INFORME TÉCNICO: SISTEMA DE NAVEGACIÓN ESTELAR

1. DESCRIPCIÓN DEL PROYECTO

La aplicación desarrollada es una **Interfaz de Mando Espacial** diseñada para gestionar misiones de exploración galáctica. Se ha estructurado como una *Single Page Application* (SPA) donde la navegación entre el Manual de Vuelo, el Simulador y el Diagnóstico se realiza de forma dinámica mediante la manipulación del DOM, sin recargar la página.

---

2. JUEGO INTERACTIVO: SIMULADOR DE EXPLORACIÓN (RA4)

Descripción funcional:

Consiste en un simulador lógico basado en la gestión de recursos de una nave espacial. El usuario selecciona destinos galácticos y el sistema procesa la viabilidad del viaje en función de la energía disponible y la probabilidad de éxito.

Justificación técnica de estructuras de datos:

- **Uso de Objetos:** Se utiliza una función constructora llamada Nave para instanciar el vehículo. Contiene propiedades de estado (combustible, integridad) y métodos funcionales como .viajar(), que encapsulan la lógica de consumo energético.
- **Arrays Multidimensionales:** La base de datos de misiones reside en el array destinosGalacticos. Cada índice contiene un sub-array con los datos: [Nombre, Distancia, Riesgo].
- **Métodos de Array y Bucles:** Se utiliza el método .push() para alimentar el historial de misiones y bucles for combinados con .length para procesar la lista de sectores.
- **Funciones Anidadas:** Dentro de la lógica de misión, existe calcularProbabilidadExito(), una función interna que utiliza Math.random() para determinar el resultado aleatorio del evento.

```javascript
// 2. Array multidimensional: [Nombre, Distancia (años luz), Nivel Riesgo] [cite: 37]
const destinosGalacticos = [
    ["Nebulosa de Orión", 1344, "Bajo"],
    ["Agujero Negro Sagitario A*", 26000, "Extremo"],
    ["Sistema Proxima Centauri", 4.2, "Medio"]
];

/**
 * 3. Función Constructora (Requisito RA4) [cite: 44]
 */
function Nave(nombre, combustible, potencia) {
    this.nombre = nombre;
    this.combustible = combustible;
    this.potencia = potencia;
    this.integridad = 100;

    // 4. Método dentro del objeto [cite: 45]
    this.viajar = function(distancia) {
        let gasto = parseInt(distancia / 10); // Función predefinida parseInt
        if (this.combustible >= gasto) {
            this.combustible -= gasto;
            return true;
        }
        return false;
    };
}
```

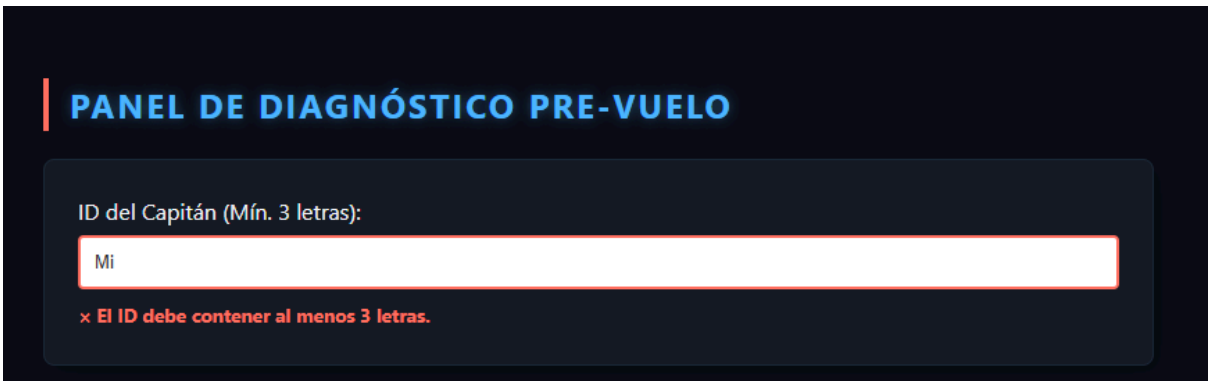
---

3. FORMULARIO DE AUTOEVALUACIÓN (RA5)

Validación y Experiencia de Usuario:

El formulario de "Diagnóstico de Sistemas" implementa una validación avanzada para asegurar la integridad de los datos de entrada.

- **Expresiones Regulares (Regex):** El campo del nombre utiliza una expresión regular para asegurar un mínimo de 3 caracteres alfabéticos.
- **Feedback en tiempo real:** Mediante eventos de input, el borde del campo cambia de color y muestra mensajes informativos instantáneos según la validez del dato.
- **Tipos de Control:** Se han implementado controles de tipo text, radio, select (combo box) y checkbox para selección múltiple.



