Database Architecture

The application uses **two PostgreSQL databases**:

1. **OSAS Database** (Primary)
   - Main operational database
   - Contains waybills, scans, users, partners, etc.
2. **ONMART Database** (Secondary)
   - Marketplace-specific operations
   - Separate schema for marketplace data

Key Database Models

Core Operational Models

Model	Purpose	Key Fields
`scans`	Tracking status history	waybill_id, status, location, timestamp
`waybill_numbers`	Shipment records	awb_number, origin, destination, weight
`partners`	3P courier partners	name, api_token, webhook_url
`resi_3p`	3P waybill mapping	internal_awb, partner_awb, partner_id
`user`	User accounts	username, password_hash, role_id, agent_id

Scanning Models

Model	Purpose
`delivery_scan`	Delivery completion records
`handover_scan`	Hub-to-hub transfers
`receiving_scan`	Package arrivals
`problem_scan`	Exception tracking
`packing_scan`	Package preparation

Financial Models

Model	Purpose

billing_vip_config	VIP customer pricing
xendit_payment	Payment gateway records
xendit_payout	Payout transactions
agent_commission	Commission calculations

Master Data Models

Model	Purpose
agent_locations	Agent location mapping
agent_customer	Customer-agent relationships
awb_books	AWB number allocation
courier_partners	Courier configurations
service_types	Available services

Additional Models (52 total)

- Manifest management (manifest_pcp, manifest_pos)
- Quality management (alter_weight, pod_record, pop_record)
- Trucking operations (trucking_shipment, trucking_config)
- Location master data (all_location_migration)
- Marketplace integration models
- Report configuration models

Database Relationships

```
User ──┐
       ├─→ Agent_Locations → Waybill_Numbers
       └─→ Role

Waybill_Numbers ──┬─→ Scans (1:many)
                  ├─→ Resi_3p
                  └─→ Delivery_Scan / Problem_Scan

Partners ─→ Resi_3p ─→ Waybill_Numbers

Agent_Customer ─→ Waybill_Numbers
```

Connection Pooling

```
// From db.config.js
pool: {
  max: DB_POOL_MAX,        // Maximum connections
  min: DB_POOL_MIN,        // Minimum connections
  acquire: DB_POOL_ACQUIRE, // Connection acquisition timeout
  idle: DB_POOL_IDLE       // Idle timeout
}
```

Revision #1
Created 24 February 2026 06:48:37 by ondeliveloper
Updated 24 February 2026 06:49:44 by ondeliveloper