5. O Cliente recebe uma notificação de confirmação e o Vendedor recebe uma notificação de novo pedido.

Diagrama de Classes

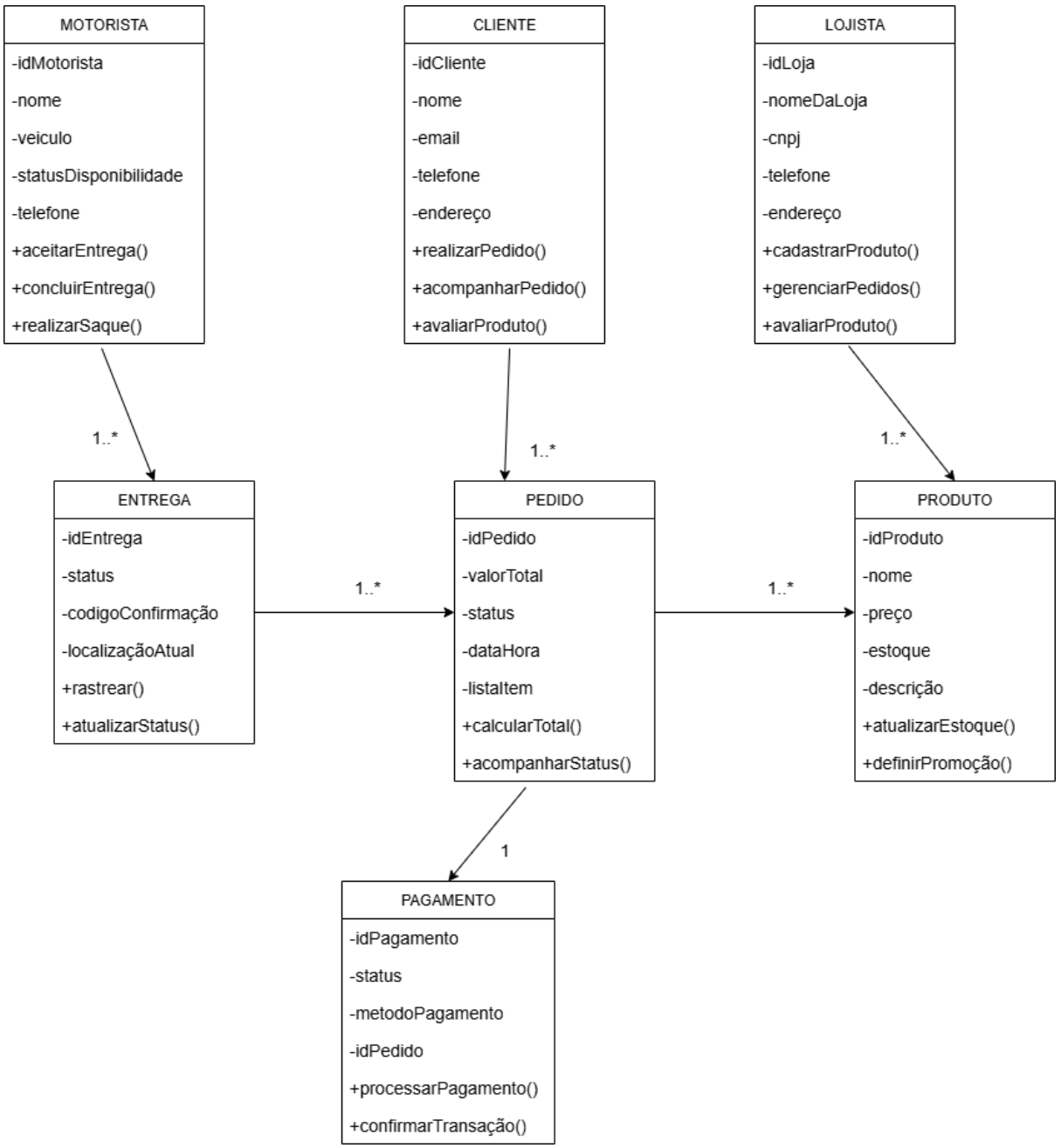

Diagrama de Classes

Visão Geral

Este documento descreve as classes principais do sistema WIN Marketplace, baseadas nas tabelas do banco de dados. Cada classe representa uma entidade, com seus atributos (propriedades) e seus métodos (comportamentos).

