# Informe sobre la Relación entre atracciones.html, atracciones.css y mostrarinformacion.js

## Introducción

El sitio web de Magik Park utiliza tres archivos principales para gestionar la página de atracciones: atracciones.html, atracciones.css y mostrarinformacion.js. Estos archivos trabajan en conjunto para proporcionar una experiencia interactiva y visualmente atractiva para los visitantes del parque. A continuación, se detalla cómo estos archivos se integran y colaboran para crear el mapa interactivo, implementar los puntos de interés y las animaciones, y gestionar los filtros y su funcionamiento.

## Creación del Mapa Interactivo

### atracciones.html

El archivo atracciones.html define la estructura básica de la página de atracciones, incluyendo el contenedor del mapa y los filtros. La sección relevante del archivo HTML es la siguiente:

<div class="map-container">

    <img src="../img/mapa.png" alt="Mapa del Magik Park" class="map-image" usemap="#parkmap">

    <div id="resultados-count">10 atracciones encontradas</div>

    <div class="resultados-lista"></div>

    <map name="parkmap" id="parkmap">

        <!-- Áreas interactivas del mapa -->

        <area shape="circle" alt="Montaña Rusa del Terror" onclick="mostrarAtraccion('Montaña Rusa del Terror'); return false;">

        <area shape="circle" alt="Torre del Vertigo" onclick="mostrarAtraccion('Torre del Vertigo'); return false;">

        <!-- Más áreas interactivas... -->

    </map>

</div>

### atracciones.css

El archivo atracciones.css define los estilos visuales para el mapa y los puntos de interés. A continuación se muestra una sección relevante del archivo CSS:

.map-container {

    flex: 1;

    position: relative;

    min-height: 600px;

    background:  #fff;

    width: 100%;

    height: auto;

    border-radius: 10px;

    padding: 15px;

    box-shadow: 0 2px 8px  rgba(0,0,0,0.1);

    display: flex;

    flex-direction: column;

    align-items: center;

    margin-bottom: 60px;

}

.map-image {

    width: 100%;

    height: auto;

    display: block;

    border-radius: 8px;

    margin: 0 auto;

}

.punto-interes {

    position: absolute;

    width: 20px;

    height: 20px;

    background-color:  #394754;

    border: 3px solid  #ffffff;

    border-radius: 50%;

    cursor: pointer;

    transition: all 0.3s ease;

    transform: translate(-50%, -50%);

    z-index: 100;

    box-shadow: 0 0 10px  rgba(57, 71, 84, 0.5);

    animation: pulse 2s infinite;

}

.punto-interes:hover {

    transform: translate(-50%, -50%) scale(1.3);

    background-color:  #ff4444;

    border-color:  #ffffff;

    box-shadow: 0 0 15px  rgba(255, 68, 68, 0.8);

    z-index: 101;

}

@keyframes pulse {

    0% {

        box-shadow: 0 0 0 0  rgba(57, 71, 84, 0.7);

    }

    70% {

        box-shadow: 0 0 0 10px  rgba(57, 71, 84, 0);

    }

    100% {

        box-shadow: 0 0 0 0  rgba(57, 71, 84, 0);

    }

}

### mostrarinformacion.js

El archivo mostrarinformacion.js maneja la lógica para inicializar los puntos de interés en el mapa y gestionar las interacciones del usuario. A continuación se muestra una sección relevante del archivo JavaScript:

const atracciones = [

    {

        "nombre": "Montaña Rusa del Terror",

        "descripción": "Una emocionante montaña rusa con giros y caídas vertiginosas.",

        "ubicación": "Zona Maquinismo",

        "edad": 12,

        "altura": 140,

        "tiempo": "2 minutos",

        "horario": "10:00 - 20:00",

        "imagen": "../img/Montaña Rusa del Terror.webp",

        "tipo": "Montaña Rusa"

    },

    // Más atracciones...

