



UNIVERSIDAD DE COSTA RICA

UNIVERSIDAD DE COSTA RICA

SEDE DEL CARIBE

CURSO:
ANÁLISIS Y DISEÑO DE SISTEMAS

DOCENTE:

ROLANDO HERRERA SÁNCHEZ

MANUAL TECNICO

INTEGRANTES DEL GRUPO:

TRISTAN MÉNDEZ MAYORGA

RODRIGO RODRIGUEZ UMAÑA

OSCAR FARGAS BRENES

KEINTH VIALES REYES

FECHA:

30/09/2025

introducción

El proyecto OKAMI es un sistema web de escritorio diseñado específicamente para optimizar y centralizar la gestión de ventas de un emprendimiento de suplementos deportivos. Construido sobre una arquitectura robusta, utilizamos SQL Server como motor de base de datos para garantizar la integridad y el manejo eficiente de datos críticos como el inventario y las transacciones. La lógica del *backend* se desarrolla con Python, aprovechando su simplicidad y potencia para gestionar las operaciones del servidor. En el *frontend*, la interfaz de usuario se construye utilizando la tríada fundamental de HTML (para la estructura), CSS (para el diseño y la estética) y JavaScript (para la interactividad y la lógica del cliente), asegurando una experiencia fluida y reactiva. El desarrollo inicial y las pruebas de funcionalidad se realizan en un entorno controlado con una dirección de acceso local (como localhost), preparando el sistema para un despliegue y uso comercial eficiente.

1. Requerimientos y Matriz de Trazabilidad

CU-ID	Requerimiento de Usuario	Tipo	Lógica de Implementación
CU01	Visualizar Catálogo (Público)	Funcional	GET /api/productos (Backend) y loadProducts() (Frontend).
CU02	Filtrar Productos por Categoría	Funcional	handleFilterClick() (Frontend, filtro en memoria).
CU03	Ver Detalles de Producto (Modal)	Funcional	openProductModal(product) (Frontend).
CU04	Seleccionar Variante y Stock	Funcional	GET /api/inventario/<id> y fillModalStock() (Frontend).
CU05	Enviar Pedido por WhatsApp	Funcional	redirectToWhatsApp() (Frontend).
CU06	Iniciar Sesión (Maestro)	Funcional	POST /login (Backend, autenticación con cookie) y handleLogin() (Frontend).
CU07	Cerrar Sesión	Funcional	POST /logout (Backend) y handleLogout() (Frontend).
CU08	CRUD: Listar Productos (Admin)	Funcional	renderAdminProductList() (Frontend) basado en loadProducts().
CU09	CRUD: Crear Nuevo Producto	Funcional	POST /api/productos (Backend, subida de imagen con FormData) y handleNewProductSubmit() (Frontend).
CU10	CRUD: Editar Producto	Funcional	PUT /api/productos/<id> (Backend, soporta FormData con/sin nueva imagen) y openEditProductForm() (Frontend).

CU-ID	Requerimiento de Usuario	Tipo	Lógica de Implementación
CU11	CRUD: Eliminar Producto	Funcional	DELETE /api/productos/<id> (Backend, borrado lógico) y apiDeleteProduct() (Frontend).
CU12	Gestión de Archivos	No Funcional	Backend: Manejo de la subida de imágenes a la carpeta /uploads.
CU13	Seguridad de Sesión	No Funcional	Backend: Uso de cookies con session para mantener el estado de autenticación.

2. Capa de Persistencia: Esquema de Base de Datos (SQL)

-- TABLA DE USUARIOS (MAESTRO)

```
CREATE TABLE usuarios (
    idusuario INTEGER PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL,
    contrasena VARCHAR(255) NOT NULL, -- Se debe almacenar el hash
    rol VARCHAR(50) DEFAULT 'admin'
);
```

-- TABLA DE CATEGORÍAS (Artículos / Filtros)

```
CREATE TABLE categorias (
    idcategoria INTEGER PRIMARY KEY,
    nombrecategoria VARCHAR(100) UNIQUE NOT NULL
);
```

-- TABLA DE PRODUCTOS

```
CREATE TABLE productos (
    idproducto INTEGER PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    preciobase REAL NOT NULL,
    descripcion TEXT,
    idcategoria INTEGER NOT NULL,
    enoferta BOOLEAN DEFAULT 0,
    esborrado BOOLEAN DEFAULT 0,
    urlimagen VARCHAR(255),
    FOREIGN KEY (idcategoria) REFERENCES categorias(idcategoria)
);
```

-- TABLA DE VARIANTES (Ej. Peso, Sabor, Contenido)

```
CREATE TABLE variantes (
    idvariante INTEGER PRIMARY KEY,
    presentacion VARCHAR(100), -- Ej. 1LB, 5LB, Chocolate, Vainilla
    contenido VARCHAR(100), -- Tipo de contenido
    sabor VARCHAR(100) -- Sabor
);
```

-- TABLA DE INVENTARIO (Relación Producto-Variante-Stock)

```
CREATE TABLE inventario (
    idproducto INTEGER NOT NULL,
    idvariante INTEGER NOT NULL,
    cantidadstock INTEGER NOT NULL DEFAULT 0,
    PRIMARY KEY (idproducto, idvariante),
```

```
FOREIGN KEY (idproducto) REFERENCES productos(idproducto) ON DELETE CASCADE,  
FOREIGN KEY (idvariante) REFERENCES variantes(idvariante)  
);
```

3. Lógica del Negocio (Backend): Código de Flask (app.py)

```
import os  
  
from flask import Flask, request, jsonify, session, redirect, url_for  
from werkzeug.security import generate_password_hash, check_password_hash  
from werkzeug.utils import secure_filename  
from datetime import timedelta  
import sqlite3  
  
# ===== CONFIGURACIÓN INICIAL =====  
  
UPLOAD_FOLDER = 'static/uploads'  
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}  
DATABASE = 'database.db'  
  
app = Flask(__name__)  
app.secret_key = 'tu_clave_secreta_aqui' # Cambiar por una clave fuerte  
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER  
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=60) # Sesión expira en 60 mins  
  
# Asegurar que la carpeta de uploads exista  
os.makedirs(UPLOAD_FOLDER, exist_ok=True)  
  
# ===== FUNCIONES DE BASE DE DATOS =====
```

```
def get_db():

    """Conecta o crea la base de datos y retorna la conexión."""

    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    return conn


def init_db():

    """Inicializa la base de datos con las tablas y datos iniciales."""

    with app.app_context():

        db = get_db()
        cursor = db.cursor()

        # Esquema (mantenido del manual tecnico)
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS usuarios (
                idusuario INTEGER PRIMARY KEY,
                nombre VARCHAR(100) NOT NULL,
                correo VARCHAR(100) UNIQUE NOT NULL,
                contrasena VARCHAR(255) NOT NULL,
                rol VARCHAR(50) DEFAULT 'admin'
            );
        """)

        cursor.execute("""
            CREATE TABLE IF NOT EXISTS categorias (
                idcategoria INTEGER PRIMARY KEY,
                nombrecategoria VARCHAR(100) UNIQUE NOT NULL
        """)
```

```
    );
  ")
cursor.execute("

CREATE TABLE IF NOT EXISTS productos (
    idproducto INTEGER PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    preciobase REAL NOT NULL,
    descripcion TEXT,
    idcategoria INTEGER NOT NULL,
    enoferta BOOLEAN DEFAULT 0,
    esborrado BOOLEAN DEFAULT 0,
    urlimagen VARCHAR(255),
    FOREIGN KEY (idcategoria) REFERENCES categorias(idcategoria)
);

")
cursor.execute("

CREATE TABLE IF NOT EXISTS variantes (
    idvariante INTEGER PRIMARY KEY,
    presentacion VARCHAR(100),
    contenido VARCHAR(100),
    sabor VARCHAR(100)
);

")
cursor.execute("

CREATE TABLE IF NOT EXISTS inventario (
    idproducto INTEGER NOT NULL,
    idvariante INTEGER NOT NULL,

