**Полная реализация приложения на Laravel с ER-диаграммой, миграциями и документацией**

**1. ER-диаграмма третьей нормальной формы (3NF)**

**Сущности и связи:**

Copy

+---------------+ +-------------+ +------------+

| Customers | | Orders | | Payments |

+---------------+ +-------------+ +------------+

| PK: id |1----N| PK: id |1----N| PK: id |

| name | | FK: customer| | FK: order |

| email (unique)| | order\_date | | amount |

| phone | | total\_amount| | payment\_date|

| created\_at | | status | | method |

+---------------+ +-------------+ +------------+

|

| N

|

+------------+ +------------------+

| OrderItems | | ProductCategory |

+------------+ +------------------+

| PK: id |N----1| PK: id |

| FK: order | | name |

| FK: product| | parent\_id (FK) |

| quantity | +------------------+

| unit\_price | |

+------------+ | 1

| |

| 1 |

| |

+------------+ |

| Products |N---------+

+------------+

| PK: id |

| name |

| description|

| price |

| stock |

| FK: category

+------------+

**2. Реализация в Laravel миграциях**

**Миграции:**

bash

Copy

php artisan make:model Customer -m

php artisan make:model ProductCategory -m

php artisan make:model Product -m

php artisan make:model Order -m

php artisan make:model OrderItem -m

php artisan make:model Payment -m

**customers таблица:**

php

Copy

Schema::create('customers', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->string('phone');

$table->timestamps();

});

**product\_categories таблица:**

php

Copy

Schema::create('product\_categories', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->foreignId('parent\_id')->nullable()->constrained('product\_categories');

$table->timestamps();

});

**products таблица:**

php

Copy

Schema::create('products', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->text('description')->nullable();

$table->decimal('price', 8, 2);

$table->integer('stock');

$table->foreignId('category\_id')->constrained('product\_categories');

$table->timestamps();

});

**orders таблица:**

php

Copy

Schema::create('orders', function (Blueprint $table) {

$table->id();

$table->foreignId('customer\_id')->constrained();

$table->dateTime('order\_date');

$table->decimal('total\_amount', 10, 2);

$table->string('status')->default('pending');

$table->timestamps();

});

**order\_items таблица:**

php

Copy

Schema::create('order\_items', function (Blueprint $table) {

$table->id();

$table->foreignId('order\_id')->constrained();

$table->foreignId('product\_id')->constrained();

$table->integer('quantity');

$table->decimal('unit\_price', 8, 2);

$table->timestamps();

});

**payments таблица:**

php

Copy

Schema::create('payments', function (Blueprint $table) {

$table->id();

$table->foreignId('order\_id')->constrained();

$table->decimal('amount', 10, 2);

$table->dateTime('payment\_date');

$table->string('method');

$table->timestamps();

});

**Калькулирующий запрос (сумма продаж по категориям):**

php

Copy

public function getSalesByCategory()

{

return DB::table('product\_categories as pc')

->select(

'pc.name as category\_name',

DB::raw('SUM(oi.quantity \* oi.unit\_price) as total\_sales'),

DB::raw('COUNT(DISTINCT o.id) as order\_count')

)

->join('products as p', 'p.category\_id', '=', 'pc.id')

->join('order\_items as oi', 'oi.product\_id', '=', 'p.id')

->join('orders as o', 'o.id', '=', 'oi.order\_id')

->where('o.status', 'completed')

->groupBy('pc.id', 'pc.name')

->orderByDesc('total\_sales')

->get();

}

**3. Реализация функционала приложения**

**Модели:**

**Customer.php**

php

Copy

class Customer extends Model

{

protected $fillable = ['name', 'email', 'phone'];

public function orders()

{

return $this->hasMany(Order::class);

}

}

**ProductCategory.php**

php

Copy

class ProductCategory extends Model

{

protected $fillable = ['name', 'parent\_id'];

public function products()

{

return $this->hasMany(Product::class);

}

public function parent()

{

return $this->belongsTo(ProductCategory::class, 'parent\_id');

}

public function children()

{

return $this->hasMany(ProductCategory::class, 'parent\_id');

}

}

**Product.php**

php

Copy

class Product extends Model

{

protected $fillable = ['name', 'description', 'price', 'stock', 'category\_id'];

public function category()

{

return $this->belongsTo(ProductCategory::class);

}

public function orderItems()

{

return $this->hasMany(OrderItem::class);

}

}

**Order.php**

php

Copy

class Order extends Model

{

protected $fillable = ['customer\_id', 'order\_date', 'total\_amount', 'status'];

public function customer()

{

return $this->belongsTo(Customer::class);

