Backend API & Models Documentation

Table of Contents

Models Overview

User Model

File: `models/User.ts`

Purpose: Manages system users with role-based access control.

Schema Fields: - `email` (String, unique, required) - `password` (String, required, hashed) - `name` (String, required) - `role` (Enum: admin, agent, driver) - `isActive` (Boolean, default: true) - `photoUrl` (String, optional) - `lastLogin` (Date, optional)

Static Methods: - `findByEmail(email)` - Find user by email address - `findByRole(role)` - Find all users with specific role

Instance Methods: - `comparePassword(password)` - Compare plain password with hashed password

Pre-save Hook: Hashes password if modified

Client Model

File: `models/Client.ts`

Purpose: Manages customer information and account balances.

Schema Fields: - `code` (String, unique, auto-generated: CLI-YYYYMMDD-XXXX) - `companyName` (String, optional) - `firstName` (String, required) - `lastName` (String, required) - `email` (String, required) - `phone` (String, required) - `address` (Object: street, city, postalCode, country) - `accountBalance` (Number, default: 0) - `notes` (String, optional) - `isActive` (Boolean, default: true)

Static Methods: - `generateClientCode()` - Generate unique client code - `findByEmail(email)` - Find client by email

Pre-save Hook: Auto-generates client code if not provided

Driver Model

File: `models/Driver.ts`

Purpose: Manages driver profiles, licenses, and availability.

Schema Fields: - `employeeId` (String, unique, auto-generated: DRV-YYYYMMDD-XXXX) - `firstName` (String, required) - `lastName` (String, required) - `email` (String, unique, required) - `phone` (String, required) - `address` (Object: street, city, postalCode, country) - `licenseNumber` (String, required) - `licenseExpiry` (Date, required) - `licenseType` (String, required) - `status` (Enum: available, on_tour, off_duty, on_leave) - `hireDate` (Date, required) - `rating` (Number, optional) - `totalToursCompleted` (Number, optional) - `notes` (String, optional) - `isActive` (Boolean, default: true)

Static Methods: - `generateEmployeeId()` - Generate unique employee ID - `findAvailable()` - Find all available drivers - `findByEmail(email)` - Find driver by email

Pre-save Hook: Auto-generates employee ID if not provided

Vehicle Model

File: `models/Vehicle.ts`

Purpose: Manages fleet vehicles, maintenance, and availability.

Schema Fields: - `registrationNumber` (String, unique, required) - `type` (Enum: van, truck, motorcycle, car) - `brand` (String, required) - `model` (String, required) - `year` (Number, required) - `capacity` (Object: weight in kg, volume in m^3) - `fuelType` (String, required) - `fuelConsumption` (Number, required) - `status` (Enum: available, in_use, maintenance, out_of_service) - `lastMaintenanceDate` (Date, optional) - `nextMaintenanceDate` (Date, optional) - `mileage` (Number, default: 0) - `notes` (String, optional) - `isActive` (Boolean, default: true)

Static Methods: - `findAvailable()` - Find all available vehicles - `findByType(type)` - Find vehicles by type - `findNeedingMaintenance()` - Find vehicles needing maintenance

---

Shipment Model

File: `models/Shipment.ts`

Purpose: Manages individual shipment records and tracking.

Schema Fields: - `shipmentNumber` (String, unique, auto-generated: SHP-YYYYMMDD-XXXX) - `client` (Reference to Client) - `serviceType`

(Reference to ServiceType) - `destination` (Reference to Destination) - `senderName`, `senderPhone`, `senderAddress` (Sender details) - `receiverName`, `receiverPhone`, `receiverAddress` (Receiver details) - `packages` (Array of package details: description, weight, volume, quantity) - `totalWeight` (Number, calculated) - `totalVolume` (Number, calculated) - `priceBreakdown` (Object: baseAmount, weightAmount, volumeAmount, additionalFees, discount) - `totalAmount` (Number, required) - `status` (Enum: pending, picked_up, in_transit, at_sorting_center, out_for_delivery, delivered, failed_delivery, returned, cancelled) - `trackingHistory` (Array of tracking entries) - `deliveryTour` (Reference to DeliveryTour, optional) - `invoice` (Reference to Invoice, optional) - `pickupDate`, `estimatedDeliveryDate`, `actualDeliveryDate` (Date fields) - `notes` (String, optional) - `isInvoiced` (Boolean, default: false)

