Manual Técnico del aplicativo Web

Área de Desarrollo Móvil
Centro De Electricidad Y Automatización Industrial, SENA
3067594: Análisis y Desarrollo de Software
Cómite Evaluador
5 de julio del 2026.

Cali, Valle del Cauca

Tabla de Contenido

Control de Versiones del Documento

La siguiente tabla registra el historial de cambios realizados sobre este documento, con el fin de mantener trazabilidad sobre su evolución.

Versión	Fecha	Autor	Cambios realizados
1.0	28/06/2026	Equipo TAM	Versión inicial de la ficha técnica: entorno, stack tecnológico, módulos, base de datos y seguridad.
2.0	05/07/2026	Equipo TAM	Se agregaron objetivo y alcance, requisitos e instalación, diagrama de arquitectura, diagrama entidad-relación, glosario, control de versiones y referencias.

Objetivo y Alcance del Manual

Objetivo

Este documento tiene como objetivo describir de forma clara y estructurada los aspectos técnicos del proyecto TAM, incluyendo su entorno de desarrollo, arquitectura, tecnologías utilizadas, módulos funcionales, estructura de la base de datos y prácticas de seguridad implementadas, con el fin de servir como referencia técnica para su comprensión, mantenimiento y evaluación.

Alcance

La ficha técnica cubre la versión web del proyecto TAM desarrollada con el framework Laravel, abarcando desde la configuración del entorno de desarrollo local hasta el despliegue de sus módulos principales (cuentas, catálogo, comparador, carrito, pedidos y soporte). No incluye el detalle del código fuente línea por línea, sino una descripción funcional y estructural del sistema.

Dirigido a

- **Instructores y evaluadores** del SENA encargados de revisar la evidencia formativa del proyecto.
- **Aprendices del equipo de desarrollo** que requieran instalar, mantener o ampliar el sistema.
- **Nuevos integrantes** que se incorporen al proyecto y necesiten comprender su funcionamiento técnico.

Entorno del Proyecto

El entorno no es el diseño visual de la página. Es todo el conjunto de programas, servidores y herramientas que se usan en la computadora para que el código funcione y se conecte correctamente a la base de datos.

En el proyecto se utiliza un Entorno de Desarrollo Local, compuesto por las siguientes herramientas:

- **XAMPP / Laragon:** programa encargado de simular el servidor web (Apache) y activar el motor de la base de datos (MySQL) en la máquina local.
- **Visual Studio Code:** editor de texto donde se escribe y gestiona el código fuente.
- **PHP:** lenguaje de programación base que necesita el framework Laravel para ejecutarse.

Resumen General del Proyecto

TAM es una página web de comercio electrónico (e-commerce) creada para la venta de productos de tecnología, componentes de hardware y teléfonos celulares. El proyecto está desarrollado usando el framework Laravel para controlar los datos en el servidor, junto con hojas de estilo CSS ordenadas y separadas por vistas para la parte visual.

Requisitos e Instalación

3.1 Requisitos Previos

-
-
-
-
-
-
-

3.2 Pasos de Instalación

1. **Clonar el repositorio:** descargar el código fuente del proyecto desde el repositorio Git.

```
git clone <URL-del-repositorio>
cd tam
```

2. **Instalar dependencias de PHP:** usar Composer para descargar las librerías del framework Laravel.

```
composer install
```

3. **Configurar el archivo de entorno:** copiar el archivo de ejemplo y ajustar las variables de conexión.

```
cp .env.example .env
php artisan key:generate
```

Dentro del archivo .env se deben configurar, como mínimo, los siguientes parámetros de conexión a la base de datos:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=tam
DB_USERNAME=root
DB_PASSWORD=
```

4. **Crear la base de datos:** desde phpMyAdmin (incluido en XAMPP/Laragon) crear una base de datos vacía llamada tam e importar el archivo tam2 (2).sql.
5. **Ejecutar las migraciones:** en caso de requerir actualizar la estructura de la base de datos con las migraciones de Laravel.

```
php artisan migrate
```

6. **Enlazar el almacenamiento:** generar el enlace simbólico para las imágenes de productos.

```
php artisan storage:link
```

7. **Levantar el servidor de desarrollo:** iniciar Apache y MySQL desde el panel de XAMPP/Laragon, y luego ejecutar el servidor embebido de Laravel.