```

```
cantidadstock INTEGER NOT NULL DEFAULT 0,  
PRIMARY KEY (idproducto, idvariante),  
FOREIGN KEY (idproducto) REFERENCES productos(idproducto) ON  
DELETE CASCADE,  
FOREIGN KEY (idvariante) REFERENCES variantes(idvariante)  
);  
")
```

```
# Insertar datos iniciales si no existen  
if cursor.execute('SELECT COUNT(*) FROM usuarios').fetchone()[0] == 0:  
    hashed_password = generate_password_hash('master123')  
    cursor.execute("INSERT INTO usuarios (nombre, correo, contrasena) VALUES  
    (?, ?, ?)",  
    ('Maestro Admin', 'admin@okami.cr', hashed_password))  
  
if cursor.execute('SELECT COUNT(*) FROM categorias').fetchone()[0] == 0:  
    categorias_iniciales = ['Proteínas', 'Creatinas', 'Mass Gainers', 'Aminoácidos',  
    'Pre-Entrenos', 'Glutaminas', 'Shaker y bebidas', 'Colageno']  
    for cat in categorias_iniciales:  
        cursor.execute("INSERT INTO categorias (nombrecategoria) VALUES (?)",  
        (cat,))  
  
if cursor.execute('SELECT COUNT(*) FROM variantes').fetchone()[0] == 0:  
    variantes_iniciales = [  
        ('1LB', 'Polvo', 'Chocolate'),  
        ('2LB', 'Polvo', 'Vainilla'),  
        ('Cápsulas', 'Píldoras', 'Neutro'),  
    ]
```

```
        for pres, cont, sab in variantes_iniciales:
            cursor.execute("INSERT INTO variantes (presentacion, contenido, sabor)
VALUES (?, ?, ?)", (pres, cont, sab))

            db.commit()

# ====== HELPER DE AUTENTICACIÓN ======
=====

def is_authenticated():
    """Verifica si el usuario está logeado y retorna True o False."""
    return 'user_id' in session

def login_required(f):
    """Decorador para rutas que requieren inicio de sesión."""
    def wrapper(*args, **kwargs):
        if not is_authenticated():
            # Devuelve JSON de error para peticiones API
            return jsonify({"success": False, "message": "Acceso denegado. Se requiere autenticación."}), 401
        return f(*args, **kwargs)

        wrapper.__name__ = f.__name__ # para que el decorador funcione correctamente en Flask
    return wrapper

# ====== RUTA PRINCIPAL Y SERVICIO DE ESTÁTICOS ======
=====

@app.route('/')

```

```

def index():
    """Ruta que renderiza la plantilla principal (el Frontend)."""
    # En un entorno real, usaríamos render_template('index.html')
    # Aquí solo servimos el archivo HTML estático (aunque el HTML usa url_for de Jinja)
    return app.send_static_file('index.html')

# ===== ENDPOINTS DE AUTENTICACIÓN (CU06, CU07)
=====

@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    correo = data.get('correo')
    contrasena = data.get('contrasena')

    if not correo or not contrasena:
        return jsonify({"success": False, "message": "Faltan correo o contraseña"}), 400

    db = get_db()
    user = db.execute("SELECT * FROM usuarios WHERE correo = ?",
                      (correo,)).fetchone()
    db.close()

    if user and check_password_hash(user['contrasena'], contrasena):
        session.clear()
        session['user_id'] = user['idusuario']
        session['user_role'] = user['rol']
        session.permanent = True # Usar el timeout configurado globalmente

```

```

        return jsonify({"success": True, "message": "Inicio de sesión exitoso."})

    else:
        return jsonify({"success": False, "message": "Credenciales incorrectas"}), 401

@app.route('/logout', methods=['POST'])

def logout():
    session.clear()
    return jsonify({"success": True, "message": "Sesión cerrada."})

@app.route('/api/validar-sesion', methods=['GET'])

def validar_sesion():
    if is_authenticated():

        return jsonify({"success": True, "message": "Sesión activa.", "user_id": session['user_id']})

    else:

        return jsonify({"success": False, "message": "No hay sesión activa."}), 401

# ===== ENDPOINTS DE CATÁLOGO (CU01, CU08)
=====

@app.route('/api/productos', methods=['GET'])

def get_productos():

    """Obtiene todos los productos para la vista pública y admin."""

    db = get_db()

    # Query para traer productos NO borrados, con su categoría
    query = """
        SELECT p.*, c.nombrecategoría
        FROM productos p
    """

    return jsonify({
        "success": True,
        "message": "Productos obtenidos",
        "data": db.execute(query).fetchall()
    })

```

```
JOIN categorias c ON p.idcategoria = c.idcategoria
WHERE p.esborrado = 0
ORDER BY p.nombre
"""

productos = db.execute(query).fetchall()
db.close()

productos_list = [dict(p) for p in productos]
return jsonify({"success": True, "data": productos_list})

@app.route('/api/categorias', methods=['GET'])
def get_categorias():
    """Obtiene la lista de categorías."""
    db = get_db()
    categorias = db.execute("SELECT * FROM categorias").fetchall()
    db.close()
    return jsonify({"success": True, "data": [dict(c) for c in categorias]})

@app.route('/api/variantes', methods=['GET'])
def get_variantes():
    """Obtiene la lista de variantes para poblar el formulario CRUD."""
    db = get_db()
    variantes = db.execute("SELECT * FROM variantes").fetchall()
    db.close()
    return jsonify({"success": True, "data": [dict(v) for v in variantes]})
```

```

@app.route('/api/inventario/<int:id_producto>', methods=['GET'])

def get_inventario_producto(id_producto):
    """Obtiene el stock por variante de un producto (CU04)."""

    db = get_db()
    query = """
        SELECT i.cantidadstock, v.presentacion, v.contenido, v.sabor
        FROM inventario i
        JOIN variantes v ON i.idvariante = v.idvariante
        WHERE i.idproducto = ?
    """

    inventario = db.execute(query, (id_producto,)).fetchall()
    db.close()

    if not inventario:
        return jsonify({"success": False, "message": "Inventario no encontrado para este producto"}), 404

    return jsonify({"success": True, "data": [dict(i) for i in inventario]})

# ===== ENDPOINTS CRUD (CU09, CU10, CU11, CU12)
=====

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/api/productos', methods=['POST'])

@login_required

def create_product():

