
Plateforme Web de Réservation de Matériel Pédagogique (Data Show)

Réalisé par :
Zeineb Mkaouar

Sommaire

Table des matières

Chapitre 1 : Cadre Technologique	4
I. Introduction	4
II. Stack MERN Moderne.....	4
1. Backend — NestJS 11.....	4
2. Frontend — React 19 & Vite	4
3. Base de données — MongoDB Atlas.....	4
III. Choix Technologiques Secondaires.....	4
IV. Conclusion.....	5
Chapitre 2 : Analyse des Besoins	6
I. Introduction	6
II. Acteurs du Système	6
III. Besoins Fonctionnels	6
IV. Besoins Non Fonctionnels	7
V. Diagramme de Cas d'Utilisation	7
1. Présentation des Acteurs.....	7
2. Présentation Graphique	7
3. Description des Cas d'Utilisation	8
VI. Conclusion.....	8
Chapitre 3 : Conception	9
I. Introduction	9
II. Architecture du Système	9
III. Diagramme de Classes	9
IV. Diagramme de Séquence	10
V. Conclusion.....	11
Chapitre 4 : Réalisation.....	12
I. Introduction	12
II. Environnements de Développement.....	12
III. Fonctionnalités Avancées.....	12
1. Notifications en Temps Réel	12
2. Internationalisation & Thème	12
3. Automatisation par Cron Jobs	12
4. Export de Données CSV.....	12
IV. Interfaces de l'Application	13
1. Interface Professeur	13
2. Interface Administrateur	14
V. Conclusion.....	15

Introduction Générale

Dans le contexte de la transformation numérique des institutions éducatives, la gestion optimale des ressources pédagogiques constitue un enjeu stratégique croissant. Les équipements tels que les projecteurs numériques (Data Shows) représentent des ressources partagées, limitées en nombre, et soumises à des contraintes d'utilisation strictes : disponibilité, état de maintenance et absence de conflits d'horaires.

La problématique centrale de ce projet est la suivante : comment concevoir un système informatisé capable de gérer de manière rigoureuse, transparente et automatisée la réservation de matériel pédagogique au sein d'un établissement d'enseignement supérieur ?

Ce mini-projet propose la conception et le développement d'une plateforme web full-stack répondant à cette problématique. Elle offre deux niveaux d'accès distincts : un espace professeur permettant la gestion de l'emploi du temps et la réservation de Data Shows, et un espace administrateur dédié à la supervision globale de l'inventaire, des réparations et du calendrier de réservation.

Le présent rapport est structuré en quatre chapitres couvrant successivement le cadre technologique adopté, l'analyse des besoins fonctionnels, la conception du système, et enfin la réalisation et les fonctionnalités avancées implémentées.

Chapitre 1 : Cadre Technologique

I. Introduction

Le choix d'une architecture technique adaptée est déterminant pour assurer la performance, la maintenabilité et l'évolutivité d'une application web moderne. Pour ce projet, nous avons opté pour une variante enrichie de la stack MERN, en sélectionnant pour chaque couche la technologie la mieux adaptée aux exigences fonctionnelles identifiées.

II. Stack MERN Moderne

L'architecture retenue repose sur quatre piliers technologiques principaux, chacun justifié par des critères techniques et fonctionnels précis.

1. Backend — NestJS 11

NestJS a été choisi pour son architecture modulaire inspirée d'Angular, facilitant la maintenabilité et l'injection de dépendances. Il permet une gestion robuste des APIs REST grâce aux DTOs (Data Transfer Objects) couplés à class-validator, et aux Guards pour la protection des routes sensibles selon le rôle de l'utilisateur.

2. Frontend — React 19 & Vite

L'utilisation de la dernière version de React combinée au bundler Vite assure des performances de rendu optimales et une expérience utilisateur fluide. React 19 introduit de nouveaux hooks et patterns qui réduisent la verbosité du code et améliorent la gestion des états asynchrones.

3. Base de données — MongoDB Atlas

Le choix d'une base NoSQL orientée documents permet une flexibilité sur les schémas, notamment pour la gestion de l'emploi du temps complexe des professeurs. L'utilisation de MongoDB Atlas (Cloud) a été déterminante pour le travail collaboratif en binôme, permettant un accès synchronisé et sécurisé aux données depuis n'importe quel environnement.

III. Choix Technologiques Secondaires

Technologie	Rôle	Justification
Tailwind CSS 4	Styling	Moteur CSS-first atomique, personnalisation via variables CSS, support natif du mode sombre.
i18next / react-i18next	Internationalisation	Bascule FR/EN dynamique sans rechargement, namespaces de traduction par module.
JWT + Passport.js	Authentification	Communication stateless client/serveur, sécurisation des routes par rôle (ADMIN, PROF).

@nestjsjs/schedule	Automatisation	Cron jobs pour l'ouverture automatique des semaines (règle J-7).
Mongoose	ODM	Modélisation et validation des données côté serveur.

