CAHIER DE RECETTE

GIA Vehicle Booking System

Auteur	Version	Contexte
Epanti Awoum Franc Junior	**v1.0.0 · Février 2026**	**Test Technique GIA Group**

1. Présentation du Projet

GIA Vehicle Booking System est une application web full stack permettant la réservation de véhicules en ligne. Elle offre une interface moderne pour les clients et un tableau de bord complet pour les administrateurs.

1.1 Fonctionnalités principales

- Inscription et authentification sécurisée avec JWT
- Catalogue de véhicules avec filtres (catégorie, carburant, transmission, prix)
- Réservation en ligne avec vérification des disponibilités — auto-confirmée immédiatement
- Dashboard client : historique, statuts, boutons Payer / Voir Ticket
- Dashboard admin : Réservations, Véhicules CRUD, Utilisateurs, Paiements (filtrable)
- Module de paiement simulé : ticket imprimable et téléchargeable en PDF
- Notifications par email (confirmation, bienvenue, réinitialisation de mot de passe)
- Interface bilingue FR / EN avec changement instantané
- Design responsive (mobile, tablette, desktop)

1.2 Charte graphique GIA Group

Élément	Valeur
Couleur principale	#2A3180— Bleu marine (primary-dark)
Couleur secondaire	#189CD9— Bleu GIA (primary-light)
Typographie	Inter (Google Fonts) — lisible, moderne
Style	Minimaliste, cartes arrondie, ombres douces

2. Architecture Technique

2.1 Vue d’ensemble

Architecture : Monorepo avec séparation frontend / backend

Communication : API REST JSON | Authentification : JWT Bearer Token

Le projet est structuré en deux applications indépendantes au sein d’un même dépôt Git :

- backend/ — API REST Node.js/Express avec TypeScript
- frontend/ — SPA React avec Vite et TypeScript

2.2 Modèle de données (schéma Prisma)

Modèle	Description
User	Compte utilisateur — rôle USER/ADMIN, statut, permissions
Vehicle	Véhicule — catégorie, tarif/jour, statut, équipements
Booking	Réservation — dates, prix total, statut (CONFIRMED dès création)
Payment	Paiement — transactionId, montant, méthode, statut, devise
Notification	Historique des emails envoyés par utilisateur

3. Choix Techniques

3.1 Backend

Technologie	Justification
Node.js + Express	Écosystème JavaScript unifié, performance I/O asynchrone
TypeScript	Typage statique — sécurité accrue, meilleure maintenabilité
Prisma ORM	Migrations typesafe, auto-complétion, requêtes sûres
PostgreSQL	SGBD relationnel robuste — intégrité des données, enums natifs
JWT (jsonwebtoken)	Authentification stateless — scalable, sécurisé
bcryptjs	Hachage des mots de passe — standard industrie (salt rounds : 12)
Nodemailer	Envoi d'emails fiable — templates HTML personnalisés
Zod	Validation des entrées — schémas déclaratifs, messages d'erreur clairs
Winston	Logger structuré (JSON) avec niveaux INFO/WARN/ERROR

3.2 Frontend

Technologie	Justification
React 18 + Vite	DX moderne, HMR rapide, build optimisé
TypeScript	Contrat de types frontend/backend cohérent
Tailwind CSS	Utility-first — design system rapide, sans CSS superflu
React Router v6	Navigation SPA avec routes protégées (ProtectedRoute)
Axios	Client HTTP — intercepteurs pour JWT et gestion d'erreurs
React Toastify	Notifications non-bloquantes — feedback UX immédiat

Context API	Gestion d'état (Auth, i18n) — pas de sur-engineering Redux
jsPDF	Génération PDF ticket de paiement côté client (chargé dynamiquement)

3.3 Sécurité

- Mots de passe hashés avec bcrypt (12 rounds)
- Tokens JWT signés — expiry 7 jours, vérification à chaque requête
- Tokens de réinitialisation : crypto.randomBytes(32) en base64url — TTL 1h
- Validation entrées côté serveur avec Zod (rejet des payloads malformés)
- CORS configuré pour le domaine frontend uniquement
- Rate limiting sur les endpoints d'authentification
- Anti-énumération : même réponse HTTP 200 si email connu ou inconnu
- Aucune donnée sensible de carte stockée — seulement les 4 derniers chiffres

3.4 Performances frontend

- Optimistic updates : les actions admin modifient l'état React immédiatement (≤0 ms de latence percçu)
- Re-fetch silencieux en arrière-plan après chaque mutation
- Images véhicules servies depuis URL externe — aucun stockage local

4. API REST — Endpoints

4.1 Authentification /api/auth

Méthode + Route	Description
POST /register	Inscription — crée compte, envoie email de bienvenue
POST /login	Connexion — retourne JWT + données utilisateur
GET /me	Profil connecté (JWT requis)
POST /forgot-password	Envoi lien de réinitialisation (anti-énumération)
POST /reset-password	Réinitialisation avec token valide (TTL 1h)