Classes, Atributos e Métodos

Class: User

- **Atributos:**
  - `id`: UUID
  - `email`: String
  - `password_hash`: String
  - `user_type`: String (enum: 'CUSTOMER', 'MERCHANT', 'DRIVER', 'ADMIN')
  - `status`: String (enum: 'ACTIVE', 'INACTIVE', 'SUSPENDED', 'PENDING')
  - `email_verified`: Boolean
  - `phone`: String
  - `created_at`: Date
  - `updated_at`: Date
  - `last_login`: Date
  - `profile_image_url`: String
- **Métodos:**
  - `authenticate(email, password)`: Boolean
  - `updateProfile(profileData)`: void
  - `changePassword(newPassword)`: Boolean
  - `deactivateAccount()`: void

Class: Customer

- **Atributos:**
  - `id`: UUID
  - `first_name`: String
  - `last_name`: String
  - `cpf`: String
  - `birth_date`: Date
  - `gender`: String
  - `preferences`: JSONB (objeto de preferências)
  - `loyalty_points`: Integer
  - `total_orders`: Integer
  - `total_spent`: Decimal

- **Métodos:**
  - `createOrder(cart, deliveryAddress)`: Order
  - `trackOrder()`: OrderStatus
  - `addAddress(addressData)`: Address
  - `reviewProduct(product, rating, comment)`: ProductReview
  - `reviewDriver(driver, rating, comment)`: DriverReview
- **Relacionamentos:**
  - `1:1` com `User` (herda ou está diretamente ligado a um `User`).
  - `1:N` com `Order` (um cliente pode ter muitos pedidos).

Class: Merchant

- **Atributos:**
  - `id`: UUID
  - `store_name`: String
  - `owner_name`: String
  - `cnpj`: String
  - `cpf_owner`: String
  - `description`: String
  - `category`: String
  - `is_approved`: Boolean
  - `commission_rate`: Decimal
  - `total_sales`: Decimal
  - `rating`: Decimal
  - `rating_count`: Integer
  - `operating_hours`: JSONB
  - `delivery_radius`: Integer
  - `minimum_order`: Decimal
- **Métodos:**
  - `addProduct(productData)`: Product
  - `updateProduct(productId, productData)`: void
  - `updateOrderStatus(orderId, newStatus)`: void
  - `viewSalesReports(period)`: ReportData
  - `processWithdrawal(amount)`: void
- **Relacionamentos:**
  - `1:1` com `User`.
  - `1:N` com `Product` (uma loja pode ter muitos produtos).
  - `1:N` com `OrderItem` (uma loja recebe muitos itens de pedidos).

Class: Driver

- **Atributos:**
  - `id`: UUID
  - `first_name`: String

    - `last_name`: String
    - `cpf`: String
    - `cnh`: String
    - `cnh_category`: String
    - `vehicle_type`: String
    - `vehicle_plate`: String
    - `is_approved`: Boolean
    - `is_available`: Boolean
    - `current_location`: Point (geometria)
    - `rating`: Decimal
    - `rating_count`: Integer
    - `total_deliveries`: Integer
    - `total_earnings`: Decimal
    - `bank_account`: JSONB
- **Métodos:**
    - `setAvailability(isAvailable)`: void
    - `acceptDelivery(deliveryId)`: void
    - `updateDeliveryStatus(deliveryId, newStatus)`: void
    - `viewEarnings(period)`: ReportData
- **Relacionamentos:**
    - `1:1` com `User`.
    - `1:N` com `Delivery` (um motorista realiza muitas entregas).

Class: Admin

- **Atributos:**
    - `id`: UUID
    - `name`: String
    - `role`: String
    - `permissions`: JSONB
    - `last_activity`: Date
- **Métodos:**
    - `manageUsers(userId, action)`: void
    - `approveMerchant(merchantId)`: void
    - `generateReports(reportType)`: ReportData
    - `setCommissionRate(newRate)`: void
- **Relacionamentos:**
    - `1:1` com `User`.

Class: Address

- **Atributos:**
    - `id`: UUID
    - `user_id`: UUID

    - `type`: String
    - `label`: String
    - `cep`: String
    - `street`: String
    - `number`: String
    - `complement`: String
    - `neighborhood`: String
    - `city`: String
    - `state`: String
    - `coordinates`: Point
    - `is_default`: Boolean
    - `created_at`: Date
- **Métodos:**
    - `validateCEP(cep)`: Boolean
- **Relacionamentos:**
    - `N:1` com `User` (muitos endereços pertencem a um único usuário).

