



WEBCRAFTERS: MANUAL TÉCNICO DE USUARIO

Solución web para MagicColombia



David Arturo Ruiz Chanchi

Miguel Ángel Fernández Samboni

Eider Alexis Camacho Palacios

SEPTIEMBRE 4 DEL 2025

UNIVERSIDAD: UNIREMINGTON

Cali, Valle del Cauca

Índice

Resumen ejecutivo.....	3
------------------------	---

- 1.1. Visión y misión
- 1.2. Objetivos del proyecto
- 1.3. Alcance y restricciones

Manual de usuario.....	8
------------------------	---

- 2.1. Registro e inicio de sesión
- 2.2. Panel Administrador
- 2.3. Catálogo y post

Manual técnico.....	10
---------------------	----

- 3.1. Arquitectura general y flujo
- 3.2. Entornos y Rutas
- 3.3. Estructura de Carpetas
- 3.4. Modelo de Datos
- 3.5. Contratos API

- 3.6. Reglas de imágenes
- 3.7. Frontend: IDs y comportamiento
- 3.8. Operaciones comunes
- 3.9. Seguridad (local)
- 3.10. Troubleshooting (solución de problemas)
- 3.11. SQL - guía práctica
- 3.12. Estándares y convenciones
- 3.13. Checklists
- 3.14. Anexos
- 3.15. Runbook express (que hacer si...)

Análisis del Sistema.....13

- 4.1. Modelos y estándares de calidad
- 4.2. Requerimientos funcionales y no funcionales

Diseño y modelado.....18

- 5.1. Diagrama de casos de uso
- 5.2. Diagrama de clases

- 5.3. Modelo relacional

6. Conclusión del proyecto.....	20
---------------------------------	----

1. Resumen ejecutivo

El presente documento detalla el diseño y desarrollo de una solución web para modernizar y optimizar MagicColombia, un emprendimiento dedicado a la comercialización de cartas coleccionables. Para este proyecto, se adoptó la norma ISO/IEC 25010 como marco de evaluación de calidad y el modelo CMMI como guía para el ciclo de desarrollo. La arquitectura del software se basa en un diseño de tres capas que separa la interfaz, la lógica de negocio y la gestión de datos, lo que garantiza escalabilidad y eficiencia operativa.

La implementación de este nuevo sistema no solo optimiza los procesos internos, sino que también mejora significativamente la experiencia del usuario final y fortalece la competitividad de MagicColombia en el mercado colombiano.

1.1 Visión

Convertirse en una empresa referente en el desarrollo y rediseño de sitios web en Colombia, destacándose por la implementación de soluciones digitales funcionales, seguras y orientadas al usuario. Se busca posicionar a Magic Colombia como un modelo de modernización tecnológica para emprendimientos comerciales, integrando buenas prácticas de ingeniería de software y adaptándose a las tendencias del comercio electrónico.

Misión

Proporcionar a nuestros clientes soluciones web integrales, centradas en la usabilidad, la eficiencia y la experiencia del usuario. Nos enfocamos en rediseñar plataformas digitales aplicando estándares de calidad como ISO/IEC 25010 y metodologías de desarrollo como CMMI, con el fin de optimizar los procesos de venta y visibilidad online de sus productos.

Objetivos

1.1 General

Rediseñar e implementar una solución web optimizada para la tienda Magic Colombia, aplicando estándares de calidad del software y metodologías de desarrollo modernas, con el fin de mejorar la experiencia del usuario, optimizar procesos internos y fortalecer la presencia digital del emprendimiento en el mercado colombiano.

1.2 Específicos

- Analizar el estado actual del sitio web de Magic Colombia, identificando sus limitaciones funcionales, técnicas y de experiencia del usuario.
- Diagnosticar las tendencias del mercado en cuanto a diseño web, experiencia de usuario, integración de herramientas externas y comercio electrónico.
- Definir y documentar los requisitos del sistema con referencia al modelo de calidad ISO/IEC 25010.
- Diseñar la arquitectura del sistema web siguiendo el modelo de tres capas, para garantizar modularidad, escalabilidad y mantenibilidad.
- Aplicar el modelo CMMI como guía para estructurar el proceso de desarrollo, de esta manera asegurar la calidad y la mejora continua del sistema.
- Representar el sistema mediante diagramas UML (casos de uso y clases) para facilitar la comprensión y la futura implementación en el proyecto.
- Proponer funcionalidades adicionales que agregan un mayor valor al negocio, tales como integración con plataformas de envío (como Rappi), catálogos avanzados, reportes automáticos.
- de pago seguras. La propuesta también considera el desarrollo de software a la medida, adaptando las necesidades específicas del negocio, así como la implementación y personalización de sistemas de gestión de contenidos (CMS) que faciliten la administración de la plataforma. Se prevé la integración con servicios externos, incluyendo pasarelas de pago y plataformas de terceros, así como la incorporación de funciones.

1.3 Alcance

El alcance de este proyecto comprende el diseño, desarrollo y optimización de una solución web orientada al comercio electrónico para la tienda Magic Colombia, integrando funcionalidades esenciales de productos, carritos de compras, procesamiento de pedidos y pasarelas, funcionalidades avanzadas como estadística de uso, gestión de usuarios, sistemas de geolocalización y herramientas interactivas para mejorar la experiencia de compra. En el ámbito de los requisitos no funcionales, el sistema estará optimizado para un alto rendimiento y velocidad, asegurando tiempos de carga reducidos y cumplimientos con estándares de posicionamiento SEO, además de incorporar medidas de ciberseguridad para la protección de datos y transacciones. El diseño será totalmente adaptable a diferentes dispositivos y navegadores, garantizando compatibilidad, accesibilidad y usabilidad para todo tipo de usuarios.

La arquitectura estará pensada para ser escalable, permitiendo incorporar nuevas funcionalidades y soportar mayor volumen de tráfico en el futuro, y se implementarán prácticas de desarrollo que favorezcan la mutabilidad, asegurando que el sistema pueda ser actualizado y mejorado de manera ágil. Finalmente, el proyecto contempla un enfoque centrado en la calidad, mediante la aplicación de metodologías y pruebas formales que validen el correcto funcionamiento y la satisfacción de los requisitos funcionales y no funcionales establecidos.

