App de ventas multiplataforma

Stack propuesto - **Backend**: Laravel 11 (PHP 8.2), MySQL/PostgreSQL, Redis (colas/cache), Laravel Sanctum (auth tokens), Laravel WebSockets (mensajería tiempo real) - **Pagos**: - **PayPal** (REST) - **Cash App Pay** (vía **Square API**) - **Tarjetas crédito/débito (Visa, MasterCard, etc.)** con **Stripe** como pasarela (cumple PCI y reduce alcance) - **Ecobank**: modo **Transferencia bancaria** con conciliación vía **webhook de confirmación** (del PSP/Bank) o conciliación manual + notificaciones. Interfaz abstracta para sustituir por el gateway regional que prefieras (ej. Flutterwave/Paystack/DPO si aplica) - **Web**: Next.js 14 (App Router), TailwindCSS, shadcn/ui - **Móvil**: React Native (Expo), Expo Router - **Mensajería**: canales tipo *conversación* (buyer↔soporte / buyer↔seller), Laravel WebSockets, Echo, Presence/Typing, adjuntos con S3/MinIO - **Infra**: Docker Compose para dev; Nginx, PHP-FPM, Supervisor (colas).

Estructura del monorepo

```
monorepo/
├─ backend/ (Laravel)
├─ web/ (Next.js)
├─ mobile/ (React Native)
├─ docker/ (nginx, php-fpm, mysql, redis, websockets)
└─ .env.example (variables compartidas)
```

Backend (Laravel)

Instalación rápida

```
cd backend
cp .env.example .env
php artisan key:generate
php artisan migrate --seed
php artisan storage:link
php artisan websockets:serve &
php artisan queue:work
```

Modelos y migraciones

- **users** (roles: admin, seller, customer)
- **products** (sku, nombre, descripción, imágenes, precio, stock)
- **carts** / **cart_items**
- **orders** (status: pending, paid, failed, shipped, refunded)
- **payments** (provider, provider_ref, amount, currency, status)
- **conversations** / **messages** (chat en tiempo real)
- **webhooks** (registro de eventos externos)

```
// backend/database/migrations/2024_01_01_000000_create_products_table.php
Schema::create('products', function (Blueprint $table) {
    $table->id();
    $table->string('sku')->unique();
    $table->string('name');
    $table->text('description')->nullable();
    $table->json('images')->nullable();
    $table->unsignedInteger('stock')->default(0);
    $table->unsignedBigInteger('price_cents');
    $table->string('currency', 3)->default('USD');
    $table->timestamps();
});
```

```
// backend/database/migrations/2024_01_01_000100_create_orders_table.php
Schema::create('orders', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained();
    $table->string('number')->unique();
    $table->string('status')->default('pending');
    $table->unsignedBigInteger('total_cents');
    $table->string('currency', 3)->default('USD');
    $table->json('shipping_address');
    $table->json('billing_address')->nullable();
    $table->timestamps();
});
```

```
// backend/database/migrations/2024_01_01_000200_create_payments_table.php
Schema::create('payments', function (Blueprint $table) {
    $table->id();
    $table->foreignId('order_id')->constrained();
    $table->string('provider'); // paypal, cashapp, stripe, ecobank_transfer
    $table->string('provider_ref')->nullable();
    $table->string('status')->default('initiated'); // initiated, authorized,
paid, failed, refunded
    $table->unsignedBigInteger('amount_cents');
    $table->string('currency', 3);
    $table->json('payload')->nullable();
    $table->timestamps();
});
```

```
// backend/database/migrations/
2024_01_01_000300_create_conversations_messages.php
Schema::create('conversations', function (Blueprint $table) {
    $table->id();
    $table->morphs('subject'); // order, store, etc.
    $table->timestamps();
});
```

```
Schema::create('conversation_participants', function (Blueprint $table) {
    $table->id();
    $table->foreignId('conversation_id')->constrained();
    $table->foreignId('user_id')->constrained();
    $table->timestamps();
});
Schema::create('messages', function (Blueprint $table) {
    $table->id();
    $table->foreignId('conversation_id')->constrained();
    $table->foreignId('user_id')->constrained();
    $table->text('body')->nullable();
    $table->string('attachment_path')->nullable();
    $table->timestamps();
});
```