```

```
"""Crea un nuevo producto (soporta JSON o FormData)."""

# Intenta obtener datos de FormData (para manejo de archivos)
nombre = request.form.get('nombre')
precio_base = request.form.get('precioBase')
descripcion = request.form.get('descripcion')
id_categoria = request.form.get('idCategoria')

# Manejo de imagen
file = request.files.get('imagenFile')
if file and allowed_file(file.filename):
    filename = secure_filename(file.filename)
    # Usamos el nombre del archivo para almacenar, la URL real es:
    /uploads/<filename>
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    file.save(filepath)
    url_imagen = filename # Almacenamos solo el nombre para referencia en la DB
else:
    return jsonify({"success": False, "message": "Archivo de imagen inválido o faltante"}), 400

if not all([nombre, precio_base, id_categoria]):
    return jsonify({"success": False, "message": "Faltan datos requeridos"}), 400

try:
    db = get_db()
    cursor = db.cursor()
    cursor.execute("""
```

```
    INSERT INTO productos (nombre, preciobase, descripcion, idcategoria,
urlimagen)

        VALUES (?, ?, ?, ?, ?)

    """", (nombre, float(precio_base), descripcion, int(id_categoria), url_imagen))

db.commit()

db.close()

return jsonify({"success": True, "message": "Producto creado correctamente",
"idproducto": cursor.lastrowid}), 201

except Exception as e:

    print(f"Error al crear producto: {e}")

    return jsonify({"success": False, "message": f"Error interno: {str(e)}"}), 500

@app.route('/api/productos/<int:id_producto>', methods=['PUT'])

@login_required

def update_product(id_producto):

    """Actualiza un producto existente (soporta JSON o FormData)."""

    # Los datos siempre vienen de FormData para soportar la imagen
    nombre = request.form.get('nombre')
    precio_base = request.form.get('precioBase')
    descripcion = request.form.get('descripcion')
    id_categoria = request.form.get('idCategoria')

    if not all([nombre, precio_base, id_categoria]):

        return jsonify({"success": False, "message": "Faltan datos requeridos"}), 400
```

```
db = get_db()
cursor = db.cursor()

# 1. Manejo de imagen (si se envía una nueva)
url_imagen = None
file = request.files.get('imagenFile')
if file and allowed_file(file.filename):
    filename = secure_filename(file.filename)
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    file.save(filepath)
    url_imagen = filename # Nuevo nombre de archivo

# 2. Construir la consulta de actualización
update_fields = [
    "nombre = ?",
    "preciobase = ?",
    "descripcion = ?",
    "idcategoria = ?"
]
update_values = [nombre, float(precio_base), descripcion, int(id_categoria)]

if url_imagen:
    update_fields.append("urlImagen = ?")
    update_values.append(url_imagen)

update_values.append(id_producto) # El ID del producto para el WHERE

query = f"UPDATE productos SET {', '.join(update_fields)} WHERE idproducto = ?"

try:
```

```
        cursor.execute(query, tuple(update_values))
        db.commit()

    if cursor.rowcount == 0:
        db.close()
        return jsonify({"success": False, "message": "Producto no encontrado"}), 404

    db.close()
    return jsonify({"success": True, "message": "Producto actualizado correctamente"})

except Exception as e:
    db.close()
    print(f"Error al actualizar producto: {e}")
    return jsonify({"success": False, "message": f"Error interno: {str(e)}"}), 500

@app.route('/api/productos/<int:id_producto>', methods=['DELETE'])
@login_required
def delete_product(id_producto):
    """Realiza un borrado lógico de un producto (CU11)."""
    try:
        db = get_db()
        cursor = db.cursor()
        # Borrado Lógico: esborrado = 1
        cursor.execute("UPDATE productos SET esborrado = 1 WHERE idproducto = ?",
                      (id_producto,))
        db.commit()
    except Exception as e:
        db.close()
        print(f"Error al realizar el borrado lógico: {e}")
        return jsonify({"success": False, "message": f"Error interno: {str(e)}"}), 500
```

```

if cursor.rowcount == 0:

    db.close()

    return jsonify({"success": False, "message": "Producto no encontrado"}), 404


db.close()

return jsonify({"success": True, "message": "Producto marcado como eliminado
(borrado lógico)."}), 200

except Exception as e:

    print(f"Error al eliminar producto: {e}")

    return jsonify({"success": False, "message": f"Error interno: {str(e)}"}), 500


# ===== INICIO DE LA APLICACIÓN
=====

if __name__ == '__main__':
    # Inicializar la DB y poblar datos si es la primera vez
    init_db()
    # Ejecutar la aplicación
    app.run(debug=True, host='127.0.0.1', port=5000)

4. Lógica del Cliente (Frontend): Código JavaScript (script.js o script2.js)

// script.js: LÓGICA PRINCIPAL DEL E-COMMERCE Y PANEL DE
ADMINISTRACIÓN (CONEXIÓN A API)

// =====
=====

// 0. CONFIG & ESTADO GLOBAL

