Plantilla de Documentación de Prompts

Campo	Contenido
ID del Prompt	P006-Fase 6 Qwen
Fecha de Uso	23/04/2025
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 Objetivo**:** Implementar la fase 6 del proyecto, centrada en integraciones API y actualización dinámica de la interfaz con AJAX en Laravel. ### Requerimientos Técnicos 1. Desarrollo de API interna * Crear un controlador en Laravel (ApiController) que proporcione los siguientes endpoints: * GET /api/productos/destacados: Obtener los productos destacados con relación a su puesto de venta. * GET /api/puestos/cercanos: Listar los puestos de venta cercanos, ordenados alfabéticamente. * GET /api/productos/buscar?query=&category=: Permitir búsqueda por nombre y categoría de productos. * GET /api/mercado/estadisticas: Devolver estadísticas del mercado (cantidad de vendedores, productos y puestos). * Implementar los modelos y relaciones necesarios en Laravel para estas funcionalidades. * Asegurar que las respuestas sean en formato JSON estructurado correctamente. 2. Integración con API de Clima (OpenWeatherMap) * Implementar un método en ApiController (GET /api/clima) que recupere los datos climáticos de San Mateo, Gran Canaria. * Utilizar la API de OpenWeatherMap con una clave de API configurable en .env. * Implementar caché en Laravel para almacenar los datos por 30 minutos y reducir llamadas innecesarias. * El resultado debe formatearse en JSON, incluyendo temperatura, condición climática, humedad y velocidad del viento. 3. Implementación de AJAX para actualización dinámica * Implementar en JavaScript (jQuery) las siguientes funcionalidades con AJAX: * Widget de clima: Recuperar datos desde /api/clima y actualizar dinámicamente la interfaz sin recargar la página. * Carga de productos destacados: Cargar dinámicamente los productos destacados desde /api/productos/destacados. * Búsqueda de productos: Implementar un formulario de búsqueda interactivo que use AJAX para consultar /api/productos/buscar. * Botón "Añadir al carrito": Implementar eventos dinámicos en botones de productos destacados y buscados para enviar la información a /carrito/agregar sin recargar la página. * Asegurar que no haya conflictos entre los scripts de featured-products.js y product-search.js. * Manejo adecuado de errores y estados de carga en la interfaz. 4. Estructura de Vistas (Blade Templates en Laravel) * app.blade.php (Layout principal): * Debe incluir la carga correcta de Bootstrap, jQuery y otros scripts de terceros en el orden adecuado. * Mantener una estructura limpia sin carga redundante de librerías. * home.blade.php (Página de inicio): * Incluir el widget de clima, la sección de productos destacados y el formulario de búsqueda de productos en la misma vista. * Asegurar que el indicador de carga solo aparezca cuando una búsqueda esté activa. * Solo cargar featured-products.js y product-search.js cuando la página de inicio esté activa. ### Consideraciones Adicionales * Depuración: Incluir mensajes de consola (console.log) en cada función AJAX para facilitar la identificación de errores. * Seguridad: Utilizar tokens CSRF en las solicitudes POST. * Optimización: Minimizar el número de llamadas API innecesarias utilizando caché y eventos bien gestionados en JavaScript. 1ºiteraciones: Modifica las funciones del controlador de la api teniendo en cuenta que la estructuras de las tablas productos , puestos (stands) y usuarios: **funciones del controlador** ```php public function getProductosDestacados() { // Supongamos que tenemos una relación entre Productos y Puestos de venta $productos = DB::table('productos') ->join('puestos', 'productos.puesto_id', '=', 'puestos.id') ->select('productos.*', 'puestos.nombre as puesto_nombre') ->where('productos.destacado', true) ->get(); return response()->json($productos); } public function getPuestosCercanos() { $puestos = DB::table('puestos')->orderBy('nombre')->get(); return response()->json($puestos); } public function buscarProductos(Request $request) { $query = $request->input('query'); $category = $request->input('category'); $productos = DB::table('productos') ->when($query, function ($q) use ($query) { return $q->where(function ($qq) use ($query) {

