Dokumentasi UML SIM4LON - Sistem Informasi Manajemen Distribusi LPG

Daftar Isi

---

1. Pendahuluan

SIM4LON adalah Sistem Informasi Manajemen Distribusi LPG yang digunakan untuk mengelola:

- **Pemesanan LPG** dari Pangkalan ke Agen/Distributor
- **Penjualan LPG** dari Pangkalan ke Konsumen
- **Manajemen Stok** LPG (3kg, 5.5kg, 12kg, 50kg)
- **Pencatatan Pembayaran** dan Invoice
- **Pengelolaan Data** Pangkalan, Driver, dan Pengguna
- **Dashboard & Laporan** operasional

Tech Stack

- **Backend**: NestJS + Prisma ORM
- **Database**: PostgreSQL
- **Frontend**: Astro + React Components

User Roles

Role	Deskripsi	Akses
ADMIN	Administrator sistem	Full access - kelola semua data
OPERATOR	Staff operasional	Kelola pesanan, pembayaran, stok
PANGKALAN	Pemilik pangkalan	Akses data milik sendiri (multi-tenant)

---

2. Use Case Diagram

2.1 Diagram Utama

```
flowchart TB
    subgraph System["SIM4LON System"]
        UC1["Login/Logout"]
        UC2["Dashboard"]
        UC3["Kelola Profil"]

        subgraph Order["Manajemen Pesanan"]
```

```
        UC4["Buat Pesanan"]
        UC5["Lihat Daftar Pesanan"]
        UC6["Update Status Pesanan"]
        UC7["Detail Pesanan"]
        UC8["Hapus Pesanan"]
    end

    subgraph Payment["Manajemen Pembayaran"]
        UC9["Catat Pembayaran"]
        UC10["Lihat Riwayat Pembayaran"]
        UC11["Generate Invoice"]
    end

    subgraph Stock["Manajemen Stok"]
        UC12["Input Stok Masuk"]
        UC13["Input Stok Keluar"]
        UC14["Lihat Ringkasan Stok"]
        UC15["Lihat Histori Stok"]
    end

    subgraph MasterData["Master Data - ADMIN Only"]
        UC16["Kelola Pengguna"]
        UC17["Kelola Pangkalan"]
        UC18["Kelola Driver"]
        UC19["Kelola Produk LPG"]
        UC20["Kelola Agen"]
    end

    subgraph Reports["Laporan"]
        UC21["Laporan Penjualan"]
        UC22["Laporan Stok"]
        UC23["Export Laporan"]
    end

    subgraph ConsumerSales["Penjualan Konsumen"]
        UC24["Catat Penjualan"]
        UC25["Kelola Konsumen"]
        UC26["Statistik Penjualan"]
    end
end

Admin((Admin))
Operator((Operator))
Pangkalan((Pangkalan))

Admin --> UC1 & UC2 & UC3
Admin --> UC4 & UC5 & UC6 & UC7 & UC8
Admin --> UC9 & UC10 & UC11
Admin --> UC12 & UC13 & UC14 & UC15
Admin --> UC16 & UC17 & UC18 & UC19 & UC20
Admin --> UC21 & UC22 & UC23
```

```
    Operator --> UC1 & UC2 & UC3
    Operator --> UC4 & UC5 & UC6 & UC7
    Operator --> UC9 & UC10 & UC11
    Operator --> UC12 & UC13 & UC14 & UC15
    Operator --> UC21 & UC22 & UC23

    Pangkalan --> UC1 & UC2 & UC3
    Pangkalan --> UC5 & UC7
    Pangkalan --> UC10
    Pangkalan --> UC14
    Pangkalan --> UC24 & UC25 & UC26
```

2.2 Access Matrix

Use Case	ADMIN	OPERATOR	PANGKALAN
Login/Logout	✅	✅	✅
Dashboard	✅	✅	✅ (own data)
Kelola Profil	✅	✅	✅
Pesanan			
Buat Pesanan	✅	✅	❌
Lihat Daftar Pesanan	✅ (all)	✅ (all)	✅ (own)
Update Status Pesanan	✅	✅	❌
Detail Pesanan	✅	✅	✅ (own)
Hapus Pesanan	✅	❌	❌
Pembayaran			
Catat Pembayaran	✅	✅	❌
Lihat Riwayat Pembayaran	✅	✅	✅ (own)
Generate Invoice	✅	✅	❌
Stok			
Input Stok Masuk	✅	✅	❌
Input Stok Keluar	✅	✅	❌
Ringkasan Stok	✅	✅	✅ (own)
Histori Stok	✅	✅	❌
Master Data			
Kelola Pengguna	✅	❌	❌
Kelola Pangkalan	✅	❌	❌

