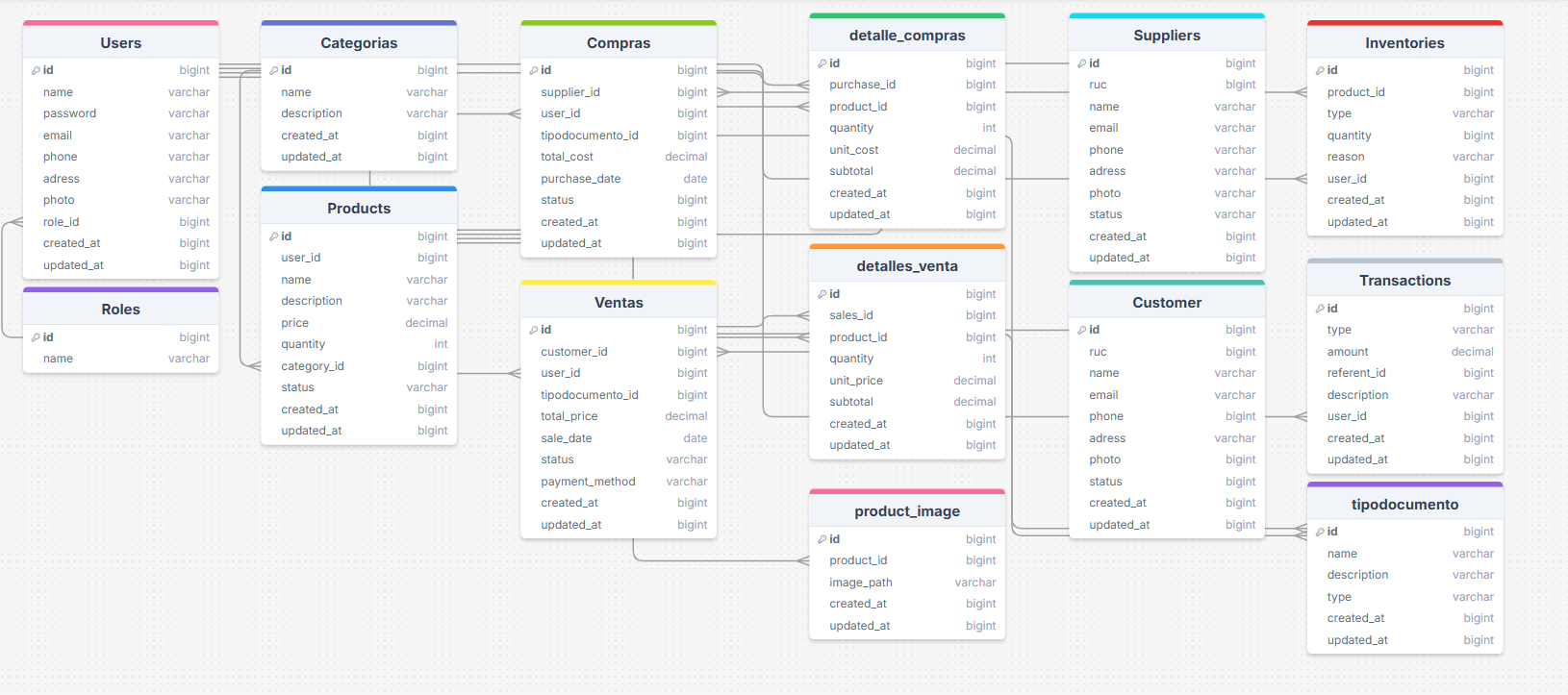
**Desarrolla un Sistema de Compra y Venta con Laravel +ChatGPT**



Diagrama de la base de datos

composer create-project laravel/laravel compraventa “10.\*”

MIGRACIONES

php artisan make:migration create\_roles\_table --create=roles

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

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

Table Users

            $table->string('phone')->nullable();

            $table->string('adress')->nullable();

            $table->string('photo')->nullable();

            $table->foreignId('role\_id')->constrained('roles')->onDelete('cascade');