```

```
const API_BASE = "http://127.0.0.1:5000";  
const WHATSAPP_NUMBER = '50688887777';  
const ALL_VARIANTS = ['Disponibles']; // si en el futuro traes variantes reales, se  
sustituye  
const AUTH_TOKEN_KEY = 'masterToken'; // ya no usado para la auth real pero lo  
mantenemos por compatibilidad  
  
// Estado en memoria (será llenado desde la API)  
let products = [];  
let categories = [];  
let variants = [];  
  
// Selectores DOM (igual que tu versión original)  
const DOM = {  
    // Cliente  
    productsGrid: document.querySelector('.products-grid'),  
    productsGridContainer: document.querySelector('.products-grid-container'),  
    filterLinks: document.querySelectorAll('.filter-nav a.filter-link'),  
    filterNavContainer: document.querySelector('.filter-nav-container'),  
    floatingActions: document.querySelector('.floating-actions'),  
    footer: document.querySelector('footer'),  
  
    // Modal de Cliente  
    modal: document.getElementById('productModal'),  
    closeBtnTop: document.querySelector('.close-btn'),  
    closeBtnBottom: document.getElementById('close-modal-bottom'),  
    stockContainer: document.getElementById('stock-container'),  
    addToCartBtn: document.getElementById('add-to-cart-btn'),
```

```
modalProductName: document.getElementById('modal-product-name'),  
modalPrice: document.getElementById('modal-price'),  
modalDescription: document.getElementById('modal-description'),
```

```
// Admin
```

```
loginFormContainer: document.getElementById('login-form-container'),  
masterLoginForm: document.getElementById('master-login-form'),  
loginUsername: document.getElementById('login-username'),  
loginPassword: document.getElementById('login-password'),  
loginMessage: document.getElementById('login-message'),
```

```
// Selectores para Botones Flotantes y Modal de Admin
```

```
adminFloatControls: document.getElementById('admin-controls-container'),  
editCatalogBtn: document.getElementById('edit-catalog-btn'),  
logoutBtn: document.getElementById('logout-btn'),  
closeLoginFormBtn: document.querySelector('.close-login-btn'),
```

```
adminModal: document.getElementById('adminModal'),
```

```
closeAdminBtn: document.querySelector('.close-admin-btn'),
```

```
adminPanel: document.getElementById('admin-panel'),
```

```
showListBtn: document.getElementById('show-list-btn'),
```

```
addProductFormContainer: document.getElementById('add-product-form-container'),
```

```
newProductForm: document.getElementById('new-product-form'),
```

```
editProductList: document.getElementById('edit-product-list'),
```

```
newProductStockInputs: document.getElementById('new-product-stock-inputs'),
```

```
newProductImageFile: document.getElementById('new-product-image-file'),
```

```
};

// Variables auxiliares del modal
let currentProduct = null;
let selectedVariant = null;

// ====== Helpers ======
function safeLowerClass(str) {
    if (!str) return "";
    return String(str).toLowerCase().replace(/\s+/g, '-').replace(/[^w-]/g, "");
}

function formatPrice(value) {
    if (value == null) return "";
    return `฿${Number(value).toLocaleString()}`;
}

function defaultImageFor(idProducto) {
    // Si no hay url en la DB, intenta una ruta por id o imagen default
    return `/images/product_${idProducto}.png`;
}

// ====== API CALLS ======
async function apiFetch(path, opts = {}) {
    const url = `${API_BASE}${path}`;
```

```
const defaultOpts = {
    credentials: "include", // enviar cookies (sesión)
    headers: {}
};

const finalOpts = Object.assign({}, defaultOpts, opts);

// Si body es objeto y no es FormData, setear JSON header
if (finalOpts.body && !(finalOpts.body instanceof FormData) &&
!finalOpts.headers['Content-Type']) {
    finalOpts.headers['Content-Type'] = 'application/json';
}

try {
    const res = await fetch(url, finalOpts);
    // Si es un redirect HTML (logout redirige al index), fetch seguirá el redirect.

    // Intentamos parsear JSON si lo hay, si no devolveremos { success: false,
    message: ... }

    const text = await res.text();
    try {
        const json = JSON.parse(text);
        return { ok: res.ok, status: res.status, json };
    } catch {
        // no JSON
        return { ok: res.ok, status: res.status, text };
    }
} catch (err) {
    console.error("apiFetch error:", err);
    return { ok: false, error: err };
}
```

```
}

async function loadProducts() {
  const r = await apiFetch("/api/productos", { method: "GET" });
  if (!r.ok) {
    console.error("Error cargando productos:", r);
    return;
  }
  const data = r.json || r.text;
  // tu API responde { success: true, data: [...] } (según lo que mostraste)
  const payload = r.json || {};
  const list = payload.data || [];
  // Mapear cada item al formato esperado por el frontend
  products = list.map(item => {
    const mapped = {
      id: item.idproducto || item.idProducto || item.id || null,
      idProducto: item.idproducto || item.idProducto || item.id || null,
      name: item.nombre || item.name || "",
      tag: item.nombrecategoría || item.tag || "",
      tagClass: safeLowerClass(item.nombrecategoría || item.tag || ""),
      price: item.preciobase || item.precioBase || 0,
      isOffer: !!item.enoferta,
      descripcion: item.descripcion || "",
      esBorrado: !!item.esborrado,
      nombreCategoria: item.nombrecategoría || "",
      // URL de imagen: usamos la propiedad urlImagen si existe (opción A), si no
      fallback
    }
  })
}
```

```

        urlImagen: item.urlImagen || item.urlImagen || defaultImageFor(item.idproducto || item.idProducto || item.id)
    };

    return mapped;
});

// Renderizar
renderProducts(products);

// También actualizar la lista admin si está visible
if (!DOM.editProductList.classList.contains('hidden')) {
    renderAdminProductList();
}

}

async function loadCategories() {
    const r = await apiFetch("/api/categorias", { method: "GET" });
    if (r.ok && r.json && Array.isArray(r.json.data)) {
        categories = r.json.data;
    } else {
        categories = [];
    }
    // poblar selects si existen (por ejemplo en el formulario add product)
    fillCategorySelects();
}

async function loadVariants() {
    const r = await apiFetch("/api/variantes", { method: "GET" });
}

```

```
if (r.ok && r.json && Array.isArray(r.json.data)) {
    variants = r.json.data;
} else {
    variants = [];
}
// generar inputs de stock según ALL_VARIANTS o variantes reales
generateStockInputs();
}

// Create product (POST)
async function apiCreateProduct(payload) {
    // payload = { nombre, precioBase, descripcion, enOferta, idCategoria, stock: [...] }
    const r = await apiFetch("/api/productos", {
        method: "POST",
        body: JSON.stringify(payload)
    });
    return r;
}

// Update product logic is integrated into handleNewProductSubmit using FormData

// Delete product (DELETE)
async function apiDeleteProduct(idProducto) {
    const r = await apiFetch(`/api/productos/${idProducto}` , {
        method: "DELETE",
        credentials: "include" // ← NECESARIO para enviar cookies de sesión
    });
}
```

```

    return r;
}

// ===== RENDER & UI =====

function createProductCardHTML(product) {
    const formattedPrice = (product.price || 0).toLocaleString();
    const priceDisplay = `<div class="product-price">${formattedPrice}</div>`;
    const imageUrl = product.urlImagen
        ? `/uploads/${product.urlImagen}`
        : defaultImageFor(product.idProducto);

    return `
        <div class="product-card" data-product-id="${product.idProducto}" data-filter-
id="${product.id}">
            <div class="product-tag ${product.tagClass}">${product.tag}</div>
            <div class="product-image-placeholder" style="background-image:
url('${imageUrl}'); background-size: cover; background-position: center;"></div>
            <div class="product-name">${product.name}</div>
            ${priceDisplay}
        </div>
    `;
}

function renderProducts(productsToDisplay) {
    if (!DOM.productsGrid) return;
    DOM.productsGrid.innerHTML =
        productsToDisplay.map(createProductCardHTML).join("");
}

```

```

attachCardListeners();
}

function attachCardListeners() {
  if (!DOM.productsGrid) return;
  DOM.productsGrid.querySelectorAll('.product-card').forEach(card => {
    card.addEventListener('click', (event) => {
      if (event.target.closest('.product-tag')) return;
      const uniqueProductId = card.getAttribute('data-product-id');
      const product = products.find(p => String(p.idProducto) ===
String(uniqueProductId));
      if (product) openProductModal(product);
    });
  });
}

function handleFilterClick(event) {
  event.preventDefault();
  const clickedLink = event.currentTarget;
  const filterId = clickedLink.getAttribute('data-filter-id');
  DOM.filterLinks.forEach(link => link.classList.remove('active'));
  clickedLink.classList.add('active');

  let filteredProducts;
  if (filterId === '0') {
    filteredProducts = products;
  } else {