IV. Conclusion

L'ensemble de ces choix technologiques forment une architecture cohérente, moderne et évolutive. Chaque composant a été sélectionné non pas par effet de mode, mais en réponse directe aux contraintes fonctionnelles identifiées lors de l'analyse des besoins. Le chapitre suivant présente ces besoins en détail.

Chapitre 2 : Analyse des Besoins

I. Introduction

Cette phase d'analyse constitue le fondement de toute démarche de développement logiciel rigoureuse. Elle permet d'identifier avec précision les acteurs impliqués, leurs besoins fonctionnels et les contraintes non fonctionnelles à respecter pour garantir la qualité du système livré.

II. Acteurs du Système

Le système distingue deux acteurs principaux, chacun disposant de droits et de responsabilités bien définis :

Professeur	Administrateur
- Gérer son emploi du temps - Réserver un Data Show - Consulter et annuler ses réservations - Soumettre des réclamations de pannes - Recevoir des notifications	- Gérer l'inventaire (Data Shows, salles) - Enregistrer et suivre les réparations - Ouvrir/fermer les semaines - Consulter toutes les réservations - Répondre aux réclamations - Exporter les données en CSV

III. Besoins Fonctionnels

Le tableau ci-dessous présente l'ensemble des fonctionnalités attendues du système, organisées par identifiant :

#	Fonctionnalité	Description
F01	Authentification	Connexion sécurisée par JWT selon le rôle (Professeur / Administrateur). Accès conditionné par le profil.
F02	Emploi du temps	Saisie des séances S1 à S6 avec horaires et salles. Pré-remplissage automatique lors de la réservation.
F03	Réservation	Choix date, séance, salle et Data Show. Réservation hors emploi du temps (rattrapage) permise.
F04	Vérifications	Blocage : semaine fermée, DS en panne, double réservation. Avertissement : salle déjà équipée.
F05	Réclamations	Signalement de panne par le professeur. Consultation et réponse par l'administrateur.
F06	Réparations	Enregistrement des réparations avec historique complet (date, description, action).
F07	Gestion des semaines	Ouverture manuelle ou automatique (J-7). Fermeture possible à tout moment.
F08	Notifications	Génération automatique lors de mise en panne d'un DS réservé. Affichage en temps réel.
F09	Export CSV	Extraction des historiques de réservations et réparations au format CSV.

IV. Besoins Non Fonctionnels

- **Sécurité : authentification JWT, Guards NestJS, validation des entrées via DTOs.**
- Performance : rendu optimisé React 19 + Vite ; requêtes MongoDB indexées.
- Disponibilité : hébergement MongoDB Atlas avec haute disponibilité cloud.
- Maintenabilité : architecture modulaire NestJS, séparation Controller / Service.
- Accessibilité multilingue : interface traduite FR/EN via react-i18next.
- Ergonomie : pré-remplissage automatique, mode sombre/clair, retours visuels clairs.

V. Diagramme de Cas d'Utilisation

1. Présentation des Acteurs

Le système implique deux acteurs distincts : le Professeur, utilisateur final des fonctionnalités de réservation et de suivi, et l'Administrateur, responsable de la supervision et de la configuration globale du système.

2. Présentation Graphique

Le diagramme ci-dessous représente les interactions entre les acteurs et les cas d'utilisation du système :