}

public function items()

{

return $this->hasMany(OrderItem::class);

}

public function payments()

{

return $this->hasMany(Payment::class);

}

public function calculateTotal()

{

return $this->items->sum(function($item) {

return $item->quantity \* $item->unit\_price;

});

}

}

**OrderItem.php**

php

Copy

class OrderItem extends Model

{

protected $fillable = ['order\_id', 'product\_id', 'quantity', 'unit\_price'];

public function order()

{

return $this->belongsTo(Order::class);

}

public function product()

{

return $this->belongsTo(Product::class);

}

}

**Payment.php**

php

Copy

class Payment extends Model

{

protected $fillable = ['order\_id', 'amount', 'payment\_date', 'method'];

public function order()

{

return $this->belongsTo(Order::class);

}

}

**Контроллеры:**

**OrderController.php (основные методы)**

php

Copy

class OrderController extends Controller

{

public function store(Request $request)

{

$validated = $request->validate([

'customer\_id' => 'required|exists:customers,id',

'items' => 'required|array',

'items.\*.product\_id' => 'required|exists:products,id',

'items.\*.quantity' => 'required|integer|min:1'

]);

DB::transaction(function() use ($validated) {

$order = Order::create([

'customer\_id' => $validated['customer\_id'],

'order\_date' => now(),

'status' => 'pending',

'total\_amount' => 0

]);

$total = 0;

foreach ($validated['items'] as $item) {

$product = Product::find($item['product\_id']);

$orderItem = OrderItem::create([

'order\_id' => $order->id,

'product\_id' => $product->id,

'quantity' => $item['quantity'],

'unit\_price' => $product->price

]);

$total += $orderItem->quantity \* $orderItem->unit\_price;

// Update product stock

$product->decrement('stock', $item['quantity']);

}

$order->update(['total\_amount' => $total]);

});

return response()->json(['message' => 'Order created successfully'], 201);

}

public function addPayment(Request $request, Order $order)

{

$validated = $request->validate([

'amount' => 'required|numeric|min:0.01',

'method' => 'required|string'

]);

$payment = $order->payments()->create([

'amount' => $validated['amount'],

'payment\_date' => now(),

'method' => $validated['method']

]);

// Update order status if fully paid

$paidAmount = $order->payments->sum('amount');

if ($paidAmount >= $order->total\_amount) {

$order->update(['status' => 'paid']);

}

return response()->json($payment, 201);

}

}

**API маршруты (routes/api.php):**

php

Copy

Route::apiResource('customers', CustomerController::class);

Route::apiResource('products', ProductController::class);

Route::apiResource('categories', ProductCategoryController::class);

Route::prefix('orders')->group(function() {

Route::get('/', [OrderController::class, 'index']);

Route::post('/', [OrderController::class, 'store']);

Route::get('/{order}', [OrderController::class, 'show']);

Route::post('/{order}/payments', [OrderController::class, 'addPayment']);

// Reports

Route::get('/reports/sales-by-category', [ReportController::class, 'salesByCategory']);

});

**Документация API**

**1. Customers**

**GET /api/customers** - List all customers  
**POST /api/customers** - Create new customer

json

Copy

{

"name": "John Doe",

"email": "john@example.com",

"phone": "1234567890"

}

**2. Products**

**GET /api/products** - List all products  
**POST /api/products** - Create new product

json

Copy

{

"name": "Laptop",

"description": "High performance laptop",

"price": 999.99,

"stock": 10,

"category\_id": 1

}

**3. Orders**

**POST /api/orders** - Create new order

json

Copy

{

"customer\_id": 1,

"items": [

{

"product\_id": 1,

"quantity": 2

}

]

}

**POST /api/orders/{order}/payments** - Add payment to order

json

Copy

{

"amount": 1999.98,

"method": "credit\_card"

}

**4. Reports**

**GET /api/orders/reports/sales-by-category** - Get sales by category report  
Response:

json

Copy

[

{

"category\_name": "Electronics",

"total\_sales": 5000.00,

"order\_count": 10

}

]

**Инструкция по запуску**

1. Установите зависимости:

bash

Copy

composer install

npm install

1. Настройте .env файл:

bash

Copy

cp .env.example .env

php artisan key:generate

1. Запустите миграции и сидеры:

bash

Copy

php artisan migrate --seed

1. Запустите сервер:

bash

Copy

php artisan serve

1. Для фронтенда (если нужно):

bash

Copy

npm run dev

Приложение будет доступно по адресу: [http://localhost:8000](http://localhost:8000/)

API endpoints доступны с префиксом /api (например: <http://localhost:8000/api/customers>)