Kelola Driver	✅	❌	❌
Kelola Produk LPG	✅	❌	❌
Kelola Agen	✅	❌	❌
Penjualan Konsumen			
Catat Penjualan	❌	❌	✅
Kelola Konsumen	❌	❌	✅
Statistik Penjualan	❌	❌	✅

2.3 Use Case Detail

UC4: Buat Pesanan

Aspek	Deskripsi
Actor	Admin, Operator
Precondition	User sudah login, ada pangkalan aktif
Main Flow	1. Pilih pangkalan 2. Pilih driver (opsional) 3. Tambah item LPG (tipe, qty, harga) 4. Simpan pesanan
Postcondition	Pesanan tersimpan dengan status DRAFT, activity log tercatat
Include	UC11 Generate Invoice (auto)

UC6: Update Status Pesanan

Aspek	Deskripsi
Actor	Admin, Operator
Precondition	Pesanan sudah ada
Main Flow	1. Pilih pesanan 2. Ubah status sesuai workflow 3. Tambah catatan (opsional)
Business Rule	Status hanya bisa diubah sesuai alur: DRAFT → MENUNGGU_PEMBAYARAN → DIPROSES → SIAP_KIRIM → DIKIRIM → SELESAI
Alternative	Status bisa diubah ke BATAL dari status manapun

---

3. Activity Diagram

3.1 Alur Pembuatan Pesanan (Order Workflow)

```
flowchart TD
    Start([Mulai]) --> Login{Sudah Login?}
    Login -- Tidak --> LoginPage[Halaman Login]
    LoginPage --> InputCredential[Input Email & Password]
    InputCredential --> Validate{Validasi}
    Validate -- Gagal --> LoginPage
    Validate -- Berhasil --> Login
    Login -- Ya --> Dashboard[Dashboard]

    Dashboard --> BuatPesanan[Buat Pesanan Baru]
    BuatPesanan --> PilihPangkalan[Pilih Pangkalan]
    PilihPangkalan --> PilihDriver[Pilih Driver - Optional]
    PilihDriver --> TambahItem[Tambah Item LPG]

    TambahItem --> InputItem[Input Tipe LPG, Qty, Harga]
    InputItem --> ItemLagi{Tambah Item Lagi?}
    ItemLagi -- Ya --> TambahItem
    ItemLagi -- Tidak --> TambahCatatan[Tambah Catatan - Optional]

    TambahCatatan --> HitungTotal[Sistem Hitung Total + PPN]
    HitungTotal --> SimpanPesanan[Simpan Pesanan]
    SimpanPesanan --> GenerateCode[Generate Order Code: ORD-XXXX]
    GenerateCode --> CreateTimeline[Buat Timeline Track - DRAFT]
    CreateTimeline --> LogActivity[Log Activity: ORDER_NEW]
    LogActivity --> CreatePaymentDetail[Buat Payment Detail]
    CreatePaymentDetail --> AutoSyncStock{Status SELESAI?}
    AutoSyncStock -- Ya --> UpdateStock[Update Stok Pangkalan]
    AutoSyncStock -- Tidak --> End1([Selesai - Status DRAFT])
    UpdateStock --> End2([Selesai])
```