```

```
        filteredProducts = products.filter(product => String(product.id) === String(filterId));

    }

    renderProducts(filteredProducts);

}

// ===== MODAL & WHATSAPP =====

function openProductModal(product) {
    currentProduct = product;
    selectedVariant = null;

    DOM.modalProductName.textContent = product.name;
    DOM.modalPrice.textContent = formatPrice(product.price);
    DOM.modalDescription.textContent = product.descripcion || `Suplemento
${product.name}`;

    // La URL de la imagen debe ser relativa a /uploads/
    const imageUrlPath = product.urlImagen.startsWith('/') ? product.urlImagen :
    `/uploads/${product.urlImagen}`;

    document.getElementById("modal-image-placeholder").style.backgroundImage =
    `url('${imageUrlPath}')`;

    fillModalStock(product);

    DOM.addToCartBtn.disabled = true;
    DOM.addToCartBtn.textContent = 'Selecciona una variante';
}
```

```
DOM.modal.style.display = 'block';
}

async function fillModalStock(product) {
  DOM.stockContainer.innerHTML = "";
  // Intentar traer stock real para este producto
  try {
    const r = await apiFetch(`/api/inventario/${product.idProducto}`, { method: "GET" });
    if (r.ok && r.json && Array.isArray(r.json.data) && r.json.data.length > 0) {
      const stockRows = r.json.data;
      // stockRows contienen: cantidadstock, presentacion, contenido, sabor
      stockRows.forEach(row => {
        // Combinamos para una etiqueta de variante útil
        const variantLabel = `${row.presentacion} (${row.sabor || 'N/A'})`;
        const count = row.cantidadstock || row.cantidadStock || 0;
        createVariantOption(variantLabel, count);
      });
      return;
    }
  } catch (e) {
    console.warn("No se pudo cargar stock via API, usando stock simulado/fallback", e);
  }
}

// Fallback: usar ALL_VARIANTS y un contador por defecto (0)
const stock = {} // no tenemos simulatedStock global fiable ahora
```

```

ALL_VARIANTS.forEach(variant => {
  const count = stock[variant] || 0;
  createVariantOption(variant, count);
});

}

function createVariantOption(variantLabel, count) {
  const isAvailable = count > 0;
  const statusClass = isAvailable ? 'available' : 'unavailable';
  const variantDiv = document.createElement('div');
  variantDiv.className = `size-option ${statusClass}`;
  if (isAvailable) {
    variantDiv.setAttribute('data-variant', variantLabel);
    variantDiv.addEventListener('click', handleVariantSelection);
  }
  variantDiv.innerHTML = `${variantLabel}<br><span style="font-size:0.75em;">${count} disponibles</span>`;
  DOM.stockContainer.appendChild(variantDiv);
}

function handleVariantSelection(event) {
  const clickedVariantOption = event.currentTarget;
  const variant = clickedVariantOption.getAttribute('data-variant');
  document.querySelectorAll('.size-option.available').forEach(opt =>
    opt.classList.remove('selected'));
  clickedVariantOption.classList.add('selected');
  selectedVariant = variant;
  DOM.addToCartBtn.disabled = false;
}

```

```
DOM.addToCartBtn.textContent = `Comprar Variante ${variant}`;
}

function closeProductModal() {
  DOM.modal.style.display = 'none';
}

function redirectToWhatsApp() {
  if (!currentProduct || !selectedVariant) {
    alert('Error: No hay producto o variante seleccionada.');
    return;
  }
  const messageLines = [
    '¡Hola! Estoy interesado/a en comprar el siguiente suplemento:',
    '',
    `Producto: ${currentProduct.name}`,
    `Variante: ${selectedVariant}`,
    `Precio: ${formatPrice(currentProduct.price)}`,
    `Código del Producto: ${currentProduct.idProducto}`,
    '',
    '¿Me confirmas la disponibilidad para hacer el pedido?',
  ];
  const encodedMessage = encodeURIComponent(messageLines.join('\n'));
  const whatsappUrl =
`https://wa.me/${WHATSAPP_NUMBER}?text=${encodedMessage}`;
  window.open(whatsappUrl, '_blank');
  closeProductModal();
}
```

```
}

// ===== AUTH (login/logout usando API)
=====

async function handleLogin(event) {
    event.preventDefault();
    const username = DOM.loginUsername.value.trim();
    const password = DOM.loginPassword.value.trim();

    try {
        const r = await apiFetch("/login", {
            method: "POST",
            body: JSON.stringify({ correo: username, contrasena: password })
        });
        if (!r.ok) {
            const message = (r.json && r.json.message) || "Error de autenticación";
            DOM.loginMessage.textContent = message;
            DOM.loginMessage.style.color = "var(--color-accent)";
            return;
        }
        const payload = r.json || {};
        if (payload.success) {
            DOM.loginMessage.textContent = "Inicio de sesión exitoso.";
            DOM.loginMessage.style.color = "green";
            // mostrar controles admin
            showAdminControls(true);
        }
    } catch (error) {
        console.error("Error al iniciar sesión:", error);
        DOM.loginMessage.textContent = "Error al iniciar sesión";
        DOM.loginMessage.style.color = "red";
    }
}
```

```
    setTimeout(() => toggleLoginForm(false), 400);
    // recargar lista de productos (para ver cambios admin)
    await loadProducts();
}

} else {
    DOM.loginMessage.textContent = payload.message || "Credenciales
incorrectas";
    DOM.loginMessage.style.color = "var(--color-accent)";
}
} catch (err) {
    console.error("Login error:", err);
    DOM.loginMessage.textContent = "No se pudo conectar al servidor";
    DOM.loginMessage.style.color = "red";
}
}
```

```
async function handleLogout() {
    try {
        await apiFetch("/logout", { method: "POST" });
    } catch (e) {
        console.warn("Logout error", e);
    } finally {
        showAdminControls(false);
        alert("Sesión cerrada. Regresando a la vista pública.");
        // recargar productos como público
        await loadProducts();
    }
}
```

```
function toggleLoginForm(show) {
    if (!DOM.masterLoginForm) return;
    if (show) {
        DOM.masterLoginForm.classList.remove('hidden');
        DOM.loginMessage.textContent = "";
    } else {
        DOM.masterLoginForm.classList.add('hidden');
    }
}

function checkAuthOnLoad() {
    // Intentamos validar la sesión con el backend pidiendo /api/validar-sesion
    (async () => {
        try {
            const r = await apiFetch("/api/validar-sesion", { method: "GET" });
            if (r.ok && r.json && r.json.success) {
                showAdminControls(true);
            } else {
                showAdminControls(false);
            }
        } catch {
            showAdminControls(false);
        }
    })();
}
```

```
// ===== ADMIN PANEL (CRUD) =====
```

```
function generateStockInputs() {
    // Si tienes variants reales (variants array), úsalas; si no, ALL_VARIANTS
    const source = (variants && variants.length) ? variants.map(v => v.presentacion || v.Presentacion || `Var${v.idvariante} || ${v.idVariante} || ${v.id}`) : ALL_VARIANTS;
    let html = "";
    source.forEach(variant => {
        const safeld = `stock-${String(variant).replace(/\s+/g, '-')}`;
        html += `
            <div class="stock-input-group">
                <label for="${safeld}">${variant}</label>
                <input type="number" id="${safeld}" name="stock_${variant}" min="0" value="0">
            </div>
        `;
    });
    DOM.newProductStockInputs.innerHTML = html;
}
```

```
function fillCategorySelects() {
    // En el HTML actual se usa un input 'idArticulo', no un select. Adaptamos la lógica para el select si existiera
    const select = DOM.newProductForm ?
        DOM.newProductForm.querySelector('select[name="categoria"]') : null;
    if (!select) return;
    select.innerHTML = `<option value="">-- Seleccione categoría --</option>`;
    categories.forEach(c => {
```

```
const id = c.idcategoria || c.idCategoria || c.id;
const name = c.nombrecategoria || c.nombreCategoria || c.nombre || c.name;
const opt = document.createElement('option');
opt.value = id;
opt.textContent = name;
select.appendChild(opt);
});
}
```

```
function renderAdminProductList() {
let listHTML = `

<div class="admin-list-header">
  <h3>Lista de Productos</h3>
  <button id="add-product-from-list-btn" class="action-btn primary">+ Añadir
  Producto</button>
</div>

<table class="admin-table">
  <thead>
    <tr>
      <th>ID</th>
      <th>Nombre</th>
      <th>Precio</th>
      <th>Categoría</th>
      <th>Acciones</th>
    </tr>
  </thead>
  <tbody>
```

```

`;

products.forEach(product => {

    // totalStock: si se quiere obtener, se puede consultar el inventario por producto;
    // aquí dejamos "-" como placeholder

    const totalStock = "-";

    listHTML += `

        <tr data-product-id="${product.idProducto}">

            <td>${product.idProducto}</td>
            <td>${product.name}</td>
            <td>${formatPrice(product.price)}</td>
            <td>${product.tag}</td>
            <td>

                <button class="action-btn small edit-btn" data-
id="${product.idProducto}">📝 Editar</button>
                <button class="action-btn small secondary delete-btn" data-
id="${product.idProducto}">🗑 Eliminar</button>
                <span class="stock-info">(Total Stock: ${totalStock})</span>
            </td>
        </tr>
    `;
});

listHTML += `</tbody></table><p class="admin-note">Nota: Use las acciones para
editar/eliminar; crear un producto subirá datos al backend.</p>`;

DOM.editProductList.innerHTML = listHTML;