Static Methods: - `generateShipmentNumber()` - Generate unique shipment number - `findByClient(clientId)` - Find shipments for a client - `findPendingInvoice(clientId)` - Find shipments ready for invoicing

Pre-save Hook: - Auto-generates shipment number - Calculates total weight and volume - Adds initial tracking entry

Indexes: - client, status, createdAt, isInvoiced, deliveryTour

---

DeliveryTour Model

File: `models/DeliveryTour.ts`

Purpose: Manages delivery routes and tour execution.

Schema Fields: - `tourNumber` (String, unique, auto-generated: TR-YYYYMMDD-XXXX) - `date` (Date, required) - `driver` (Reference to Driver, required) - `vehicle` (Reference to Vehicle, required) - `shipments` (Array of references to Shipment) - `status` (Enum: planned, in_progress, completed, cancelled) - `plannedRoute` (Object: startLocation, endLocation, estimatedDistance, estimatedDuration) - `actualRoute` (Object: startTime, endTime, actualDistance, actualDuration, fuelConsumed) - `deliveriesCompleted` (Number, default: 0) - `deliveriesFailed` (Number, default: 0) - `incidents` (Array of references to Incident) - `notes` (String, optional)

Static Methods: - `generateTourNumber()` - Generate unique tour number - `findByDriver(driverId)` - Find tours by driver - `findByDate(date)` - Find tours on specific date

Pre-save Hook: Auto-generates tour number if not provided

Indexes: - driver, date, status

Invoice Model

File: `models/Invoice.ts`

Purpose: Manages billing and invoicing for clients.

Schema Fields: - `invoiceNumber` (String, unique, auto-generated: INV-YYYYMMDD-XXXX) - `client` (Reference to Client, required) - `shipments` (Array of references to Shipment) - `amountHT` (Number, before tax) - `tvaRate` (Number, default: 19%) - `tvaAmount` (Number, calculated) - `totalTTC` (Number, after tax) - `amountPaid` (Number, default: 0) - `amountDue` (Number, calculated) - `status` (Enum: draft, pending, partially_paid, paid, overdue, cancelled) - `issueDate` (Date, required) - `dueDate` (Date, required) - `notes` (String, optional)

Static Methods: - `generateInvoiceNumber()` - Generate unique invoice number - `findByClient(clientId)` - Find invoices for a client - `findOverdue()` - Find overdue invoices

Pre-save Hook: Auto-generates invoice number and calculates amounts

Indexes: - client, status, issueDate, dueDate

Payment Model

File: `models/Payment.ts`

Purpose: Tracks payments received for invoices.

Schema Fields: - `paymentNumber` (String, unique, auto-generated: PAY-YYYYMMDD-XXXX) - `invoice` (Reference to Invoice, required) - `client` (Reference to Client, required) - `amount` (Number, required) - `paymentMethod` (Enum: cash, bank_transfer, check, card) - `paymentDate` (Date, required) - `reference` (String, optional - check number, transfer reference) - `notes` (String, optional)

Static Methods: - `generatePaymentNumber()` - Generate unique payment number - `findByInvoice(invoiceId)` - Find payments for an invoice - `findByClient(clientId)` - Find payments from a client

Pre-save Hook: Auto-generates payment number

Post-save Hook: Updates invoice status and amounts

Incident Model

File: `models/Incident.ts`

Purpose: Tracks incidents during deliveries (accidents, delays, damages).