3.2 Alur Perubahan Status Pesanan

```
flowchart TD
    Start([Mulai]) --> CurrentStatus{Status Saat Ini}

    CurrentStatus --> DRAFT[DRAFT]
    CurrentStatus --> MENUNGGU["MENUNGGU PEMBAYARAN"]
    CurrentStatus --> DIPROSES[DIPROSES]
    CurrentStatus --> SIAP_KIRIM[SIAP KIRIM]
    CurrentStatus --> DIKIRIM[DIKIRIM]

    DRAFT --> |"Ajukan"| MENUNGGU
    DRAFT --> |"Batalkan"| BATAL1[BATAL]

    MENUNGGU --> |"Proses"| DIPROSES
    MENUNGGU --> |"Batalkan"| BATAL2[BATAL]

    DIPROSES --> |"Siap Kirim"| SIAP_KIRIM
    DIPROSES --> |"Batalkan"| BATAL3[BATAL]
```

```
    SIAP_KIRIM --> |"Kirim"| DIKIRIM
    SIAP_KIRIM --> |"Batalkan"| BATAL4[BATAL]

    DIKIRIM --> |"Selesai"| SELESAI[SELESAI]
    DIKIRIM --> |"Batalkan"| BATAL5[BATAL]

    SELESAI --> SyncStock[Auto-Sync Stok Pangkalan]
    SyncStock --> End1([Selesai])

    BATAL1 & BATAL2 & BATAL3 & BATAL4 & BATAL5 --> End2([Pesanan Dibatalkan])
```

3.3 Alur Pencatatan Pembayaran

```
flowchart TD
    Start([Mulai]) --> PilihPesanan[Pilih Pesanan]
    PilihPesanan --> CekStatus{Status Pesanan?}

    CekStatus -- DRAFT --> Tolak1[Tidak Bisa Bayar - Status DRAFT]
    CekStatus -- BATAL --> Tolak2[Tidak Bisa Bayar - Dibatalkan]
    CekStatus -- Valid --> InputPayment[Input Data Pembayaran]

    InputPayment --> PilihMetode[Pilih Metode: TUNAI/TRANSFER]
    PilihMetode --> InputJumlah[Input Jumlah Bayar]
    InputJumlah --> IsDP{Apakah DP?}

    IsDP -- Ya --> MarkDP[Tandai sebagai DP]
    IsDP -- Tidak --> FullPayment[Pembayaran Penuh]

    MarkDP & FullPayment --> UploadBukti{Upload Bukti?}
    UploadBukti -- Ya --> Upload[Upload ke Storage]
    UploadBukti -- Tidak --> Skip[Skip]

    Upload & Skip --> SaveRecord[Simpan Payment Record]
    SaveRecord --> UpdatePaymentDetail[Update Order Payment Detail]
    UpdatePaymentDetail --> CekLunas{Total Bayar >= Total?}

    CekLunas -- Ya --> MarkPaid[Tandai LUNAS]
    CekLunas -- Tidak --> MarkPartial[Tandai Belum Lunas]

    MarkPaid & MarkPartial --> LogActivity[Log Activity: PAYMENT_RECEIVED]
    LogActivity --> End([Selesai])
```

3.4 Alur Manajemen Stok

```
flowchart TD
    Start([Mulai]) --> JenisAksi{Jenis Aksi?}

    JenisAksi -- "Stok Masuk" --> InputMasuk[Input Stok Masuk]
    JenisAksi -- "Stok Keluar" --> InputKeluar[Input Stok Keluar]
```

```
    InputMasuk --> PilihProduk1[Pilih Produk LPG]
    InputKeluar --> PilihProduk2[Pilih Produk LPG]

    PilihProduk1 --> InputQty1[Input Jumlah]
    PilihProduk2 --> InputQty2[Input Jumlah]

    InputQty1 --> TambahNote1[Tambah Catatan - Optional]
    InputQty2 --> CekStok{Stok Cukup?}

    CekStok -- Tidak --> Error[Error: Stok Tidak Cukup]
    CekStok -- Ya --> TambahNote2[Tambah Catatan - Optional]

    TambahNote1 --> SimpanMasuk["Simpan (movement_type: MASUK)"]
    TambahNote2 --> SimpanKeluar["Simpan (movement_type: KELUAR)"]

    SimpanMasuk & SimpanKeluar --> UpdateSummary[Update Stock Summary]
    UpdateSummary --> LogActivity[Log Activity: STOCK_IN / STOCK_OUT]
    LogActivity --> End([Selesai])
```