$qq->where('nombre', 'like', '%' . $query . '%') ->orWhere('descripcion', 'like', '%' . $query . '%'); }); }) ->when($category, function ($q) use ($category) { return $q->where('categoria_id', $category); }) ->get(); return response()->json($productos); } public function getMercadoEstadisticas() { $vendedores = DB::table('vendedores')->count(); $productos = DB::table('productos')->count(); $puestos = DB::table('puestos')->count(); return response()->json(['vendedores' => $vendedores, 'productos' => $productos, 'puestos' => $puestos]); } ``` Tablas de datos **Tabla users** Esta tabla almacena la información de todos los usuarios del sistema, independientemente de su rol. Campo Tipo Descripción Restricciones id INT Identificador único del usuario PK, AUTO_INCREMENT name VARCHAR(255) Nombre completo del usuario NOT NULL email VARCHAR(255) Correo electrónico UNIQUE, NOT NULL password VARCHAR(255) Contraseña encriptada NOT NULL phone VARCHAR(20) Número de teléfono NULL profile_picture VARCHAR(255) Ruta a la imagen de perfil NULL role ENUM Rol del usuario: ‘cliente’, ‘vendedor’, ‘proveedor’ NOT NULL remember_token VARCHAR(100) Token para “recordar sesión” NULL created_at TIMESTAMP Fecha de creación del registro NOT NULL updated_at TIMESTAMP Fecha de última actualización NOT NULL **Relaciones:** - Un usuario no cliente puede tener un puesto (stand)(1:0/1) - Un usuario no cliente puede crear múltiples productos (1:N) - Un usuario cliente puede realizar múltiples pedidos (1:N) - Un usuario vendedor puede recibir múltiples pedidos (1:N) - Un usuario proveedor puede realizar múltiples ventas a vendedores (1:N) - Un usuario vendedor puede realizar múltiples compras a proveedores (1:N) - Un usuario puede crear múltiples foros (1:N) - Un usuario puede publicar múltiples temas (1:N) - Un usuario puede escribir múltiples comentarios (1:N) - Un usuario puede enviar y recibir múltiples mensajes (1:N) ### **2.2 Tabla stands** Almacena la información de los puestos que gestionan los vendedores en el mercado. Campo Tipo Descripción Restricciones id INT Identificador único del puesto PK, AUTO_INCREMENT user_id INT ID del usuario vendedor propietario FK (users.id), NOT NULL name VARCHAR(255) Nombre del puesto NOT NULL description TEXT Descripción detallada del puesto NULL location VARCHAR(255) Ubicación física dentro del mercado NOT NULL category VARCHAR(255) Categoría del puesto (ej. frutas, artesanía) NULL stand_picture VARCHAR(255) Ruta a la imagen de perfil NULL created_at TIMESTAMP Fecha de creación del registro NOT NULL updated_at TIMESTAMP Fecha de última actualización NOT NULL **Relaciones:** - Un puesto pertenece a un único usuario vendedor/proveedor (N:1) - Un puesto puede tener múltiples productos (1:N) ### **2.3 Tabla products** Contiene todos los productos disponibles en el sistema, tanto de vendedores como de proveedores. Campo Tipo Descripción Restricciones id INT Identificador único del producto PK, AUTO_INCREMENT user_id INT ID del usuario que creó el producto FK (users.id), NOT NULL stand_id INT ID del puesto al que pertenece (si aplica) FK (stands.id), NULL name VARCHAR(255) Nombre del producto NOT NULL description TEXT Descripción detallada del producto NULL price DECIMAL(8,2) Precio del producto NOT NULL stock INT Cantidad disponible NOT NULL, DEFAULT 0 image VARCHAR(255) Ruta a la imagen del producto NULL category VARCHAR(255) Categoría del producto NULL is_available BOOLEAN Indica si el producto está disponible NOT NULL, DEFAULT TRUE created_at TIMESTAMP Fecha de creación del registro NOT NULL updated_at TIMESTAMP Fecha de última actualización NOT NULL **Relaciones:** - Un producto pertenece a un único usuario (N:1) - Un producto puede pertenecer a un puesto (N:0/1) - Un producto puede estar en múltiples items de pedido (1:N) - Un producto puede estar en múltiples items de venta de proveedor (1:N)

