications de l'utilisateur connecté, avec une icône et une couleur distinctes selon le type (approuvée = vert, refusée = rouge, nouvelle demande = bleu). L'utilisateur peut marquer les notifications comme lues individuellement ou toutes d'un coup.



7.12 Paramètres du compte

Tous les rôles ont accès à une page de paramètres affichant leurs informations de profil. Le changement de mot de passe se fait via un modal sécurisé qui vérifie l'ancien mot de passe avant mise à jour. Des indicateurs visuels affichent en temps réel la conformité du nouveau mot de passe (8 caractères minimum, majuscule, chiffre).





8. Versioning et accès a la production

Le projet a été versionné avec Git. Le dépôt est hébergé sur Gitlab et accessible à l'adresse suivante :

https://gitlab.com/sco-chartreux/slam2-25-26/zhao-emma/pharmaequipement

Le dépôt est organisé avec deux dossiers principaux (backend/ et mobile/) ainsi qu'un docker-compose.yml à la racine.

La production est déployée sur une VM Ubuntu Server 24.04 hébergée sur un ESXi de l'établissement. Le backend est conteneurisé avec Docker Compose (services mysql et backend). L'APK Android est distribué via EAS Build (Expo Application Services).

Les liens pour télecharger l'apk de l'application :

- version développement, utilisé avec tunnel expo : https://expo.dev/accounts/em186/projects/mobile/builds/0795ee40-76ba-4148-bffe-c23549dd53fc

- version production : https://expo.dev/accounts/em186/projects/mobile/builds/2cfa4013-f685-4135-908f-b0bb4543c0bc

Logins :

- admin : administrateur@pharmaequip.test, pharmaEquip2026!

- employe : unemploye@pharmaequip.test, pharmaEquip2026!

- technicien : untechnicien@pharmaequip.test, pharmaEquip2026!

9. Conclusion

Ce projet m'a permis de développer une application mobile full stack complète en autonomie, de la conception de la base de données jusqu'au déploiement en production. J'ai notamment approfondi mes compétences en architecture REST (middlewares, autorisation par rôle, gestion des transactions SQL) et en navigation conditionnelle avec Expo Router.

Les principales difficultés rencontrées ont été la gestion de la cohérence des statuts d'équipements en fonction des créneaux actifs, la mise en place du port forwarding entre WSL2 et l'émulateur Android lors des tests, ainsi que la configuration du déploiement Docker sur le réseau de l'établissement.

Des axes d'amélioration sont identifiés pour une prochaine version : la mise en place d'un cron job pour la synchronisation automatique des statuts, l'ajout de notifications push via Expo Notifications, et le développement d'une interface web d'administration complémentaire à l'application mobile.