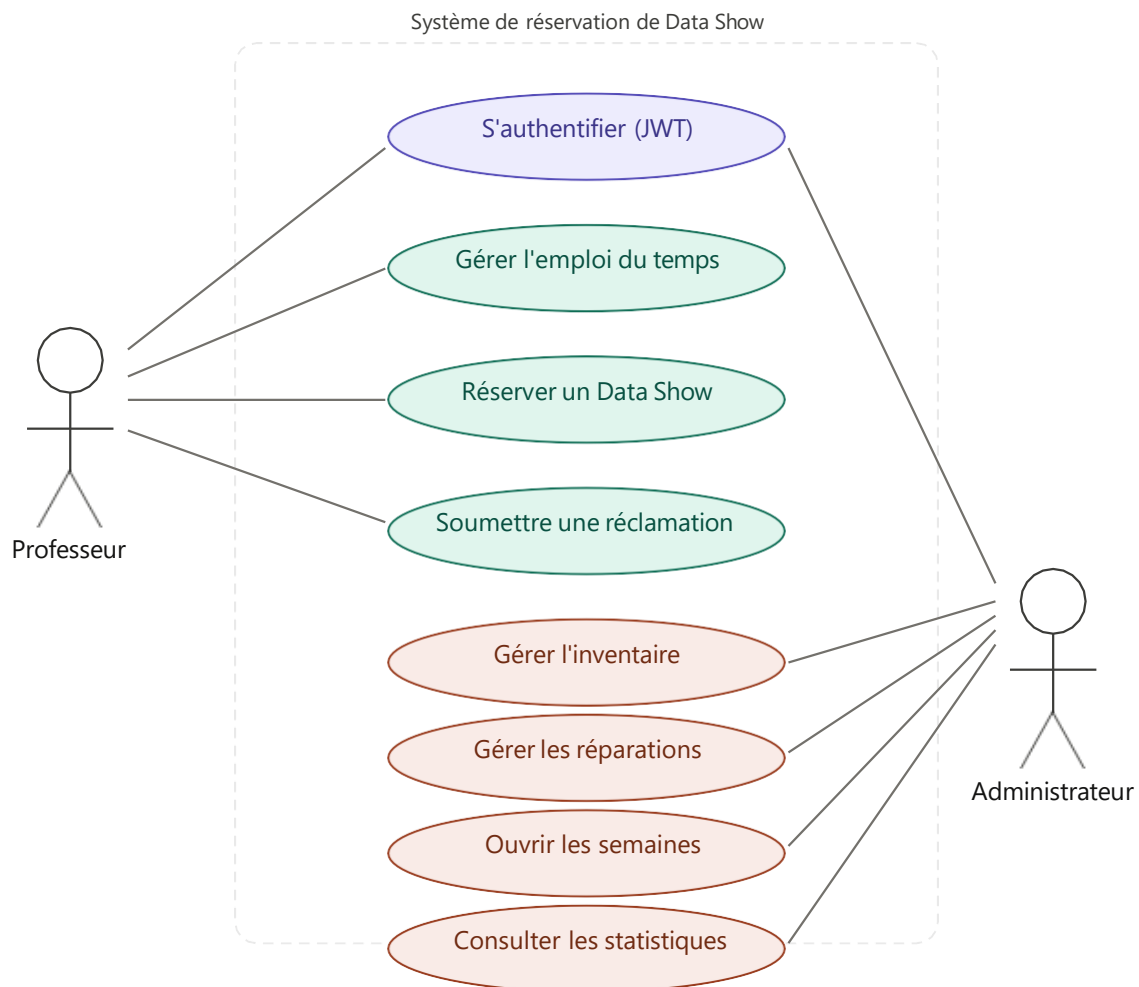


Figure 1 — Diagramme de cas d'utilisation du système de réservation

3. Description des Cas d'Utilisation

Chaque cas d'utilisation correspond à une fonctionnalité métier identifiée lors de l'analyse. L'authentification (UC1) est commune aux deux acteurs. Les cas UC2 à UC4 sont propres au Professeur, tandis que UC5 à UC8 relèvent exclusivement de l'Administrateur.

VI. Conclusion

L'analyse des besoins a permis de délimiter précisément le périmètre fonctionnel du système et d'identifier les contraintes techniques à respecter. Ces éléments servent de base solide à la phase de conception présentée dans le chapitre suivant.

Chapitre 3 : Conception

I. Introduction

La phase de conception traduit les besoins spécifiés en modèles formels servant de référence lors de l'implémentation. Elle couvre l'architecture globale, la modélisation des données et la logique applicative centrale du système.

II. Architecture du Système

L'application repose sur une architecture Client-Serveur à trois couches communiquant via une API REST sécurisée :

- **Couche Présentation (Frontend) : application React 19 + Vite, consommant l'API via Axios.**
- **Couche Métier (Backend) : serveur NestJS exposant des endpoints REST par module fonctionnel.**
- **Couche Données : MongoDB Atlas stockant les documents selon les schémas Mongoose définis.**

La sécurité est assurée à deux niveaux : les Guards NestJS filtrent les requêtes non authentifiées, tandis que les DTOs couplés à class-validator assurent la validation stricte des données en entrée.

III. Diagramme de Classes

Le modèle de données s'articule autour de cinq entités principales : User, DataShow, Reservation, Repair et Notification. Le diagramme suivant illustre leurs attributs et leurs relations :