];

function inicializarPuntos() {

    const mapContainer = document.querySelector('.map-container');

    if (!mapContainer) {

        console.error('No se encontró el contenedor del mapa');

        return;

    }

    const imagen = mapContainer.querySelector('.map-image');

    if (!imagen) {

        console.error('No se encontró la imagen del mapa');

        return;

    }

    function crearPuntos() {

        // Limpiar puntos existentes

        const puntosExistentes = document.querySelectorAll('.punto-interes');

        puntosExistentes.forEach(p => p.remove());

        atracciones.forEach(atraccion => {

            const coordenadas = obtenerCoordenadas(atraccion.nombre);

            if (!coordenadas) {

                console.error(`No se encontraron coordenadas para ${atraccion.nombre}`);

                return;

            }

            const punto = document.createElement('div');

            punto.className = 'punto-interes';

            punto.setAttribute('data-tipo', atraccion.tipo);

            punto.style.left = `${coordenadas.x}%`;

            punto.style.top = `${coordenadas.y}%`;

            const tooltip = document.createElement('div');

            tooltip.className = 'tooltip';

            tooltip.innerHTML = `

                <strong>${atraccion.nombre}</strong>

                <br>

                <span style="font-size: 0.8em">${atraccion.tipo}</span>

            `;

            punto.appendChild(tooltip);

            punto.addEventListener('click', () => mostrarAtraccion(atraccion.nombre));

            punto.addEventListener('mouseenter', () => tooltip.style.opacity = '1');

            punto.addEventListener('mouseleave', () => tooltip.style.opacity = '0');

            mapContainer.appendChild(punto);

        });

    }

    function obtenerCoordenadas(nombre) {

        const coordenadas = {

            "Montaña Rusa del Terror": { x: 15, y: 63 },

            "Torre del Vértigo": { x: 42, y: 30 },

            // Más coordenadas...

        };

        return coordenadas[nombre];

    }

    if (imagen.complete) {

        crearPuntos();

    } else {

        imagen.addEventListener('load', crearPuntos);

    }

}

if (document.readyState === 'loading') {

    document.addEventListener('DOMContentLoaded', inicializarPuntos);

} else {

    inicializarPuntos();

}

## Implementación de Puntos y Animaciones

### Creación de Puntos de Interés

Los puntos de interés se crean dinámicamente en el archivo mostrarinformacion.js utilizando la función inicializarPuntos(). Esta función recorre el array atracciones y crea un elemento div para cada atracción, posicionándolo en el mapa según las coordenadas predefinidas.

### Animaciones de Puntos de Interés

Las animaciones de los puntos de interés se definen en el archivo atracciones.css utilizando la regla @keyframes pulse. Esta animación crea un efecto de pulsación continuo que hace que los puntos sean más visibles y atractivos para los usuarios.

### Interacción con Puntos de Interés

Los puntos de interés tienen eventos de clic y hover definidos en mostrarinformacion.js. Al hacer clic en un punto, se muestra un popup con información detallada de la atracción. Al pasar el cursor sobre un punto, se muestra un tooltip con el nombre y el tipo de la atracción.

## Apartado de Filtros y su Funcionamiento

### atracciones.html

El archivo atracciones.html define los botones de filtro que permiten a los usuarios seleccionar atracciones basadas en criterios como edad, duración y altura mínima. La sección relevante del archivo HTML es la siguiente:

<div class="filtros-container">

    <!-- Filtros por edad -->

    <div class="filtro-grupo">

        <h4>Edad</h4>

        <ion-button class="filtro-btn active" data-tipo="edad" data-valor="todas" onclick="actualizarFiltro('edad', 'todas')">Todas</ion-button>

        <ion-button class="filtro-btn" data-tipo="edad" data-valor="niños" onclick="actualizarFiltro('edad', 'niños')">Niños (0-8)</ion-button>

        <ion-button class="filtro-btn" data-tipo="edad" data-valor="jovenes" onclick="actualizarFiltro('edad', 'jovenes')">Jóvenes (9-12)</ion-button>