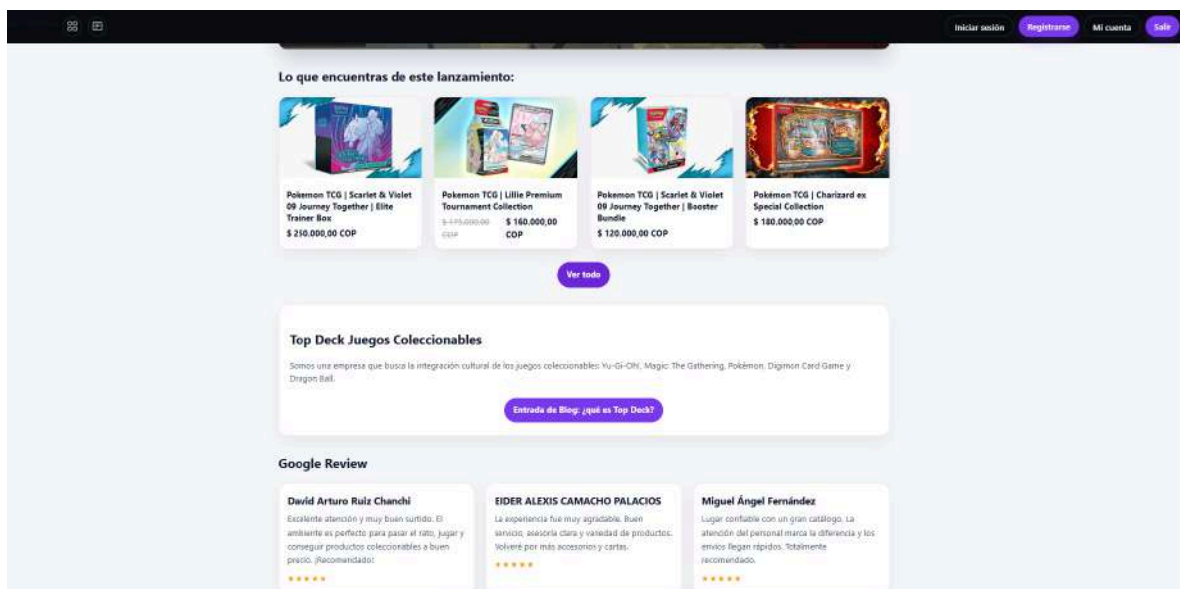
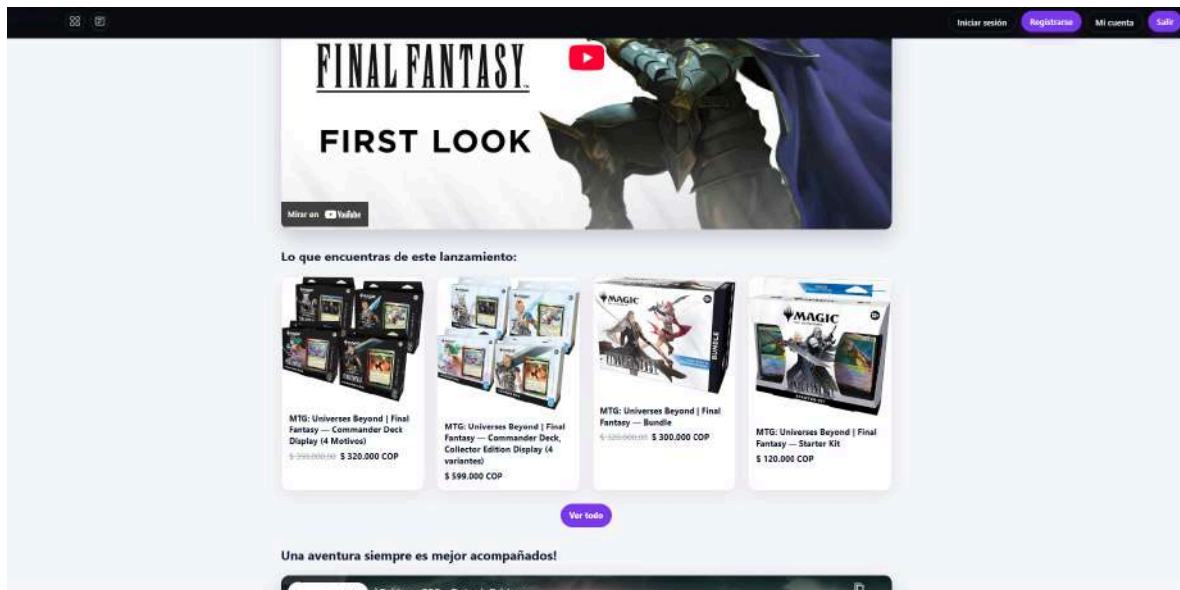
Restricciones

El desarrollo de la nueva plataforma web para Magic Colombia está sujeto a un conjunto de restricciones funcionales y no funcionales que delimitan su alcance y condiciones de ejecución. En primer lugar, el sistema solo incluirá las funcionalidades acordadas en la fase de planificación, por lo que cualquier característica adicional no contemplada requerirá una reevaluación de tiempos, costos y recursos. Las integraciones con sistemas de terceros estarán condicionadas a la disponibilidad de APIs públicas y bien documentadas, pudiendo descartarse aquellas conexiones que resulten técnica o económicamente inviables. Asimismo, se excluyen desarrollos que no se ajusten a la naturaleza y experiencia del equipo, así como proyectos con presupuestos o plazos no realistas, o que involucren actividades ilegales o contrarias a las políticas del negocio. Desde el punto de vista técnico, podrían existir limitaciones en la cantidad y el tipo de datos gestionados, según la infraestructura de hosting y las licencias de software utilizadas. En cuanto a las restricciones no funcionales, el sistema deberá cumplir parámetros de rendimiento que aseguren tiempos de carga óptimos y soporte para un número determinado de usuarios concurrentes sin degradación. Se adoptarán medidas de seguridad alineadas con normativas vigentes, incluyendo cifrado de datos, autenticación segura y auditorías periódicas. El proyecto deberá ajustarse al presupuesto y cronograma definidos, siendo necesaria una renegociación para cambios significativos. Además, se garantizará compatibilidad con navegadores y sistemas operativos específicos, manteniendo un diseño responsivo que asegure la usabilidad y accesibilidad conforme a estándares como WCAG. Por último, el código deberá desarrollarse con un enfoque modular y documentado, para facilitar su mantenimiento y evolución a largo plazo.

2. Manual de usuario

Este manual está diseñado para ayudarte a usar la nueva plataforma web de MagicColombia de forma sencilla e intuitiva.

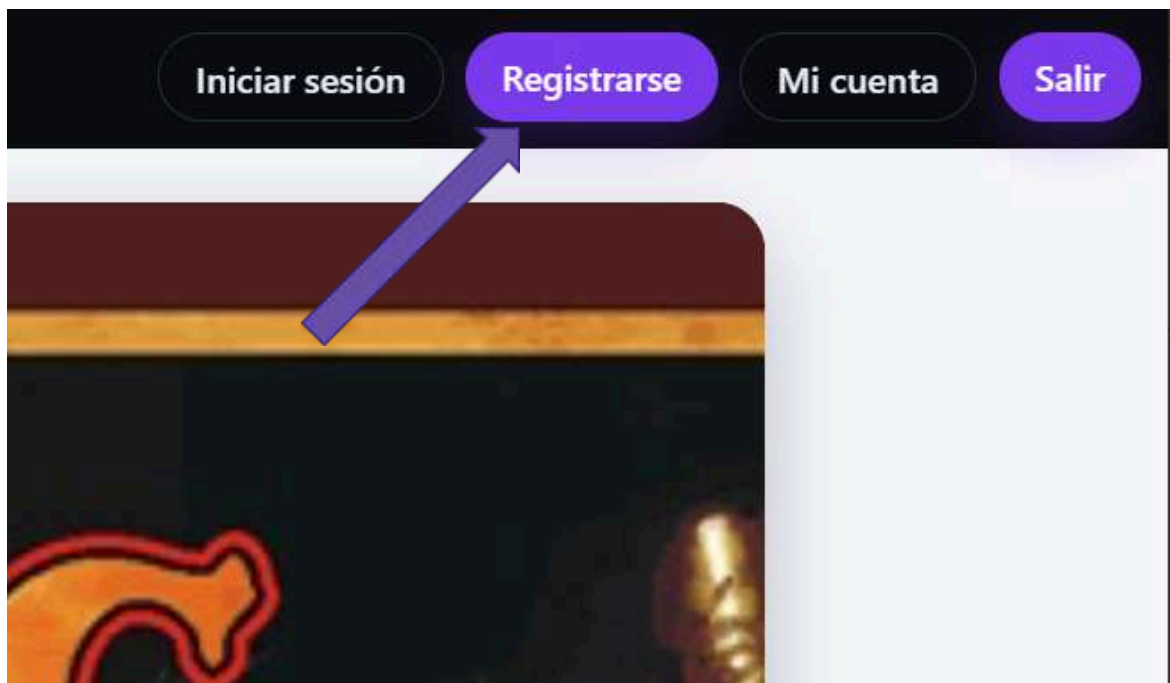




2.1. Cómo empezar: Registro e inicio de sesión

Para acceder a todas las funciones, primero debes registrarte. Una vez que te registres en línea, podrás iniciar sesión para acceder al Panel de administrador.

Botón para registrarse:



Se cargará formulario de registro:

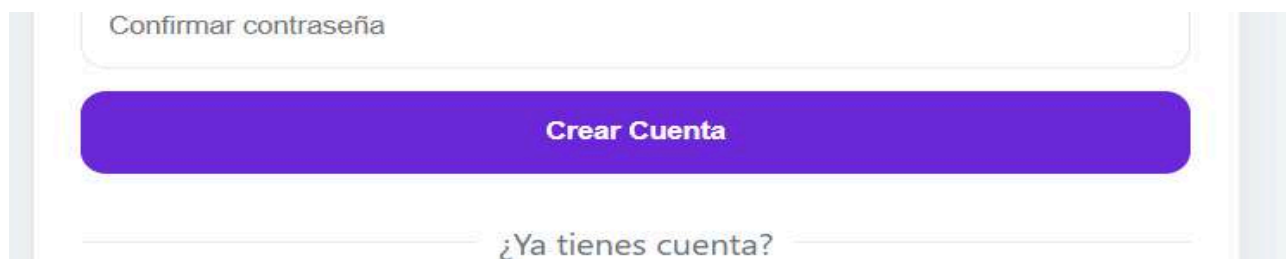
A screenshot of a registration form titled 'Crear Cuenta'. The form is centered on a light blue background. It includes the following fields: 'Nombre completo:', 'Correo electrónico:', 'País:' (with a dropdown menu showing 'Colombia (+57)'), 'Teléfono:' (with a label 'Número de teléfono'), 'Contraseña:', and 'Confirmar Contraseña:'. Below the password fields, there is a note 'Mínimo 8 caracteres'. At the bottom of the form is a blue button labeled 'Crear Cuenta' and a link that says '¿Ya tienes cuenta?'. The top of the page shows the 'MAGIC' logo and some social media icons.

