Plantilla de Documentación de Prompts

Campo	Contenido
ID del Prompt	P004-Fase 4 Qwen
Fecha de Uso	25/04/2025
Modelo de IA	Qwen2.5-coder:32b
Categoría de la Tarea	Frontend/Backend/Base de Datos/Autenticación
Tecnología	HTML/CSS/JavaScript/PHP/SQL

Texto Completo del Prompt	V.1 _Analiza los documentos adjuntos y proporciona una implementación detallada para la Fase 4 de la Tienda Virtual del CMS del Mercado Local San Mateo. En esta fase, se deben abordar los siguientes aspectos:_ 1. **Carrito de Compra**: * Definir la estructura de datos necesaria en base a la base de datos proporcionada. * Implementar un controlador que gestione la adición, eliminación y actualización de productos en el carrito. * Utilizar sesiones para almacenar temporalmente el carrito antes de confirmar la compra. * Proporcionar una vista para mostrar el carrito con opciones de modificación. 2. **Proceso de Checkout**: * Implementar un flujo de compra que incluya validación de datos de envío y método de pago. * Garantizar la verificación de stock antes de la confirmación del pedido. * Manejar múltiples vendedores si aplica, asegurando la correcta asociación de productos a cada pedido. * Proponer una estructura para gestionar cambios en el estado de los pedidos. 3. **Integración de Pasarela de Pago**: * Determinar opciones viables de pasarela de pago considerando el mercado español. * Proporcionar un controlador para gestionar pagos y callbacks de confirmación. * Garantizar la seguridad en la transmisión de datos sensibles. * Proponer una estrategia de gestión de errores en pagos y posibles reembolsos. **Consideraciones adicionales:** * Cumplimiento de normativas españolas de comercio electrónico (IVA, derecho de desistimiento, protección de datos). * Notificaciones para clientes y vendedores sobre cambios en pedidos. * Implementación de un historial de pedidos accesible para los clientes. * Recomendaciones para la optimización de la experiencia de usuario y mejoras futuras. 1ºiteracion: dame la migracion de la tabla orders, además dame las vistas, rutas y funciones de los controladores para poder mostrar el carrito, gestionar el estado de los pedidos, ver el **Historial** de los pedidos, vista del checkout y del pago. **Tabla orders** Registra los pedidos realizados por clientes a vendedores. Campo Tipo Descripción Restricciones id INT Identificador único del pedido PK, AUTO_INCREMENT buyer_id INT ID del usuario cliente FK (users.id), NOT NULL seller_id INT ID del usuario vendedor FK (users.id), NOT NULL total DECIMAL(10,2) Monto total del pedido NOT NULL status ENUM Estado: ‘pendiente’, ‘procesando’, ‘enviado’, ‘entregado’, ‘cancelado’ NOT NULL, DEFAULT ‘pendiente’ payment_method VARCHAR(255) Método de pago utilizado NOT NULL payment_status ENUM Estado del pago: ‘pendiente’, ‘pagado’, ‘reembolsado’ NOT NULL, DEFAULT ‘pendiente’ shipping_address TEXT Dirección de envío NULL created_at TIMESTAMP Fecha de creación del registro NOT NULL updated_at TIMESTAMP Fecha de última actualización NOT NULL **Relaciones:** - Un pedido pertenece a un único cliente (rol cliente o vendedor)(N:1) - Un pedido es atendido por un único vendedor (rol proveedor o vendedor)(N:1) - Un pedido contiene múltiples items (1:N) 2ºiteracion: Modifica la vista para que muestre el carrito con los datos pasados por la función show el controlador CartController, además dame la función show del controlador: ```php @extends('layouts.app') @section('content') **Carrito de Compras** @if (session('success')) {{ session('success') }} @endif @if (count(session()->get('cart', [])) == 0) Tu carrito está vacío. @else @foreach (session()->get('cart', []) as $productId => $quantity) @endforeach **Producto** **Cantidad** **Precio Unitario** **Total** **Acciones** {{ App\Models\Product::find($productId)->name }} @csrf Actualizar {{ App\Models\Product::find($productId)->price }} {{ App\Models\Product::find($productId)->price * $quantity }} Eliminar Ir a Checkout @endif @endsection ``` 3ºiteracion: Dame un botón para ponerlo en la vista de un producto, el boton debe servir para añadir el producto al carrito