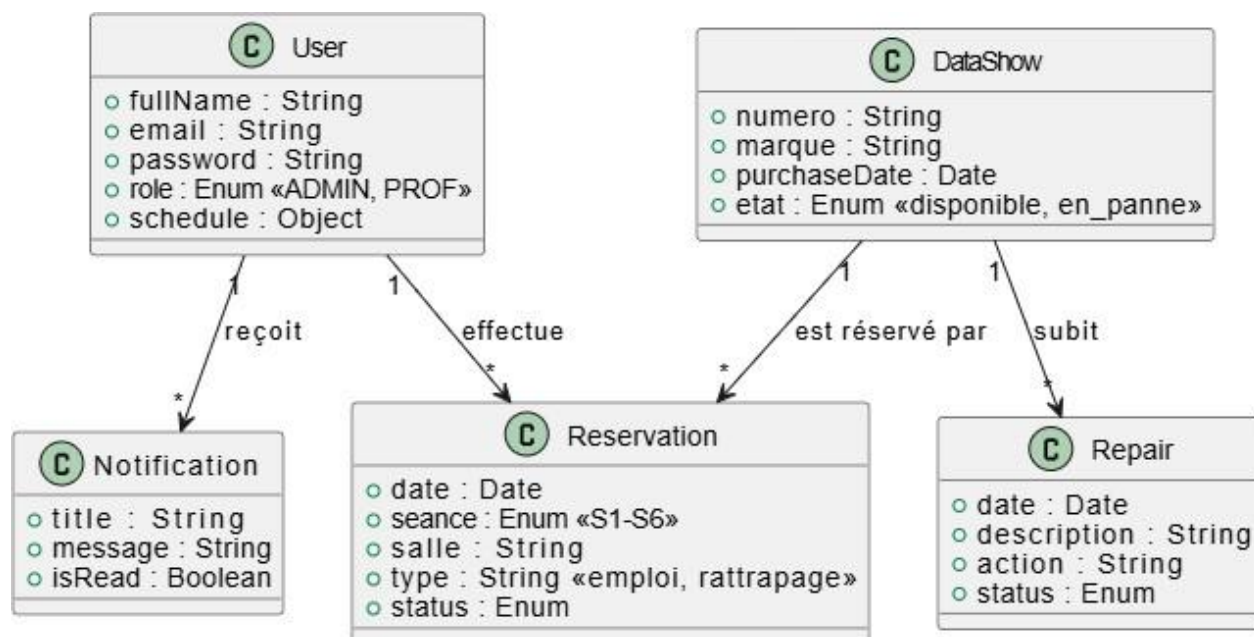


Figure 2 — Diagramme de classes (schémas Mongoose)

Les relations clés sont : un User effectue plusieurs Reservations et reçoit plusieurs Notifications ; un DataShow est lié à plusieurs Reservations et subit plusieurs Repairs.

IV. Diagramme de Séquence

Ce diagramme illustre la robustesse du système lors d'une tentative de réservation. Trois vérifications séquentielles sont effectuées avant tout enregistrement :

1. La semaine cible doit être ouverte par l'administrateur — sinon : `ForbiddenException` (HTTP 403).
2. Le Data Show sélectionné doit être disponible — sinon : `BadRequestException` (HTTP 400).
3. Aucun conflit de séance (DS ou Professeur) — sinon : `ConflictException` (HTTP 409).