---

4. Sequence Diagram

4.1 Login & Authentication

```
sequenceDiagram
    autonumber
    actor User
    participant FE as Frontend
    participant API as AuthController
    participant Service as AuthService
    participant DB as PostgreSQL
    participant JWT as JwtService

    User->>FE: Input email & password
    FE->>API: POST /api/auth/login
    API->>Service: login(dto)

    Service->>DB: findUnique(email)
    DB-->>Service: User data / null

    alt User tidak ditemukan
        Service-->>API: UnauthorizedException
        API-->>FE: 401 Error
        FE-->>User: "Email atau password salah"
    else User tidak aktif
        Service-->>API: UnauthorizedException
        API-->>FE: 401 Error
        FE-->>User: "Akun tidak aktif"
    else User valid
        Service->>Service: bcrypt.compare(password)
```

```
    alt Password salah
        Service-->>API: UnauthorizedException
        API-->>FE: 401 Error
        FE-->>User: "Email atau password salah"
    else Password benar
        Service->>JWT: sign(payload)
        JWT-->>Service: access_token
        Service-->>API: { access_token, user }
        API-->>FE: 200 OK
        FE->>FE: Simpan token di localStorage
        FE-->>User: Redirect ke Dashboard
    end
end
```

4.2 Buat Pesanan Baru

```
sequenceDiagram
    autonumber
    actor User
    participant FE as Frontend
    participant API as OrderController
    participant Service as OrderService
    participant Activity as ActivityService
    participant DB as PostgreSQL

    User->>FE: Isi form pesanan
    FE->>API: POST /api/orders
    Note over API: JwtAuthGuard validates token

    API->>Service: create(dto)

    Service->>DB: Generate order code
    DB-->>Service: ORD-XXXX

    Service->>Service: Calculate total + tax

    Service->>DB: Create order
    DB-->>Service: Order created

    loop For each item
        Service->>DB: Create order_item
        DB-->>Service: Item created
    end

    Service->>DB: Create order_payment_details
    DB-->>Service: Payment detail created

    Service->>DB: Create timeline_track (DRAFT)
    DB-->>Service: Timeline created

    Service->>Activity: logActivity('ORDER_NEW')
```

```
    Activity->>DB: Create activity_log
    DB-->>Activity: Log created

    Service-->>API: Order response
    API-->>FE: 201 Created
    FE-->>User: Tampilkan notifikasi sukses
```

4.3 Update Status Pesanan

```
sequenceDiagram
    autonumber
    actor User
    participant FE as Frontend
    participant API as OrderController
    participant Service as OrderService
    participant PangkalanStock as PangkalanStockService
    participant Activity as ActivityService
    participant DB as PostgreSQL

    User->>FE: Klik update status
    FE->>API: PATCH /api/orders/:id/status
    API->>Service: updateStatus(id, dto)

    Service->>DB: findOne(order)
    DB-->>Service: Order data

    Service->>Service: validateStatusTransition()

    alt Invalid transition
        Service-->>API: BadRequestException
        API-->>FE: 400 Error
        FE-->>User: "Transisi status tidak valid"
    else Valid transition
        Service->>DB: Update order status
        DB-->>Service: Updated

        Service->>DB: Create timeline_track
        DB-->>Service: Timeline created

        alt Status = SELESAI
            Service->>PangkalanStock: syncStock(order_items)
            PangkalanStock->>DB: Update pangkalan_stocks
            DB-->>PangkalanStock: Updated
            PangkalanStock->>DB: Create pangkalan_stock_movement
            DB-->>PangkalanStock: Created
        end

        Service->>Activity: logActivity('STATUS_CHANGED')
        Activity->>DB: Create activity_log

        Service-->>API: Updated order
```

```
        API-->>FE: 200 OK
        FE-->>User: Tampilkan status baru
    end
```

4.4 Catat Pembayaran

```
sequenceDiagram
    autonumber
    actor User
    participant FE as Frontend
    participant API as PaymentController
    participant Service as PaymentService
    participant DB as PostgreSQL

    User->>FE: Input data pembayaran
    FE->>API: POST /api/payments/records
    API->>Service: createRecord(dto, userId)

    Service->>DB: Create payment_record
    DB-->>Service: Record created

    Service->>DB: Upsert order_payment_details
    Note over DB: Update amount_paid with increment
    DB-->>Service: Updated

    Service->>DB: Get order total_amount
    DB-->>Service: Total amount

    Service->>Service: Check if fully paid

    alt Fully paid
        Service->>DB: Update is_paid = true
    end

    Service-->>API: Payment record
    API-->>FE: 201 Created
    FE-->>User: Tampilkan konfirmasi pembayaran
```