Class: Category

- **Atributos:**
    - `id`: UUID
    - `name`: String
    - `slug`: String
    - `parent_id`: UUID
    - `icon`: String
    - `description`: String
    - `is_active`: Boolean
    - `sort_order`: Integer
    - `created_at`: Date
- **Relacionamentos:**
    - `1:N` com `Product` (uma categoria pode ter muitos produtos).
    - `1:N` consigo mesma (categorias podem ter subcategorias).

Class: Product

- **Atributos:**
    - `id`: UUID
    - `merchant_id`: UUID
    - `category_id`: UUID
    - `name`: String
    - `slug`: String
    - `description`: String
    - `short_description`: String

    - `sku`: String
    - `barcode`: String
    - `price`: Decimal
    - `compare_price`: Decimal
    - `cost_price`: Decimal
    - `stock_quantity`: Integer
    - `low_stock_alert`: Integer
    - `track_stock`: Boolean
    - `weight`: Decimal
    - `length`: Decimal
    - `width`: Decimal
    - `height`: Decimal
    - `status`: String
    - `is_featured`: Boolean
    - `meta_title`: String
    - `meta_description`: String
    - `keywords`: String
    - `views_count`: Integer
    - `sales_count`: Integer
    - `rating`: Decimal
    - `rating_count`: Integer
    - `created_at`: Date
    - `updated_at`: Date
- **Métodos:**
    - `calculatePriceWithDiscount()`: Decimal
    - `updateStock(quantityChange)`: void
    - `addRating(rating)`: void
- **Relacionamentos:**
    - `N:1` com `Merchant` (muitos produtos pertencem a uma loja).
    - `N:1` com `Category` (muitos produtos pertencem a uma categoria).
    - `1:N` com `ProductImage` (um produto tem muitas imagens).
    - `1:N` com `ProductVariant` (um produto pode ter muitas variações).
    - `1:N` com `ProductReview` (um produto pode ter muitas avaliações).
    - `1:N` com `OrderItem` (um produto pode estar em muitos itens de pedido).

Class: ProductImage

- **Atributos:**
    - `id`: UUID
    - `product_id`: UUID
    - `image_url`: String
    - `alt_text`: String
    - `sort_order`: Integer

    - `is_primary`: Boolean
- **Relacionamentos:**
    - `N:1` com `Product`.

Class: ProductVariant

- **Atributos:**
    - `id`: UUID
    - `product_id`: UUID
    - `name`: String
    - `sku`: String
    - `price`: Decimal
    - `stock_quantity`: Integer
    - `attributes`: JSONB
    - `is_active`: Boolean
    - `created_at`: Date
- **Relacionamentos:**
    - `N:1` com `Product`.

Class: Order

- **Atributos:**
    - `id`: UUID
    - `order_number`: String
    - `customer_id`: UUID
    - `status`: String
    - `subtotal`: Decimal
    - `discount_amount`: Decimal
    - `shipping_amount`: Decimal
    - `tax_amount`: Decimal
    - `total_amount`: Decimal
    - `delivery_address`: JSONB
    - `created_at`: Date
    - `confirmed_at`: Date
    - `delivered_at`: Date
    - `delivery_code`: String
    - `tracking_number`: String
    - `customer_notes`: String
    - `internal_notes`: String
- **Métodos:**
    - `calculateTotal()`: Decimal
    - `cancelOrder()`: void
- **Relacionamentos:**
    - `N:1` com `Customer`.

    - `1:N` com `OrderItem` (um pedido contém muitos itens).
    - `1:1` com `Payment` (um pedido tem um pagamento).
    - `1:1` com `Delivery` (um pedido tem uma entrega).

Class: OrderItem

- **Atributos:**
    - `id`: UUID
    - `order_id`: UUID
    - `product_id`: UUID
    - `variant_id`: UUID
    - `merchant_id`: UUID
    - `product_name`: String
    - `product_sku`: String
    - `quantity`: Integer
    - `unit_price`: Decimal
    - `total_price`: Decimal
    - `status`: String
    - `created_at`: Date
- **Relacionamentos:**
    - `N:1` com `Order`.
    - `N:1` com `Product`.
    - `N:1` com `Merchant`.

