IES GASPAR MELCHOR DE Jovellanos
Calle Móstoles, 64
28941 Fuenlabrada
Madrid
Teléfono: 916971565

Comunidad de Madrid
Consejería de educación e investigación

INFORME TÉCNICO: SISTEMA DE NAVEGACIÓN ESTELAR

2ºDaw
Sergio Sánchez Plaza
Miguel Sánchez Martín

1. INTRODUCCIÓN Y ARQUITECTURA

El proyecto consiste en una aplicación web interactiva diseñada bajo el paradigma de **Single Page Application (SPA)**. La navegación se gestiona íntegramente mediante la manipulación del árbol **DOM (Document Object Model)**, permitiendo una transición fluida entre el manual, el simulador y el diagnóstico sin recargar el navegador. Se ha seguido una estructura modular separando la lógica de negocio (juego.js), la lógica de interacción (formulario.js) y el controlador principal (app.js).

1.1 PARTICIPACIÓN REAL Y ROLES DEL EQUIPO (RA4/RA5)

El desarrollo del proyecto se ha estructurado siguiendo un modelo de trabajo colaborativo dividido en dos grandes bloques de ingeniería: **Lógica de Sistemas (RA4)** e **Interfaz y Validación (RA5)**.

1.2 Desglose de Responsabilidades

Miembro del Grupo	**Rol**	**Áreas de Contribución Técnica**
Sergio	**Ingeniero de Lógica y Sistemas (RA4)**	Programación del motor de juego en juego.js. Creación del objeto constructor Nave. Gestión de la matriz multidimensional destinos galácticos. Implementación de algoritmos de probabilidad para eventos aleatorios. Maquetación base del simulador.
Miguel	**Arquitecto de Interfaz y Validación (RA5)**	Programación completa de formulario.js. Diseño de validaciones avanzadas con Regex. Estructuración estética con CSS Grid 3x2. Gestión del feedback dinámico al usuario. Adaptación del HTML para integrar los nuevos 7 tipos de entrada de datos.

2. RA4: ESTRUCTURAS DE DATOS Y LÓGICA DE PROGRAMACIÓN

En esta sección se demuestra el dominio de la programación orientada a objetos (basada en prototipos) y la gestión de colecciones de datos.

2.1. El Objeto Constructor Nave

En lugar de usar objetos literales planos, se ha implementado una **función constructora**. Esto permite instanciar múltiples naves si el proyecto escalara:

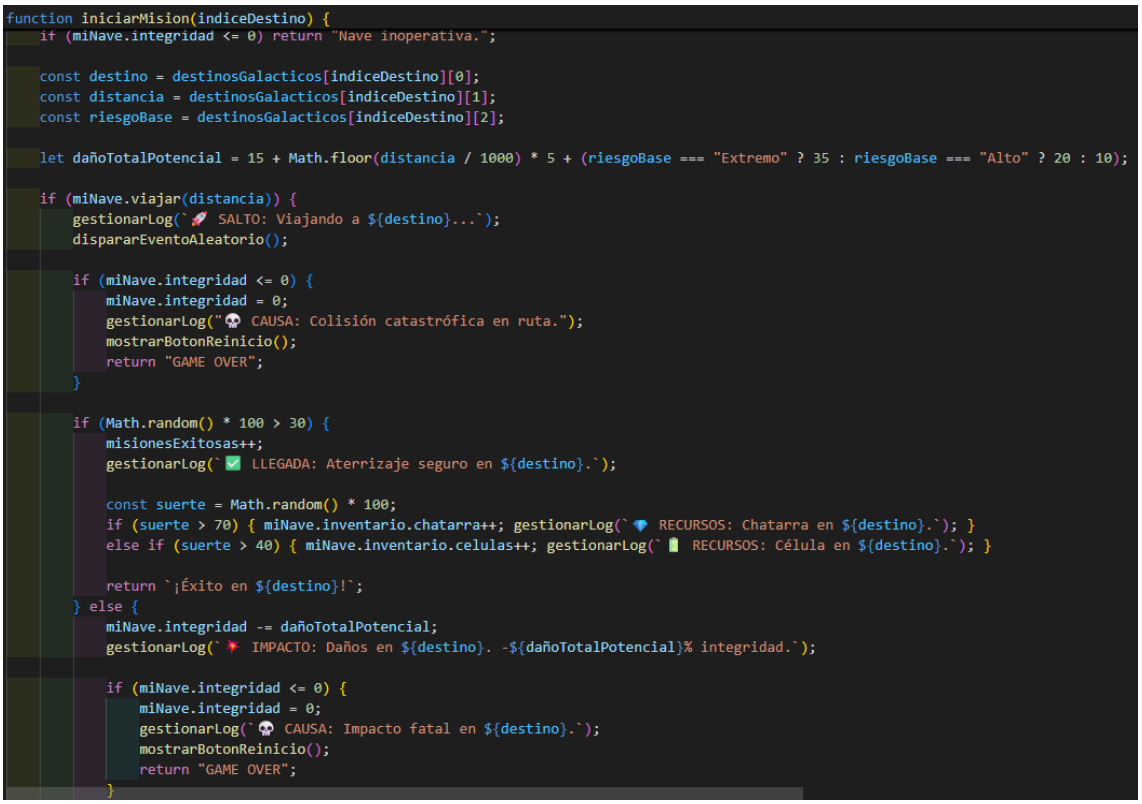
- **Encapsulamiento de Propiedades**: combustible, integridad e inventario (este último es un objeto anidado para mayor complejidad estructural).
- **Métodos Funcionales**:
    - **viajar(distancia):** Calcula el gasto energético mediante una fórmula matemática ponderada por la potencia de la nave.
    - **usarRecursos():** Lógica condicional que prioriza la reparación del casco o la recarga de plasma según la disponibilidad en el inventario.

```
function Nave(nombre, combustible, potencia) {
    this.nombre = nombre;
    this.combustible = combustible;
    this.combustibleMax = combustible;
    this.potencia = potencia;
    this.integridad = 100;
    this.estaCritica = false;

    this.inventario = {
        chatarra: 0,
        celulas: 0
    };
```

```
this.viajar = function (distancia) {
    let gasto = Math.floor(distancia * (100 / this.potencia));
    if (this.combustible >= gasto) {
        this.combustible -= gasto;
        return true;
    }
    return false;
};

this.usarRecursos = function () {
    if (this.integridad <= 0) return;

    if (this.inventario.chatarra > 0) {
        if (this.integridad < 100) {
            this.integridad = Math.min(this.integridad + 20, 100);
            this.inventario.chatarra--;
            if (this.integridad > 40) this.estaCritica = false;
            gestionarLog("🔧 REPARACIÓN: +20% casco.");
        } else {
            gestionarLog("🛡️ AVISO: Casco al 100%.");
        }
    }

    if (this.inventario.celulas > 0) {
        this.combustible = Math.min(this.combustible + 1500, this.combustibleMax);
        this.inventario.celulas--;
        gestionarLog("🔋 RECARGA: +1500 plasma.");
    }
    actualizarInterfaz("Sistemas de mantenimiento");
};
```

2.2. Gestión de Arrays y Matrices

- **Array Multidimensional**: El sistema utiliza una matriz para definir los destinos. Cada entrada es un array que contiene datos de distinto tipo (string, number), lo que demuestra la capacidad de gestionar estructuras de datos mixtas.
- **Iteración Dinámica**: Se emplea el método .forEach() para recorrer el array de destinos y generar las "cards" en la interfaz, asegurando que cualquier cambio en la base de datos se refleje automáticamente en el HTML.



```
function iniciarMision(indiceDestino) {
    if (miNave.integridad <= 0) return "Nave inoperativa.";

    const destino = destinosGalacticos[indiceDestino][0];
    const distancia = destinosGalacticos[indiceDestino][1];
    const riesgoBase = destinosGalacticos[indiceDestino][2];

    let dañoTotalPotencial = 15 + Math.floor(distancia / 1000) * 5 + (riesgoBase === "Extremo" ? 35 : riesgoBase === "Alto" ? 20 : 10);

    if (miNave.viajar(distancia)) {
        gestionarLog(`🚀 SALTO: Viajando a ${destino}...`);
        dispararEventoAleatorio();

        if (miNave.integridad <= 0) {
            miNave.integridad = 0;
            gestionarLog("💀 CAUSA: Colisión catastrófica en ruta.");
            mostrarBotonReinicio();
            return "GAME OVER";
        }

        if (Math.random() * 100 > 30) {
            misionesExitosas++;
            gestionarLog(`✅ LLEGADA: Aterrizaje seguro en ${destino}.`);

            const suerte = Math.random() * 100;
            if (suerte > 70) { miNave.inventario.chatarra++; gestionarLog(`💎 RECURSOS: Chatarra en ${destino}.`); }
            else if (suerte > 40) { miNave.inventario.celulas++; gestionarLog(`🔋 RECURSOS: Célula en ${destino}.`); }

            return `¡Éxito en ${destino}!`;
        } else {
            miNave.integridad -= dañoTotalPotencial;
            gestionarLog(`💥 IMPACTO: Daños en ${destino}. -${dañoTotalPotencial}% integridad.`);

            if (miNave.integridad <= 0) {
                miNave.integridad = 0;
                gestionarLog(`💀 CAUSA: Impacto fatal en ${destino}.`);
                mostrarBotonReinicio();
                return "GAME OVER";
            }
```