---

4. GESTIÓN DE EVENTOS Y DOM (RA5)

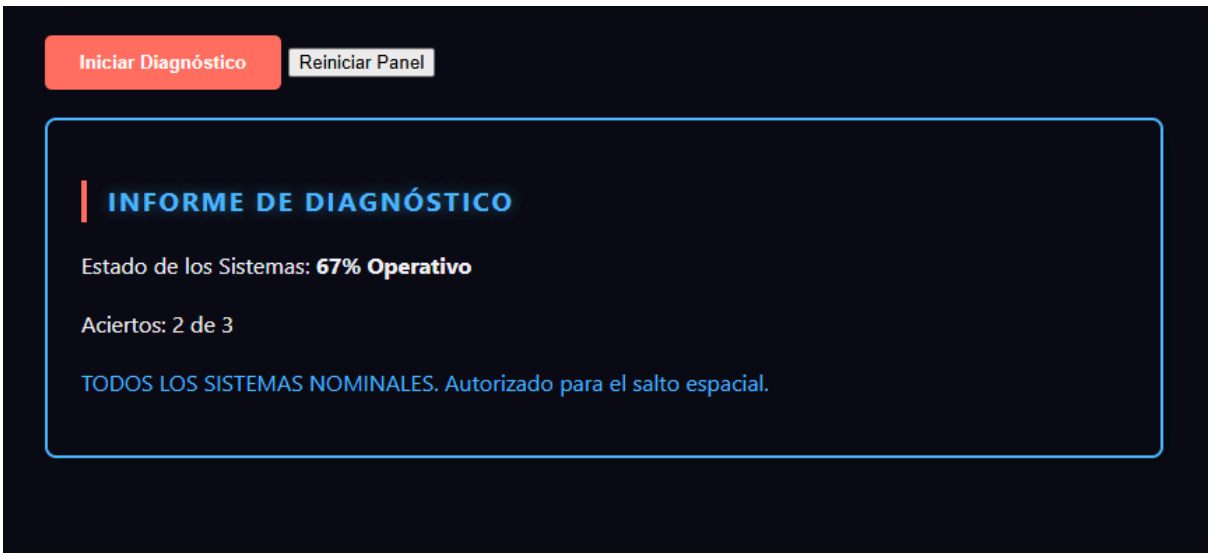
Se han implementado y personalizado los siguientes eventos para garantizar una interfaz interactiva y funcional:

1. **Eventos de Ventana (load):** Controla la inicialización de todos los módulos y la generación dinámica de contenido explicativo al cargar la aplicación.
2. **Eventos de Ratón (click):** Gestionan la navegación entre secciones (cambio de clases CSS .activa) y la ejecución de acciones en el simulador.
3. **Eventos de Formulario (submit, reset):** El evento submit procesa los datos, calcula la nota final sobre 100 y genera un informe visual detallado.
4. **Eventos de Teclado (keydown):** Se ha habilitado la tecla Escape como comando rápido para resetear el panel de control.

```javascript
/**
 * Gestión de la navegación entre secciones (Juego, Explicación, Formulario)
 */
function inicializarNavegacion() {
    const botonesNav = document.querySelectorAll('.nav-link');
    const secciones = document.querySelectorAll('.seccion-app');

    botonesNav.forEach(boton => {
        boton.addEventListener('click', (e) => {
            // Usamos currentTarget para asegurar que siempre leemos el data-target del botón
            const target = e.currentTarget.dataset.target;

            if (!target) return; // Seguridad

            secciones.forEach(sec => {
                sec.classList.remove('activa');
                if (sec.id === target) {
                    sec.classList.add('activa');
                    console.log(`Navegando a: ${target}`); // Para depuración
                }
            });
        });
    });
}
```

5. TABLA DE AUTOEVALUACIÓN TÉCNICA

ELEMENTO	TIPO DE CONTROL	VALIDACIÓN / ESTADO
Caja de texto	Input Text	Validado con Regex (Mín. 3 letras)
Radio Button	Input Radio	Evaluación de opción única (Sonido)
Combo Box	Select	Selección de diagnóstico de motores
Check Button	Input Checkbox	Selección múltiple de protocolos
Resultado	Div Dinámico	Feedback calculado mediante JavaScript



---

6. REFLEXIÓN TÉCNICA FINAL

El desarrollo de esta aplicación ha permitido integrar la manipulación del DOM con estructuras de datos complejas. La separación de la lógica en archivos modulares (app.js, juego.js, formulario.js) facilita el mantenimiento y cumple con los estándares de desarrollo profesional exigidos, garantizando una interfaz de usuario coherente y una lógica de programación robusta.