Se encuentran los siguientes campos:

- Nombre completo
- Correo Electrónico
- Selección de prefijos telefónicos
- Numero de telefono
- Contraseña
- Confirmar Contraseña

Los campos se deben completar todos para la creación de cuenta.

Una vez llenado todos los campos, el siguiente paso es dar click en Botón “Crear Cuenta”



The image shows a registration form with a light gray background. At the top, there is a text input field labeled "Confirmar contraseña". Below this field is a large, rounded rectangular button with a solid purple background and the text "Crear Cuenta" in white. At the bottom of the form, there is a link that says "¿Ya tienes cuenta?" in a light gray font, preceded and followed by horizontal lines.

Habrà una recarga de la página y se cargará el login de Iniciar Sesión, donde podremos rellenar los campos para el ingreso:

Inicio de sesión

Accede a tu cuenta para ver pedidos y guardar favoritos.

Correo electrónico

Contraseña [Ver](#)

☐ Recordarme [¿Olvidaste tu contraseña?](#)

Iniciar sesión

[¿No tienes cuenta?](#) [Crear cuenta](#)

MagicColombia
Desde 2002 con la comunidad TCG.

Links Rápidos
[Búsqueda](#)
[Términos del servicio](#)

Sobre Nosotros
Carrera 65 # 5-107 · Cali · Colombia

Rellenamos los campos (Correo Electrónico y contraseña) y luego dar click en Botón “Iniciar Sesión”

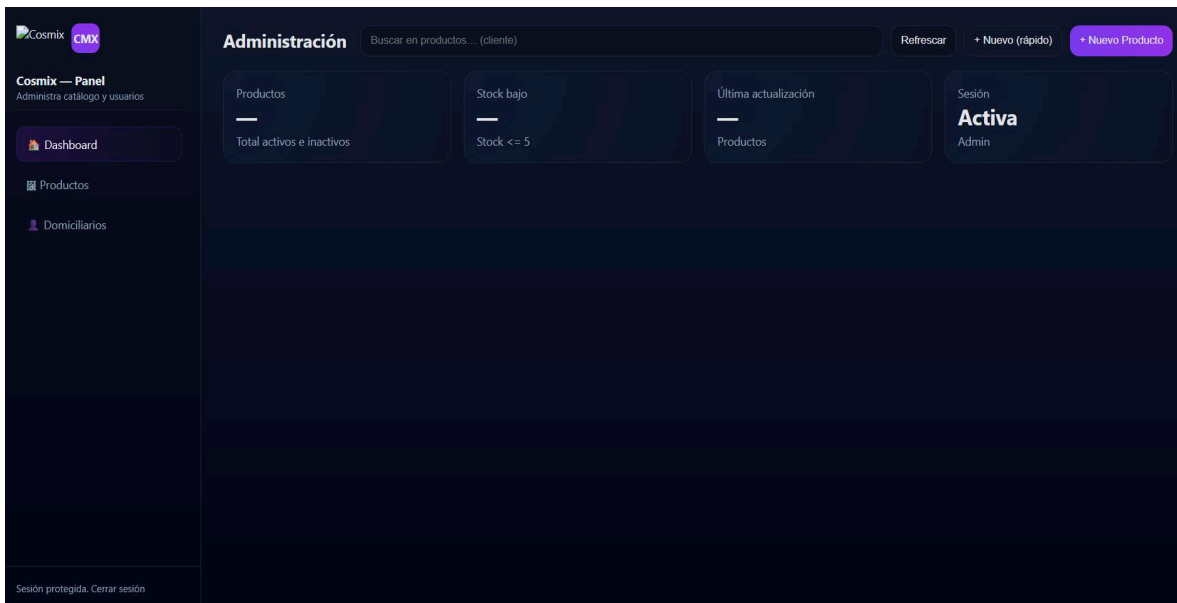
☐ Recordarme [¿Olvidaste tu contraseña?](#)

Iniciar sesión

[¿No tienes cuenta?](#) [Crear cuenta](#)

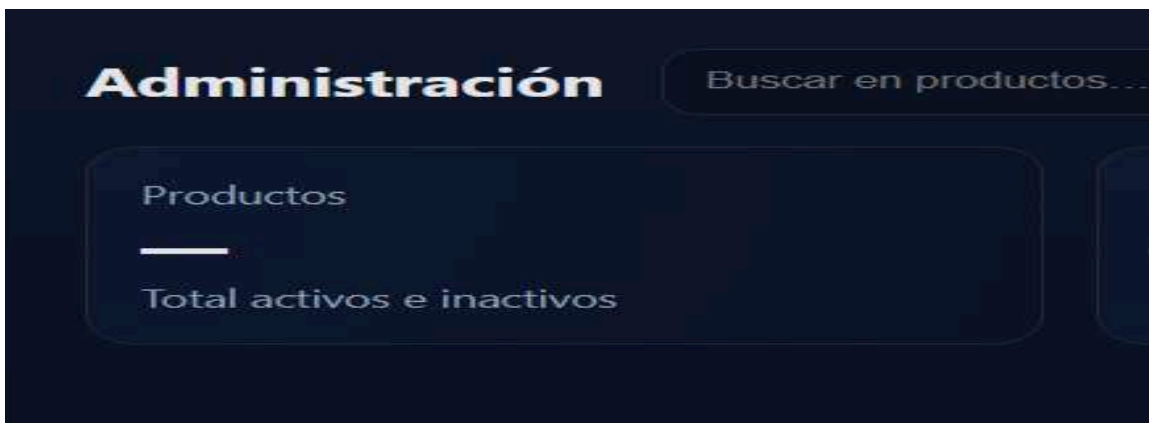
2.2. Panel Administrador

Dashboard de administrador



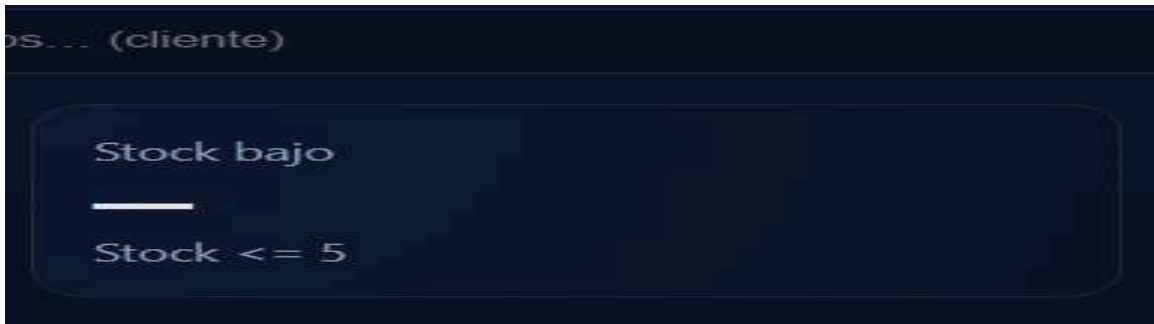
Campos:

- Productos:



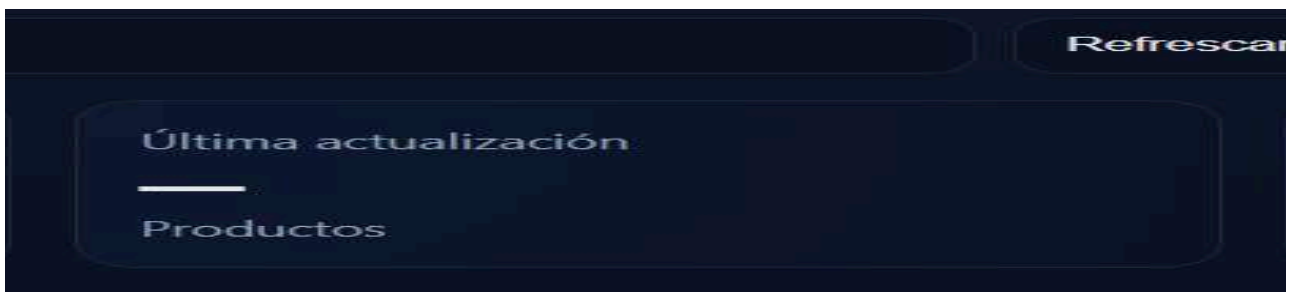
Apartado de Productos, donde se identificará el total de productos (activos e inactivos)