Rutas API

```
// backend/routes/api.php
Route::post('/auth/register', [AuthController::class, 'register']);
Route::post('/auth/login', [AuthController::class, 'login']);

Route::middleware('auth:sanctum')->group(function(){
  Route::get('/me', [AuthController::class, 'me']);
  Route::apiResource('products', ProductController::class);
  Route::get('/cart', [CartController::class, 'show']);
  Route::post('/cart/items', [CartController::class, 'add']);
  Route::delete('/cart/items/{item}', [CartController::class, 'remove']);

  Route::post('/checkout', [CheckoutController::class, 'createOrder']);
  Route::post('/payments/intent', [PaymentController::class,
'createPaymentIntent']);
  Route::post('/payments/{payment}/capture', [PaymentController::class,
'capture']);

  Route::get('/conversations', [ConversationController::class, 'index']);
  Route::post('/conversations', [ConversationController::class, 'store']);
  Route::get('/conversations/{id}/messages', [MessageController::class,
'index']);
  Route::post('/conversations/{id}/messages', [MessageController::class,
'store']);
});

// Webhooks
Route::post('/webhooks/paypal', [WebhookController::class, 'paypal']);
Route::post('/webhooks/stripe', [WebhookController::class, 'stripe']);
Route::post('/webhooks/square', [WebhookController::class, 'square']);
Route::post('/webhooks/ecobank', [WebhookController::class, 'ecobank']);
```

Controladores clave

```
// backend/app/Http/Controllers/PaymentController.php
class PaymentController extends Controller
{
    public function createPaymentIntent(CreatePaymentRequest $request)
    {
        $order = Order::findOrFail($request->order_id);
        $provider = app(\App\Payments\PaymentFactory::class)
            ->for($request->provider); // 'paypal' | 'stripe' | 'cashapp' |
'ecobank_transfer'

        $intent = $provider->createPaymentIntent($order);

        $payment = Payment::create([
            'order_id' => $order->id,
            'provider' => $request->provider,
            'status'   => 'initiated',
            'amount_cents' => $order->total_cents,
            'currency' => $order->currency,
            'payload'  => $intent,
        ]);

        return response()->json(['client' => $intent, 'payment_id' =>
$payment->id]);
    }

    public function capture(Payment $payment)
    {
        $provider = app(\App\Payments\PaymentFactory::class)->for($payment-
>provider);
        $result = $provider->capture($payment);
        $payment->update(['status' => $result['status'], 'provider_ref' =>
$result['id'] ?? null]);
        if ($result['status'] === 'paid') {
            $payment->order->update(['status' => 'paid']);
        }
        return response()->json($result);
    }
}
```

Abstracción de pagos

```
// backend/app/Payments/Contracts/PaymentProvider.php
interface PaymentProvider {
    public function createPaymentIntent(Order $order): array; // retorna
tokens/client_secret/links
    public function capture(Payment $payment): array; // cambia a 'paid' o
'failed'
```

```
    public function handleWebhook(Request $request): void; // idempotente
}
```

```
// backend/app/Payments/PaymentFactory.php
class PaymentFactory {
    public function for(string $provider): PaymentProvider {
        return match($provider){
            'paypal' => app(PayPalProvider::class),
            'stripe' => app(StripeCardProvider::class),
            'cashapp' => app(CashAppPayProvider::class),
            'ecobank_transfer' => app(EcobankTransferProvider::class),
            default => throw new InvalidArgumentException('Proveedor no
soportado')
        };
    }
}
```

Implementación: PayPal

```
// backend/app/Payments/PayPalProvider.php
class PayPalProvider implements PaymentProvider {
    public function createPaymentIntent(Order $order): array {
        // Crear Order en PayPal y devolver approve link / id
        // ... usar PayPal SDK
        return ['id' => 'PAYPAL_ORDER_ID', 'approve_url' => 'https://...'];
    }
    public function capture(Payment $payment): array {
        // Capturar Order en PayPal por ID
        return ['status' => 'paid', 'id' => 'PAYPAL_CAPTURE_ID'];
    }
    public function handleWebhook(Request $request): void {
        // Validar firma y actualizar pago/orden
    }
}
```