2.3. Algoritmos de Azar y Probabilidad

Se ha implementado un motor de eventos aleatorios mediante **Math.random().** Este calcula la probabilidad de encontrar recursos o sufrir daños por meteoritos durante el salto, añadiendo una capa de "Gamificación" y complejidad algorítmica al proyecto.

```
if (miNave.viajar(distancia)) {
    gestionarLog(`🚀 SALTO: Viajando a ${destino}...`);
    dispararEventoAleatorio();

    if (miNave.integridad <= 0) {
        miNave.integridad = 0;
        gestionarLog("💀 CAUSA: Colisión catastrófica en ruta.");
        mostrarBotonReinicio();
        return "GAME OVER";
    }

    if (Math.random() * 100 > 30) {
        misionesExitosas++;
        gestionarLog(`✅ LLEGADA: Aterrizaje seguro en ${destino}.`);

        const suerte = Math.random() * 100;
        if (suerte > 70) { miNave.inventario.chatarra++; gestionarLog(`💎 RECURSOS: Chatarra en ${destino}.`); }
        else if (suerte > 40) { miNave.inventario.celulas++; gestionarLog(`🔋 RECURSOS: Célula en ${destino}.`); }

        return `¡Éxito en ${destino}!`;
    } else {
        miNave.integridad -= dañoTotalPotencial;
        gestionarLog(`💥 IMPACTO: Daños en ${destino}. -${dañoTotalPotencial}% integridad.`);

        if (miNave.integridad <= 0) {
            miNave.integridad = 0;
            gestionarLog(`💀 CAUSA: Impacto fatal en ${destino}.`);
            mostrarBotonReinicio();
            return "GAME OVER";
        }

        if (miNave.integridad <= 40) miNave.estaCritica = true;
        return `¡Daños estructurales!`;
    }
```

2.4 Interfaz del juego:

La interfaz del juego se divide en las siguientes partes:

- Imagen que cambia de forma dinámica al cambiar el estado de la integridad de la nave.
- Grid que muestra los posibles destinos.
- Información del estado de la nave
- Inventario de los recursos de la nave.
- Bitácora del capitán al mando de la nave.

3. RA5: INTERACCIÓN Y VALIDACIÓN AVANZADA

El Panel de Diagnóstico ha sido diseñado para demostrar el control total sobre los eventos del navegador y la captura de datos del usuario.

3.1. Validación mediante Expresiones Regulares (Regex)

La validación del ID del capitán no es una simple comprobación de longitud. Se utiliza la expresión **/^[a-zA-ZÁÉÍÓÚáéíóúñÑ]{3,}$/** para:

1. Garantizar un mínimo de 3 caracteres.
2. Permitir únicamente letras (incluyendo tildes y eñes).
3. Excluir números y caracteres especiales en el nombre del oficial.

```
inputNombre.addEventListener("input", (e) => {
  // Expresión regular: Validar que el campo tenga al menos 3 letras y solo sean caracteres alfabéticos
  const regexNombre = /^[a-zA-ZÁÉÍÓÚáéíóúñÑ]{3,}$/;

  if (regexNombre.test(e.target.value)) {
    inputNombre.style.border = "2px solid #4db8ff";
    feedbackNombre.textContent = "✓ ID de Capitán validado.";
    feedbackNombre.style.color = "#4db8ff";
  } else {
    inputNombre.style.border = "2px solid #ff6f61";
    feedbackNombre.textContent = "✗ El ID debe contener al menos 3 letras.";
    feedbackNombre.style.color = "#ff6f61";
  }
});
```

Comprobación del cumplimiento de la introducción del nombre de usuario:

3.2. Análisis Detallado de los 7 Niveles de Entrada

Para cumplir con la rúbrica de "variedad de controles", se han implementado:

- **Control de Rango (range)**: Con un *event listene*r de tipo input que actualiza una etiqueta en tiempo real, proporcionando feedback visual constante.
- **Lógica de Checkboxes**: A diferencia de los radios, aquí el JS debe validar una selección múltiple específica (2 opciones correctas), penalizando si se marcan más o menos de las requeridas.