- Stock Bajo:



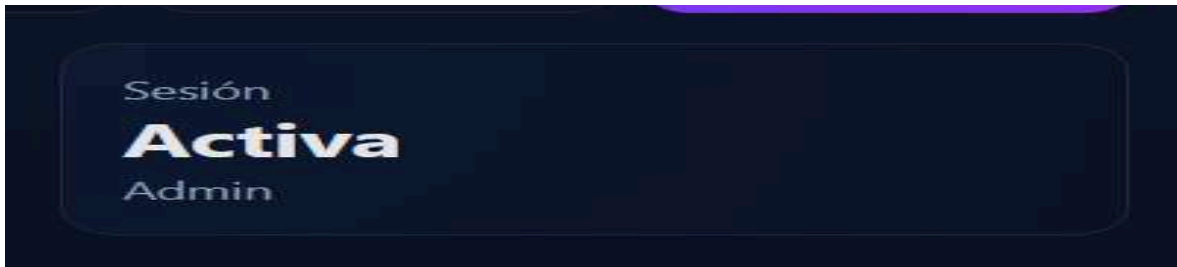
Apartado Stock bajo, donde se identifica el stock de productos que sean menor o igual a 5.

- Última actualización:



Apartado de Última actualización, donde se puede observar la hora de la última vez que se realizó cambios en los productos.

- Sesión:



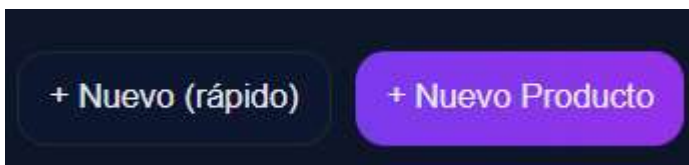
Apartado Sesión, indica el estado de la sesión del administrador.

Barra de búsqueda:



Realiza búsqueda de productos.

Botones “+Nuevo”



→ El boton + Nuevo (rápido) realiza una creación de un nuevo producto a partir de su código SKU, como solo cuenta con un solo campo, se considera una manera rapida de agregar un producto.

→ El boton + Nuevo Producto, agrega una nuevo producto partiendo del diligenciamiento de los siguientes campos:

Producto Cerrar

Nombre

SKU

Categoría

Tipo de carta

Edición

Precio

Stock

URL imagen

Activo ☒ SI

Cancelar Guardar

- Nombre (producto)
- SKU (producto)
- Categoría (producto)
- Tipo de carta (producto)
- Edición (producto)
- Precio
- Stock (cantidad)
- URL imagen (ruta de imagen para el producto)

Botón “Guardar” para guardar información del producto y agregarlo.

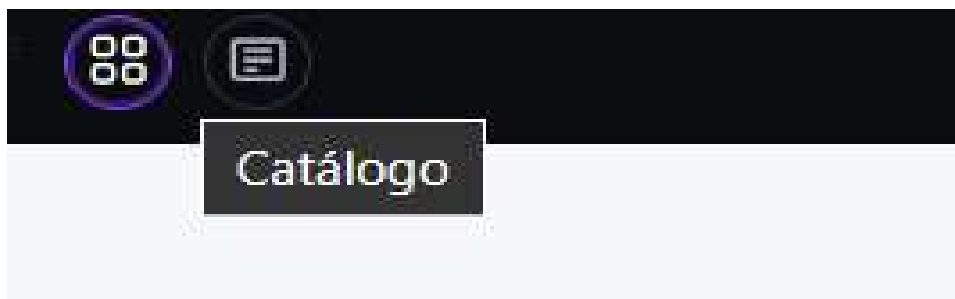
2.3. Catálogo y post

Catálogo

La plataforma cuenta con un catálogo avanzado que facilita la búsqueda de cartas.

Puedes usar filtros de búsqueda por tipo de carta, Juegos de mesa y Accesorios.

Botón para abrir el catálogo



Catálogo con todos los Productos:


Todos los productos

Alfabéticamente, A-Z


☐ Solo disponibles
 Mostrando 12 productos

Limpiar filtros


Todas Juegos de mesa TCG / Cartas Accesorios Lanzamiento
 Min \$ Máx \$




Aqualin
\$ 129.000




Binder Collection | Black Bolt & White Flare
\$ 160.000




Booster Box - ULTRA LIMIT - [FB04]
\$ 450.000




Booster Box Cyber Eden Digimon
\$ 500.000




Booster Box Duelist's Advance
\$ 500.000




Booster Box Justice Hunter
\$ 500.000




Booster Box Versus Monsters
\$ 500.000



BOOSTER PACK: Treasure Boosters Set
\$ 100.000



MTG: Universes Beyond | Final Fantasy Bundle
~~\$ 320.000~~ \$ 300.000




MTG: Universes Beyond | Final Fantasy Commander Deck Display (4 Motivos)
~~\$ 350.000~~ \$ 320.000

Producto seleccionado:

[← Volver al catálogo](#)

Disponible



TCG

Booster Box Duelist's Advance

\$ 500.000 COP

Cantidad

[Añadir al carrito](#)
[Comprar ahora](#)

[Descripción](#)
[Especificaciones](#)
[Envíos y devoluciones](#)

Expansión Duelist's Advance. Ideal para renovar tu deck.

Muestra información del producto (Tipo, estado, Precio, imagen) opciones de compra y añadir a carrito.

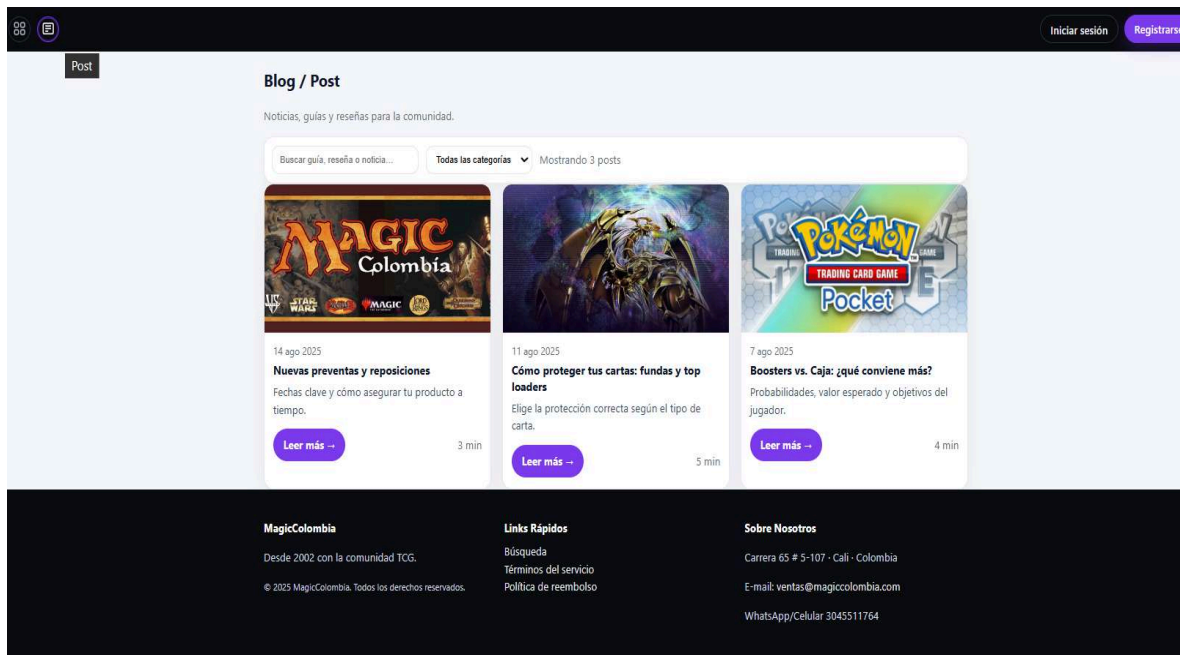
Post

La plataforma cuenta con un blog para mantener a la comunidad informada sobre lanzamientos, eventos y torneos. Este apartado lo llamamos “Post”

Botón de Post



Post con todas las Publicaciones:



3. Manual técnico

Este manual proporciona una visión detallada de los componentes, la arquitectura, el entorno, rutas, estructura de carpetas, modelos de datos, contrato de API, funciones-operaciones, reglas de imágenes, seguridad, Troubleshooting (solución de problemas), guía práctica-SQL, estándares-convenciones, checklists, anexos y runbook express (que hacer si...) del sistema web de MagicColombia.