4.2 Véhicules /api/vehicles

Méthode + Route	Description
GET /	Liste avec filtres (catégorie, prix, transmission...)
GET /:id	Détails d'un véhicule
POST /	Créer un véhicule (Admin)

PATCH /:id	Modifier un véhicule (Admin)
DELETE /:id	Supprimer un véhicule (Admin)

4.3 Réservations /api/bookings

Méthode + Route	Description
POST /	Créer une réservation — statut CONFIRMED immédiat (disponibilité validée en amont)
GET /my-bookings	Réservations du client connecté
GET / (admin)	Toutes les réservations
PATCH /:id/status	Changer le statut (Admin)
DELETE /:id	Supprimer une réservation (Admin)
GET /stats	Statistiques globales (Admin)

4.4 Utilisateurs /api/users

Méthode + Route	Description
GET /	Liste tous les utilisateurs (Admin)
PATCH /:id/status	Bloquer / débloquer (Admin)
PATCH /:id/role	Promouvoir / rétrograder (Admin)
DELETE /:id	Supprimer un compte (Admin)

4.5 Paiements /api/payments

Méthode + Route	Accès
POST /process	**JWT (propriétaire)**
GET /:bookingId	**JWT (propriétaire ou Admin)**
GET /	**Admin uniquement**

Filtres de GET /api/payments (query params) :

- dateFrom / dateTo : plage de dates ISO 8601 (dateTo étendu à 23 h 59 inclus)
- status : PENDING | COMPLETED | FAILED | REFUNDED
- paymentMethod : CARD | MOBILE_MONEY | CASH

5. Fonctionnalités Détaillées

5.1 Gestion des réservations

- Vérification automatique des conflits de dates avant création (requête Prisma atomique)
- Calcul automatique du nombre de jours et du prix total
- Auto-confirmation : le statut est définitivement CONFIRMED à la création (aucune validation admin requise)
- Envoi de confirmation par email avec détails de la réservation
- L'admin peut modifier n'importe quel statut depuis le dashboard (CONFIRMED → COMPLETED | CANCELLED, etc.)

5.2 Dashboard Administrateur

- Vue synthétique : total véhicules, réservations, revenus, en attente
- Onglet Réservations : actions contextuelles selon le statut (icônes confirm/terminer/annuler/remettre en attente)
- Onglet Véhicules : CRUD complet avec modal (création / modification / vue détaillée)
- Onglet Utilisateurs : bloquer/débloquer, promouvoir admin, supprimer
- Onglet Paiements : tableau filtrable (plage de dates, statut, moyen de paiement) + ruban résumé (total, encaissé, complétés)
- Toutes les actions utilisent des optimistic updates (0 ms de latence visible)

5.3 Module de Paiement Simulé

- Accessible depuis le dashboard client sur toute réservation CONFIRMED non encore payée
- Navigation 3 étapes : Récapitulatif → Formulaire carte → Ticket/Confirmation
- Carte de test refusée : tout numéro se terminant par 0002 → DECLINED (code 402)
- Toute autre carte valide (16 chiffres) → ACCEPTED avec délai de traitement simulé (1.5–2 s)
- Le paiement crée une ligne Payment en base et passe la réservation en COMPLETED
- Aucune donnée sensible stockée — seuls les 4 derniers chiffres (masqués) et le nom du titulaire
- Ticket de paiement généré : n° transaction, client, véhicule, dates, montant, mode, statut PAYÉ
- Impression : window.print() avec styles @media print
- Téléchargement PDF : jsPDF chargé dynamiquement — ticket formaté aux couleurs GIA Group

5.4 Internationalisation (i18n)

- Deux langues : Français (défaut) et Anglais
- Changement instantané sans rechargement de page (React Context)
- ~200 clés de traduction couvrant toute l'interface
- Persistance du choix de langue dans localStorage

6. Tests

Tests fonctionnels manuels effectués sur l'ensemble des parcours utilisateur (voir Cahier de Recette). 67 cas de test — 100 % PASS.

7. Perspectives d'Évolution

- Paiement réel : remplacement du simulateur par Stripe ou Mobile Money (MTN/Orange Cameroun)
- Upload d'images véhicules : stockage Cloudinary ou AWS S3
- Notifications push : WebSockets pour alertes temps réel admin
- Application mobile : React Native avec partage du code métier
- Déploiement cloud : VPS Ubuntu + Nginx + PM2 + certificat SSL Let's Encrypt
- Tests automatisés : Jest (backend) + Cypress E2E (frontend)

8. Résumé de la Stack

Couche	Technologies
Runtime	Node.js 20 LTS
Langage	TypeScript 5 (frontend + backend)
Framework API	Express.js 4
ORM	Prisma 5
Base de données	PostgreSQL 15
Frontend	React 18 + Vite 5
Styles	Tailwind CSS 3
Authentification	JWT (jsonwebtoken) + bcryptjs
Emails	Nodemailer (SMTP)
Validation	Zod
Logger	Winston
Gestion d'état	React Context API
HTTP Client	Axios
Génération PDF	jsPDF 2.5 (chargé dynamiquement côté client)
Serveur local	Laragon (Windows)

— Fin du document technique —