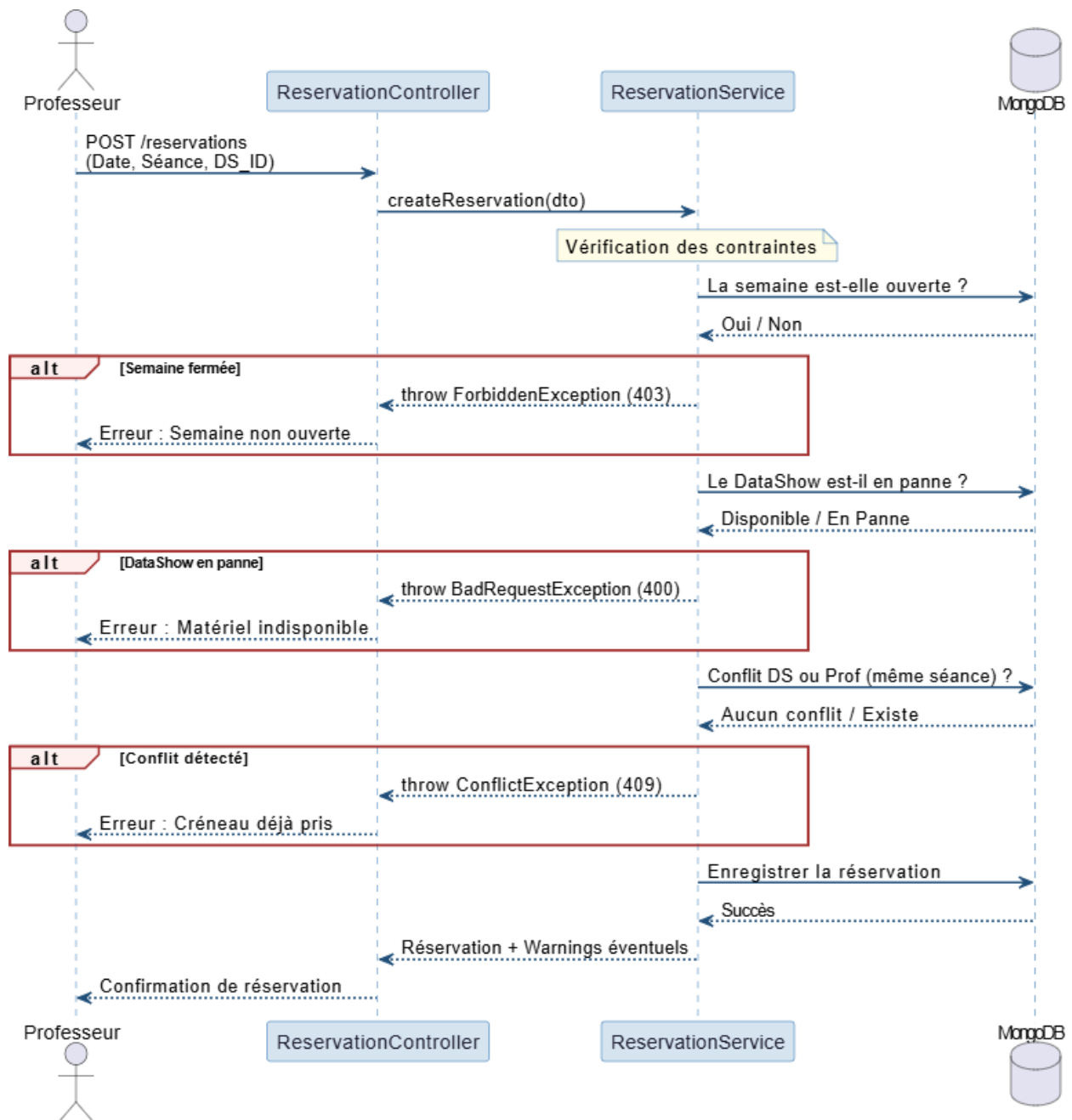


Figure 3 — Diagramme de séquence : logique de réservation

V. Conclusion

La phase de conception a permis de formaliser l'ensemble des choix architecturaux et de modélisation. Les diagrammes produits constituent la référence technique utilisée tout au long du développement. Le chapitre suivant présente la réalisation concrète du système.

Chapitre 4 : Réalisation

I. Introduction

Ce chapitre présente la réalisation concrète du système, en couvrant successivement l'environnement de développement, les fonctionnalités avancées implémentées, et les principales interfaces de l'application.

II. Environnements de Développement

Le développement a été réalisé en binôme dans l'environnement suivant :

- Éditeur : Visual Studio Code avec extensions ESLint, Prettier et NestJS Snippets.
- Gestion de version : Git / GitHub avec branches feature séparées par fonctionnalité.
- Base de données : MongoDB Atlas (Cloud) pour un accès collaboratif synchronisé.
- Tests API : Postman pour la validation des endpoints REST.
- Déploiement local : Node.js 20 LTS, npm, ports 3000 (frontend) et 3001 (backend).

III. Fonctionnalités Avancées

1. Notifications en Temps Réel

Lorsqu'un administrateur déclare un Data Show « en panne », le système identifie automatiquement toutes les réservations futures liées à cet appareil et génère des notifications personnalisées pour chaque professeur concerné. Le frontend interroge régulièrement l'API (polling) pour mettre à jour le badge de notification en temps réel.

2. Internationalisation & Thème

L'intégralité de l'interface est traduite en français et en anglais via react-i18next. La bascule s'effectue dynamiquement sans rechargement de page. Un système de thème Clair / Sombre est implémenté via les variables CSS de Tailwind 4, avec mémorisation de la préférence utilisateur dans le stockage local.

3. Automatisation par Cron Jobs

Conformément au cahier des charges, une tâche planifiée s'exécute chaque lundi à minuit via @nestjs/schedule pour ouvrir automatiquement les réservations de la semaine suivante (règle J-7). Cette automatisation supprime toute intervention administrative manuelle hebdomadaire.

4. Export de Données CSV

Un module d'exportation permet à l'administrateur de télécharger en un clic l'historique des réservations et des réparations au format CSV, directement exploitable dans Excel ou Google Sheets pour le reporting périodique.

IV. Interfaces de l'Application

Les captures d'écran suivantes illustrent les principales vues de l'application dans les deux rôles disponibles.

1. Interface Professeur

L'interface professeur comprend le tableau de bord, la gestion de l'emploi du temps, le formulaire de réservation avec pré-remplissage automatique, et le suivi des réservations en cours.

