## Plantilla de Documentación de Prompts

| Campo | Contenido |
|---|---|
| ID del Prompt | P002-Fase 2 Qwen |
| Fecha de Uso | 22/04/2025 |
| Modelo de IA | Qwen2.5-coder:32b |
| Categoría de la Tarea | Frontend/Backend/Base de Datos/Autenticacion |
| Tecnología | HTML/CSS/Javascript/PHP/SQL |
| Texto Completo del Prompt | V.1 Necesito una implementación completa de **autenticación y gestión de perfiles** en **Laravel 5.2.1** para un **CMS de Mercado Local**. El sistema debe permitir el **registro e inicio de sesión de usuarios** con diferentes roles (**cliente, vendedor y proveedor**) y la **gestión de perfiles según el rol**. También se debe incluir **middleware de autorización** para restringir accesos.<br>**Requisitos detallados:**<br>1. **Autenticación**<br>* Implementar el sistema de autenticación con Laravel UI.<br>* Personalización del modelo User para incluir los campos phone, profile_picture y role (cliente, vendedor, proveedor).<br>* Migración de la tabla users con estos campos adicionales.<br>* Implementación de validaciones seguras en el RegisterController y LoginController.<br>2. **Middleware de Autorización**<br>* Crear un middleware CheckRole para restringir acceso según el rol.<br>* Registrar el middleware en Kernel.php.<br>* Definir rutas protegidas para cada tipo de usuario en routes/web.php.<br>3. **Gestión de Perfiles por Rol**<br>* Crear controladores ClientController, VendorController, ProviderController con sus respectivas funcionalidades:<br>* Cliente: ver y actualizar perfil.<br>* Vendedor: ver y actualizar perfil, gestionar su puesto (stand), gestionar productos..<br>* Proveedor: ver y actualizar perfil, gestionar su puesto (stand), gestionar productos.<br>* Implementar vistas en Blade para la edición de perfil de cada usuario, siguiendo un diseño basado en Bootstrap.<br>4. **Flujo de autenticación y redirecciones** |

| | |
|---|---|
| | * Modificar LoginController para redirigir según el rol.<br><br>* Implementar logout y protección contra ataques CSRF.<br><br>5. **Corrección de errores comunes**<br><br>* Garantizar que las rutas están correctamente definidas (App\Http\Controllers…).<br><br>* Incluir comandos de Laravel (php artisan migrate, db:seed) para inicializar correctamente el sistema.<br><br>6. **Configuraciones y datos de prueba para facilitar la ejecución**<br><br>* Incluir ejemplos de usuarios en UsersTableSeeder.<br><br>* Explicar cómo probar la funcionalidad (rutas relevantes, credenciales de prueba).<br><br>* Incluir el layout base (app.blade.php) con navegación adecuada según el rol del usuario.<br><br>1ºiteracion: Dame el código para middleware CheckRole<br><br>2ºiteracion: Dame el código de los controladores ClientController, VendorController y ProviderController, así como sus vistas y rutas.<br><br>3ºIteracion: dame el comando para resolver este error: **In order to use the Auth::routes() method, please install the laravel/ui package.**<br><br>4ºIteracion: añade la opción de añadir una foto de perfil a la vista profile:<br>```php<br>@extends('layouts.app')<br>@section('content')<br> <br><h1>Perfil del Cliente</h1><br>@if (session('success'))<br> <br>{{ session('success') }}<br> <br>@endif<br><form method="POST" action="{{ route('client.updateProfile') }}"><br>@csrf<br>@method('PUT')<br> <br><label for="name">Nombre</label><br><input type="text" id="name" name="name" class="form-control" value="{{ old('name', $user->name) }}" required><br> <br> <br><label for="email">Email</label><br><input type="email" id="email" name="email" class="form-control" value="{{ old('email', $user->email) }}" required><br> <br> <br><label for="phone">Teléfono</label><br><input type="text" id="phone" name="phone" class="form-control" value="{{ old('phone', $user->phone) }}"> |

```
        </div>
        <button type="submit" class="btn btn-primary">Actualizar Perfil</button>
    </form>
</div>
@endsection
```
5ºIteracion: modifica la vista y las funciones del controlador RegisterController para que usuario al registrarse pueda elegir el rol:
**Funciones del controlador**
```php
 protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
        ]);
    }
    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\Models\User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);
    }
```
**Vista**
```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">{{ __('Register') }}</div>
                <div class="card-body">
                    <form method="POST" action="{{ route('register') }}">
                        @csrf
                        <div class="row mb-3">
                            <label for="name" class="col-md-4 col-form-label
text-md-end">{{ __('Name') }}</label>
                            <div class="col-md-6">
                                <input id="name" type="text" class="form-control
@error('name') is-invalid @enderror" name="name" value="{{ old('name') }}"
required autocomplete="name" autofocus>
                                @error('name')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
```

```
                    </div>
                    <div class="row mb-3">
                        <label for="email" class="col-md-4 col-form-label
text-md-end">{{ __('Email Address') }}</label>
                        <div class="col-md-6">
                            <input id="email" type="email" class="form-control
@error('email') is-invalid @enderror" name="email" value="{{ old('email') }}"
required autocomplete="email">
                            @error('email')
                                <span class="invalid-feedback" role="alert">
                                    <strong>{{ $message }}</strong>
                                </span>
                            @enderror
                        </div>
                    </div>
                    <div class="row mb-3">
                        <label for="password" class="col-md-4 col-form-label
text-md-end">{{ __('Password') }}</label>
                        <div class="col-md-6">
                            <input id="password" type="password" class="form-control
@error('password') is-invalid @enderror" name="password" required
autocomplete="new-password">
                            @error('password')
                                <span class="invalid-feedback" role="alert">
                                    <strong>{{ $message }}</strong>
                                </span>
                            @enderror
                        </div>
                    </div>
                    <div class="row mb-3">
                        <label for="password-confirm" class="col-md-4 col-form-label
text-md-end">{{ __('Confirm Password') }}</label>
                        <div class="col-md-6">
                            <input id="password-confirm" type="password"
class="form-control" name="password_confirmation" required
autocomplete="new-password">
                        </div>
                    </div>
                    <div class="row mb-0">
                        <div class="col-md-6 offset-md-4">
                            <button type="submit" class="btn btn-primary">
                                {{ __('Register') }}
                            </button>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
@endsection
```