```
php artisan serve
```

8. **Acceder a la aplicación:** abrir el navegador en http://localhost:8000 (o el puerto configurado).

3.3 Verificación de la Instalación

Para confirmar que el entorno quedó correctamente configurado, se recomienda verificar que: el catálogo de productos cargue datos desde la base de datos, el registro/login funcione junto con el CAPTCHA, y el envío de correos de prueba (recuperación de contraseña) se ejecute sin errores en la cola de Mailer.

Tecnologías Utilizadas (Stack Tecnológico)

- **Backend:** framework Laravel (PHP) para manejar la lógica de la tienda, el control de las cuentas de usuario y la seguridad del sistema.
- **Servicios de comunicación:** componente Mailer / Mail Transfer integrado en Laravel para el envío automatizado de correos electrónicos (notificaciones de compras y recuperación de cuentas).
- **Seguridad de accesos:** sistema de validación CAPTCHA para la protección de formularios, evitando registros falsos y ataques automatizados de bots.
- **Frontend:** estructura básica en HTML5 y diseño visual con CSS3 nativo. Los archivos CSS están separados por páginas específicas (como catalogo.css y carrito.css) para mantener un código limpio.
- **Base de datos:** sistema MySQL / MariaDB para guardar y relacionar toda la información de la tienda.
- **Control de versiones:** herramienta Git para almacenar y controlar el historial de modificaciones del código.

Arquitectura del Sistema

La versión web de TAM implementa una arquitectura cliente-servidor desarrollada con Laravel bajo el patrón Modelo-Vista-Controlador (MVC). Esta arquitectura organiza el sistema en componentes independientes que facilitan el desarrollo, el mantenimiento y la escalabilidad de la aplicación.

Cuando un usuario realiza una acción desde el navegador, la solicitud es enviada al servidor web Apache, donde Laravel procesa la petición mediante sus controladores. Posteriormente, el sistema interactúa con la base de datos MySQL utilizando el ORM Eloquent para consultar o almacenar la información necesaria, y finalmente genera la respuesta que será presentada al usuario.

Esta estructura permite mantener separada la interfaz gráfica, la lógica del negocio y el acceso a los datos, favoreciendo un código más organizado, reutilizable y seguro.

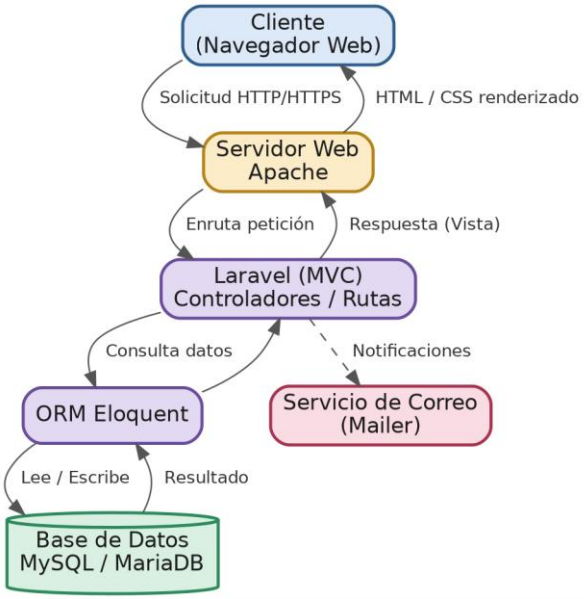

Figura 1. Flujo cliente – servidor – base de datos.

6.1 Capas de la Arquitectura

Capa de Presentación

Corresponde a las vistas desarrolladas en HTML y CSS, encargadas de mostrar la información al usuario y permitir la interacción mediante formularios, catálogos, comparadores de productos, carrito de compras y demás interfaces del sistema.

Capa de Lógica de Negocio

Implementada mediante Laravel, contiene los controladores, reglas de negocio, validaciones, autenticación de usuarios, gestión de productos, pedidos, carrito de compras, comparador y demás procesos internos de la aplicación.

Capa de Datos

Está compuesta por la base de datos MySQL y el ORM Eloquent de Laravel, responsables del almacenamiento, consulta y actualización de la información relacionada con usuarios, productos, categorías, pedidos, inventario y demás módulos del sistema.

6.2 Infraestructura

Componente	Tecnología
Cliente	Navegador Web
Servidor Web	Apache
Framework	Laravel
Lenguaje Backend	PHP
Base de Datos	MySQL / MariaDB
Comunicación	HTTP / HTTPS
Entorno de Desarrollo	XAMPP o Laragon

Módulos y Páginas de la Aplicación

A continuación se describe cada módulo funcional de la aplicación, indicando su propósito, los archivos de estilo asociados y su comportamiento en el servidor.

Módulo	Pantalla Visual (CSS)	Lógica de Servidor (Laravel)