Class: Delivery

- **Atributos:**
    - `id`: UUID
    - `order_id`: UUID
    - `driver_id`: UUID
    - `status`: String
    - `pickup_address`: JSONB
    - `delivery_address`: JSONB
    - `current_location`: Point
    - `assigned_at`: Date
    - `picked_up_at`: Date
    - `delivered_at`: Date
    - `delivery_fee`: Decimal
    - `driver_earnings`: Decimal
    - `confirmation_code`: String
    - `proof_of_delivery`: JSONB
    - `delivery_notes`: String
    - `created_at`: Date

- **Métodos:**
  - `updateLocation(newLocation)`: void
- **Relacionamentos:**
  - `N:1` com `Order`.
  - `N:1` com `Driver`.
  - `1:N` com `DeliveryTracking` (uma entrega pode ter muitos pontos de rastreamento).

Class: DeliveryTracking

- **Atributos:**
  - `id`: UUID
  - `delivery_id`: UUID
  - `location`: Point
  - `timestamp`: Date
  - `status`: String
  - `notes`: String
- **Relacionamentos:**
  - `N:1` com `Delivery`.

Class: Payment

- **Atributos:**
  - `id`: UUID
  - `order_id`: UUID
  - `payment_method`: String
  - `status`: String
  - `amount`: Decimal
  - `fee_amount`: Decimal
  - `net_amount`: Decimal
  - `gateway_provider`: String
  - `transaction_id`: String
  - `gateway_response`: JSONB
  - `splits`: JSONB
  - `processed_at`: Date
  - `refunded_at`: Date
  - `created_at`: Date
- **Métodos:**
  - `processPayment()`: Boolean
  - `refund()`: Boolean
- **Relacionamentos:**
  - `N:1` com `Order`.

Class: ProductReview

- **Atributos:**
  - `id`: UUID
  - `product_id`: UUID
  - `customer_id`: UUID
  - `order_id`: UUID
  - `rating`: Integer
  - `comment`: String
- **Relacionamentos:**
  - `N:1` com `Product`.
  - `N:1` com `Customer`.
  - `N:1` com `Order`.

Class: DriverReview

- **Atributos:**
  - `id`: UUID
  - `driver_id`: UUID
  - `customer_id`: UUID
  - `order_id`: UUID
  - `rating`: Integer
  - `comment`: String
- **Relacionamentos:**
  - `N:1` com `Driver`.
  - `N:1` com `Customer`.
  - `N:1` com `Order`.

Explicação sobre a Class: `ProductVariant`

A classe **`ProductVariant`** serve para gerenciar as diferentes versões de um mesmo produto. Em vez de criar um novo produto para cada variação (por exemplo, uma camiseta azul e uma camiseta vermelha), você cria um único produto principal (a camiseta) e utiliza a classe `ProductVariant` para registrar suas opções.

Para entender melhor, imagine que um lojista de roupas vende a "Camiseta Básica WIN". Essa camiseta existe em duas cores e três tamanhos. Sem a classe `ProductVariant`, o lojista teria que cadastrar seis produtos diferentes:

- Camiseta Básica WIN - Azul - P
- Camiseta Básica WIN - Azul - M
- Camiseta Básica WIN - Azul - G

- Camiseta Básica WIN - Vermelha - P
- Camiseta Básica WIN - Vermelha - M
- Camiseta Básica WIN - Vermelha - G

Isso tornaria o gerenciamento do catálogo e do estoque muito difícil.

Com a classe `ProductVariant`, o processo fica mais organizado:

1. Você tem a classe principal **`Product`** com os atributos genéricos do produto (nome: "Camiseta Básica WIN", descrição, etc.).
2. Cada opção específica (Azul, Vermelha, P, M, G) é uma instância da classe **`ProductVariant`**.

Essa classe é essencial para:

- **Organização do Catálogo:** Agrupar produtos semelhantes.
- **Gestão de Estoque:** Controlar o estoque de cada variação de forma independente.
- **Flexibilidade:** Permitir que o cliente escolha as opções que deseja (tamanho, cor, etc.) em uma única página de produto.

Os atributos que você incluiu no diagrama, como **`name`**, **`sku`**, **`price`** e **`stock_quantity`**, são cruciais porque permitem que cada variação tenha seu próprio preço e seu próprio controle de estoque. O atributo **`attributes`** (`{"color": "red", "size": "M"}`) é fundamental para descrever as características de cada variação de forma dinâmica.