4ºiteracion: Junto al botón pon algo para elegir la cantidad, ten en cuenta que el método del controlador es: ```php public function addToCart($productId, $quantity) { // Verificar stock y agregar al carrito en sesión if (session()->has('cart')) { $cart = session()->get('cart'); if (isset($cart[$productId])) { $cart[$productId] += $quantity; } else { $cart[$productId] = $quantity; } } else { $cart = [$productId => $quantity]; } session()->put('cart', $cart); return redirect()->back()->with('success', 'Producto agregado al carrito!'); } ```

5ºiteracion: Asegura en la funcion que las operaciones que se hacen con quantity sean int: ```php public function addToCart($productId, $quantity) { // Verificar stock y agregar al carrito en sesión if (session()->has('cart')) { $cart = session()->get('cart'); if (isset($cart[$productId])) { $cart[$productId] += $quantity; } else { $cart[$productId] = $quantity; } } else { $cart = [$productId => $quantity]; } session()->put('cart', $cart); return redirect()->back()->with('success', 'Producto agregado al carrito!'); } ```

6ºiteracion: Asegura que se pueda actualizar los productos del carro: **Funcion del CartController** ```php public function updateCart(Request $request) { if ($request->quantity <= 0) { return redirect()->route('show.cart')->with('error', 'Cantidad debe ser mayor a cero.'); } if ($request->id && $request->quantity) { $cart = session()->get('cart'); $cart[$request->id]["quantity"] = intval($request->quantity); session()->put('cart', $cart); return redirect()->route('show.cart')->with('success', 'Carrito actualizado.'); } } ``` **Vista** ```php @extends('layouts.app') @section('content')

Carrito de Compras

@if (session('success'))
{{ session('success') }}
@endif @if (count($productsInCart) == 0)

Tu carrito está vacío.

@else @foreach ($productsInCart as $productId => $item) @endforeach

Producto	Cantidad	Precio Unitario	Total	Acciones
{{ $item['product']->name }}	@csrf Actualizar	{{ $item['product']->price }}	{{ $item['product']->price * $item['quantity'] }}	Eliminar

Ir a Checkout @endif @endsection ``` **Ruta** ```php Route::post('/cart/update', [CartController::class, 'updateCart'])->name('update.cart'); ```

7ºiteracion: Para el checkout, como un cliente puede comprar productos de varios vendedores se generará una orden por cada vendedor con los productos que cogió el cliente que pertenezcan al vendedor. Por ejemplo si el cliente tiene en el carrito tres productos del vendedor 1 y uno del vendedor 2, al comprar se generara dos ordenes, una con los productos del vendedor 1 con sus productos y otro con el producto del vendedor 2, aunque el cliente pague todo junto. Modifica la vista y controlador para cumplir con lo anterior: **Controlador** ```php validate(['name' => 'required|string', 'email' => 'required|email', 'shipping_address' => 'required|string']); $cart = session()->get('cart'); if (!$cart) { return redirect()->route('show.cart')->with('error', 'Tu carrito está vacío.'); } $total = 0; foreach ($cart as $item) { $total += $item['quantity'] * $item['price']; } // Aquí debes obtener el ID del comprador y vendedor $buyerId = auth()->user()->id; // Suponiendo que el usuario autenticado es el comprador $sellerId = 1; // Debes definir cómo obtendrás el ID del vendedor $order = Order::create(['buyer_id' => $buyerId, 'seller_id' => $sellerId, 'total' => $total, 'status' => 'pendiente', 'payment_method' => 'tarjeta', // Debes definir cómo obtendrás el método de pago 'payment_status' => 'pendiente', 'shipping_address' => $validatedData['shipping_address']]); session()->forget('cart'); return redirect()->route('order.show', $order->id)->with('success', 'Pedido procesado exitosamente.'); } } ``` **Vista** ```php @extends('layouts.app') @section('content')

Checkout

@if (session('success'))
{{ session('success') }}
@endif @if (count(session()->get('cart', [])) == 0)

Tu carrito está vacío.