```

```
const addProductFromListBtn = document.getElementById('add-product-from-list-btn');

if (addProductFromListBtn) addProductFromListBtn.addEventListener('click', () =>
showAdminSection('add'));

// editar

DOM.editProductList.querySelectorAll('.edit-btn').forEach(btn => {

  btn.addEventListener('click', async (event) => {

    const id = event.currentTarget.getAttribute('data-id');

    await openEditProductForm(id);

  });
});

// eliminar

DOM.editProductList.querySelectorAll('.delete-btn').forEach(btn => {

  btn.addEventListener('click', async (event) => {

    const id = event.currentTarget.getAttribute('data-id');

    if (!confirm(`¿Estás seguro de eliminar el producto ${id}?`)) return;

    const r = await apiDeleteProduct(id);

    if (r.ok && r.json && r.json.success) {

      alert("Producto eliminado correctamente.");

      await loadProducts();

      renderAdminProductList();

    } else {

      alert("Error eliminando producto: " + ((r.json && r.json.message) || r.status || 'Error'));

    }
  });
});
```

```

    });
}

async function openEditProductForm(idProducto) {
    // Cargar datos del producto y llenar el formulario; cambiar el modo a edit
    const product = products.find(p => String(p.idProducto) === String(idProducto));
    if (!product) {
        alert("Producto no encontrado en memoria");
        return;
    }
    showAdminSection('add', { mode: 'edit', id: idProducto });
    // llenar campos del formulario si existen
    if (!DOM.newProductForm) return;
    DOM.newProductForm.querySelector('[name="nombre"]').value = product.name || "";
    DOM.newProductForm.querySelector('[name="precio"]').value = product.price || "";
    DOM.newProductForm.querySelector('[name="descripcion"]').value =
        product.descripcion || "";
    // El HTML usa un input para el ID de artículo.
    const idArticuloInput = DOM.newProductForm.querySelector('[name="idArticulo"]');
    if (idArticuloInput) idArticuloInput.value = product.idCategoria || "";
    // almacenar idProducto en el form para saber que estamos editando
    DOM.newProductForm.setAttribute('data-edit-id', idProducto);
    // Cambiar el campo de imagen a no requerido al editar
    DOM.newProductImageFile.removeAttribute('required');
}

function showAdminControls(show) {

```

```
if (!DOM.adminFloatControls) return;

if (show) DOM.adminFloatControls.classList.remove('hidden'); else
DOM.adminFloatControls.classList.add('hidden');

}

function showAdminSection(section, options = {}) {
    DOM.addProductFormContainer.classList.add('hidden');
    DOM.editProductList.classList.add('hidden');

    if (section === 'add') {
        DOM.addProductFormContainer.classList.remove('hidden');
        const formTitle = DOM.addProductFormContainer.querySelector('h3');
        if (options.mode === 'edit') {
            formTitle.textContent = `Editar Producto #${options.id}`;
            // Al editar, la imagen ya no es obligatoria
            DOM.newProductImageFile.removeAttribute('required');
        } else {
            formTitle.textContent = 'Agregar Nuevo Producto';
            // limpiar marker de edit
            if (DOM.newProductForm) {
                DOM.newProductForm.removeAttribute('data-edit-id');
                DOM.newProductForm.reset(); // Limpiar formulario al crear uno nuevo
            }
            // Al agregar, la imagen es obligatoria
            DOM.newProductImageFile.setAttribute('required', '');
        }
    } else if (section === 'list') {
```

```
DOM.editProductList.classList.remove('hidden');
renderAdminProductList();
}

}

// ===== FORM HANDLERS (Crear / Editar)
=====

async function handleNewProductSubmit(event) {
event.preventDefault();
if (!DOM.newProductForm) return;

const form = DOM.newProductForm;

const nombre = form.querySelector('[name="nombre"]').value.trim();
const precioBase = form.querySelector('[name="precio"]').value;
const descripcion = form.querySelector('[name="descripcion"]').value.trim();
const idCategoria = form.querySelector('[name="idArticulo"]').value;
const fileInput = document.getElementById("new-product-image-file");
const file = fileInput.files[0];

const editId = form.getAttribute('data-edit-id');

// Validación simplificada: La imagen es requerida solo si es 'Crear' (no editId) o si se
// subió un archivo nuevo.

if (!nombre || !precioBase || !idCategoria) {
alert("Faltan campos de texto (nombre, precio, categoría).");
}
```

```
        return;
    }

    if (!editId && !file) {
        alert("Faltan la imagen del producto (requerida al crear).");
        return;
    }

    // -----
    // ARMAR EL FormData COMPLETO
    // -----

    const fd = new FormData();
    fd.append("nombre", nombre);
    fd.append("precioBase", precioBase);
    fd.append("descripcion", descripcion);
    fd.append("enOferta", "false"); // Valor por defecto
    fd.append("idCategoria", idCategoria);

    if (file) {
        fd.append("imagenFile", file); // Solo si se selecciona una nueva imagen
    }

    try {
        let url = "/api/productos";
        let method = "POST";

        if (editId) {
```

```
url = `/api/productos/${editId}`;
method = "PUT";
}

const res = await fetch(url, {
  method,
  body: fd,
  credentials: "include"
});

// Intentar parsear el JSON
const text = await res.text();
let json;
try {
  json = JSON.parse(text);
} catch {
  json = { success: false, message: "Respuesta no es JSON. Error del servidor." };
}

if (json.success) {
  alert(editId ? "Producto actualizado." : "Producto creado.");
  form.reset();
  await loadProducts();
  showAdminSection('list');
} else {
  alert("Error: " + json.message);
}
```

```
        } catch (err) {
            console.error("Error guardando producto:", err);
            alert("Error en el servidor o al conectar con la API.");
        }
    }
```

```
// ===== INICIALIZACIÓN =====
```

```
function initializeApp() {
    // Cargar datos iniciales
    checkAuthOnLoad();
    loadCategories(); // cargar categorias para selects
    loadVariants(); // cargar variantes si existen
    loadProducts(); // finalmente cargar productos

    // Listeners
    if (DOM.masterLoginForm) DOM.masterLoginForm.addEventListener('submit', handleLogin);
    if (DOM.closeLoginFormBtn) DOM.closeLoginFormBtn.addEventListener('click', (e) => { e.preventDefault(); toggleLoginForm(false); });

    if (DOM.editCatalogBtn) DOM.editCatalogBtn.addEventListener('click', () => {
        openAdminModal();
    });
}
```

```
if (DOM.logoutBtn) DOM.logoutBtn.addEventListener('click', () => { handleLogout();});
```

```
if (DOM.closeAdminBtn) DOM.closeAdminBtn.addEventListener('click', closeAdminModal);
```

```
window.addEventListener('click', (event) => {
  if (event.target === DOM.modal) closeProductModal();
  if (event.target === DOM.adminModal) closeAdminModal();
});
```

```
DOM.filterLinks.forEach(link => link.addEventListener('click', handleFilterClick));
```

```
if (DOM.closeBtnTop) DOM.closeBtnTop.addEventListener('click', closeProductModal);
```

```
if (DOM.closeBtnBottom) DOM.closeBtnBottom.addEventListener('click', closeProductModal);
```

```
if (DOM.addToCartBtn) DOM.addToCartBtn.addEventListener('click', redirectToWhatsApp);
```

```
if (DOM.newProductForm) DOM.newProductForm.addEventListener('submit', handleNewProductSubmit);
```

```
if (DOM.showListBtn) DOM.showListBtn.addEventListener('click', () => showAdminSection('list'));
```

```
// Floating button (perfil maestro)
```

```
const floatingButton = document.querySelector('.floating-actions .action-btn:nth-child(1)');
```

```
if (floatingButton) {
```

```
  floatingButton.addEventListener('click', (event) => {
```

```
event.preventDefault();

// Si el admin no está logueado (controls hidden), mostramos el login form
if (DOM.adminFloatControls.classList.contains('hidden')) {

    toggleLoginForm(true);

} else {

    // Si ya está logueado, abrimos el modal admin directamente
    openAdminModal();

}

});

}

// Admin modal open/close

function openAdminModal() {

    DOM.productsGridContainer.classList.add('hidden');

    DOM.filterNavContainer.classList.add('hidden');

    DOM.footer.classList.add('hidden');

    DOM.adminModal.style.display = 'block';

    generateStockInputs(); // Asegura que los inputs de stock estén listos
    showAdminSection('list');

}

function closeAdminModal() {

    DOM.adminModal.style.display = 'none';

    DOM.productsGridContainer.classList.remove('hidden');

    DOM.filterNavContainer.classList.remove('hidden');

    DOM.footer.classList.remove('hidden');

}
```

```
document.addEventListener('DOMContentLoaded', initializeApp);
```