1) Resumen

Cosmix es un mini-e-commerce para TCG con:

- ****Tienda**** (`/cosmix/html/tienda.html`) que lista productos desde una API pública.
- ****Panel Admin**** (`/cosmix/html/admin.html`) para crear/editar/borrar productos.
- ****APIs en PHP**** (bajo `/cosmix/magic_auth_php`) que leen/escriben en MySQL.
- ****Imágenes estáticas**** en `/cosmix/images/productos/` reforzadas por reglas simples de nombres.

****Idea clave:**** Todo gira en torno a la tabla `productos` y dos endpoints:

- ****Público (lista)**:** `/cosmix/magic_auth_php/productos_list.php`
- ****Privado (CRUD)**:** `/cosmix/magic_auth_php/admin/productos_api.php`

3.1. Arquitectura general y flujo (Admin-Productos-Tienda)

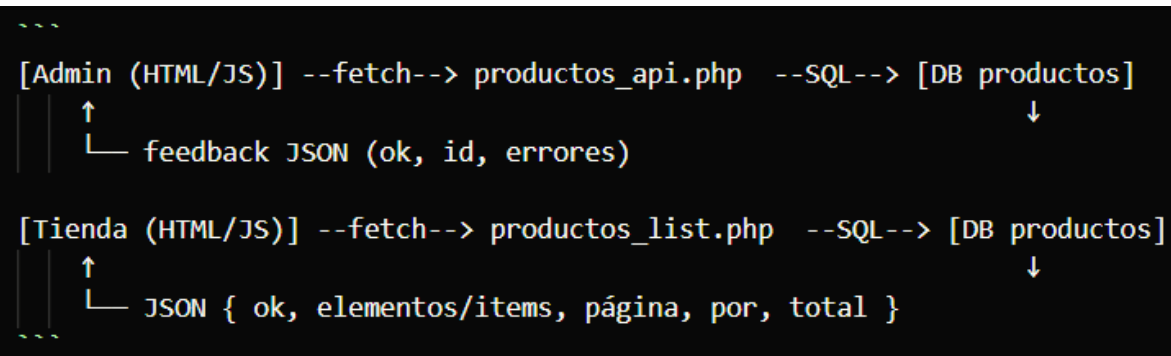


Diagrama de flujo para ilustrar los distintos componentes del software y su comunicación entre sí.

- Capa de presentación (Frontend): Interfaz con la que el cliente interactúa. **[Admin (HTML/JS)]** y la **[Tienda (HTML/JS)]**
- Capa de lógica de negocio (Backend/Servidor de aplicaciones): Lógica principal, las reglas de negocio y el procesamiento. **productos_api.php** y **productos_list.php**

- Capa de datos (Backend/Base de datos): Almacena y gestiona la información. **[DB productos]**

```

### Componentes
- **Frontend**: HTML + JS sin framework. La tienda renderiza un *grid* a partir del JSON. El panel abre un modal para CRUD.
- **Backend**: PHP puro con PDO. Un endpoint "público" (solo lectura) y otro "admin" (alta/edición/baja).
- **Base de datos**: MySQL/DB (XAMPP). Índice único en `sku`.
- **Archivos estáticos**: Imágenes en `/cosmix/images/productos/`.

```

3.2. Entornos y Rutas

Configuración y las rutas del proyecto web.

```

- **SO**: Windows (XAMPP).
- **Docroot**: `C:\xampp\htdocs\cosmix\`
- **Apps principales**:
  - Tienda: `http://localhost/cosmix/html/tienda.html`
  - Admin: `http://localhost/cosmix/html/admin.html`
- **APIs**:
  - Pública (GET): `http://localhost/cosmix/magic_auth_php/productos_list.php`
  - Admin (POST/GET): `http://localhost/cosmix/magic_auth_php/admin/productos_api.php`
- **Imágenes**: `C:\xampp\htdocs\cosmix\images\productos\` → **URL**: `/cosmix/images/productos/...`
> **Importante**: En **BD** siempre guarda **rutas web relativas** (p. ej. `/cosmix/images/productos/img.5.jpg`), **no** rutas de Windows.

```

→ Entorno de Servidor Local:

SO: Windows (XAMPP): desarrollo en sistema operativo Windows, utilizando XAMPP.

Docroot: **C:\xampp\htdocs\cosmix**: El "Docroot" (document root) carpeta principal donde se almacenan todos los archivos del sitio web.

→ Puntos de Acceso de la Aplicación (Rutas Web):

Tienda: **http://localhost/cosmix/html/tienda.html**: URL de la página principal de la tienda.

Admin: **http://localhost/cosmix/html/admin.html**: URL para la interfaz de administrador, donde puede gestionar los productos.

→ API y su Lógica:

Comunicación entre el frontend y el backend:

Pública (GET):

http://localhost/cosmix/magic_auth_php/productos_list.php: API para obtener la lista de productos.

Admin (POST/GET):

http://localhost/cosmix/magic_auth_php/admin/productos_api.php: Esta API se usa para la gestión de productos desde el panel de administración (crear, actualizar y eliminar productos)

→ Gestión de Archivos:

Imágenes: **C:\xampp\htdocs\cosmix\images\productos**: Ruta del sistema de archivos de Windows, para guardar físicamente las imágenes de los productos.

→ Concepto Clave (¡Importante!)

"En BD siempre guarda rutas web relativas (**p. ej. /cosmix/images/productos/img.5.jpg**), no rutas de Windows.": Las rutas web

relativas. Se aseguran que los enlaces funcionen independientemente del sistema operativo o la ubicación del servidor.

3.3. Estructura de Carpetas

```
cosmix/  
├── html/  
│   ├── tienda.html      # Tienda (pública)  
│   └── admin.html       # Panel de administración  
├── js/  
│   ├── tienda.js        # Lógica de la tienda (fetch, filtros, render)  
│   ├── data.products.js # Datos/semillas (legacy/fallback)  
│   ├── products.lanzamiento.js # Etiquetas/lanzamientos (opcional)  
│   └── site.enhancements.js # Ajustes UI comunes (opcional)  
├── images/  
│   └── productos/  
│       ├── img.1.jpg  
│       ├── img.2.jpg  
│       └── ...  
└── magic_auth_php/  
    ├── productos_list.php # API pública (GET)  
    └── admin/  
        └── productos_api.php # API admin (CRUD)
```

Mapa visual que organiza todos los componentes de la aplicación. Se detalla separación clara del frontend (**html**, **js**, **images**) y del backend (**magic_auth_php**).

3.4. Modelo de Datos

****Tabla `productos`** (campos usados hoy):**

Campo	Tipo	Notas
-----	-----	-----
`id`	INT AI PK	Identificador
`sku`	VARCHAR(50) UNIQUE	Clave de negocio, única y estable
`nombre`	VARCHAR(255)	Nombre mostrado
`precio`	DECIMAL(10,2)	En COP
`stock`	INT	Unidades disponibles
`img_url`	VARCHAR(255) NULL	Ruta web relativa a la imagen
`is_active`	TINYINT(1) DEFAULT 1	1 activo, 0 inactivo

> Si `img_url` es `NULL` o empty, la tienda puede construir una ruta por defecto con `img.{id}.jpg`.

Descripción de estructura de los datos de cada producto que la tienda vende.

3.5. Contratos API

```
### 6.1 Pública - **GET** `/cosmix/magic_auth_php/productos_list.php`

**Parámetros (opcional):**
- `page` (int), `per` (int).

**Respuesta real actual (ES):**
```json
{
 "ok": true,
 "elementos": [
 { "id": 3, "sku": "DM-START-01", "nombre": "Deck Inicio Digimon 01",
 "precio": "95000.00", "existencias": 8, "está_activo": 1, "img_url": "/cosmix/images/productos/img.1.jpg" }
],
 "página": 1,
 "por": 12,
 "total": 3
}
```

**Notas de compatibilidad (tienda.js):**
- Soporta tanto `elementos` (ES) como `items` (EN).
- Mapea claves a formato interno:
  - `stock` ← `existencias` | `stock`
  - `is_active` ← `está_activo` | `is_active`
```

```

### 6.2 Admin — **POST** `/cosmix/magic_auth_php/admin/productos_api.php`

**Acciones:**
- `action=list` (GET/POST): lista con paginación (mismo esquema que el público).
- `action=save` (POST): inserta o actualiza.
  - **Payload esperado:** `id?`, `sku`, `nombre`, `precio`, `stock`, `img_url?`, `is_active`
  - **Respuestas típicas:**
    - `200 { "ok": true, "id": <id_nuevo_o_existente> }`
    - `409 { "ok": false, "error": "duplicate", "msg": "SKU ya existe" }`
    - `500 { "ok": false, "error": "sql" | "server" }`
- `action=delete` (POST): elimina por `id` o `sku`.

> El modal de **admin.html** usa el campo `name="imagen_url"` pero al enviar se mapea a `img_url`.

---
```

Reglas de comunicación entre el frontend (el navegador) y el backend (los archivos PHP).

