UNIVERSIDAD DEL VALLE

PROYECTO FINAL DE PROGRAMACIÓN WEB I : ……

MANUAL TECNICO

CARRERA: Ingeniería de Sistemas Informáticos

SEMESTRE: Tercer Semestre

DOCENTE: Ing. Carla A. Claure Melgar

ALUMNO: MIRKO VARGAS FIRMINO

TRINIDAD-BENI-BOLIVIA – FECHA….

# MANUAL TECNICO

# HTML

Este manual técnico proporciona una descripción detallada de la estructura del código HTML de la página web CROSS WORLD. Además, se detallan las funciones JavaScript asociadas para mejorar la comprensión de la interactividad del sitio.

**2. Estructura del Código**

**2.1 Encabezado (<header>)**

El encabezado contiene el logotipo de CROSS WORLD y la barra de navegación con enlaces a las secciones principales del sitio.

**2.2 Sección de Nosotros (<section class="nosotros">)**

Esta sección presenta información sobre quiénes son en CROSS WORLD y su relación con el motocross.

**2.3 Sección de Tuning (<section class="tuning">)**

En esta sección, se describe el tuning con un texto de marcado y se proporciona un botón de acción.

**2.4 Sección de Productos (<section class="containerProductos">)**

Presenta una serie de productos con sus imágenes, precios y botones para agregar al carrito.

**2.5 Secciones de Imágenes Publicitarias (<div class="espacioImg">)**

Dos secciones que contienen imágenes publicitarias con títulos llamativos.

**2.6 Secciones de Ofertas y Categorías (<section class="conteiner\_ofertas"> y <section class="conteiner\_categoria">)**

Estas secciones están actualmente vacías y pueden utilizarse para futuras ofertas y categorías.

**2.7 Pie de Página (<footer class="pie\_pagina">)**

Contiene información de contacto y enlaces a redes sociales.

**3. Funcionalidades JavaScript**

**3.1 showCart(x)**

Una función JavaScript que muestra el contenido del carrito cuando se hace clic en el icono de carrito.

**3.2 closeBtn()**

Una función JavaScript que cierra el contenido del carrito cuando se hace clic en el botón de cierre.

# CSS

Este manual técnico describe los estilos CSS utilizados en el sitio web CROSS WORLD para proporcionar una experiencia visual atractiva y coherente. Los estilos se dividen en secciones para facilitar la comprensión y el mantenimiento del código.

1. **Estilos del Cuerpo (body)**

body { font-family: sans-serif; margin: 0; }

1. **Estilos de Enlaces (a)**

a { text-decoration: none; color: #35A29F; }

1. **Estilos de Encabezado (header)**

header { display: flex; min-height: 65px; background-color: #071952; justify-content: space-between; align-items: center; padding: 10px 0; }

1. **Estilos de Secciones**

**4.1 Sección de Encabezado de Imagen (div.encabezadoImg)**

.encabezadoImg { width: 100%; height: 500px; background-image: url(...); /\* Otros estilos... \*/ }

**4.2 Sección de Nosotros (section.nosotros)**

.nosotros\_text { text-align: center; margin-bottom: 30px; }

**4.3 Espacio Publicitario (div.espacioImg)**

.espacioImg { position: relative; width: 100%; /\* Otros estilos... \*/ }

**4.4 Sección de Tuning (section.tuning)**

.conteiner\_tuning { display: flex; align-items: center; justify-content: center; margin: 0 auto; background-color: #333; }

**4.5 Sección de Productos (section.containerProductos)**

.products { padding-top: 170px; display: flex; flex-wrap: wrap; justify-content: space-between; background-color: #222323; }

**5. Estilos de Productos y Carrito**

**5.1 Estilos de Productos (div.carts)**

.carts { max-width: 250px; box-shadow: 0 3px 10px rgb(0 0 0 / 20%); /\* Otros estilos... \*/ }

**5.2 Estilos de Carrito (div.cart-products)**

.cart-products { position: absolute!important; top: 82px; right: -18px; background-color: #fff; /\* Otros estilos... \*/ }

**6. Estilos de Categorías y Pie de Página**

**6.1 Estilos de Categoría (div.categoria)**

.categoria { text-align: center; background-color: #333; padding: 20px; /\* Otros estilos... \*/ }

**6.2 Estilos de Pie de Página (footer.pie\_pagina)**

.pie\_pagina { background-color: #333; color: #fff; padding: 30px; text-align: center; }

# JS

Este manual técnico describe el código JavaScript utilizado para gestionar un carrito de compras en el sitio web CROSS WORLD. Las funciones y variables aquí descritas facilitan la interactividad del usuario con la funcionalidad de compra en la plataforma.

