

Cahier des charges – Système de réservation en ligne

1. Contexte & objectif

Le client souhaite disposer d'une **plateforme de réservation en ligne sécurisée**, accessible via navigateur, permettant à des utilisateurs authentifiés de **réserver, modifier et annuler des créneaux** pour des ressources définies.

La plateforme devra proposer une **expérience utilisateur complète**, incluant **des notifications email automatiques** à chaque étape clé du cycle de réservation.

2. Typologie des utilisateurs

Utilisateur standard

- Créer un compte et se connecter
- Consulter les ressources disponibles
- Créer, modifier et annuler ses réservations
- Consulter l'historique de ses réservations
- **Recevoir des notifications email liées à ses réservations**

Administrateur

- Tous les droits utilisateur
- Gérer les ressources réservables
- Gérer toutes les réservations
- Accéder à des statistiques globales
- **Recevoir des notifications sur les événements critiques**

3. Authentification & gestion des comptes

Fonctionnalités obligatoires

- Inscription via **magic link**
- Connexion via **magic link**
- Gestion de session sécurisée
- Déconnexion
- Double authentification (2FA)
- Récupération automatique du profil utilisateur
- Suppression de compte utilisateur

Contraintes

- Utilisation d'un **système user-managed** (ex : Clerk)
- Aucun mot de passe géré par l'application
- Protection des routes front-end et back-end
- Gestion explicite des erreurs d'authentification

4. Pages attendues (front-end)

Pages publiques

1. Landing page
2. Page de connexion (magic link)

Pages utilisateur authentifié

3. Dashboard utilisateur
4. Liste des ressources
5. Création de réservation
6. Détail d'une réservation
7. Modification de réservation
8. Historique des réservations
9. Profil utilisateur & sécurité

Pages administrateur

10. Dashboard admin
11. Gestion des ressources
12. Gestion des réservations

Pages système

13. Pages d'erreur (401, 403, 404, 500)
14. Page de chargement global

Total attendu : 12 à 14 pages

5. Fonctionnalités métier détaillées

Réservations

- Créer une réservation
- Modifier une réservation

- Annuler une réservation
- Vérifier la disponibilité en temps réel
- Empêcher les conflits de créneaux
- Gestion des statuts :
 - active
 - modifiée
 - annulée
 - passée

Ressources réservables

- Nom
- Description
- Disponibilité (jours / horaires)
- Activation / désactivation
- Capacité (optionnel)
- Règles spécifiques (durée min/max)

6. Notifications email

Événements déclencheurs

Des emails doivent être envoyés automatiquement lors des événements suivants :

Pour l'utilisateur

- Confirmation de création de réservation
- Modification de réservation

- Annulation de réservation
- Rappel avant la réservation (ex : J-1 ou H-1)
- Suppression de compte

Pour l'administrateur

- Création d'une nouvelle réservation
- Annulation d'une réservation
- Incident critique (optionnel)

Contenu attendu des emails

- Objet explicite
- Informations clés :
 - ressource
 - date et heure
 - statut
- Ton professionnel
- Pas d'informations sensibles
- Lien vers l'application (si pertinent)

Contraintes techniques

- Envoi d'email **côté back-end uniquement**
- Utilisation d'un service d'envoi (SMTP, API email)
- Gestion des erreurs d'envoi :
 - retry

- logs
- L'échec d'envoi d'email **ne doit pas bloquer** la réservation
- Emails testables en environnement de développement

7. API & back-end

Exigences fonctionnelles

- API REST sécurisée ou GraphQL
- Authentification par token
- Gestion des rôles
- Validation serveur
- Logique métier centralisée
- Déclenchement des emails côté API

Endpoints attendus (exemples)

- POST /reservations → crée + email
- PUT /reservations/:id → modifie + email
- DELETE /reservations/:id → annule + email
- GET /reservations/me
- GET /resources
- POST /resources (admin)

Gestion des erreurs API

Chaque endpoint doit retourner :

- code HTTP correct
- message exploitable par le front
- erreur métier distincte de l'erreur technique

8. États applicatifs front-end

Pour chaque action réseau :

- chargement
- succès
- erreur
- données vides

Les notifications visuelles ne remplacent **pas** les emails.

9. Contraintes non fonctionnelles

- Séparation front / back
- Variables d'environnement
- Aucun secret exposé
- Code maintenable
- Architecture documentée
- Application déployable

10. Critères de validation client et évaluation

Le projet est considéré conforme si :

- toutes les fonctionnalités CRUD fonctionnent
- l'authentification est sécurisée
- **les emails sont envoyés sur chaque événement clé**
- les erreurs sont gérées correctement
- l'expérience utilisateur est fluide