@else
@csrf Nombre Email Dirección de Envío Procesar Pago
@endif @endsection ```

8ºiteracion: la funcion process del controlador del checkout da error: **Error** **Trying to access array offset on value of type int** **Linea** ```php 'quantity' => $item['quantity'], ```

9ºiteracion: modifica cart.blade.php para que reciba los datos del controlador

10ºiteracion: modifica el index de OrderController para que solo cargue las ordenes que le hayan hecho (seller_id) o pedido (buyer_id) el usuario autenticado: ```php public function index() { $orders = Order::with('client', 'vendor')->get(); return view('orders.index', compact('orders')); } ```

12ºiteracion:modifica el controlador y vista del pago para que permita pagar mediante stripe y cuando se pague cambie el estado de la orden a pagado: **Controlador** ```php validate(['payment_method' => 'required|string']); $order = Order::findOrFail($orderId); // Aquí debes implementar la lógica del procesamiento de pago // Por ahora solo actualizamos el método de pago y el estado del pago $order->update(['payment_method' => $validatedData['payment_method'], 'payment_status' => 'pagado']); return redirect()->route('order.show', $orderId)->with('success', 'Pago procesado exitosamente.'); } } ``` **vista** ```php @extends('layouts.app') @section('content')

Pago

@if (session('success'))
{{ session('success') }}
@endif
@csrf Método de Pago Tarjeta PayPal Procesar Pago
@endsection ```

13ºiteracion: añade a la vista del historial de pedidos un boton a la pasarela de pago a cada pedido en el que el usuario registrado sea el comprador(buyer_id) y el status del pago sea pendiente: ```php @extends('layouts.app') @section('content')

Pedidos

@if (session('success'))
{{ session('success') }}
@endif @foreach ($orders as $order) @endforeach

ID	Comprador	Vendedor	Total	Estado	Método de Pago	Estatus del Pago	Dirección de Envío	Acciones
{{ $order->id }}	{{ $order->client->name }}	{{ $order->vendor->name }}	{{ $order->total }}	{{ $order->status }}	{{ $order->payment_method }}	{{ $order->payment_status }}	{{ $order->shipping_address }}	Ver @if (auth()->id() === $order->vendor->id) Editar @csrf @method('DELETE') Eliminar @endif

@endsection ```

14ºiteracion: al pulsar el boton de pagar me sale el mensaje: The stripe token field is required.

V.1.1 **Objetivo del ejercicio:**
Analizar, depurar y optimizar una implementación recursiva de un algoritmo, aplicando criterios de eficiencia computacional y buenas prácticas de programación.

Instrucciones detalladas:

1. **Análisis inicial del código:**
    - Explica en detalle qué hace el algoritmo actual.
    - Determina su complejidad temporal y espacial utilizando notación Big O.
    - Identifica cuellos de botella, redundancias y oportunidades de optimización.

2. **Detección y corrección de errores:**
    - Lista y explica errores lógicos, sintácticos o de estilo.
    - Describe cómo estos errores afectan al resultado o rendimiento.
    - Aplica correcciones documentadas (comentarios inline).

3. **Optimización de código:**
    - Refactoriza el código para mejorar su eficiencia sin cambiar su funcionalidad.
    - Compara la versión original y la optimizada en términos de tiempo de ejecución, uso de memoria y legibilidad.
    - Incluye técnicas como *memoization*, poda de ramas, o reescritura iterativa si es adecuado.