Tabla Categorias

php artisan make:migration create\_categories\_table --create-categories

table Categories

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

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

Tabla Productos

php artisan make:migration create\_products\_table --create=products

$table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

$table->string('name');

$table->text('description');

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

$table->integer('quantity');

$table->foreignId('category\_id')->constrained('categories')->onDelete('cascade');

$table->enum('status',['available', 'sold', 'archived'])->default('available');

Tabla Produts\_image

php artisan make:migration create\_products\_image\_\_table --create=product\_images

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

            $table->string('image\_path');

Tabla Suppliers

php artisan make:migration create\_suppliers\_table --create=suppliers

 $table->id();

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

            $table->string('name');

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

            $table->string('phone')->nullable();

            $table->text('adress');

            $table->string('photo')->nullable();

            $table->enum('status',['active','inactive'])->default('active');

            $table->timestamps();

Tabla Costumers

php artisan make:migration create\_customers\_table --create=customers

 $table->string('ruc')->unique();

            $table->string('name');

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

            $table->string('phone')->nullable();

            $table->text('adress');

            $table->string('photo')->nullable();

            $table->enum('status',['active','inactive'])->default('active');

Tabla tipodocuemento

php artisan make:migration create\_tipdocumento\_table --create=tipdocumento

$table->string('name');

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

            $table->enum('type',['venta', 'compra']);

Tabla Compras

php artisan make:migration create\_compras\_table --create=compras

 $table->foreignId('supplier\_id')->constrained('suppliers')->onDelete('set null');

            $table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

            $table->foreignId('tipodocumento\_id')->constrained('tipodocumento')->onDelete('cascade');

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

            $table->date('purchase\_date');

            $table->enum('status',['pendient', 'completed', 'canceled'])->default('completed');

Tabla detallecompra

php artisan make:migration create\_detalle\_compras\_table --create=detalle\_compras

 $table->foreignId('purchase\_id')->constrained('compras')->onDelete('cascade'); ;

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

            $table->integer('quantity');

            $table->decimal('unit\_cost', 10, 2);

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

Tabla Ventas

php artisan make:migration create\_ventas\_table --create=ventas

 $table->foreignId('customer\_id')->nullable()->constrained('customers')->onDelete('set null');

            $table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

            $table->foreignId('tipodocumento\_id')->constrained('tipodocumento')->onDelete('cascade');

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

            $table->date('sale\_date');

            $table->enum('status',['pendient', 'completed', 'canceled'])->default('completed');

            $table->enum('payment\_method',['cash', 'credit\_card', 'bank\_transfer'])->default('cash');

Tabla detalle\_venta

php artisan make:migration create\_detalle\_ventas\_table --create=detalle\_ventas

 $table->foreignId('sale\_id')->constrained('ventas')->onDelete('cascade'); ;

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

            $table->integer('quantity');

            $table->decimal('unit\_price', 10, 2);

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

Tabla Inventario

php artisan make:migration create\_inventories\_table --create=inventories

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

            $table->enum('type',['purchase', 'sale', 'adjustment']);

            $table->integer('quantity');

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

            $table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

Tabla transaction

php artisan make:migration create\_transaction\_table --create=transaction

  $table->enum('type',['purchase', 'payment', 'sale', 'adjustment', 'refund']);

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

            $table->bigInteger('reference\_id')->nullable();

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

            $table->foreignId('user\_id')->constrained('users')->onDelete('cascade');

Php artisan migrate

CREACION DE MODELOS

Model Role

php artisan make:model Role

* campos a trabajar

    protected $fillable = [

        'name',

        'description',

    ];

* Relaciones

    public function users()

    {

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

    }

Model User

Campos modificables

    protected $fillable = [

        'name',

        'email',

        'password',

        'phone',

        'adress',

        'photo',

        'role\_id',

    ];

Relaciones

    public function role()

    {

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

    }

Model Category

php artisan make:model Category

campos

    protected $fillable = [

        'name',

        'description',

    ];

Relaciones

   public function products()

    {

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

    }

Model Product

php artisan make:model Product

Campos

    protected $fillable = [

        'user\_id',

        'name',

        'description',

        'price',

        'quantity',

        'category\_id',

        'status',

    ];