4.5 Consumer Order (Penjualan ke Konsumen)

```
sequenceDiagram
    autonumber
    actor Pangkalan as User Pangkalan
    participant FE as Frontend
    participant API as ConsumerOrderController
    participant Service as ConsumerOrderService
    participant StockService as PangkalanStockService
    participant DB as PostgreSQL
```

```
    Pangkalan->>FE: Input penjualan
    FE->>API: POST /api/consumer-orders
    Note over API: Extract pangkalan_id from JWT

    API->>Service: create(pangkalanId, dto)

    Service->>DB: Get default price (lpg_prices)
    DB-->>Service: Price data

    Service->>DB: Check pangkalan_stocks
    DB-->>Service: Current stock qty

    alt Stok tidak cukup
        Service-->>API: BadRequestException
        API-->>FE: 400 Error
        FE-->>Pangkalan: "Stok tidak mencukupi"
    else Stok cukup
        Service->>DB: Generate code PORD-XXXX
        DB-->>Service: Code generated

        Service->>DB: Create consumer_order
        DB-->>Service: Order created

        Service->>StockService: decreaseStock(lpg_type, qty)
        StockService->>DB: Update pangkalan_stocks
        StockService->>DB: Create stock_movement

        Service-->>API: Consumer order
        API-->>FE: 201 Created
        FE-->>Pangkalan: Tampilkan nota
    end
```

---

5. Class Diagram

5.1 Backend Service Architecture

```
classDiagram
    class PrismaService {
        <<Injectable>>
        +users
        +orders
        +pangkalans
        +drivers
        +payments
        +$connect()
        +$disconnect()
    }

    class AuthService {
        <<Injectable>>
```

```
    -prisma: PrismaService
    -jwtService: JwtService
    +register(dto: RegisterDto)
    +login(dto: LoginDto)
    +getProfile(userId: string)
    +updateProfile(userId: string, dto)
    +changePassword(userId: string, oldPwd, newPwd)
}

class UserService {
    <<Injectable>>
    -prisma: PrismaService
    +findAll(page, limit, search?)
    +findOne(id: string)
    +create(dto: CreateUserDto)
    +update(id: string, dto: UpdateUserDto)
    +remove(id: string)
    +resetPassword(id: string)
}

class OrderService {
    <<Injectable>>
    -prisma: PrismaService
    -activityService: ActivityService
    +findAll(page, limit, status?, pangkalanId?, driverId?)
    +findOne(id: string)
    +create(dto: CreateOrderDto)
    +update(id: string, dto: UpdateOrderDto)
    +updateStatus(id: string, dto: UpdateOrderStatusDto)
    +remove(id: string)
    +validateStatusTransition(current, new)
    +getStats(todayOnly: boolean)
}

class PaymentService {
    <<Injectable>>
    -prisma: PrismaService
    +findAllRecords(page, limit, orderId?, invoiceId?, method?)
    +findOneRecord(id: string)
    +createRecord(dto: CreatePaymentRecordDto, userId)
    +updateOrderPayment(orderId, dto: UpdateOrderPaymentDto)
    +getOrderPayment(orderId: string)
}

class StockService {
    <<Injectable>>
    -prisma: PrismaService
    +getHistory(page, limit, lpgType?, movementType?)
    +createMovement(dto: CreateStockMovementDto, userId)
    +getSummary()
    +getHistoryByType(lpgType, limit)
}
```

```
class PangkalanService {
    <<Injectable>>
    -prisma: PrismaService
    +findAll(page, limit, isActive?, search?)
    +findOne(id: string)
    +create(dto: CreatePangkalanDto)
    +update(id: string, dto: UpdatePangkalanDto)
    +remove(id: string)
}

class DriverService {
    <<Injectable>>
    -prisma: PrismaService
    +findAll(page, limit, isActive?)
    +findOne(id: string)
    +create(dto: CreateDriverDto)
    +update(id: string, dto: UpdateDriverDto)
    +remove(id: string)
}

class ConsumerOrderService {
    <<Injectable>>
    -prisma: PrismaService
    +findAll(pangkalanId, page, limit, options?)
    +findOne(id, pangkalanId)
    +create(pangkalanId, dto: CreateConsumerOrderDto)
    +update(id, pangkalanId, dto)
    +remove(id, pangkalanId)
    +getStats(pangkalanId, todayOnly?)
    +getRecentSales(pangkalanId, limit)
    +getChartData(pangkalanId)
}

class ActivityService {
    <<Injectable>>
    -prisma: PrismaService
    +findAll(page, limit, type?, userId?)
    +create(dto: CreateActivityLogDto)
    +getRecent(limit)
    +getByType(type, limit)
    +logActivity(type, title, options?)
}

class DashboardService {
    <<Injectable>>
    -prisma: PrismaService
    +getStats()
    +getStockSummary()
    +getDynamicProductsStock()
    +getSalesChart()
    +getStockChart()
```

```
        +getProfitChart()
        +getTopPangkalan()
        +getStockConsumption()
        +getRecentActivities()
    }

    AuthService --> PrismaService
    UserService --> PrismaService
    OrderService --> PrismaService
    OrderService --> ActivityService
    PaymentService --> PrismaService
    StockService --> PrismaService
    PangkalanService --> PrismaService
    DriverService --> PrismaService
    ConsumerOrderService --> PrismaService
    ActivityService --> PrismaService
    DashboardService --> PrismaService
```