Schema Fields: - `incidentNumber` (String, unique, auto-generated: INC-YYYYMMDD-XXXX) - `type` (Enum: delay, loss, damage, technical_issue, accident, other) - `shipment` (Reference to Shipment, optional) - `deliveryTour` (Reference to DeliveryTour, optional) - `vehicle` (Reference to Vehicle, optional) - `driver` (Reference to Driver, optional) - `description` (String, required) - `location` (String, optional) - `occurredAt` (Date, required) - `documents` (Array of URLs - Cloudinary) - `photos` (Array of URLs - Cloudinary) - `status` (Enum: reported, under_investigation, resolved, closed) - `resolution` (String, optional) - `resolvedAt` (Date, optional) - `resolvedBy` (Reference to User, optional) - `reportedBy` (Reference to User, required)

Static Methods: - `generateIncidentNumber()` - Generate unique incident number - `findByTour(tourId)` - Find incidents for a tour - `findByDriver(driverId)` - Find incidents involving a driver

Pre-save Hook: Auto-generates incident number

Validation: At least one of shipment, deliveryTour, vehicle, or driver must be provided

---

Complaint Model

File: `models/Complaint.ts`

Purpose: Manages customer complaints and feedback.

Schema Fields: - `complaintNumber` (String, unique, auto-generated: CMP-YYYYMMDD-XXXX) - `client` (Reference to Client, optional) - `deliveryTour` (Reference to DeliveryTour, optional) - `shipments` (Array of references to Shipment, optional) - `invoice` (Reference to Invoice, optional) - `nature` (Enum: delay, damage, loss, billing, service_quality, driver_behavior, other) - `description` (String, required) - `status` (Enum: pending, in_progress, resolved, cancelled) - `priority` (Enum: low, medium, high, urgent) - `resolution` (String, optional) - `resolvedAt` (Date, optional) - `resolvedBy` (Reference to User, optional) - `attachments` (Array of URLs - Cloudinary) - `assignedTo` (Reference to User, optional)

Static Methods: - `generateComplaintNumber()` - Generate unique complaint number - `findByClient(clientId)` - Find complaints from a client - `findPending()` - Find pending complaints

Pre-save Hook: Auto-generates complaint number

Validation: At least client or deliveryTour must be provided

API Routes

Authentication Routes

POST `/api/auth/signup` **Purpose:** Register a new user account

Request Body:

```
{
  "name": "string",
  "email": "string",
  "password": "string",
  "confirmPassword": "string",
  "role": "admin|agent|driver"
}
```

Response: User object (without password)

POST `/api/auth/signin` **Purpose:** Authenticate user and create session

Request Body:

```
{
  "email": "string",
  "password": "string"
}
```

Response: Session token

Client Routes

GET `/api/clients` **Purpose:** List all clients with pagination and filtering

Query Parameters: - `page` (number, default: 1) - `limit` (number, default: 10) - `status` (active|inactive) - `search` (string)

Response: Array of clients with pagination metadata

POST `/api/clients` **Purpose:** Create a new client

Request Body:

```
{
  "companyName": "string (optional)",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "phone": "string",
  "address": {
    "street": "string",
```

```
    "city": "string",
    "postalCode": "string",
    "country": "string"
  },
  "notes": "string (optional)"
}
```

Response: Created client object

GET `/api/clients/[id]` **Purpose:** Get a specific client by ID

Response: Client object with populated relationships

PUT `/api/clients/[id]` **Purpose:** Update a client

Request Body: Partial client object

Response: Updated client

DELETE `/api/clients/[id]` **Purpose:** Delete a client (soft delete)

Response: Success message

Driver Routes

GET `/api/drivers` **Purpose:** List all drivers with filtering

Query Parameters: - `status` (available|on_tour|off_duty|on_leave) - `isActive` (boolean)

Response: Array of drivers

POST `/api/drivers` **Purpose:** Create a new driver

Request Body:

```
{
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "phone": "string",
  "address": { ... },
  "licenseNumber": "string",
  "licenseExpiry": "date",
  "licenseType": "string",
  "hireDate": "date"
}
```