2ºiteracion: Añade a la vista home los elementos que interactuan con las apis: clima, productos
destacasdos, busqueda de productos y estadisticas
Vista home
```php
@extends('layouts.app')
@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Dashboard') }}</div>
```

```
            <div class="card-body">
                @if (session('status'))
                    <div class="alert alert-success" role="alert">
                        {{ session('status') }}
                    </div>
                @endif
                {{ __('You are logged in!') }}
            </div>
        </div>
    </div>
</div>
@endsection
```

Vista con los elementos de api

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Proyecto Fase 6</title>
                                                                                <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <div class="container mt-5">
        <h1>Proyecto Fase 6</h1>
        <!-- Clima -->
        <div id="clima" class="mb-4">
            <h2>Clima en San Mateo</h2>
            <p id="temperatura"></p>
            <p id="descripcion"></p>
            <p>Humedad: <span id="humedad"></span>%</p>
            <p>Viento: <span id="viento"></span> m/s</p>
        </div>
        <!-- Productos Destacados -->
        <h2>Productos Destacados</h2>
        <ul id="productos-destacados"></ul>
        <!-- Puestos Cercanos -->
        <h2>Puestos Cercanos</h2>
        <ul id="puestos-cercanos"></ul>
        <!-- Búsqueda de Productos -->
        <form id="busqueda-form" class="mb-4">
            <div class="input-group mb-3">
                    <input type="text" id="query" name="query" class="form-control"
placeholder="Buscar
productos...">
                <button class="btn btn-primary">Buscar</button>
            </div>
        </form>
        <ul id="resultado-busqueda"></ul>
    </div>
    <script>
        $(document).ready(function() {
            // Obtener clima
            $.get('/api/clima', function(data) {
                $('#temperatura').text('Temperatura: ' + data.temperatura + '°C');
                $('#descripcion').text('Descripción: ' + data.descripcion);
                $('#humedad').text(data.humedad);
                $('#viento').text(data.viento);
            });
            // Obtener productos destacados
            $.get('/api/productos/destacados', function(data) {
                var ul = $('#productos-destacados');
                data.forEach(function(producto) {
```

	ul.append(' ' + producto.nombre + ' - Puesto: ' + producto.puesto_nombre + ' '); }); }); // Obtener puestos cercanos $.get('/api/puestos/cercanos', function(data) { var ul = $('#puestos-cercanos'); data.forEach(function(puesto) { ul.append(' ' + puesto.nombre + ' '); }); }); // Buscar productos $('#busqueda-form').on('submit', function(e) { e.preventDefault(); var query = $('#query').val(); $.get('/api/productos/buscar', { query: query }, function(data) { var ul = $('#resultado-busqueda'); ul.empty(); data.forEach(function(producto) { ul.append(' ' + producto.nombre + ' '); }); }); }); }); </script> </body> </html> 3ºiteracion: No aparecen los datos de las apis en la vista y en la consola sale: jQuery.Deferred exception: $.get is not a function TypeError: $.get is not a function at HTMLDocument. (http://mercado_local_qwen.test/home?query=Puesto+1:99:15) at e (https://code.jquery.com/jquery-3.6.0.min.js:2:30038) at https://code.jquery.com/jquery-3.6.0.min.js:2:30340 undefined jquery-3.6.0.min.js:2 Uncaught TypeError: $.get is not a function at HTMLDocument. (home?query=Puesto+1:99:15) at e (jquery-3.6.0.min.js:2:30038) at jquery-3.6.0.min.js:2:30340 (anonymous)@home?query=Puesto+1:99e@jquery-3.6.0.min.js:2(anonymous)@jquery-3.6.0.m in.js:2**setTimeout**(anonymous)@jquery-3.6.0.min.js:2c@jquery-3.6.0.min.js:2fireWith@jquer y-3.6.0.min.js:2fire@jquery-3.6.0.min.js:2c@jquery-3.6.0.min.js:2fireWith@jquery-3.6.0.min.js:2r eady@jquery-3.6.0.min.js:2B@jquery-3.6.0.min.js:2 4ºiteracion: Al parecer era un conflicto de dependencias, pero ahora sale en la consola: home?query=Puesto+1:98 Uncaught ReferenceError: $ is not defined at home?query=Puesto+1:98:9 (anonymous)@home?query=Puesto+1:98 señalando $(document).ready(function() { 5ºiteracion: ahora da error: Uncaught TypeError: $.get is not a function at HTMLDocument. (home?query=Puesto+1:101:15) at e (jquery-3.6.0.min.js:2:30038) at jquery-3.6.0.min.js:2:30340 señalando $.get('/api/clima', function(data) { 6ºiteracion: Ya lo arregle, añade también el script para ver las estadísticas del mercado: ```php @extends('layouts.app') @section('content') Proyecto Fase 6 <!-- Clima --> Clima en San Mateo