Detalla que se puede pedir, cómo se debe pedir y qué tipo de respuesta se puede esperar.

Con los contratos API es posible una comunicación correcta para que la aplicación funcione.

3.6. Reglas de imágenes

7) Reglas de imágenes

```

- **Carpeta**: `C:\xampp\htdocs\cosmix\images\productos\`
- **URL pública**: `/cosmix/images/productos/`
- **Convención recomendada**: `img.{id}.jpg` (minúsculas, sin espacios).
- En BD siempre **rutas web** (no caminos `C:\...`).

**Normalización de `img_url` (SQL útil):**
```sql
-- 0) Vacíos a NULL (higiene)
UPDATE productos SET img_url = NULL WHERE img_url IS NULL OR TRIM(img_url) = '';

-- 1) Backslashes de Windows → web
UPDATE productos SET img_url = REPLACE(img_url, '\\', '/') WHERE img_url LIKE '%\\%';

-- 2) Quita el prefijo local (si quedó pegado)
-- Con REGEXP_REPLACE (si tu motor lo soporta):
UPDATE productos SET img_url = REGEXP_REPLACE(img_url, '^[A-Za-z]:/xampp/htdocs', '')
WHERE img_url REGEXP '^[A-Za-z]:/xampp/htdocs';

-- Alternativa sin REGEXP (más simple):
UPDATE productos SET img_url = REPLACE(img_url, 'c:/xampp/htdocs', '')
WHERE img_url LIKE 'c:/xampp/htdocs%';

-- 3) Asegura un único slash inicial
UPDATE productos SET img_url = CONCAT('/', TRIM(LEADING '/' FROM img_url))
WHERE img_url IS NOT NULL AND img_url <> '';

-- 4) Homogeneiza nombres tipo img_12.jpg → img.12.jpg (si aplica)
UPDATE productos SET img_url = REPLACE(img_url, 'img_', 'img.')
WHERE img_url LIKE '%/img_%';
```

```

Detalle de las reglas de almacenamiento de imágenes y proporciona las consultas SQL necesarias para mantener la consistencia e integridad de las rutas de las imágenes en la base de datos.

3.7. Frontend: IDs y comportamiento

```

### 8.1 Tienda (`tienda.html` + `js/tienda.js`)
- Elementos clave: `#q` (buscador), `#sort` (orden), `#onlyInStock` (toggle), `#refresh` (recargar), `#product-grid` (contenedor).
- Carga de datos: intenta API → si falla, usa fallbacks locales (`data.products.js`, etc.).
- Mapeo de JSON: convierte las llaves ES6EN y asegura `price:number`, `stock:number`.
- Imagen: usa `row.img_url` si viene; si no, construye `/cosmix/images/productos/img.{id}.jpg`; y si tampoco existe, usa un placeholder (opcional).

### 8.2 Admin (`admin.html`)
- Modal: campos `nombre`, `sku`, `precio`, `stock`, `imagen_url` (→ `img_url`), `activo` (→ `is_active`).
- Flujo: `Guardar` → `productos_api.php?action=save` → si `ok`, refresca tabla y opcionalmente redirige a tienda.
- Validaciones mínimas: SKU único, números en precio/stock.

```

Hoja de ruta para entender el comportamiento y la lógica de las dos interfaces de usuario del proyecto "Cosmix": la tienda pública y el panel de administración.

3.8. Operaciones comunes

```

## 9) Operaciones comunes

### 9.1 Crear un producto (Admin)
1. Abre admin.html → + Nuevo Producto.
2. Completa el formulario:
   - SKU (único): ejemplo OP-TREASURE-001.
   - Nombre, Precio (sin separador de miles en BD), Stock.
   - URL imagen: p. ej. /cosmix/images/productos/img.20.jpg.
   - Activo: marcado.
3. Guardar. Si ves duplicate, cambia el SKU.

### 9.2 Editar
- Abre el mismo modal con los valores existentes, ajusta y guarda.

### 9.3 Eliminar
- Desde Admin ejecuta action=delete (o usa SQL más abajo).

### 9.4 Ver en Tienda
- Abre tienda.html y pulsa Refrescar. Desactiva "Solo disponibles" si el stock es 0.

```

Explicación de cómo realizar las operaciones más comunes en la aplicación "Cosmix" a través del panel de administración.

3.9. Seguridad (local)

10) Seguridad (local)

- **Admin** vive bajo `/magic_auth_php/admin/` y el **fetch** se hace con `credentials:'include'`.
- No se exponen credenciales ni cookies a dominios externos (misma `*origin*`).
- Mantén XAMPP sólo local; para producción usarías HTTPS, sesiones endurecidas, WAF, etc.

Manejo de la seguridad básica del panel de administración en un entorno local, al tiempo una advertencia de que esta configuración no es adecuada para un entorno de producción.

3.10. Troubleshooting (solución de problemas)

11) Troubleshooting rápido

| Síntoma | Causa probable | Solución |
|---|--|--|
| ***No sale la imagen** | <code>`img_url`</code> guarda ruta de Windows o sin <code>`/`</code> inicial | Ejecuta el bloque de <i>Normalización de `img_url`</i> arriba. Verifica archivo en <code>/cosmix/images/productos/`</code> . |
| ***"Unexpected token <" al parsear JSON** | PHP lanzó un HTML de error (500) | Abre <i>Network</i> → respuesta de <code>productos_api.php`</code> , revisa <code>php_error_log`</code> . Confirma <code>`action`</code> y payloads. |
| ***duplicate (409)** | SKU ya existe | Cambia SKU o edita el existente. |
| ***No veo el producto** | Filtro "Solo disponibles" activado y <code>`stock=0`</code> | Desmarca el toggle o aumenta stock. |
| ***404 en `productos_list.php`** | Ruta mal escrita | Asegura <code>/cosmix/magic_auth_php/productos_list.php`</code> |
| ***CORS/cookies** | Llamas desde otro origen | Trabaja siempre en <code>`localhost`</code> . |

Guía rápida de solución de problemas (troubleshooting) para errores comunes que pueden ocurrir en el proyecto web.

Formato tabla de tres columnas: Síntoma (el problema), Causa probable y Solución.

Esta guía es una herramienta práctica, para que el desarrollador pueda diagnosticar y resolver rápidamente los fallos más comunes durante el desarrollo del proyecto.

3.11. SQL - guía práctica

12) SQL – Guía práctica

> Ejecuta en ****phpMyAdmin**** sobre la BD **`cosmix`**.