Response: Created driver

GET `/api/drivers/[id]` **Purpose:** Get specific driver details

PUT `/api/drivers/[id]` **Purpose:** Update driver information

DELETE `/api/drivers/[id]` **Purpose:** Deactivate driver

GET `/api/driver/tours` **Purpose:** Get tours assigned to current logged-in driver

Authentication: Required (driver role)

Response: Array of tours with populated shipments

Vehicle Routes

GET `/api/vehicles` **Purpose:** List all vehicles

Query Parameters: - `status` (available|in_use|maintenance|out_of_service) - `type` (van|truck|motorcycle|car)

POST `/api/vehicles` **Purpose:** Add a new vehicle to fleet

GET `/api/vehicles/[id]` **Purpose:** Get specific vehicle details

PUT `/api/vehicles/[id]` **Purpose:** Update vehicle information

DELETE `/api/vehicles/[id]` **Purpose:** Remove vehicle from fleet

Shipment Routes

GET `/api/shipments` **Purpose:** List all shipments with filtering

Query Parameters: - `status` (shipment status) - `client` (client ID) - `dateFrom`, `dateTo` (date range)

POST `/api/shipments` **Purpose:** Create a new shipment

Request Body:

```
{
  "client": "objectId",
  "serviceType": "objectId",
  "destination": "objectId",
  "senderName": "string",
  "senderPhone": "string",
```

```
  "senderAddress": { ... },
  "receiverName": "string",
  "receiverPhone": "string",
  "receiverAddress": { ... },
  "packages": [
    {
      "description": "string",
      "weight": "number",
      "volume": "number",
      "quantity": "number"
    }
  ]
}
```

Response: Created shipment with auto-calculated pricing

GET `/api/shipments/[id]` **Purpose:** Get shipment details with full tracking history

PUT `/api/shipments/[id]` **Purpose:** Update shipment information

PATCH `/api/shipments/[id]/status` **Purpose:** Update shipment status and add tracking entry

Request Body:

```
{
  "status": "shipment_status",
  "notes": "string (optional)",
  "receiverName": "string (optional)",
  "proofOfDelivery": {
    "photos": ["url1", "url2"],
    "signature": "string",
    "timestamp": "date"
  }
}
```

Tour Routes

GET `/api/tours` **Purpose:** List all delivery tours

Query Parameters: - `status` (planned|in_progress|completed|cancelled) - `driver` (driver ID) - `vehicle` (vehicle ID) - `dateFrom`, `dateTo`

POST `/api/tours` **Purpose:** Create a new delivery tour

Request Body:

```
{
  "date": "date",
  "driver": "objectId",
  "vehicle": "objectId",
  "shipments": ["objectId1", "objectId2"],
  "plannedRoute": {
    "startLocation": "string",
    "endLocation": "string",
    "estimatedDistance": "number",
    "estimatedDuration": "number"
  }
}
```

Auto-actions: - Updates driver status to ON_TOUR - Updates vehicle status to IN_USE - Assigns shipments to tour

GET `/api/tours/[id]` **Purpose:** Get tour details with populated driver, vehicle, and shipments

PUT `/api/tours/[id]` **Purpose:** Update tour (only for PLANNED status)

DELETE `/api/tours/[id]` **Purpose:** Cancel/delete tour and restore driver/vehicle status

PATCH `/api/tours/[id]/start` **Purpose:** Mark tour as IN_PROGRESS

Auto-actions: - Updates tour status to IN_PROGRESS - Updates driver status to ON_TOUR - Updates vehicle status to IN_USE - Records actual-Route.startTime

PATCH `/api/tours/[id]/complete` **Purpose:** Complete a tour

Request Body:

```
{
  "actualRoute": {
    "startTime": "date",
    "endTime": "date",
    "actualDistance": "number",
    "actualDuration": "number",
    "fuelConsumed": "number"
  },
  "deliveriesCompleted": "number",
  "deliveriesFailed": "number",
```

```
  "notes": "string (optional)"
}
```