```
        <p id="descripcion"></p>
        <p>Humedad: <span id="humedad"></span>%</p>
        <p>Viento: <span id="viento"></span> m/s</p>
    </div>
    <!-- Productos Destacados -->
    <h2>Productos Destacados</h2>
    <ul id="productos-destacados"></ul>
    <!-- Puestos Cercanos -->
    <h2>Puestos Cercanos</h2>
    <ul id="puestos-cercanos"></ul>
    <!-- Búsqueda de Productos -->
    <form id="busqueda-form" class="mb-4">
        <div class="input-group mb-3">
                <input type="text" id="query" name="query" class="form-control"
placeholder="Buscar
productos...">
            <button class="btn btn-primary">Buscar</button>
        </div>
    </form>
    <ul id="resultado-busqueda"></ul>
    <!-- Estadísticas del Mercado -->
    <h2>Estadísticas del Mercado</h2>
    <div id="estadisticas">
        <p>Vendedores: <span id="vendedores"></span></p>
        <p>Productos: <span id="productos"></span></p>
        <p>Puestos: <span id="puestos"></span></p>
    </div>
    <!-- Incluir jQuery directamente -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
        $(document).ready(function() {
            // Obtener clima
            $.get('/api/clima', function(data) {
                $('#temperatura').text('Temperatura: ' + data.temperatura + '°C');
                $('#descripcion').text('Descripción: ' + data.descripcion);
                $('#humedad').text(data.humedad);
                $('#viento').text(data.viento);
            });
            // Obtener productos destacados
            $.get('/api/productos/destacados', function(data) {
                var ul = $('#productos-destacados');
                data.forEach(function(producto) {
                                ul.append('<li>' + producto.name + ' - Puesto: ' +
producto.stand_nombre + '</li>');
                });
            });
            // Obtener puestos cercanos
            $.get('/api/puestos/cercanos', function(data) {
                var ul = $('#puestos-cercanos');
                data.forEach(function(puesto) {
                    ul.append('<li>' + puesto.name + '</li>');
                });
            });
            // Buscar productos
            $('#busqueda-form').on('submit', function(e) {
                e.preventDefault();
                var query = $('#query').val();
                $.get('/api/productos/buscar', { query: query }, function(data) {
                    var ul = $('#resultado-busqueda');
                    ul.empty();
                    data.forEach(function(producto) {
                        ul.append('<li>' + producto.name + '</li>');
                    });
                });
            });
            // Obtener estadísticas del mercado
            $.get('/api/mercado-estadisticas', function(data) {
```