Implementación: Tarjetas (Stripe)

```
// backend/app/Payments/StripeCardProvider.php
class StripeCardProvider implements PaymentProvider {
    public function createPaymentIntent(Order $order): array {
        // Crear PaymentIntent para tarjetas
        return ['client_secret' => 'pi_123_secret_abc'];
    }
    public function capture(Payment $payment): array {
        return ['status' => 'paid', 'id' => 'pi_123'];
    }
    public function handleWebhook(Request $request): void {
        // events: payment_intent.succeeded, charge.refunded, etc.
```

```
    }
}
```

Implementación: Cash App Pay (Square)

```
// backend/app/Payments/CashAppPayProvider.php
class CashAppPayProvider implements PaymentProvider {
    public function createPaymentIntent(Order $order): array {
        // Crear Checkout con Square y habilitar Cash App Pay
        return ['checkout_url' => 'https://squareup.com/checkout/...'];
    }
    public function capture(Payment $payment): array {
        // Confirmación vía webhook → marcar como paid
        return ['status' => 'paid', 'id' => 'square_payment_id'];
    }
    public function handleWebhook(Request $request): void {
        // Validar firma de Square
    }
}
```

Implementación: Ecobank (Transferencia)

```
// backend/app/Payments/EcobankTransferProvider.php
class EcobankTransferProvider implements PaymentProvider {
    public function createPaymentIntent(Order $order): array {
        // Generar referencia única y mostrar datos de cuenta
        return [
          'reference' => 'ECO-'.Str::uuid(),
          'instructions' => [
            'bank' => 'Ecobank',
            'account_name' => config('payments.ecobank.account_name'),
            'account_number' => config('payments.ecobank.account_number'),
            'swift' => config('payments.ecobank.swift'),
          ],
        ];
    }
    public function capture(Payment $payment): array {
        // Normalmente se confirma por webhook del PSP o conciliación manual
        return ['status' => 'initiated'];
    }
    public function handleWebhook(Request $request): void {
        // Marcar pago como 'paid' al recibir confirmación del banco/PSP
    }
}
```

Mensajería en tiempo real

- **Paquetes**: `beyondcode/laravel-websockets`, `pusher/pusher-php-server`, `laravel/echo`
- **Eventos**: `MessageSent`, `UserTyping`, `MessageRead`
- **Seguridad**: Canales privados con *guards* de conversación

```
// backend/app/Events/MessageSent.php
class MessageSent implements ShouldBroadcast {
    use SerializesModels;
    public function __construct(public Message $message) {}
    public function broadcastOn(){
        return new PrivateChannel('conversations.'.$this->message->conversation_id);
    }
}
```

Web (Next.js)

Páginas clave

- `/` listado de productos, búsqueda, filtros
- `/product/[slug]` ficha, galería, añadir al carrito
- `/cart` carrito editable
- `/checkout` elegir método de pago: **PayPal / Tarjeta (Visa) / Cash App / Ecobank**
- Widget de **chat** persistente (soporte)

```
// web/app/checkout/page.tsx
export default function Checkout(){
  // obtiene métodos desde /api/payment-methods
  // al elegir Stripe → render Elemento de tarjeta
  // al elegir PayPal → redirigir approve_url
  // Cash App → abrir checkout_url de Square
  // Ecobank → mostrar instrucciones + subir comprobante opcional
  return <div className="container mx-auto p-6">...</div>
}
```

Cliente Echo (chat)

```
// web/lib/echo.ts
import Echo from 'laravel-echo';
export const echo = new Echo({
  broadcaster: 'pusher',
  key: process.env.NEXT_PUBLIC_PUSHER_KEY,
  wsHost: process.env.NEXT_PUBLIC_WS_HOST,
  wsPort: Number(process.env.NEXT_PUBLIC_WS_PORT || 6001),
  forceTLS: false,
```

```
  disableStats: true,
  authEndpoint: process.env.NEXT_PUBLIC_API_URL + '/broadcasting/auth',
});
```

