TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ECATEPEC

INGENIERÍA INFORMÁTICA

Proyecto 1er Departamental

Presenta:

JOSÉ IGNACIO SALINAS LÓPEZ

Profesor:

LEONARDO MIGUEL MORENO VILLALBA

Grupo:

15601

Septiembre 2026

Objetivo

plicar los conocimientos de HTML5, CSS3, JavaScript y Node.js para crear un juego web completo que incluye una experiencia interactiva, un sistema de puntuaciones y un servidor para guardar datos de manera permanente.

Descripción del Proyecto

RunnerJS es un juego web de tipo endless runner donde un personaje debe saltar para evitar obstáculos, a medida que avanza el tiempo, la dificultad aumenta gradualmente para desafiar al jugador, el juego cuenta con un contador de puntuación y un sistema de Leaderboard que almacena las mejores puntuaciones en un servidor, permitiendo a los jugadores competir entre sí.

Explicación de los temas aplicados.

HTML5: Se usó para la estructura del juego, con un elemento <canvas> que sirve como el área de juego.

CSS3: Se utilizó para dar estilo a la interfaz de usuario (el menú, el game over y la tabla de puntuaciones), usando Flexbox y Grid para organizar los elementos.

JavaScript: Es el motor del juego. Se encarga de la lógica de los saltos, el movimiento de los obstáculos, la detección de colisiones y la gestión de las puntuaciones. También se usa para enviar y recibir datos del servidor mediante la API.

Node.js con Express: Se usó para crear el servidor del backend. Express gestiona las peticiones y las respuestas de la API, permitiendo que el juego guarde y obtenga las puntuaciones del archivo JSON.

Diagramas de maquetado.

─ /frontend

│ ├─ index.html # Landing (enlaza prácticas y juego)

│ ├─ /practicas/... # Todas tus prácticas previas

│ ├─ /proyecto

│ │ ├─ game.html

│ │ ├─ game.css

```
│  │  ├─ game.js              # loop, niveles, colisiones, HUD
│  │  ├─ api.js               # fetch a la API (scores)
│  │  └─ assets/              # sprites opcionales (o shapes)
├─ /backend
│  ├─ server.js
│  ├─ scores.routes.js
│  ├─ data/scores.json
│  └─ package.json
└─ /docs
   └─ RunnerJS_Proyecto.pdf      # Documento final.
```

Código fuente comentado.