Cuentas y Login	login.css, forgot_password.css	Inicia sesión, cierra sesión, valida el sistema CAPTCHA e interactúa con el Mailer para enviar el enlace de recuperación. Encripta contraseñas.
Catálogo	catalogo.css, base.css	Trae los productos de la base de datos y los muestra organizados en pantalla.
Comparador	comparador.css	Compara las características técnicas de dos o más productos elegidos.
Carrito	carrito.css	Guarda temporalmente los productos que el cliente quiere comprar.
Pedidos	pedidos.css	Registra las compras hechas por los clientes, guarda su historial y dispara el envío de correos de confirmación por Mailer.
Soporte	contacto.css	Valida el CAPTCHA de seguridad, recibe los mensajes de los usuarios y los almacena en la base de datos.

Estructura de la Base de Datos

A partir del archivo original tam2 (2).sql, la base de datos relacional se organiza de la siguiente manera:

8.1 Diagrama Entidad-Relación

El siguiente esquema resume las tablas principales de la tienda y sus relaciones de cardinalidad (1:N), facilitando la comprensión visual del modelo de datos.

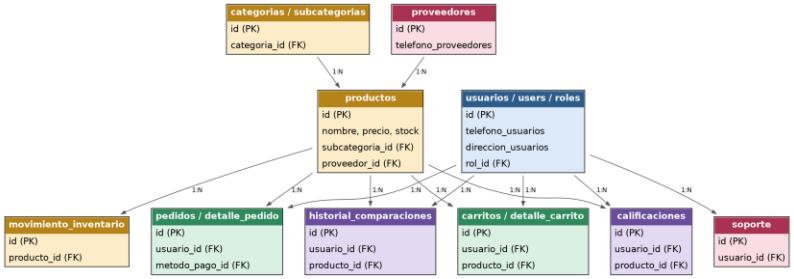

Figura 2. Esquema entidad-relación simplificado de la base de datos TAM.

8.2 Tablas Principales de la Tienda

- **usuarios, users, roles:** guarda los datos de los clientes, administradores y sus permisos de acceso, incluyendo telefono_usuarios y direccion_usuarios.
- **productos:** almacena el nombre, precio, stock, descripción y fotos de los artículos tecnológicos (marcas como Asus, Acer, entre otras).
- **categorias, subcategorias:** organiza jerárquicamente los productos (ejemplo: laptops, accesorios).
- **carritos, detalle_carrito:** guarda los productos que el usuario añade a su carrito antes de pagar.
- **pedidos, detalle_pedido, metodo_pago_usuarios:** almacena las facturas, compras terminadas y el método de pago utilizado por el cliente.
- **movimiento_inventario:** controla las entradas y salidas de mercancía (Kardex).
- **historial_comparaciones:** registra los artículos que los usuarios han comparado en la plataforma.
- **proveedores, proveedores_categorias, telefono_proveedores:** información de contacto de las empresas que surten el hardware.
- **calificaciones:** almacena las puntuaciones y opiniones de los compradores sobre los productos.
- **soporte:** guarda los mensajes de ayuda enviados desde el formulario de contacto.

8.3 Tablas Automáticas e Infraestructura de Laravel

- **migrations:** controla el orden y la creación de las tablas de la base de datos.
- **sessions:** almacenamiento y manejo de las sesiones activas de los usuarios en la tienda.
- **jobs, failed_jobs, job_batches:** cola de procesos encargada de gestionar de manera asíncrona las tareas en segundo plano, como el envío masivo de correos de Mailer sin bloquear la interfaz.
- **cache, cache_locks:** sistema de optimización para que las búsquedas repetidas carguen más rápido.
- **password_resets, password_reset_tokens:** códigos de seguridad efímeros para restablecer contraseñas olvidadas de forma segura por correo electrónico.

Seguridad y Buenas Prácticas

1. **Filtros antibots (CAPTCHA):** verificación en los formularios críticos (login, registro y contacto) para bloquear intentos de acceso automáticos o maliciosos.
2. **Seguridad contra inyecciones SQL:** uso automático del ORM Eloquent de Laravel para proteger la base de datos y evitar que personas externas alteren el sistema.
3. **Protección de formularios (tokens @csrf):** directivas de protección obligatorias para asegurar que nadie envíe datos falsos a la aplicación desde sitios externos.
4. **Integridad referencial:** estructura unida mediante llaves foráneas para evitar registros sueltos o inconsistencias de datos en los procesos de compra.