```
            $('#vendedores').text(data.vendedores);
            $('#productos').text(data.productos);
            $('#puestos').text(data.puestos);
        });
    });
  </script>
</div>
@endsection
```

7ºiteracion: Modificame la funcion del controlador de productos para mostrar los productos de vendedores
si el usuario tiene rol de cliente, en caso de tener rol de vendedor o proveedor que se muestren
productos de vendedores y proveedores:

```php
public function verproductos()
  {
            if(auth()->user()->role === 'vendedor' || auth()->user()->role === 'proveedor'){
                $owners = User::where('role', 'vendedor')->orWhere('role', 'proveedor')->get();
        $products = Product::where('user_id', $owners->id)->orWhere('role', 'proveedor')->get()->id;
        }else{
         $owners = User::where('role', 'vendedor')->get();
         $stands = Product::where('user_id', $owners->id)->get();
        }
        return view('products.index', compact('products'));
  }
```

8ºiteracion: Teniendo en cuenta la estructura de la tabla de productos y puestos, añade a las vistas index
los elementos necesarios para filtrar y buscar su correspondiente elemento, en el index de
productos buscar y filtrar productos y en el de puestos los puestos:
Estructura de las tablas
Productos

```php
$table->id();
        $table->string('name');
        $table->string('description')->nullable();
        $table->decimal('price', 8, 2)->nullable(false);
        $table->integer('stock')->default(0)->nullable(false);
        $table->string('image')->nullable();
        $table->string('category')->nullable();
        $table->boolean('is_available')->default(true)->nullable(false);
        $table->timestamp('created_at')->useCurrent();

$table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
        // Relaciones
        $table->foreignId('user_id')->constrained('users')->onDelete('cascade');

$table->foreignId('stand_id')->nullable()->constrained('stands')->onDelete('set null');
```

Puesto

```php
 $table->id();
      $table->string('name')->nullable(false);
      $table->text('description')->nullable();
      $table->string('location')->nullable(false);
      $table->timestamp('created_at')->useCurrent();
      $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();

      // Relaciones
      $table->foreignId('user_id')->constrained('users')->onDelete('cascade');
```

Vistas
index productos
```php
@extends('layouts.app')
@section('content')
<div class="container">
    <h1>Products</h1>
    @if (!is_null(auth()->user()->stand) && (auth()->user()->role === 'vendedor' ||
auth()->user()->role === 'proveedor'))
     <a href="{{ route('products.create') }}" class="btn btn-primary mb-3">Create
New Product</a>
            @elseif(is_null(auth()->user()->stand) && (auth()->user()->role ===
'vendedor' ||
auth()->user()->role === 'proveedor'))
    <p>Crea un puesto para poder añadir productos</p>
       <a href="{{ route('stands.create') }}" class="btn btn-primary mb-3">Crear
puesto</a>
    @endif
    @if(session()->has('success'))
       <div class="alert alert-success">{{ session()->get('success') }}</div>
    @endif
    <table class="table">
       <thead>
          <tr>
             <th>ID</th>
             <th>Name</th>
             <th>Price</th>
             <th>Stock</th>
             <th>Available</th>
             <th>Actions</th>
          </tr>
       </thead>
       <tbody>
          @foreach($products as $product)
          <tr>
             <td>{{ $product->id }}</td>
             <td>{{ $product->name }}</td>
             <td>${{ number_format($product->price, 2) }}</td>
             <td>{{ $product->stock }}</td>
             <td>{{ $product->is_available ? 'Yes' : 'No' }}</td>
             <td>
                          <a href="{{ route('products.show', $product) }}" class="btn
btn-info">Show</a>
                            <a href="{{ route('products.edit', $product) }}" class="btn
btn-primary">Edit</a>
                              <form action="{{ route('products.destroy', $product) }}"
method="POST"
style="display:inline-block;">
                    @csrf
                    @method('DELETE')
                        <button type="submit" class="btn btn-danger" onclick="return
confirm('Are you
sure?')">Delete</button>
                </form>
             </td>
          </tr>
          @endforeach
       </tbody>
    </table>
</div>
@endsection
```
index puestos
```php
@extends('layouts.app')
@section('content')
<div class="container">
```