5. Capa de Presentación (Frontend): Código HTML (index.html)

```
<!DOCTYPE html>
```

```
<html lang="es">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Okami Fit - Suplementos</title>
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/base.css') }}>
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/layout.css') }}>
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/header-nav.css') }}>
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/product-card.css') }}>
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/modal.css') }}>
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/utilities.css') }}>
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/media-queries.css') }}>
```

```
    <link
```

```
        href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"
```

```
        rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
    <header class="main-header">
```

```
<div class="header-banner">
    
</div>
</header>

<div id="adminModal" class="modal-admin">
    <div class="modal-content-admin">
        <span class="close-admin-btn">&times;</span>

        <section id="admin-panel" class="admin-panel-content">
            <h2>Gestión de Catálogo <img alt="grid icon" style="vertical-align: middle;"></h2>
            <div class="admin-menu">
                <button id="show-list-btn" class="action-btn primary"> <img alt="document with pencil icon" style="vertical-align: middle;"> Gestionar
                    Productos</button>
            </div>

            <div id="edit-product-list" class="admin-list-container hidden">
                <h3>Lista de Productos</h3>
                <p>Se está cargando el listado...</p>
            </div>

            <div id="add-product-form-container" class="admin-form-container hidden">
                <h3>Agregar/Editar Producto</h3>
                <form id="new-product-form" class="crud-form" enctype="multipart/form-data">
                    <label for="new-product-name">Nombre:</label>
```

```
<input type="text" id="new-product-name" name="nombre" required>

<label for="new-product-price">Precio Base (€):</label>
<input type="number" id="new-product-price" name="precio" required
min="0">

<label for="new-product-tag">Tag (Ej: Proteína, Creatina,
Oferta):</label>
<input type="text" id="new-product-tag" name="tag" required>

<label for="new-product-filter-id">ID de Artículo (Categoría 1-9):</label>
<input type="number" id="new-product-filter-id" name="idArticulo"
required min="1" max="9">

<label for="new-product-description">Descripción:</label>
<textarea id="new-product-description" name="descripcion"
required></textarea>

<label for="new-product-image-file">Subir Imagen:</label>
<input type="file" id="new-product-image-file" name="imagen"
accept="image/*" required>

<h4>Stock Inicial por Variante (Peso/Sabor):</h4>
<div id="new-product-stock-inputs" class="stock-inputs-grid">
</div>

<button type="submit" class="action-btn primary">Guardar
Producto</button>
```

```
</form>

</div>

</section>

</div>

</div>

<div id="login-form-container">

<form id="master-login-form" class="login-form hidden">

<h3>Acceso Maestro</h3>

<input type="text" id="login-username" placeholder="Usuario Maestro" required>

<input type="password" id="login-password" placeholder="Contraseña" required>

<button type="submit">Iniciar Sesión</button>

<button class="close-login-btn">X </button>

<p id="login-message" class="error-message"></p>

</form>

</div>

<nav class="filter-nav-container">

<div class="filter-nav">

<a href="#" class="filter-link active" data-filter-id="0">Todos</a>

<a href="#" class="filter-link" data-filter-id="1">Proteínas</a>

<a href="#" class="filter-link" data-filter-id="2">Creatinas</a>

<a href="#" class="filter-link" data-filter-id="3">Mass Gainers</a>

<a href="#" class="filter-link" data-filter-id="4">Aminoácidos</a>

<a href="#" class="filter-link" data-filter-id="5">Pre-Entrenos</a>

<a href="#" class="filter-link" data-filter-id="6">Glutaminas</a>


```

```
<a href="#" class="filter-link" data-filter-id="7">Shaker y bebidas</a>
<a href="#" class="filter-link" data-filter-id="8">Colageno</a>
```

```
</div>
</nav>
```

```
<main class="products-grid-container">
  <h2>Nuestros Productos Destacados</h2>
  <div class="products-grid">
    </div>
</main>
```

```
<hr>
```

```
<footer>
  <div class="footer-container">
    <div class="footer-col brand-contact">
      <h2>OKAMI FIT</h2>
      <br><br>
      <h3>Tu tienda de suplementos de confianza.</h3>
      <h3>Contacto: +506 8756-8991</h3>
      <br><br>
      <br>
      <ul class="social-links">
        <li><a href="https://www.facebook.com/share/1Fv6RHfne/?mibextid=wwXlfr" aria-label="Facebook">Facebook<i class="fab fa-facebook-f"></i></a></li>
        <li><a href="https://www.instagram.com/okamifit_cr?igsh=MTZqbjd2ajRncmxhdA%3D%3D&ut
  </div>
</div>
```

```
m_source=qr" aria-label="Instagram">Instagram<i class="fab fa-instagram"></i></a></li>

        </ul>
    </div>

<div class="footer-col okami-links">
    <h2 class="hidden-title">Okami</h2> <ul class="link-list">
        <br>
        <li><a href="#" onclick="openModal('quienes')">Quienes somos</a></li>
        <br><br>
        <li><a href="#" onclick="openModal('vision')">Visión</a></li>
        <br><br>
        <h2>Ubicación:</h2>
        <h3>Nos ubicamos en Limón Centro y en San José, San Pedro.</h3>
    </ul>
</div>
</div>

</footer>

<div class="floating-actions">
    <button class="action-btn">Perfil Maestro</button>
    <div id="admin-controls-container" class="admin-float-controls hidden">
        <button id="edit-catalog-btn" class="action-btn secondary">📝 Editar
        Catálogo</button>