Auto-actions: - Updates tour status to COMPLETED - Restores driver status to AVAILABLE - Restores vehicle status to AVAILABLE - Updates vehicle mileage

PATCH `/api/tours/[id]/reorder` **Purpose:** Reorder shipments in a tour

Request Body:

```
{
  "shipmentIds": ["id1", "id2", "id3"]
}
```

Invoice Routes

POST `/api/invoices/generate` **Purpose:** Generate invoice for client shipments

Request Body:

```
{
  "clientId": "objectId",
  "shipmentIds": ["id1", "id2"],
  "dueInDays": 30,
  "notes": "string (optional)"
}
```

Auto-actions: - Creates invoice - Calculates totals with TVA - Marks shipments as invoiced

GET `/api/invoices` **Purpose:** List all invoices

GET `/api/invoices/[id]` **Purpose:** Get invoice details

PUT `/api/invoices/[id]` **Purpose:** Update invoice

Payment Routes

POST `/api/payments` **Purpose:** Record a payment for an invoice

Request Body:

```
{
  "invoice": "objectId",
  "amount": "number",
```

```
  "paymentMethod": "cash|bank_transfer|check|card",
  "paymentDate": "date",
  "reference": "string (optional)"
}
```

Auto-actions: - Updates invoice amountPaid and amountDue - Updates invoice status if fully paid

---

Incident Routes

POST `/api/incidents` **Purpose:** Report a new incident

Request Body:

```
{
  "type": "delay|loss|damage|technical_issue|accident|other",
  "deliveryTour": "objectId (optional)",
  "vehicle": "objectId (optional)",
  "description": "string",
  "location": "string (optional)",
  "occurredAt": "date",
  "photos": ["url1", "url2"]
}
```

GET `/api/incidents` **Purpose:** List all incidents

GET `/api/incidents/[id]` **Purpose:** Get incident details

PUT `/api/incidents/[id]` **Purpose:** Update incident or resolve

Complaint Routes

POST `/api/complaints` **Purpose:** Submit a new complaint

Request Body:

```
{
  "client": "objectId (optional)",
  "deliveryTour": "objectId (optional)",
  "nature": "delay|damage|loss|billing|service_quality|driver_behavior|other",
  "description": "string",
  "priority": "low|medium|high|urgent",
  "attachments": ["url1"]
}
```

Upload Routes

POST `/api/upload` **Purpose:** Upload files to Cloudinary

Request Body:

```
{
  "file": "base64_string",
  "folder": "string (optional)"
}
```

Response:

```
{
  "url": "cloudinary_url",
  "publicId": "string"
}
```

Authentication

Provider: NextAuth v5

Session Strategy: JWT

Roles: - `admin`: Full system access - `agent`: Manage shipments, tours, clients - `driver`: Execute tours, update shipments

Protected Routes: All API routes except `/api/auth/*`

Authorization Middleware: `requireAuth()` function validates JWT and attaches user to request

---

Error Handling

Wrapper Function: `withErrorHandler(handler)`

Standard Error Responses:

```
{
  "error": "Error message",
  "statusCode": 400|404|500
}
```

Success Responses:

```
{
  "data": { ... },
  "message": "Success message",
```

```
  "pagination": { ... } // if applicable
}
```

Validation: Zod schemas for all request bodies

Pagination

Query Parameters: - `page` (default: 1) - `limit` (default: 10, max: 100)

Pagination Metadata:

```
{
  "page": 1,
  "limit": 10,
  "total": 150,
  "totalPages": 15
}
```

File Uploads

Provider: Cloudinary

Supported Types: Images, PDFs

Folders: - `incidents/` - Incident documentation - `pod/` - Proof of delivery photos - `complaints/` - Complaint attachments

Max File Size: 10MB per file