5.2 Controller Layer

```
classDiagram
    class AuthController {
        <<Controller>>
        +register(dto: RegisterDto)
        +login(dto: LoginDto)
        +getProfile(userId)
        +updateProfile(userId, dto)
        +changePassword(userId, dto)
    }

    class OrderController {
        <<Controller>>
        +findAll(query)
        +findOne(id)
        +create(dto)
        +update(id, dto)
        +updateStatus(id, dto)
        +remove(id)
        +getStats(query)
    }

    class PaymentController {
        <<Controller>>
        +findAllRecords(query)
        +findOneRecord(id)
        +createRecord(dto, userId)
        +updateOrderPayment(orderId, dto)
        +getOrderPayment(orderId)
    }

    class UserController {
```

```
    <<Controller>>
    +findAll(query)
    +findOne(id)
    +create(dto)
    +update(id, dto)
    +remove(id)
    +resetPassword(id)
}

class PangkalanController {
    <<Controller>>
    +findAll(query)
    +findOne(id)
    +create(dto)
    +update(id, dto)
    +remove(id)
}

class DriverController {
    <<Controller>>
    +findAll(query)
    +findOne(id)
    +create(dto)
    +update(id, dto)
    +remove(id)
}

class StockController {
    <<Controller>>
    +getHistory(query)
    +createMovement(dto, userId)
    +getSummary()
}

class ConsumerOrderController {
    <<Controller>>
    +findAll(pangkalanId, query)
    +findOne(id, pangkalanId)
    +create(pangkalanId, dto)
    +update(id, pangkalanId, dto)
    +remove(id, pangkalanId)
    +getStats(pangkalanId)
    +getRecentSales(pangkalanId)
    +getChartData(pangkalanId)
}

class DashboardController {
    <<Controller>>
    +getStats()
    +getSalesChart()
    +getStockChart()
    +getProfitChart()
```

```
        +getTopPangkalan()
        +getRecentActivities()
    }
```

5.3 Guards & Middleware

```
classDiagram
    class JwtAuthGuard {
        <<CanActivate>>
        +canActivate(context: ExecutionContext): boolean
    }

    class JwtStrategy {
        <<PassportStrategy>>
        -prisma: PrismaService
        +validate(payload: JwtPayload): User
    }

    class RolesGuard {
        <<CanActivate>>
        +canActivate(context: ExecutionContext): boolean
    }

    JwtAuthGuard ..> JwtStrategy
    RolesGuard ..> JwtPayload
```