Relaciones

    public function category()

    {

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

    }

    public function user()

    {

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

    }

    public function images()

    {

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

    }

Model ProductImage

php artisan make:model ProductImage

    protected $fillable = [

        'product\_id',

        'image\_path',

    ];

    public function product()

    {

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

    }

Model Supplier

php artisan make:model Suppliers

    protected $fillable = [

        'ruc',

        'name',

        'email',

        'image\_path',

        'phone',

        'address',

        'photo',

        'status',

    ];

    public function compras()

    {

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

    }

Model Customer

php artisan make:model Customer

    protected $fillable = [

        'ruc',

        'name',

        'email',

        'phone',

        'address',

        'photo',

        'status',

    ];

    public function ventas()

    {

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

    }

Model Compra

php artisan make:model Compra

    protected $fillable = [

        'supplier\_id',

        'user\_id',

        'tipodocumento\_id',

        'total\_cost',

        'purchase\_date',

        'status',

    ];

    public function supplier()

    {

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

    }

    public function user()

    {

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

    }

    public function tipodocumento()

    {

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

    }

    public function detalles()

    {

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

    }

Model DetalleCompra

php artisan make:model DetalleCompra

    protected $fillable = [

        'purchase\_id',

        'product\_id',

        'quantity',

        'unit\_cost',

        'subtotal',

    ];

    public function compra()

    {

        return $this->belongsTo(Compra::class, 'purchase\_id');

    }

    public function producto()

    {

        return $this->belongsTo(Product::class, 'product\_id');

    }

Model Venta

php artisan make:model Venta

    protected $fillable = [

        'customer\_id',

        'user\_id',

        'tipodocumento\_id',

        'total\_price',

        'sales\_date',

        'status',

        'payment\_method',

    ];

    public function customer()

    {

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

    }

    public function user()

    {

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

    }

    public function tipodocumento()

    {

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

    }

    public function detalles()

    {

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

    }

Model DetalleVenta

php artisan make:model DetalleVenta

    protected $fillable = [

        'sale\_id',

        'product\_id',

        'quantity',

        'unit\_price',

        'subtotal',

    ];

    public function venta()

    {

        return $this->belongsTo(Venta::class, 'sale\_id');

    }

    public function producto()

    {

        return $this->belongsTo(Product::class, 'product\_id');

    }

Model TipoDocumento

php artisan make:model TipoDocumento

    protected $fillable = [

        'name',

        'description',

        'type',

    ];

    public function compras()

    {

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

    }

    public function ventas()

    {

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

    }

Model Inventory

php artisan make:model Inventory

    protected $fillable = [

        'product\_id',

        'quantity',

        'type',

        'reason',

        'user\_id',

    ];

    public function producto()

    {

        return $this->belongsTo(Product::class, 'product\_id');

    }

    public function user()

    {

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

    }

Model Transaction

php artisan make:model Transaction

    protected $fillable = [

        'type',

        'amount',

        'reference\_id',

        'description',

        'user\_id',

    ];

    public function user()

    {

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

    }

FACTORIES Y SEEDER

Role

php artisan make:factory RoleFactory

 protected $model = \App\Models\Role::class;

    public function definition(): array

    {

        return [

            'name' => $this->faker->unique()->word(),

            'description' => $this->faker->unique()->sentence(),

        ];

    }

php artisan make:seeder RoleSeeder

public function run(): void

    {

        $roles = [

            ['name' => 'Administrador', 'description' => 'Administrator'],

            ['name' => 'Vendedor', 'description' => 'Editor'],

            ['name' => 'Comprador', 'description' => 'Viewer'],

            ['name' => 'Mantenedor', 'description' => 'Viewer'],

            ['name' => 'Analista', 'description' => 'Viewer'],

            ['name' => 'Almacen', 'description' => 'Viewer'],

        ];

        foreach ($roles as $role) {

            Role::create($role);