1. **Variables y Selección de Elementos HTML**

javascriptCopy code

// Variables para la selección de elementos HTML let allContainerCart = document.querySelector('.products'); let containerBuyCart = document.querySelector('.card-items'); let priceTotal = document.querySelector('.price-total'); let amountProduct = document.querySelector('.count-product'); // Arreglo que almacena los productos en el carrito let buyThings = []; let totalCard = 0; let countProduct = 0;

1. **Funciones**

**2.1 loadEventListenrs()**

Esta función inicializa los event listeners necesarios para el funcionamiento del carrito.

javascriptCopy code

function loadEventListenrs() { allContainerCart.addEventListener('click', addProduct); containerBuyCart.addEventListener('click', deleteProduct); }

**2.2 addProduct(e)**

Esta función maneja la lógica de agregar un producto al carrito cuando se hace clic en el botón "Agregar al carrito".

javascriptCopy code

function addProduct(e) { e.preventDefault(); if (e.target.classList.contains('btn-add-cart')) { const selectProduct = e.target.parentElement; readTheContent(selectProduct); } }

**2.3 deleteProduct(e)**

Esta función gestiona la eliminación de un producto del carrito cuando se hace clic en el botón "X".

javascriptCopy code

function deleteProduct(e) { if (e.target.classList.contains('delete-product')) { const deleteId = e.target.getAttribute('data-id'); // Lógica para calcular el total del carrito al eliminar un producto buyThings.forEach(value => { if (value.id == deleteId) { let priceReduce = parseFloat(value.price) \* parseFloat(value.amount); totalCard = totalCard - priceReduce; totalCard = totalCard.toFixed(2); } }); // Filtrar el producto eliminado del arreglo buyThings = buyThings.filter(product => product.id !== deleteId); countProduct--; // Manejar el caso cuando no hay productos en el carrito if (buyThings.length === 0) { priceTotal.innerHTML = 0; amountProduct.innerHTML = 0; } loadHtml(); } }

**2.4 readTheContent(product)**

Esta función extrae la información del producto seleccionado y la agrega al arreglo **buyThings**.

javascriptCopy code

function readTheContent(product) { const infoProduct = { image: product.querySelector('div img').src, title: product.querySelector('.title').textContent, price: product.querySelector('div p span').textContent, id: product.querySelector('a').getAttribute('data-id'), amount: 1 } // Actualizar el total del carrito y verificar si el producto ya existe totalCard = parseFloat(totalCard) + parseFloat(infoProduct.price); totalCard = totalCard.toFixed(2); const exist = buyThings.some(product => product.id === infoProduct.id); if (exist) { // Incrementar la cantidad si el producto ya existe en el carrito const pro = buyThings.map(product => { if (product.id === infoProduct.id) { product.amount++; return product; } else { return product } }); buyThings = [...pro]; } else { // Agregar el producto al carrito buyThings = [...buyThings, infoProduct] countProduct++; } loadHtml(); }

**2.5 loadHtml()**

Esta función carga y actualiza la interfaz de usuario con la información del carrito.

javascriptCopy code

function loadHtml() { clearHtml(); buyThings.forEach(product => { const { image, title, price, amount, id } = product; const row = document.createElement('div'); row.classList.add('item'); row.innerHTML = ` <img src="${image}" alt=""> <div class="item-content"> <h5>${title}</h5> <h5 class="cart-price">${price}$</h5> <h6>Amount: ${amount}</h6> </div> <span class="delete-product" data-id="${id}">X</span> `; containerBuyCart.appendChild(row); // Actualizar el total y la cantidad de productos en el carrito priceTotal.innerHTML = totalCard; amountProduct.innerHTML = countProduct; }); }

**2.6 clearHtml()**

Esta función limpia el contenido actual del carrito en la interfaz de usuario.

javascriptCopy code

function clearHtml() { containerBuyCart.innerHTML = ''; }