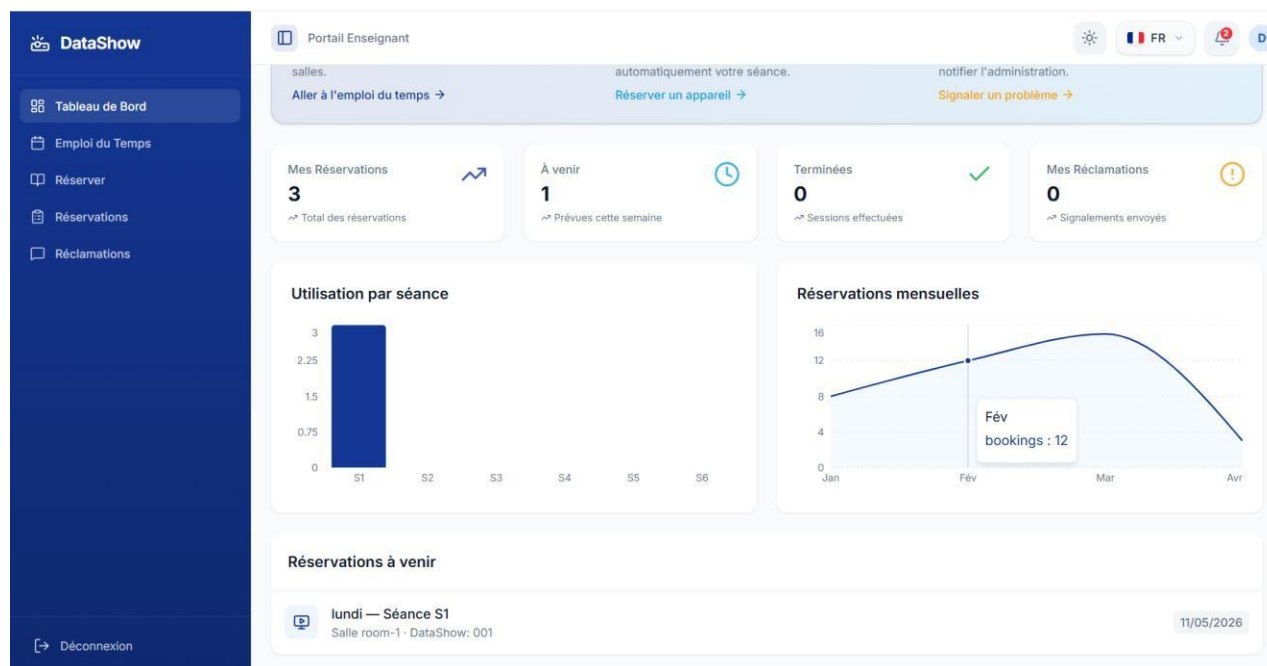


Figure 4 — Tableau de bord Professeur (vue réservations)

The screenshot shows the 'Portail Enseignant' reservation form. The sidebar is identical to the previous figure. The main content area has a top section titled 'Comment ça marche ?' (How it works?) with three bullet points explaining the reservation process. Below this are two tabs: 'Réservation Rapide' (selected) and 'Manuel (Rattrapage)' (Manual Catch-up). The 'Réservation Rapide' form has two steps: '1. Choisir la Date' (Choose the Date) with a date picker set to '11/05/2026' and a note 'Semaine du 11/05/2026 est OUVERTE' (Week of 11/05/2026 is OPEN), and '2. Choisir la Séance' (Choose the Session) with a dropdown menu showing 'Séance S1 (Salle room-1)'. A green message box states 'Séance détectée automatiquement ! Nous avons trouvé votre cours en Salle room-1 pour la Séance S1.' (Session detected automatically! We found your course in Salle room-1 for Session S1.). Step 3, 'Choisir l'Appareil' (Choose the Device), shows '0 disponibles' (0 available) and a dropdown with 'Aucun datashow disponible pour ce créneau' (No datashow available for this slot). A 'Confirmer la Réservation' (Confirm Reservation) button is at the bottom.

Figure 5 — Formulaire de réservation avec pré-remplissage automatique

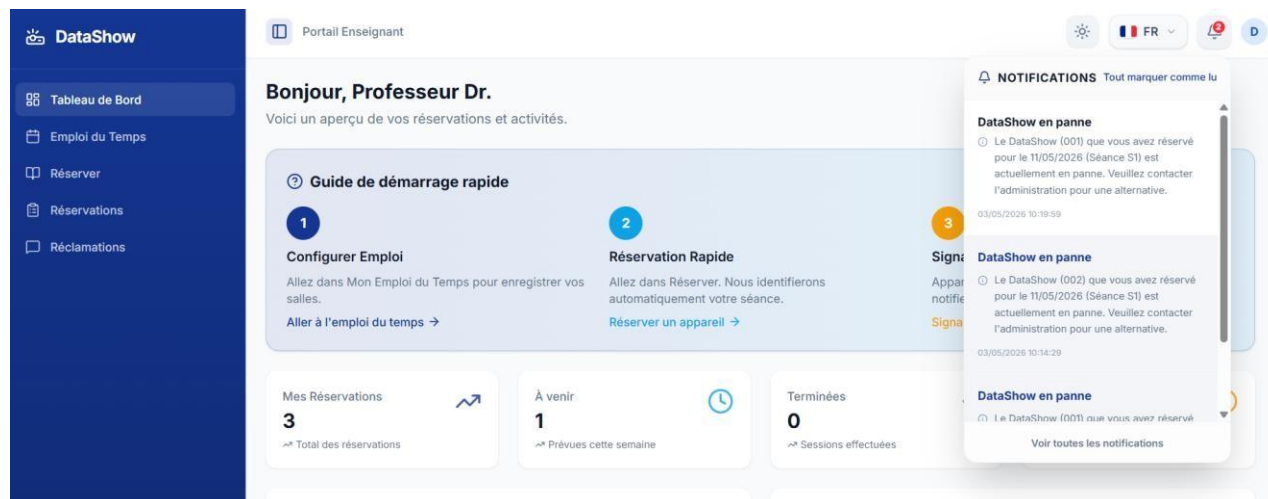


Figure 6 — Centre de notifications (alerte Data Show en panne)

2. Interface Administrateur

L'interface administrateur donne accès à la gestion de l'inventaire, au calendrier des semaines ouvertes, au tableau de bord des statistiques et à la gestion des réclamations.