```

```
<button id="logout-btn" class="action-btn accent">  Cerrar Sesión</button>

</div>
</div>

<div id="productModal" class="modal">

  <div class="modal-content">

    <span class="close-btn">&times;</span>

    <div class="modal-product-details">

      <div id="modal-image-placeholder" class="modal-image-placeholder"></div>

      <div class="modal-info-text">

        <h3 id="modal-product-name"></h3>

        <p class="modal-price-display">Precio: <span id="modal-price"></span></p>

        <p class="modal-stock-label">Variante (Stock):</p>

        <div id="stock-container" class="stock-container"></div>

        <div class="product-description">

          <h4>Detalles del Producto:</h4>

          <p id="modal-description">Descripción detallada del producto.</p>

        </div>

      </div>

    </div>

    <div class="modal-actions">

      <button class="action-btn secondary" id="close-modal-bottom">Cerrar</button>

      <button class="action-btn primary" id="add-to-cart-btn">Comprar</button>

    </div>

  </div>

</div>
```

```
<div id="infoModal" class="modal">  
  <div class="modal-content">  
    <span class="close" onclick="closeModal()">&times;</span>  
  
    <h2 id="modalTitle"></h2>  
    <p id="modalText"></p>  
  
  </div>  
</div>
```

```
<script src="{{ url_for('static', filename='js/script2.js') }}"></script>
```

```
<script>  
function openModal(section) {  
  
  const title = document.getElementById("modalTitle");  
  const text = document.getElementById("modalText");  
  
  if (section === "quienes") {  
    title.textContent = "¿Quiénes somos?";  
    text.innerHTML =  
      "En Okami Fit ofrecemos los mejores precios del país y una amplia variedad de  
      suplementos de calidad garantizada. Nuestro compromiso es ayudarte a alcanzar tus  
      metas fitness mediante asesoría personalizada y productos confiables que impulsan tu  
      rendimiento y bienestar.<br><br>" +
```

"En Okami Fit trabajamos con pasión, compromiso y transparencia, siempre priorizando tu salud y confianza. Queremos motivarte, impulsarte y demostrarte que con disciplina, constancia y buenos productos tu mejor versión siempre es posible.";

}

```
if (section === "vision") {
```

```
    title.textContent = "Visión";
```

```
    text.textContent =
```

"Ser la tienda líder en suplementos y bienestar en Costa Rica, reconocida por la calidad, " +

"excelente servicio al cliente, precios competitivos y compromiso con el crecimiento personal y deportivo de cada persona.";

}

```
document.getElementById("infoModal").style.display = "block";
```

}

```
function closeModal() {
```

```
    document.getElementById("infoModal").style.display = "none";
```

}

```
// Cerrar haciendo clic fuera del modal
```

```
window.onclick = function (event) {
```

```
    const modal = document.getElementById("infoModal");
```

```
    if (event.target === modal) {
```

```
        closeModal();
```

}

```
};
```

```
</script>
```

```
</body>
```

```
</html>
```

6. Documentación del Frontend (Análisis Técnico)

6.1. Lógica del Cliente (JavaScript)

Esta sección documenta la arquitectura de la lógica del lado del cliente, implementada en script.js / script2.js.

- Conexión API:
 - const API_BASE = "http://127.0.0.1:5000";
 - La función apiFetch asegura que las cookies de sesión se envíen con credentials: "include", indispensable para la autenticación del Backend.
- Módulos Clave:
 - Catálogo y Filtros: loadProducts() (consulta GET /api/productos) y handleFilterClick() (filtro en memoria).
 - Autenticación: handleLogin() (POST /login) y handleLogout() (POST /logout), que controlan la visibilidad de admin-controls-container.
 - Pedidos (CU05): La función redirectToWhatsApp() construye dinámicamente el mensaje de pedido con el nombre del producto, la variante y el precio, y redirige al usuario a https://wa.me/.
 - CRUD (Admin): handleNewProductSubmit soporta la creación (POST) y edición (PUT) de productos utilizando el objeto FormData para la correcta gestión del archivo de imagen. apiDeleteProduct realiza el borrado lógico.

6.2. Capa de Presentación (HTML)

- Entorno: Utiliza Jinja2 (en Flask) para manejar rutas estáticas como {{ url_for('static', filename='...') }}.
- Estructura de la Interfaz:

- Catálogo: Contenedores principales para la grilla de productos (products-grid) y la navegación de filtros (filter-nav).
- Panel de Administración: Implementado en el modal adminModal, contiene el listado (edit-product-list) y el formulario CRUD (new-product-form).
- Flotantes: El botón "Perfil Maestro" gestiona la apertura del master-login-form o del adminModal, dependiendo del estado de autenticación.

7. Resumen Final del Manual Técnico

Capa/Documento	Componentes Clave	Arquitectura y Tecnologías
Capa de Persistencia	Esquema de Base de Datos	SQLite: Tablas para Productos, Usuarios (Admin), Categorías e Inventory (Stock por Variante).
Lógica del Negocio (Backend/API)	API RESTful (Flask)	Python / Flask: CRUD completo (/api/productos). Autenticación basada en Cookies de Sesión . Manejo de subida de archivos de imagen.
Capa de Presentación	HTML, CSS, JavaScript	HTML (Jinja2): Estructura con modales para detalles/admin. JavaScript: Interacción con la API, filtros, y la funcionalidad clave de Pedido por WhatsApp .
Funcionalidad Crítica	Pedidos y CRUD	Flujo de Pedido simple (WhatsApp) y gestión completa de productos (Crear, Listar, Editar, Eliminar) mediante el Panel Maestro.

CONCLUSION

La elaboración de este Manual Técnico no solo marca la finalización de la fase inicial del proyecto OKAMI, sino que también establece un cimiento sólido para su vida operativa y futura evolución. La documentación detallada de la arquitectura, las dependencias (*Python, SQL Server, JavaScript, HTML y CSS*) y los procedimientos de configuración en un entorno localhost es de valor incalculable.

Este manual sirve como una guía técnica exhaustiva para cualquier programador o equipo de desarrollo que asuma responsabilidades futuras. Proporciona una hoja de ruta clara de cómo funciona internamente el sistema, desde el flujo de datos entre el *frontend* y el *backend* hasta la estructura precisa de la base de datos en SQL Server.

Gracias a esta documentación, las tareas de mantenimiento, la corrección de errores y la implementación de cambios o nuevas funcionalidades serán procesos más rápidos y menos propensos a errores. Se reduce drásticamente la curva de aprendizaje, asegurando que el sistema OKAMI pueda ser sostenido y evolucionado de manera eficiente a lo largo del tiempo, protegiendo la inversión inicial del emprendimiento en su plataforma de ventas.