12.1 Búsquedas de productos

```sql

-- General (los más recientes primero)

```
SELECT id, sku, nombre, precio, stock, img_url, is_active
FROM productos
ORDER BY id DESC
LIMIT 50;
```

-- Por SKU exacto

```
SELECT *
FROM productos
WHERE sku = 'OP-TREASURE-001';
```

-- Por nombre (contiene, insensible a may/min)

```
SELECT *
FROM productos
WHERE nombre LIKE '%booster%'
ORDER BY nombre ASC;
```

-- Sólo activos y con stock

```
SELECT *
FROM productos
WHERE is_active = 1 AND stock > 0
ORDER BY id DESC;
```
```


12.2 Insertar / actualizar / borrar

```

```sql
-- Insert directo
INSERT INTO productos (sku, nombre, precio, stock, img_url, is_active)
VALUES ('OP-TREASURE-001', 'Booster Pack Treasure', 100000, 5,
| | | | '/cosmix/images/productos/img.20.jpg', 1);

-- Update por SKU
UPDATE productos
SET precio = 99000, stock = 3
WHERE sku = 'OP-TREASURE-001';

-- Borrar (por SKU)
DELETE FROM productos
WHERE sku = 'OP-TREASURE-001';

-- Borrar (por ID)
DELETE FROM productos
WHERE id = 123;
```

```

12.3 Usuarios (tabla `usuarios`)

```

```sql
-- Vista rápida (asumiendo campos comunes)
SELECT *
FROM usuarios
ORDER BY id DESC
LIMIT 100;
```

```

12.4 Mantenimiento y *hardening*

```

```sql
-- Asegura columnas (idempotente; ignora si ya existen)
ALTER TABLE productos
| ADD COLUMN IF NOT EXISTS img_url VARCHAR(255) NULL AFTER stock,
| ADD COLUMN IF NOT EXISTS is_active TINYINT(1) NOT NULL DEFAULT 1 AFTER img_url;

-- Índice único en SKU (idempotente)
ALTER TABLE productos
| ADD UNIQUE INDEX IF NOT EXISTS uq_productos_sku (sku);
```

```

3.12. Estándares y convenciones

13) Estándares y convenciones

- ****SKU****: mayúsculas, guiones medios, fijo en el tiempo. Ej.: `MTG-MOD-ETB`.
- ****Precios****: en BD como `DECIMAL(10,2)` (sin puntos de miles).
- ****Rutas****: siempre ****web relativas**** con ****slash inicial****: `/cosmix/images/productos/img.1.jpg`.
- ****Imágenes****: minúsculas, sin espacios, `img.{id}.jpg` preferido.

3.13. Checklists

14) Checklists

Antes de “subir” productos

- [] Imagen copiada a `/cosmix/images/productos/` con nombre correcto.
- [] `img_url` con slash inicial y sin `C:\`.
- [] SKU único.
- [] `is_active=1` y stock correcto.

Al cerrar una sesión de alta masiva

- [] Ejecutar ****Normalización de `img_url`****.
- [] Verificar en tienda con ****Refrescar**** y filtros.

Listas de verificación (checklists) que un desarrollador o un administrador debe seguir para asegurar que la gestión de productos se realiza de forma correcta.

Se encuentra un conjunto de buenas prácticas para garantizar la calidad y la coherencia de los datos del proyecto.

3.14. Anexos

```
## 15) Anexos

### 15.1 Ejemplos de `fetch`

**Tienda (GET lista pública):**
```js
fetch('/cosmix/magic_auth_php/productos_list.php', { credentials: 'include' })
 .then(r => r.json())
 .then(j => console.log(j));
```

**Admin (save):**
```js
const fd = new FormData();
fd.append('action', 'save');
fd.append('sku', 'OP-TREASURE-001');
fd.append('nombre', 'Booster Pack Treasure');
fd.append('precio', '100000');
fd.append('stock', '5');
fd.append('img_url', '/cosmix/images/productos/img.20.jpg');
fd.append('is_active', '1');

fetch('/cosmix/magic_auth_php/admin/productos_api.php', {
 method: 'POST',
 body: fd,
 credentials: 'include'
}).then(r => r.json()).then(console.log);
```
```

```
### 15.2 Glosario mínimo
- **SKU**: Identificador comercial único por producto.
- **Stock**: Unidades disponibles; si está en 0 y activado “Solo disponibles”, no se muestra.
- **is_active**: Visible (=1) o no visible (=0) en el catálogo.
```

Ejemplos de código JavaScript y un pequeño glosario.

- Ejemplos de fetch: Tienda (GET) y Admin (save)
- Glosario Mínimo: SKU, Stock y is_active

El documento proporciona ejemplos prácticos de cómo el código del frontend se comunica con el backend y define algunos términos clave del proyecto.

3.15. Runbook express (que hacer si...)

16) Runbook express (qué hacer si...)

- 1) ****Subí imagen y no se ve**** → comprobar que existe en `/cosmix/images/productos/`, ejecutar normalización de `img_url`, refrescar tienda.
- 2) ****Admin dice duplicate**** → buscar por SKU, decidir actualizar o cambiar SKU.
- 3) ****La tienda quedó vacía**** → abrir consola/red, probar `productos_list.php`, revisar que responde `{ ok:true }`.
- 4) ****Error JSON (Unexpected `<`)**** → el PHP devolvió HTML de error; abrir respuesta en Network, revisar logs PHP.

Guía rápida de acciones a seguir para resolver cuatro problemas comunes del proyecto.

4.1. Modelos y estándares de calidad

- ISO/IEC 25010: Utilizada como marco de evaluación de calidad del software.

Define los requisitos no funcionales del sistema, como la usabilidad, la seguridad y la eficiencia de rendimiento.

- Modelo CMMI: Se eligió porque está diseñado para el desarrollo de software y se adapta perfectamente a un proyecto ya existente que se planea mejorar. Este modelo nos permite centrarnos en la calidad, planificación y mejora continua del sitio web.

4.2. Requerimientos funcionales y no funcionales

El sistema fue diseñado para cumplir con los siguientes requerimientos, derivados del análisis del negocio y la norma ISO 25010:

Funcionales (RF): Permite el registro de clientes (RF1), la gestión de usuarios (RF5), administración de inventario dinámico (RF12), proporcionar un catálogo con filtros avanzados por tipo de producto (RF7), un blog (RF9), gestionar inventario dinámico para evitar la venta de productos sin stock (RF11) y un panel administrativo (RF10).

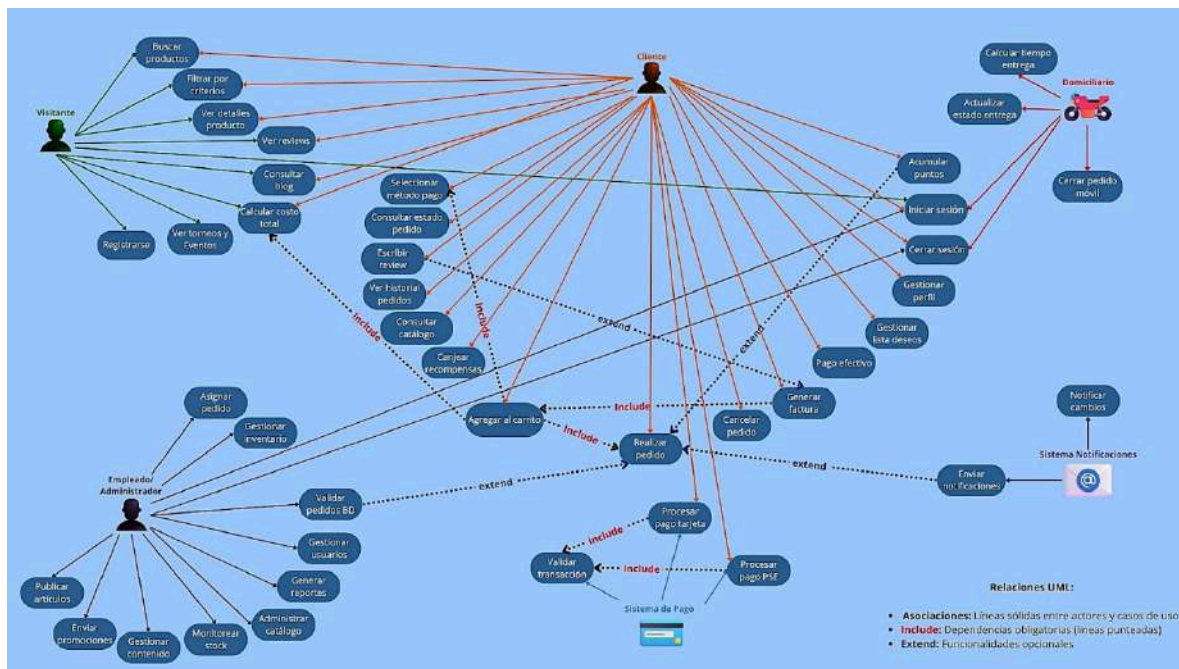
No funcionales (RNF): El sistema debe ser fácil de usar (RNF1), responsivo (RNF2), disponible 24/7 (RNF3), permitir mantenimiento y ajustes futuros (RNF6), y el diseño debe reflejar la identidad visual y los valores de la marca (RNF7). Se garantiza una carga de páginas en menos de 3 segundos (RNF11) y se cuenta con filtros de catálogo que deben responder en menos de 2 segundos (RNF9).

5. Diseño y modelado

Para facilitar la comprensión del sistema, se crearon los siguientes diagramas.