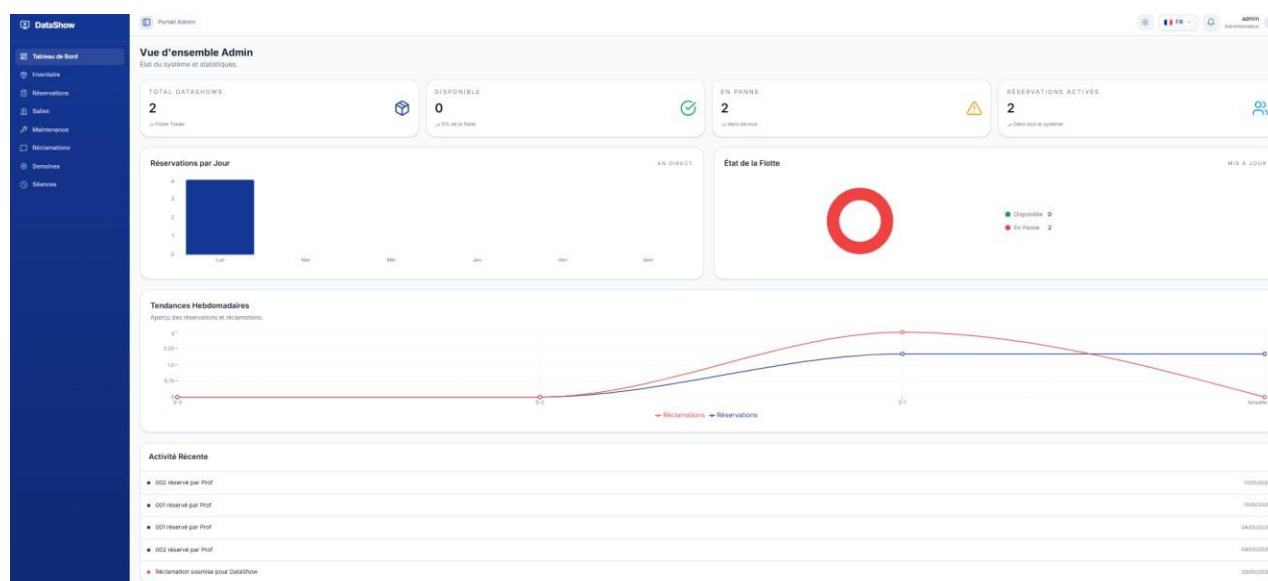


Figure 7 — Tableau de bord Administrateur (statistiques et graphiques)