```
    <h1>Stands</h1>
    @if (is_null(auth()->user()->stand) && (auth()->user()->role === 'vendedor' ||
auth()->user()->role === 'proveedor'))
    <p>Crea un puesto para poder añadir productos</p>
        <a href="{{ route('stands.create') }}" class="btn btn-primary mb-3">Crear
puesto</a>
    @endif
    @if(session()->has('success'))
        <div class="alert alert-success">{{ session()->get('success') }}</div>
    @endif
    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Location</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            @foreach($stands as $stand)
            <tr>
                <td>{{ $stand->id }}</td>
                <td>{{ $stand->name }}</td>
                <td>{{ $stand->location }}</td>
                <td>
                                <a href="{{ route('stands.show', $stand) }}" class="btn
btn-info">Show</a>
                    @if (auth()->id() === $stand->user_id)
                                    <a href="{{ route('stands.edit', $stand) }}" class="btn
btn-primary">Edit</a>
                                        <form action="{{ route('stands.destroy', $stand) }}"
method="POST"
style="display:inline-block;">
                        @csrf
                        @method('DELETE')
                            <button type="submit" class="btn btn-danger" onclick="return
confirm('Are you
sure?')">Delete</button>
                    </form>
                    @endif

                </td>
            </tr>
            @endforeach
        </tbody>
    </table>
</div>
@endsection
```

9ºiteracion: Modifica la funcion para que los usuarios con rol de clientes solo vean los puestos de los que
tienen rol de vendedor, y los usuarios con rol de vendedor o proveedor los puestos de
vendedores y proveedores:
Funcion
```php
public function verpuestos(Request $request)
    {
        $user = auth()->user();
        if ($user->role === 'vendedor' || $user->role === 'proveedor') {
            // Obtener IDs de vendedores y proveedores
            $owners = User::whereIn('role', ['vendedor', 'proveedor'])->pluck('id');

            // Obtener productos de vendedores y proveedores
            $stands = Stand::whereIn('user_id', $owners);
```

```
        } else {
            // Obtener IDs de vendedores
            $owners = User::where('role', 'vendedor')->pluck('id');