        <ion-button class="filtro-btn" data-tipo="edad" data-valor="adultos" onclick="actualizarFiltro('edad', 'adultos')">Adultos (13+)</ion-button>

    </div>

    <!-- Filtros por duración -->

    <div class="filtro-grupo">

        <h4>Duración</h4>

        <ion-button class="filtro-btn active" data-tipo="duracion" data-valor="todas" onclick="actualizarFiltro('duracion', 'todas')">Todas</ion-button>

        <ion-button class="filtro-btn" data-tipo="duracion" data-valor="corta" onclick="actualizarFiltro('duracion', 'corta')">Corta (≤5min)</ion-button>

        <ion-button class="filtro-btn" data-tipo="duracion" data-valor="media" onclick="actualizarFiltro('duracion', 'media')">Media (6-15min)</ion-button>

        <ion-button class="filtro-btn" data-tipo="duracion" data-valor="larga" onclick="actualizarFiltro('duracion', 'larga')">Larga (>15min)</ion-button>

    </div>

    <!-- Filtros por altura -->

    <div class="filtro-grupo">

        <h4>Altura mínima</h4>

        <ion-button class="filtro-btn active" data-tipo="altura" data-valor="todas" onclick="actualizarFiltro('altura', 'todas')">Todas</ion-button>

        <ion-button class="filtro-btn" data-tipo="altura" data-valor="baja" onclick="actualizarFiltro('altura', 'baja')">Baja (≤100cm)</ion-button>

        <ion-button class="filtro-btn" data-tipo="altura" data-valor="media" onclick="actualizarFiltro('altura', 'media')">Media (101-130cm)</ion-button>

        <ion-button class="filtro-btn" data-tipo="altura" data-valor="alta" onclick="actualizarFiltro('altura', 'alta')">Alta (>130cm)</ion-button>

    </div>

    <div class="reset-container">

        <ion-button class="reset-btn" onclick="resetearFiltros()">

            <ion-icon slot="start" name="refresh-circle"></ion-icon>

            Reiniciar Filtros

        </ion-button>

    </div>

</div>

### atracciones.css

El archivo atracciones.css define los estilos visuales para los botones de filtro, asegurando que sean intuitivos y fáciles de usar. A continuación se muestra una sección relevante del archivo CSS:

.filtros-container {

    width: 280px;

    flex-shrink: 0;

    display: flex;

    flex-direction: column;

    gap: 12px;

    padding: 15px;

    background:  #f5f5f5;

    border-radius: 10px;

    height: fit-content;

    position: sticky;

    top: 20px;

}

.filtro-grupo {

    display: flex;

    flex-direction: column;

    gap: 4px;

    padding: 4px;

}

.filtro-grupo h4 {

    margin: 0 0 4px 0;

    color:  #394e60;

    font-size: 1em;

    font-weight: 600;

    padding-bottom: 4px;

}

.filtro-btn {

    --background:  #394e60;

    --background-hover:  #2d3f4d;

    --color:  white;

    --border-radius: 25px;

    --padding-start: 12px;

    --padding-end: 12px;

    font-weight: 600;

    font-size: 0.95em;

    transition: all 0.3s ease;

    position: relative;

    overflow: hidden;

    cursor: pointer;

    user-select: none;

    -webkit-tap-highlight-color: transparent;

    touch-action: manipulation;

    z-index: 1;

}

.filtro-btn.active {

    --background:  #c3cfd9;

    --color:  #394e60;

    transform: scale(1.05);

    box-shadow: 0 2px 8px  rgba(0,0,0,0.1);

    pointer-events: auto;

}

.filtro-btn:not(.active):hover {

    --background:  #2d3f4d;

    transform: translateY(-1px);