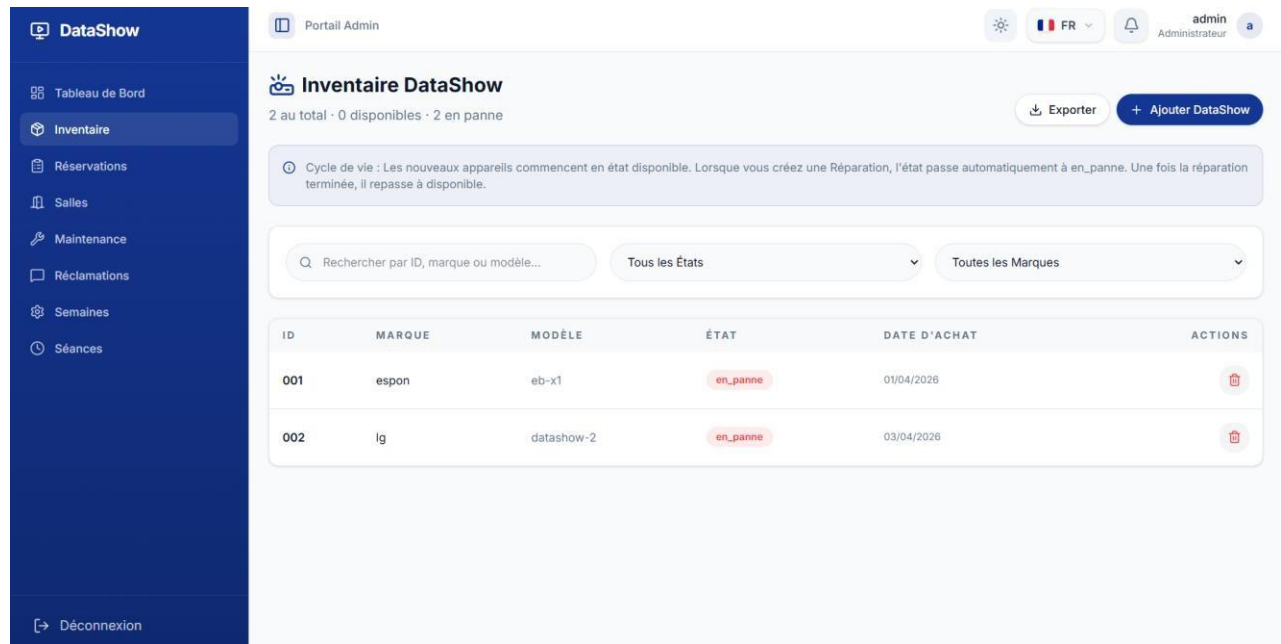


Figure 8 — Gestion de l'inventaire Data Show

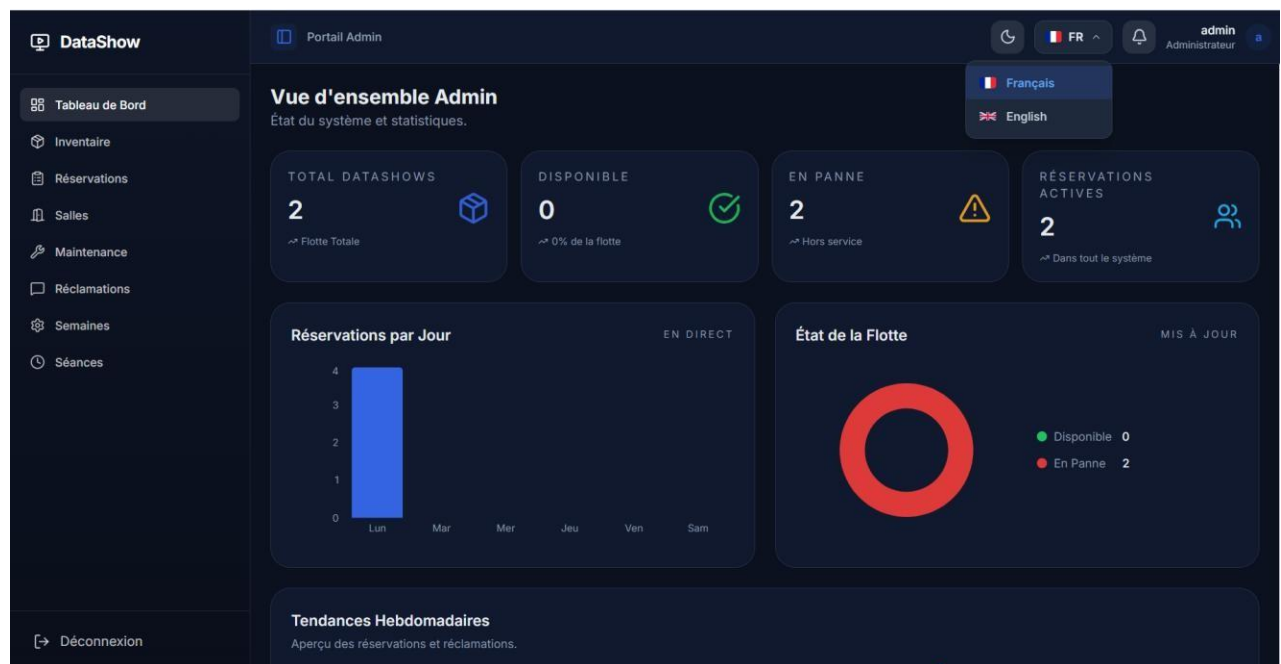


Figure 9 — Mode sombre (dark mode) — même vue en thème sombre

V. Conclusion

La réalisation du système a démontré la cohérence de l'architecture choisie : le backend NestJS a fourni une structure rigoureuse et extensible, tandis que React 19 a permis de livrer une interface moderne et réactive. Les fonctionnalités avancées telles que les notifications automatiques, l'i18n et les Cron Jobs témoignent d'une maîtrise solide des outils du développement full-stack contemporain.

Conclusion Générale

Ce mini-projet a permis de concevoir et de développer une plateforme web complète répondant à une problématique réelle de gestion de ressources pédagogiques partagées. La solution couvre l'intégralité du cycle de vie d'une réservation : de la demande du professeur jusqu'au suivi administratif post-utilisation.

Sur le plan technique, l'adoption de la stack NestJS / React 19 / MongoDB Atlas a démontré sa pertinence. L'architecture modulaire facilite la maintenabilité, tandis que le frontend réactif offre une expérience fluide et moderne. Les fonctionnalités avancées — notifications automatiques, internationalisation dynamique et Cron Jobs — attestent d'une maîtrise solide du développement full-stack.

Le système est aujourd'hui opérationnel, sécurisé et prêt à être déployé. Plusieurs évolutions peuvent être envisagées à court terme :

Évolution	Description
WebSockets	Remplacement du polling par des notifications push instantanées via Socket.io.
Application Mobile	Application cross-platform via React Native partageant la même API REST.
Analytics avancés	Enrichissement des statistiques avec graphiques Recharts par séance et salle.
Multi-établissements	Extension de l'architecture pour plusieurs campus ou départements.

En définitive, ce projet représente une expérience formatrice complète, alliant rigueur d'analyse, maîtrise des technologies modernes et sens de l'ingénierie logicielle. Il constitue une base solide et extensible pour tout développement futur.