            // Obtener productos de vendedores
            $stands = Stand::whereIn('user_id', $owners);
        }
        // Obtener parámetro de búsqueda
        $query = $request->input('query', '');
        // Aplicar filtro de búsqueda si existe
        if (!empty($query)) {
            $stands->where(function ($q) use ($query) {
                $q->where('name', 'like', '%' . $query . '%')
                  ->orWhere('location', 'like', '%' . $query . '%');
            });
        }
        // Obtener puestos finales
        $stands = $stands->get();
        return view('stands.index', compact('stands'));
    }
```

V.1.1 **Objetivo:**

Diseñar, implementar y evaluar agentes conversacionales inteligentes basados en modelos de lenguaje, enfocados en tareas complejas de múltiples turnos, personalización y retención de memoria de largo plazo. El propósito es identificar fortalezas, debilidades y oportunidades de mejora en las capacidades de interacción contextual.

🔹 **Requisitos Específicos**

1. **Diseño del Agente:**
   - Desarrolla un agente especializado en un dominio concreto (educación, soporte técnico, medicina, etc.).
   - Define su **personalidad**, **estilo de comunicación** y **nivel de experticia**.
2. **Capacidades Esperadas:**
   - Gestión de conversaciones multi-turno coherentes.
   - Uso y actualización de memoria de largo plazo.
   - Personalización de respuestas con base en interacciones previas.
3. **Contextos de Evaluación:**
   - Simula al menos **tres escenarios de usuario** con distintos objetivos, niveles de conocimiento y tono.
   - Incluye ejemplos de recuperación contextual (“¿Qué te dije hace 10 minutos?”, “Recuerda mis preferencias”).
4. **Criterios de Evaluación:**
   - Coherencia narrativa.
   - Capacidad de personalización.
   - Precisión en la recuperación de memoria.
   - Adaptabilidad a diferentes perfiles de usuario.
   - Manejo ético y seguro de la información sensible.
5. **Entregables:**
   - Script de simulación de diálogos.
   - Informe técnico con análisis cualitativo y cuantitativo.
   - Recomendaciones de mejora con base en los resultados.

🔹 **Restricciones Técnicas**

- El agente debe ser evaluado en un entorno controlado (sandbox o simulador).
- Se prohíbe el uso de datos reales de usuarios sin anonimización.

	• Se debe documentar el comportamiento del agente en condiciones límite (ambigüedad, contradicciones).	
Long. del Prompt	V.1 sin iteracion 3097 con iteracion: 24323	V.1.1 1642
Documentos incluidos	Sí, específicos de la base de datos y especificaciones	
Ejemplos Incluidos	Sí, codigos a modificar y errores a solucionar	
Calidad de la Respuesta	V.1 4	V.1.1 5
P. Ponderada	V.1	V.1.1
Prob. de Código	manejo de dependencias uso de javascript manejo de datos recibidos por API	
Iteraciones	9	
Mejoras Recomendadas	Mayor especificación para evitar futuras iteraciones	
Resultados de aprendizaje vinculados	V1 R1-R4: Diseño y desarrollo de servicios inteligentes con lenguaje natural (diseño de agentes, memoria, instrucciones). R5, R11, R12: Interés por programación, trabajo colaborativo, búsqueda autónoma de información. R6-R10: Comunicación escrita y oral, documentación técnica. R15: Aplicación autónoma de competencias técnicas.	
	V1.1 Resultado Descripción R1, R2, R3, R4 Desarrollo y despliegue de servicios inteligentes personalizados. R5, R11, R12 Trabajo autónomo y colaborativo, análisis de comportamiento. R6-R10, R13-R15 Comunicación de resultados, planificación y ejecución autónoma.	
Competencias relacionadas	V1 CETM-1, CETM-4, CETM-6, CETM-7: Diseño y desarrollo de arquitecturas y servicios telemáticos. CG-5: Programación y uso de aplicaciones en telecomunicaciones. CB-3, CB-4, CB-5: Interpretación de datos, comunicación y autonomía. CT-1, CT-2, CT-3: Comunicación efectiva, colaboración y mejora continua.	
	V1.1 Tipo Competencias Básicas CB3, CB4, CB5 Generales CG5 Transversales CT1, CT2, CT3 Específicas CETM-1, CETM-4, CETM-6, CETM-7	
Objetivos de la asignatura	V1 **OBJ-1**: Tecnologías web del cliente/servidor (base de la implementación de agentes). **OBJ-2 y OBJ-4**: Implementación de sistemas web con IA, redes sociales, o asistentes personalizados. **OBJ-5**: Fundamentos semánticos presentes en la manipulación de memoria y contexto en el agente.	
	V1.1 **OBJ-1**: Aplicación de tecnologías del lado cliente/servidor. • **OBJ-2**: Desarrollo e implementación de un sistema web inteligente. • **OBJ-4 y OBJ-5**: Redes sociales, Web semántica y personalización.	

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	4	4,3	-
1.1	Se agregaron especificaciones técnicas	5	5	5	5	5	5	5	0,67

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.** = Mejor