        }

    }

DatabaseSeeder

$this->call(RoleSeeder::class);

php artisan db:seed

User factory - seeder

factory

use App\Models\Role;

use App\Models\User;

 protected $model = User::class;

    public function definition(): array

    {

        return [

            'name' => $this->fake()->name(),

            'email' => $this->fake()->unique()->safeEmail(),

            'password' => Hash::make('123456'),

            'phone' => $this->fake()->phoneNumber(),

            'address' => $this->fake()->address(),

            'photo' => $this->fake()->imageUrl(100,100, 'people'),

            'role\_id' => Role::inRandomOrder()->first()->id,

        ];

    }

seeder

php artisan make:seeder UserSeeder

use App\Models\Role;

use App\Models\User;

use Illuminate\Support\Facades\Hash;

 public function run(): void

    {

        $roles = Role::all();

        foreach ($roles as $role) {

            User::create([

                'name' => $role->name . ' User',

                'email' => strtolower($role->name) . '@example.com',              'email' => $this->fake()->unique()->safeEmail(),

                'password' => Hash::make('123456'),

                'phone' => fake()->phoneNumber(),

                'address' => fake()->address(),

                'photo' => fake()->imageUrl(100,100, 'people'),

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

            ]);

        }

    }

databaseSeeder

       $this->call([

            RoleSeeder::class,

            UserSeeder::class,

        ]);

Php artisan db:seed

Category Seeder

php artisan make:seeder CategorySeeder

importamos

use App\Models\Category;

 public function run(): void

    {

        $categories = [

            ['name' => 'Laptops', 'description' => 'Laptops'],

            ['name' => 'Desktops', 'description' => 'Desktops'],

            ['name' => 'Mobile Phones', 'description' => 'Mobile Phones'],

            ['name' => 'Tablets', 'description' => 'Tablets'],

            ['name' => 'Accessories', 'description' => 'Accessories'],

        ];

        foreach ($categories as $category) {

            Category::create($category);

        }

    }

databaseSeeder

    public function run(): void

    {

        $this->call([

            // RoleSeeder::class,

            // UserSeeder::class,

            CategorySeeder::class,

        ]);

    }

php artisan db:seed

Product Seeder

php artisan make:seeder ProductSeeder

use App\Models\Product;

use App\Models\Category;

use App\Models\User;

 public function run(): void

    {

        $categories = Category::all();

        $users = User::whereHas('role', function ($query) {

            $query->where('name', 'Administrador')->orWhere('name', 'Mantenedor');

        })->get();

        $products = [

            ['name' => 'Laptop', 'description' => 'Laptop', 'price' => 1000, 'quantity' => 10],

            ['name' => 'Desktop', 'description' => 'Desktop', 'price' => 800, 'quantity' => 10],

            ['name' => 'Mobile Phone', 'description' => 'Mobile Phone', 'price' => 500, 'quantity' => 10],

            ['name' => 'Tablet', 'description' => 'Tablet', 'price' => 300, 'quantity' => 10],

            ['name' => 'Headphones', 'description' => 'Headphones', 'price' => 100, 'quantity' => 10],

        ];

        foreach ($products as $product) {

            Product::create([

                'name' => $product['name'],

                'description' => $product['description'],

                'price' => $product['price'],

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

                'category\_id' => $categories->random()->id,

                'user\_id' => $users->random()->id,

                'status' => 'available',

            ]);

        }

    }

databaseSeeder

            ProductSeeder::class,

php artisan db:seed

Customer Factory Seeder

php artisan make:factory CustomerFactory

Factory

    public function definition(): array

    {

        return [

            'name' => $this->fake()->name(),

            'email' => $this->fake()->unique()->safeEmail(),

            'phone' => $this->fake()->phoneNumber(),

            'address' => $this->fake()->address(),

            'photo' => $this->fake()->imageUrl(100,100, 'people'),

            'status' => $this->fake()->randomElement(['active', 'inactive']),

            'ruc' => $this->fake()->unique()->numerify('###########'),

        ];

    }

Seeder

php artisan make:seeder CustumerSeeder

    public function run(): void

    {

        Custumer::factory(10)->create();

    }

databaseSeeder

            CustomerSeeder::class,