- **Normalización de Strings**: En la pregunta de texto "SALTO", el código utiliza .trim().toUpperCase() para evitar errores por espacios accidentales o uso de minúsculas, mejorando la robustez del sistema.

```
// 2. ACTUALIZACIÓN DINÁMICA DEL RANGO (RA5)
// Muestra el porcentaje del slider en tiempo real
rangeInput.addEventListener("input", (e) => {
  rangeValue.textContent = `${e.target.value}%`;
});

// 3. EVENTOS DE FOCO (RA5)
inputNombre.addEventListener("focus", () => {
  inputNombre.style.backgroundColor = "rgb(196, 196, 196)";
  // inputNombre.style.color = "white";
});

inputNombre.addEventListener("blur", () => {
  inputNombre.style.backgroundColor = "white";
});

// 4. EVENTO DE TECLADO (RA5)
document.addEventListener("keydown", (e) => {
  if (e.key === "Escape") {
    formulario.reset();
    console.log("Panel de diagnóstico reiniciado por comando de emergencia (ESC).");
  }
});
```

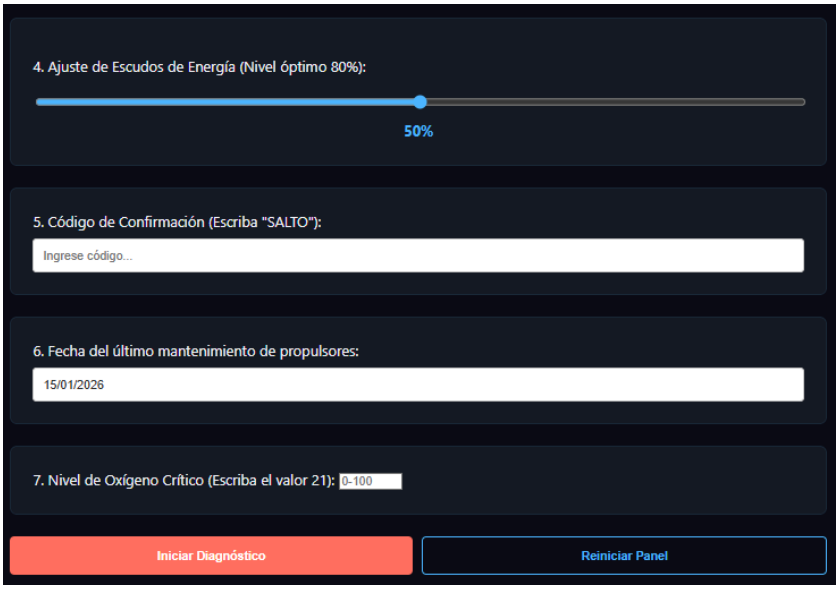
3.3 Interfaz del formulario

Parte superior:

- Nombre del capitán
- Pregunta de tipo radio.
- Pregunta de tipo select
- Pregunta de tipo checkbox

Parte inferior:

- Slider para ajustar la energía
- Campo input para introducir la palabra SALTO
- Campo input con el widget datepicker de JQuery para seleccionar una fecha
- Campo numérico para seleccionar el nivel de oxígeno.



---

4. DISEÑO Y MAQUETACIÓN (CSS3)

- **Layout Grid 3x2**: Se ha utilizado una rejilla de CSS Grid con grid-template-columns: repeat(3, 1fr) para organizar los destinos de forma simétrica.
- **Estética Cyberpunk/Espacial**: Uso de variables CSS (:root) para gestionar una paleta de colores coherente basada en el "Azul Galáctico" (#4db8ff) y "Naranja Astro" (#ff6f61).
- **Feedback Visual de Estado**: La nave cambia su imagen y aplica filtros CSS (grayscale, sepia, drop-shadow) dinámicamente según el estado de salud de la nave (Normal, Crítico o Destruido).

5. TABLA DE CUMPLIMIENTO TÉCNICO (OBLIGATORIA)

Requisito RA	Elemento Implementado	Descripción Técnica
RA4	Función Constructora	**function Nave(...)** con métodos internos.
RA4	Array Multidimensional	Matriz **destinosGalacticos** con datos mixtos.
RA5	Evento de Teclado	**keydown** (Escape) para resetear el sistema.
RA5	Evento de Foco	**focus** y **blur** para iluminar campos activos.
RA5	Validación Regex	Control estricto de caracteres en el nombre del piloto.
RA5	Feedback Dinámico	Uso de **scrollIntoView** y modificación de **innerHTML** para el informe final.

6. REFLEXIÓN TÉCNICA FINAL

Este proyecto destaca por su **originalidad** al no limitarse a un formulario estático, sino al crear un ecosistema donde el éxito en el juego depende del conocimiento técnico (manual) y la habilidad de gestión. Se han incluido comentarios exhaustivos en cada bloque de código para facilitar su mantenimiento y comprensión, cumpliendo así con los estándares de calidad profesional exigidos.

7. BIBLIOGRAFÍA

Apuntes RA4
Apuntes RA5
Apuntes JQUERY