V.1.1 Implementación de un Sistema de Autenticación y Gestión de Perfiles con Roles en Laravel 5.2.1 para un CMS de Mercado Local

**Contexto:**

Se requiere el desarrollo de una funcionalidad central para un sistema de gestión de contenidos (CMS) orientado a un **Mercado Local**. Este CMS será

utilizado por tres tipos de usuarios: **clientes, vendedores y proveedores**, cada uno con accesos y funcionalidades específicas.

**Objetivo:**
Implementar un **módulo de autenticación y gestión de perfiles** que contemple:

1. Registro e inicio de sesión con selección de rol.
2. Middleware de autorización por rol.
3. Funcionalidad de edición de perfil adaptada a cada tipo de usuario.
4. Interfaz responsiva basada en Blade y Bootstrap.
5. Flujo de navegación con redirección según rol tras login.

---

📋 **Requisitos Detallados**

**1. Autenticación:**

- Implementar sistema de autenticación utilizando Laravel (`make:auth` o Laravel UI según compatibilidad con la versión 5.2.1).
- Añadir campos personalizados al modelo `User`: `phone`, `profile_picture`, `role`.
- Migrar tabla `users` con los nuevos campos.
- Personalizar `RegisterController` para incluir selección de rol y validaciones correspondientes.

**2. Middleware de Autorización:**

- Crear un middleware `CheckRole` que permita el acceso basado en roles.
- Registrar este middleware en `Kernel.php`.
- Proteger rutas con el middleware.

**3. Gestión de Perfiles:**

- Crear controladores: `ClientController`, `VendorController`, `ProviderController`.
- Funcionalidades:
  - Cliente: ver/editar perfil.
  - Vendedor y Proveedor: ver/editar perfil, gestionar puesto (stand), gestionar productos.
- Implementar vistas con Blade y Bootstrap.

**4. Flujo y Seguridad**

- Modificar `LoginController` para redirigir según rol.
- Incluir funcionalidades de logout y protección CSRF.

**5. Inicialización del Sistema:**

- Proveer migraciones, seeders con usuarios de ejemplo y layout base `app.blade.php`.
- Documentar rutas clave y credenciales de prueba.

| | | |
|---|---|---|
| Long. del Prompt | V.1 sin iteracion: 2462<br>con iteracion: 8511 caracteres | V.1.1 1896 |
| Documentos incluidos | Sí, específicos de la base de datos y especificaciones | |
| Ejemplos Incluidos | Si, mensajes de errores a solucionar y código a modificar | |
| Calidad de la Respuesta | V.1 5 | V.1.1 5 |
| P. Ponderada | V.1 4,83 | V.1.1 5 |
| Prob. de Código | Falta de instalación de dependencias clave | |
| Iteraciones | 5 | |
| Mejoras Recomendadas | Mayor especificación para evitar iteraciones | |
| Resultados de aprendizaje vinculados | V1 **R2**: Uso de arquitecturas y modelos web (MVC, middleware).<br>**R3**: Utiliza tecnologías y estándares (Laravel, Blade, PHP).<br>**R4**: Implementa servicios web (registro, perfiles, control de acceso).<br>**R15**: Aplica lo aprendido de forma autónoma. | |
| | V1.1. **R1, R2, R3, R4:** Modelado e implementación técnica de servicios web por roles.<br>**R5, R6, R11, R12, R15:** Aplicación autónoma de competencias técnicas y comunicación de soluciones. | |
| Competencias relacionadas | V1 **CETM6**: Diseño de arquitecturas de servicios telemáticos.<br>**CETM1**: Construcción y gestión de servicios de telecomunicación.<br>**CG5**: Programación y verificación de aplicaciones.<br>**CB4**: Capacidad de comunicar soluciones a diferentes públicos.<br>**CT2**: Colaboración y comprensión de competencias profesionales. | |
| | V1.1 **CETM1, CETM4, CETM6, CETM7:** Diseño e implementación de servicios telemáticos.<br>**CG5:** Programación de aplicaciones web.<br>**CB3, CB4, CB5:** Comunicación, reflexión técnica y aprendizaje autónomo.<br>**CT1, CT2:** Comunicación con diferentes audiencias y trabajo colaborativo. | |
| Objetivos de la asignatura | V1 **OBJ-1**: Conocer tecnologías web del lado del servidor.<br>**OBJ-2**: Implementar un sistema web completo con base de datos. | |
| | V1.1 **OBJ-1, OBJ-2, OBJ-3:** Desarrollo web lado servidor, integración con base de datos, y gestión de sistemas de comercio electrónico. | |

---

## Tabla de Comparación de Versiones de Prompt

| Versión | Camb. Clave | Cla. | Esp. | Prec. Téc. | Ctx. | Estruc. | Cal. Cód. | Punt. Pond. | Mej. |
|---|---|---|---|---|---|---|---|---|---|
| 1.0 | Inicial | 5 | 5 | 5 | 4 | 5 | 5 | 4,83 | - |
| 1.1 | Se agregaron especificaciones técnicas | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 0,167 |

**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