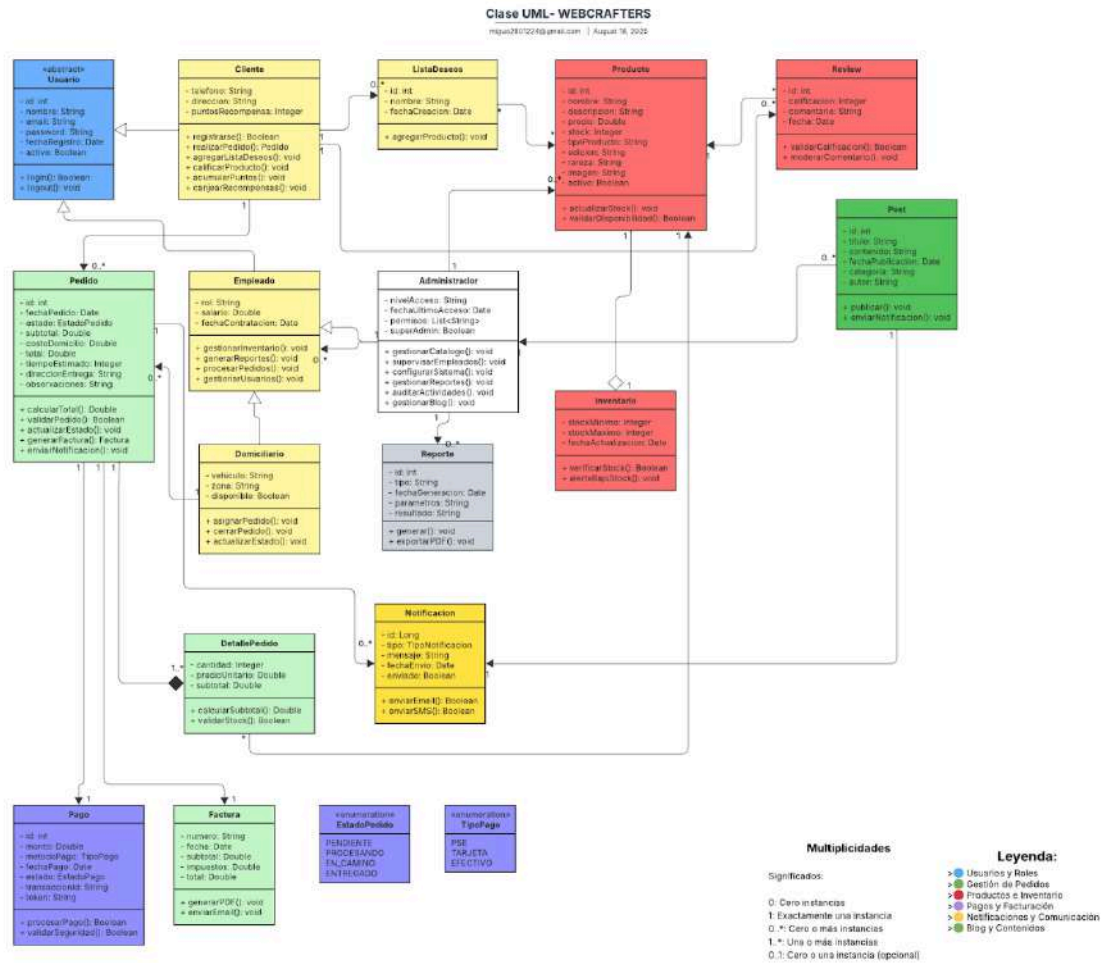
Diagrama de Casos de Uso: Muestra las interacciones entre los actores (cliente, visitante, domiciliario, empleado/administrador) y las funcionalidades del sistema. * Diagrama de Clases: Presenta la estructura estática del sistema, mostrando las clases, sus atributos, métodos y relaciones. * Modelo relacional: Representa la estructura de la base de datos, detallando las tablas, los campos y sus relaciones. * Wireframes: Muestran la estructura visual de las interfaces principales, como la página de inicio y el formulario de inicio de sesión.

Diagrama de uso



Link de diagrama: [DIAGRAMA DE CASOS DE USO webcrafters.pdf](#)

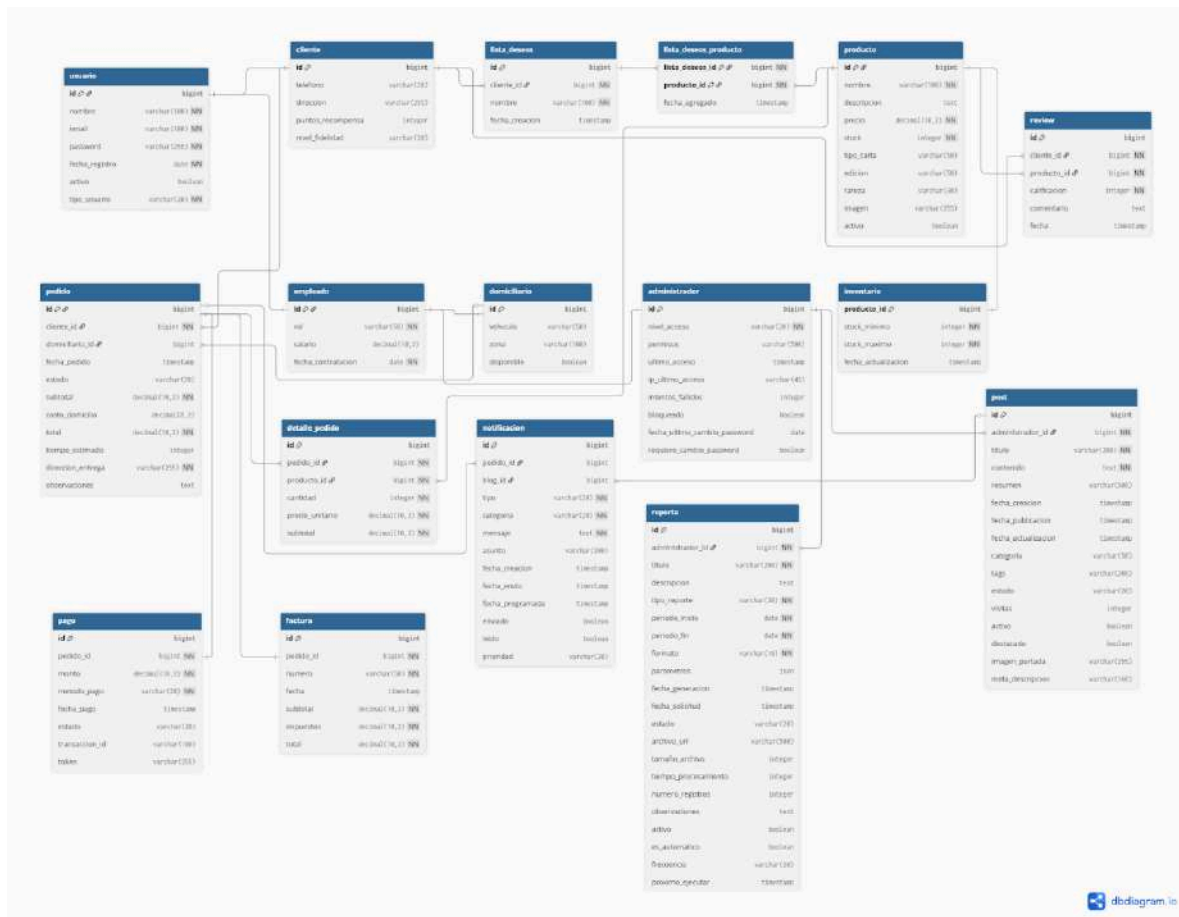
Diagrama de clase



Link lucidchart -Diagrama de clase: [Lucidchart document](#)

Link Diagrama de Clase PDF: [Clase UML Nuevo.pdf](#)

MR Modelo Relacional



Link del MR ME BD: [MODELO ME.pdf](#)

6. Conclusión

La implementación de la nueva plataforma web para Magic Colombia representa un paso significativo hacia la modernización tecnológica del emprendimiento. Al adoptar un enfoque metodológico riguroso basado en estándares como ISO/IEC 25000 y CMMI, se ha garantizado la calidad del software y una mejora continua del proceso de desarrollo.

El rediseño no solo optimiza los procesos internos y la gestión de datos, sino que también mejora la experiencia del usuario final, aumentando la competitividad de MagicColombia en el mercado. Con funcionalidades esenciales como el catálogo avanzado y los múltiples métodos de pago, el sistema está preparado para ofrecer una experiencia de compra segura y eficiente, consolidando a MagicColombia como un modelo de modernización en su sector.