5. **Encriptación de contraseñas:** las contraseñas de los usuarios se almacenan utilizando algoritmos de hash seguros, evitando que la información sensible quede expuesta ante un eventual acceso indebido.
6. **Gestión segura de sesiones:** control del tiempo de vida y la validez de las sesiones activas para reducir el riesgo de suplantación de usuarios.

Glosario de Términos Técnicos

A continuación se definen los principales conceptos técnicos utilizados a lo largo de este documento.

Término	Definición
MVC	Modelo-Vista-Controlador. Patrón de arquitectura de software que separa la lógica de negocio (Modelo), la interfaz de usuario (Vista) y el control del flujo de la aplicación (Controlador).
ORM	Object-Relational Mapping. Técnica que permite manipular datos de una base de datos relacional usando objetos y código orientado a objetos, sin escribir SQL directamente. En Laravel se implementa mediante Eloquent.
Eloquent	ORM incluido por defecto en Laravel, que representa cada tabla de la base de datos como una clase PHP (modelo).
CAPTCHA	Prueba de desafío-respuesta utilizada para determinar si el usuario que interactúa con un formulario es humano, evitando el registro de bots.
CSRF	Cross-Site Request Forgery. Tipo de ataque que fuerza a un usuario autenticado a ejecutar acciones no deseadas. Laravel lo previene mediante tokens @csrf en los formularios.
Migración	Archivo de Laravel que define cambios sobre la estructura de la base de datos (crear o modificar tablas) de forma controlada y versionada.
Kardex	Registro que controla las entradas y salidas de inventario de un producto.
API / Endpoint	Punto de acceso de un servicio web mediante el cual el cliente y el servidor intercambian información.
Framework	Conjunto de herramientas, librerías y convenciones que facilitan el desarrollo de software, como Laravel para aplicaciones PHP.
Composer	Gestor de dependencias para PHP, utilizado para instalar y administrar las librerías que requiere Laravel.
Entorno de desarrollo local	Conjunto de programas (servidor, base de datos, editor) instalados en la máquina del desarrollador para ejecutar y probar la aplicación antes de publicarla en producción.

Término	Definición
HTTPS	Protocolo de comunicación segura entre el cliente y el servidor mediante cifrado SSL/TLS.

Conclusión

La plataforma web TAM integra herramientas modernas para el desarrollo de aplicaciones de comercio electrónico, ofreciendo una arquitectura organizada, segura y escalable. La implementación de Laravel, junto con MySQL y buenas prácticas de desarrollo, permite administrar eficientemente la información del sistema y brindar una experiencia confiable a los usuarios.

Además, la separación de responsabilidades mediante el patrón MVC facilita el mantenimiento, la incorporación de nuevas funcionalidades y el crecimiento futuro de la plataforma, consolidando a TAM como una solución técnica sólida y preparada para escalar junto con las necesidades del negocio.

Referencias y Bibliografía

Laravel. (2026). Laravel 11.x Documentation. https://laravel.com/docs

Oracle Corporation. (2026). MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/

MariaDB Foundation. (2026). MariaDB Knowledge Base. https://mariadb.com/kb/

PHP Group. (2026). Manual de PHP. https://www.php.net/manual/es/

The Apache Software Foundation. (2026). Apache HTTP Server Documentation. https://httpd.apache.org/docs/

Git. (2026). Pro Git Book. https://git-scm.com/book

Composer. (2026). Composer Documentation. https://getcomposer.org/doc/

Servicio Nacional de Aprendizaje (SENA). (2026). Programa de Formación 3067594: Análisis y Desarrollo de Software.