---

6. Entity Relationship Diagram (ERD)

6.1 ERD Lengkap

```
erDiagram
    users ||--o{ activity_logs : "creates"
    users ||--o{ payment_records : "records"
    users ||--o{ stock_histories : "records"
    users }o--|| pangkalans : "belongs_to"

    pangkalans ||--o{ orders : "has"
    pangkalans ||--o{ consumers : "has"
    pangkalans ||--o{ consumer_orders : "has"
    pangkalans ||--o{ pangkalan_stocks : "has"
    pangkalans ||--o{ pangkalan_stock_movements : "has"
    pangkalans ||--o{ lpg_prices : "has"
    pangkalans ||--o{ expenses : "has"
    pangkalans }o--|| agen : "supplied_by"

    drivers ||--o{ orders : "delivers"

    orders ||--|{ order_items : "contains"
    orders ||--|| order_payment_details : "has"
```

```
orders ||--o{ timeline_tracks : "has"
orders ||--o{ invoices : "has"
orders ||--o{ payment_records : "has"
orders ||--o{ activity_logs : "has"

invoices ||--o{ payment_records : "has"

lpg_products ||--o{ stock_histories : "tracked_by"

consumers ||--o{ consumer_orders : "makes"
consumers ||--o{ consumer_pricing : "has"

users {
    uuid id PK
    string code UK "USR-001"
    string email UK
    string password
    string name
    string phone
    string avatar_url
    user_role role "ADMIN|OPERATOR|PANGKALAN"
    uuid pangkalan_id FK
    boolean is_active
    timestamp created_at
    timestamp updated_at
    timestamp deleted_at
}

agen {
    uuid id PK
    string code UK "AGN-001"
    string name
    string address
    string pic_name
    string phone
    string email
    string note
    boolean is_active
    timestamp created_at
    timestamp updated_at
    timestamp deleted_at
}

pangkalans {
    uuid id PK
    string code UK "PKL-001"
    string name
    string address
    string region
    string pic_name
    string phone
    string email
```

```
    int capacity
    string note
    uuid agen_id FK
    boolean is_active
    timestamp created_at
    timestamp updated_at
    timestamp deleted_at
}

drivers {
    uuid id PK
    string code UK "DRV-001"
    string name
    string phone
    string vehicle_id
    string note
    boolean is_active
    timestamp created_at
    timestamp updated_at
    timestamp deleted_at
}

orders {
    uuid id PK
    string code UK "ORD-0001"
    uuid pangkalan_id FK
    uuid driver_id FK
    date order_date
    status_pesanan current_status
    decimal subtotal
    decimal tax_amount
    decimal total_amount
    string note
    timestamp created_at
    timestamp updated_at
    timestamp deleted_at
}

order_items {
    uuid id PK
    uuid order_id FK
    lpg_type lpg_type
    string label
    decimal price_per_unit
    int qty
    decimal sub_total
    boolean is_taxable
    decimal tax_amount
    timestamp created_at
    timestamp updated_at
}
```

```
order_payment_details {
    uuid id PK
    uuid order_id FK UK
    boolean is_paid
    boolean is_dp
    payment_method payment_method
    decimal amount_paid
    timestamp payment_date
    string proof_url
    timestamp created_at
    timestamp updated_at
}

timeline_tracks {
    uuid id PK
    uuid order_id FK
    status_pesanan status
    string description
    string note
    timestamp created_at
}

invoices {
    uuid id PK
    uuid order_id FK
    string invoice_number UK
    date invoice_date
    date due_date
    string billing_address
    string billed_to_name
    decimal sub_total
    decimal tax_rate
    decimal tax_amount
    decimal grand_total
    string payment_status
    timestamp created_at
    timestamp updated_at
    timestamp deleted_at
}

payment_records {
    uuid id PK
    uuid order_id FK
    uuid invoice_id FK
    payment_method method
    decimal amount
    timestamp payment_time
    string proof_url
    uuid recorded_by_user_id FK
    string note
    timestamp created_at
}
```

```
lpg_products {
    uuid id PK
    string name
    decimal size_kg
    lpg_category category
    string color
    string description
    decimal selling_price
    decimal cost_price
    boolean is_active
    timestamp created_at
    timestamp updated_at
    timestamp deleted_at
}

stock_histories {
    uuid id PK
    uuid lpg_product_id FK
    lpg_type lpg_type
    stock_movement_type movement_type
    int qty
    string note
    uuid recorded_by_user_id FK
    timestamp timestamp
    timestamp created_at
}

activity_logs {
    uuid id PK
    uuid user_id FK
    uuid order_id FK
    string type
    string title
    string description
    string pangkalan_name
    decimal detail_numeric
    string icon_name
    status_pesanan order_status
    timestamp timestamp
    timestamp created_at
}

consumers {
    uuid id PK
    uuid pangkalan_id FK
    string name
    string nik
    string kk
    consumer_type consumer_type
    string phone
    string address
```

```
    string note
    boolean is_active
    timestamp created_at
    timestamp updated_at
}

consumer_orders {
    uuid id PK
    string code UK "PORD-0001"
    uuid pangkalan_id FK
    uuid consumer_id FK
    string consumer_name
    lpg_type lpg_type
    int qty
    decimal price_per_unit
    decimal cost_price
    decimal total_amount
    consumer_payment_status payment_status
    string note
    timestamp sale_date
    timestamp created_at
    timestamp updated_at
}

pangkalan_stocks {
    uuid id PK
    uuid pangkalan_id FK
    lpg_type lpg_type
    int qty
    int warning_level
    int critical_level
    timestamp created_at
    timestamp updated_at
}

pangkalan_stock_movements {
    uuid id PK
    uuid pangkalan_id FK
    lpg_type lpg_type
    string movement_type
    int qty
    string source
    uuid reference_id
    string note
    timestamp movement_date
    timestamp created_at
}

lpg_prices {
    uuid id PK
    uuid pangkalan_id FK
    lpg_type lpg_type
```

```
        decimal cost_price
        decimal selling_price
        boolean is_active
        timestamp created_at
        timestamp updated_at
    }

    consumer_pricing {
        uuid id PK
        uuid pangkalan_id
        uuid consumer_id FK
        lpg_type lpg_type
        decimal price
        timestamp created_at
        timestamp updated_at
    }

    expenses {
        uuid id PK
        uuid pangkalan_id FK
        string category
        decimal amount
        string description
        timestamp expense_date
        timestamp created_at
        timestamp updated_at
    }
```