4. **Validación del rendimiento:**
    - Diseña e implementa un conjunto de pruebas unitarias que cubran casos típicos, extremos y de error.
    - Presenta una tabla comparativa del rendimiento de ambas versiones.
    - Justifica por qué la nueva versión es superior.

5. **Documentación:**
    - Escribe una memoria técnica que incluya:
        - Objetivos del ejercicio
        - Metodología seguida
        - Análisis antes y después de la optimización
        - Decisiones técnicas tomadas y su justificación
        - Conclusiones y posibles mejoras futuras

Formato de entrega:

- Código fuente en Python (.py)
- Informe técnico en PDF (máximo 4 páginas)
- Archivo .zip con pruebas automatizadas si aplica

Criterios de evaluación:

- Corrección funcional del algoritmo
- Calidad de la optimización
- Claridad y solidez del análisis
- Estructura y claridad de la documentación
- Profesionalismo en la presentación de resultados

Long. del Prompt	V.1 sin iteraciones: 1700 con iteraciones: 10028	V.1.1 1810
Documentos incluidos	Sí, específicos de la base de datos y especificaciones	
Ejemplos Incluidos	Sí, mensajes de errores a solucionar y código a modificar	
Calidad de la Respuesta	V.1 5	V.1.1 5
P. Ponderada	V.1 4,5	V.1.1 5
Prob. de Código	manejo de propiedades	
Iteraciones	13	
Mejoras Recomendadas	Mayor especificación para evitar iteraciones	
Resultados de aprendizaje vinculados	V1 R1, R3, R4: Comprensión de relaciones y tecnologías, diseño e implementación de soluciones. R5: Interés por la programación aplicada. R6, R10, R15: Comunicación escrita, redacción de memoria técnica, autonomía.	
	V1.1 **R1, R3, R4:** Análisis de relaciones, uso de tecnologías, diseño y validación de servicios. **R5, R6, R10, R15:** Interés, comunicación escrita, elaboración de memoria técnica, autonomía.	
Competencias relacionadas	V1 CB2, CB3, CB5: Aplicación profesional del conocimiento, juicio crítico, aprendizaje autónomo. CG5: Programación y verificación de aplicaciones. CT1, CT2: Comunicación efectiva y cooperación profesional. CETM6, CETM7: Diseño de arquitecturas y programación de servicios telemáticos.	
	V1.1 **CB3, CB4, CB5:** Interpretación de datos, comunicación efectiva, aprendizaje autónomo. **CG5:** Diseño y verificación de aplicaciones. **CT1, CT2, CT3:** Comunicación profesional, colaboración, mejora continua. **CETM6, CETM7:** Arquitectura y programación de sistemas distribuidos.	
Objetivos de la asignatura	V1 OBJ-2: Desarrollo e implementación de sistemas web o similares. OBJ-4: Desarrollo de conceptos y herramientas de redes sociales (por analogía metodológica). OBJ-5: Conceptos básicos de estructuras tecnológicas (aplicado aquí en forma de búsqueda de eficiencia en algoritmos).	
	V1.1 **OBJ-2:** Diseño e implementación de sistemas completos. ● **OBJ-4:** Aplicación de herramientas de desarrollo en sistemas funcionales. ● **OBJ-5:** Enfoque semántico y técnico en programación avanzada.	

Tabla de Comparación de Versiones de Prompt

Versión	Camb. Clave	Cla.	Esp.	Prec. Téc.	Ctx.	Estruc.	Cal. Cód.	Punt. Pond.	Mej.
1.0	Inicial	4	4	5	4	5	5	4,5	-
1.1	Se agregaron especificaciones técnicas	5	5	5	5	5	5	5	0,5

Leyenda de Abreviaturas

- **Camb. Clave** = Cambios Clave
- **Cla.** = Claridad
- **Esp.** = Especificidad
- **Prec. Téc.** = Precisión Técnica
- **Ctx.** = Contexto
- **Estruc.** = Estructura
- **Cal. Cód.** = Calidad del Código
- **Punt. Pond.** = Puntuación Ponderada
- **Mej.** = Mejora