}

### mostrarinformacion.js

El archivo mostrarinformacion.js maneja la lógica para aplicar los filtros y actualizar la visualización del mapa. A continuación se muestra una sección relevante del archivo JavaScript:

const filtrosActuales = {

    edad: 'todas',

    duracion: 'todas',

    altura: 'todas'

};

function filtrarAtracciones(filtros) {

    return atracciones.filter(atraccion => {

        const cumpleEdad = filtros.edad === 'todas' || (

            (filtros.edad === 'niños' && atraccion.edad <= 8) ||

            (filtros.edad === 'jovenes' && atraccion.edad > 8 && atraccion.edad <= 12) ||

            (filtros.edad === 'adultos' && atraccion.edad > 12)

        );

        const duracionNum = parseInt(atraccion.tiempo);

        const cumpleDuracion = filtros.duracion === 'todas' || (

            (filtros.duracion === 'corta' && duracionNum <= 5) ||

            (filtros.duracion === 'media' && duracionNum > 5 && duracionNum <= 15) ||

            (filtros.duracion === 'larga' && duracionNum > 15)

        );

        const cumpleAltura = filtros.altura === 'todas' || (

            (filtros.altura === 'baja' && atraccion.altura <= 100) ||

            (filtros.altura === 'media' && atraccion.altura > 100 && atraccion.altura <= 130) ||

            (filtros.altura === 'alta' && atraccion.altura > 130)

        );

        return cumpleEdad && cumpleDuracion && cumpleAltura;

    });

}

function actualizarFiltro(tipo, valor) {

    filtrosActuales[tipo] = valor;

    const atraccionesFiltradas = filtrarAtracciones(filtrosActuales);

    actualizarVisualizacionMapa(atraccionesFiltradas);

    actualizarBotonesActivos();

    actualizarContadorResultados(atraccionesFiltradas.length);

}

function actualizarVisualizacionMapa(atraccionesFiltradas) {

    const puntos = document.querySelectorAll('.punto-interes');

    puntos.forEach(punto => {

        const tooltip = punto.querySelector('.tooltip');

        const nombreAtraccion = tooltip.querySelector('strong').textContent;

        const estaFiltrada = atraccionesFiltradas.some(a => a.nombre === nombreAtraccion);

        if (estaFiltrada) {

            punto.style.opacity = '1';

            punto.style.pointerEvents = 'auto';

            punto.style.transform = 'translate(-50%, -50%) scale(1)';

            punto.classList.remove('filtered');

        } else {

            punto.style.opacity = '0.3';

            punto.style.pointerEvents = 'none';

            punto.style.transform = 'translate(-50%, -50%) scale(0.8)';

            punto.classList.add('filtered');

        }

    });

}

function actualizarContadorResultados(cantidad) {

    const resultadosCount = document.getElementById('resultados-count');

    if (resultadosCount) {

        resultadosCount.textContent = `${cantidad} atracciones encontradas`;

    }

}

function actualizarBotonesActivos() {

    document.querySelectorAll('.filtro-btn').forEach(btn => {

        const tipo = btn.dataset.tipo;

        const valor = btn.dataset.valor;

        btn.classList.toggle('active', filtrosActuales[tipo] === valor);

    });

}

function resetearFiltros() {

    filtrosActuales.edad = 'todas';

    filtrosActuales.duracion = 'todas';

    filtrosActuales.altura = 'todas';

    actualizarVisualizacionMapa(atracciones);

    actualizarBotonesActivos();

    actualizarContadorResultados(atracciones.length);

}

## Conclusión

Los archivos atracciones.html, atracciones.css y mostrarinformacion.js trabajan en conjunto para proporcionar una experiencia interactiva y visualmente atractiva en la página de atracciones del sitio web de Magik Park. atracciones.html define la estructura de la página, atracciones.css asegura que la página se vea bien y mostrarinformacion.js maneja la lógica y la interacción del usuario. Juntos, estos archivos permiten a los usuarios explorar las atracciones del parque, aplicar filtros y obtener información detallada de manera eficiente y atractiva.