6.2 Enum Types

```
classDiagram
    class user_role {
        <<enumeration>>
        ADMIN
        OPERATOR
        PANGKALAN
    }

    class status_pesanan {
        <<enumeration>>
        DRAFT
        MENUNGGU_PEMBAYARAN
        DIPROSES
        SIAP_KIRIM
        DIKIRIM
        SELESAI
        BATAL
    }

    class lpg_type {
        <<enumeration>>
```

```
    kg3 : 3kg
    kg5 : 5.5kg
    kg12 : 12kg
    kg50 : 50kg
}

class lpg_category {
    <<enumeration>>
    SUBSIDI
    NON_SUBSIDI
}

class payment_method {
    <<enumeration>>
    TUNAI
    TRANSFER
}

class stock_movement_type {
    <<enumeration>>
    MASUK
    KELUAR
}

class consumer_type {
    <<enumeration>>
    RUMAH_TANGGA
    WARUNG
}

class consumer_payment_status {
    <<enumeration>>
    LUNAS
}
```

6.3 Table Relationships Summary

Parent Table	Child Table	Relationship	Foreign Key
users	activity_logs	1:N	user_id
users	payment_records	1:N	recorded_by_user_id
users	stock_histories	1:N	recorded_by_user_id
pangkalans	users	1:N	pangkalan_id
pangkalans	orders	1:N	pangkalan_id
pangkalans	consumers	1:N	pangkalan_id
pangkalans	consumer_orders	1:N	pangkalan_id

pangkalans	pangkalan_stocks	1:N	pangkalan_id
pangkalans	lpg_prices	1:N	pangkalan_id
pangkalans	expenses	1:N	pangkalan_id
agen	pangkalans	1:N	agen_id
drivers	orders	1:N	driver_id
orders	order_items	1:N	order_id
orders	order_payment_details	1:1	order_id
orders	timeline_tracks	1:N	order_id
orders	invoices	1:N	order_id
orders	payment_records	1:N	order_id
orders	activity_logs	1:N	order_id
invoices	payment_records	1:N	invoice_id
lpg_products	stock_histories	1:N	lpg_product_id
consumers	consumer_orders	1:N	consumer_id
consumers	consumer_pricing	1:N	consumer_id

Catatan Penting

Status Workflow Pesanan

```
DRAFT → MENUNGGU_PEMBAYARAN → DIPROSES → SIAP_KIRIM → DIKIRIM → SELESAI
  ↓              ↓              ↓           ↓          ↓
BATAL          BATAL          BATAL       BATAL      BATAL
```

Multi-Tenant Architecture (Pangkalan)

- User dengan role `PANGKALAN` hanya bisa mengakses data milik pangkalan sendiri
- `pangkalan_id` di JWT digunakan untuk filter data
- Consumer orders dan stock movements di-scope per pangkalan

Auto-Sync Stock

- Ketika status pesanan berubah ke `SELESAI`, stok pangkalan otomatis bertambah
- Movement type: `MASUK` dari pesanan agen ke pangkalan