Auto 95 Clean — Technical Deployment Overview

This document describes the architecture, deployment strategy, and payment flow for the Auto 95 Clean booking platform.

1. Technology Stack

- Frontend: React (Vite)
- Backend: NestJS (Node.js)
- Database: PostgreSQL
- Payments: SumUp or Qonto API
- Hosting: Vercel (Frontend), Render (Backend + DB)

2. Monorepo Structure

- /apps/web — React Frontend
- /apps/api — NestJS Backend
- /docs — Project documentation
- /assets — Design and branding assets

3. Deployment Strategy

The frontend is deployed on Vercel from the apps/web directory. The backend API is deployed on Render from the apps/api directory. PostgreSQL is hosted on Render or an external managed provider such as Neon or Supabase.

Required Environment Variables:
- DATABASE_URL
- JWT_SECRET
- PAYMENT_API_KEY
- WEBHOOK_SECRET

4. Payment Flow (Soft-Hold Model)

1. User selects formula, vehicle and time slot.
2. API creates a reservation with status PENDING_PAYMENT and temporarily locks the slot.
3. User completes payment via SumUp or Qonto widget.
4. Payment webhook notifies the API.
5. If successful, reservation becomes CONFIRMED. If failed, it expires and the slot is released.

5. Security & Reliability

- HTTPS enforced for all endpoints
- JWT-based authentication
- Webhook signature verification
- Automatic cleanup of expired pending reservations