Móvil (React Native / Expo)

- Pantallas: Home, Product, Cart, Checkout, Chat
- Autenticación con Sanctum (cookie-less con tokens)

```
// mobile/app/checkout.tsx
export default function CheckoutScreen(){
  // similar a la web, con opciones de pago adaptadas
  return null;
}
```

Configuración de pagos (.env)

```
# PayPal
PAYPAL_CLIENT_ID=...
PAYPAL_CLIENT_SECRET=...
PAYPAL_WEBHOOK_ID=...

# Stripe
STRIPE_SECRET=...
STRIPE_WEBHOOK_SECRET=...

# Square (Cash App Pay)
SQUARE_ACCESS_TOKEN=...
SQUARE_LOCATION_ID=...
SQUARE_WEBHOOK_SIGNATURE_KEY=...

# Ecobank (transferencia)
ECOBANK_ACCOUNT_NAME=...
ECOBANK_ACCOUNT_NUMBER=...
ECOBANK_SWIFT=...
```

Webhooks

- **/webhooks/paypal**: `CHECKOUT.ORDER.APPROVED`, `PAYMENT.CAPTURE.COMPLETED`
- **/webhooks/stripe**: `payment_intent.succeeded`, `charge.refunded`
- **/webhooks/square**: `payment.updated`
- **/webhooks/ecobank**: endpoint opcional para confirmar transferencias (si hay integración con PSP/banco)

Todos deben ser **idempotentes** y registrar payloads en tabla `webhooks`.

Seguridad y cumplimiento

- Tokens con **Sanctum**
- Validación exhaustiva de webhooks (firmas)
- Productos/órdenes bajo políticas de autorización (Gates/Policies)
- Logs de auditoría para cambios críticos

Docker Compose (dev) – extracto

```
services:
  app:
    build: ./docker/php
    volumes: ["./backend:/var/www/html"]
  nginx:
    image: nginx:alpine
    volumes:
      - ./docker/nginx/conf.d:/etc/nginx/conf.d
      - ./backend/public:/var/www/html/public
    ports: ["8080:80"]
  db:
    image: mysql:8
    environment:
      MYSQL_DATABASE: shop
      MYSQL_USER: shop
      MYSQL_PASSWORD: secret
      MYSQL_ROOT_PASSWORD: secret
    ports: ["3306:3306"]
  redis:
    image: redis:alpine
  websockets:
    image: node:18
    working_dir: /var/www/html
    volumes: ["./backend:/var/www/html"]
    command: ["php", "artisan", "websockets:serve"]
```

Flujo de checkout

1. Usuario crea orden → `/checkout`
2. Elige **proveedor** (PayPal | Tarjeta (Visa) | Cash App | Ecobank)
3. Backend crea **intent** con el proveedor (o instrucciones Ecobank)
4. Front realiza **captura** o espera **webhook** → `order.status = paid`
5. Email/Push de confirmación; stock decrementado; factura

Tareas pendientes para producción

- Completar SDKs reales (PayPal/Stripe/Square) y pruebas en *sandbox*
- Asegurar KYC/Compliance según país y moneda
- Multi-moneda e impuestos
- Panel Admin (productos, pedidos, chats)
- Tests (Pest/PHPUnit) y CI/CD

Notas sobre Ecobank

Como Ecobank no ofrece un SDK público unificado para e-commerce en todos los países, se incluye **proveedor de transferencia bancaria** con referencia única y conciliación por webhook (si tu PSP o el banco lo permite) o validación manual desde backoffice. Si luego eliges un gateway regional compatible con Ecobank, basta con crear un `EcobankGatewayProvider` que implemente `PaymentProvider`.

Licencia y recomendaciones

- Código de ejemplo bajo MIT.
- Para producción, usa **Stripe** para tarjetas (Visa) y **PayPal** para PayPal. Para **Cash App Pay**, habilítalo en Square. Ecobank: transferencia o PSP regional.

  Este documento incluye esqueletos de código listos para copiar/pegar y expandir. Puedo generar los archivos completos del proyecto si lo necesitas.