INDEX

```
<!DOCTYPE html>
<html lang="es">
<head>
  <!-- Configuración de metadatos básicos -->
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Proyecto Integrador</title>

  <!-- Importación de Tailwind CSS desde CDN -->
  <script src="https://cdn.tailwindcss.com"></script>

  <!-- Estilos personalizados -->
  <style>
```

```
/* Importación de fuente Google Fonts - Inter */
@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=swap');

/* Estilos del cuerpo principal */
body {
  font-family: 'Inter', sans-serif;
  /* Fondo con imagen GIF animada, centrada y fija */
  background: url('proyecto/assets/fondoIndex.gif') no-repeat center center fixed;
  background-size: cover;
  color: #8db7ed; /* Color de texto azul claro */
}

/* Clase para efecto de vidrio (glassmorphism) */
.glass-card {
  /* Fondo semi-transparente oscuro */
  background: rgba(20, 25, 40, 0.75);
  /* Borde azul semi-transparente */
  border: 1px solid rgba(8, 44, 247, 0.768);
  /* Efecto de desenfoque para glassmorphism */
  backdrop-filter: blur(8px);
  /* Transiciones suaves para hover */
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

/* Efectos hover para las tarjetas de vidrio */
```

```
    .glass-card:hover {
      /* Elevación y escalado al pasar el mouse */
      transform: translateY(-4px) scale(1.02);
      /* Sombra brillante azul */
      box-shadow: 0 0 20px rgba(10, 21, 229, 0.777);
    }
  </style>
</head>

<body class="min-h-screen flex flex-col items-center">

  <!-- ========== SECCIÓN HEADER ========== -->
  <header class="w-full max-w-5xl p-6 rounded-xl glass-card mb-10 mt-6">
    <!-- Navegación principal -->
    <nav class="flex justify-between items-center">
      <!-- Logo/Título principal -->
      <h1 class="text-3xl font-bold text-cyan-400 drop-shadow-lg">RunnerJS</h1>

      <!-- Menú de navegación -->
      <div class="space-x-6">
        <!-- Enlaces de navegación con efectos hover -->
        <a href="#inicio" class="text-cyan-300 hover:text-cyan-400 transition">Inicio</a>
        <a href="practicas/practicasIndex.html" class="text-cyan-300 hover:text-cyan-400 transition">Prácticas</a>
        <a href="proyecto/game.html" class="text-cyan-300 hover:text-cyan-400 transition">Proyecto</a>
```

```
      <a href="proyecto/leaderboard.html" class="text-cyan-300 hover:text-cyan-400 transition">Puntuaciones(Leaderboard)</a>
    </div>
  </nav>
 </header>

 <!-- ========== SECCIÓN MAIN ========== -->
 <main class="w-full max-w-5xl grid gap-6 md:grid-cols-2 lg:grid-cols-3 px-4">

  <!-- ========== SECCIÓN DE PRÁCTICAS ========== -->
  <section id="practicas-section" class="md:col-span-2 lg:col-span-3 grid gap-6 grid-cols-1 md:grid-cols-2 lg:grid-cols-3">

    <!-- Tarjeta Práctica 1 -->
    <div class="p-6 rounded-xl glass-card">
      <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 1</h2>
      <p class="text-gray-300">Maquetación y estilos de pantalla de Login Fantástico</p>
      <a href="practicas/Practica 1 Maquetacion y estilos/html/index.html"
        class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
    </div>

    <!-- Tarjeta Práctica 2 -->
    <div class="p-6 rounded-xl glass-card">
      <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 2</h2>
      <p class="text-gray-300">Página web interactiva de "Tarjeta de Presentación"</p>
      <a href="practicas/Practica 1 y 2 Prácticas desarrollo Web/Practica 1/html/index.html"
```

```
    class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
</div>

<!-- Tarjeta Práctica 2.1 -->
<div class="p-6 rounded-xl glass-card">
  <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 2.1</h2>
  <p class="text-gray-300">Construcción de un Juego Interactivo Simple</p>
  <a href="practicas/Practica 1 y 2 Prácticas desarrollo Web/Practica 2/html/index.html"
    class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
</div>

<!-- Tarjeta Práctica 3 -->
<div class="p-6 rounded-xl glass-card">
  <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 3</h2>
  <p class="text-gray-300">Landing Page Interactiva con Validación de Formulario</p>
  <a href="practicas/Practica 3/html/index.html"
    class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
</div>

<!-- Tarjeta Práctica 4 -->
<div class="p-6 rounded-xl glass-card">
  <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 4</h2>
  <p class="text-gray-300">Juego de Memoria con Cartas Personalizadas</p>
  <a href="practicas/Practica 4/html/index.html"
    class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
```

```
    </div>

    <!-- Tarjeta Práctica 5 -->
    <div class="p-6 rounded-xl glass-card">
      <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 5</h2>
      <p class="text-gray-300">App de Clima Local con API y Geolocalización</p>
      <a href="practicas/Practica 5/html/index.html"
        class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
    </div>
  </section>

  <!-- ========== SECCIÓN DEL PROYECTO PRINCIPAL ========== -->
  <section id="proyecto" class="md:col-span-2 lg:col-span-3">
    <!-- Tarjeta destacada del proyecto principal -->
    <div class="p-10 rounded-xl glass-card text-center">
      <!-- Título del proyecto -->
      <h2 class="text-3xl font-bold text-cyan-400 mb-4">Proyecto </h2>
      <!-- Descripción del proyecto (nota: hay texto fuera del párrafo) -->
      Un juego endless runner con un sistema de puntuaciones y niveles progresivos de dificultad
      <p class="text-lg text-gray-300 mb-6">.</p>

      <!-- Botón principal para acceder al juego -->
      <a href="proyecto/game.html"
        class="bg-cyan-400 text-gray-900 font-bold py-3 px-8 rounded-full shadow-lg hover:bg-cyan-300 transition-transform transform hover:scale-105">
```

```
          Jugar Ahora
        </a>
      </div>
    </section>
  </main>
</body>
</html>
```

PRACTICA INDEX

```
<!DOCTYPE html>
<html lang="es">
<head>
    <!-- Configuración de metadatos básicos -->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Menú de Prácticas</title>

    <!-- Importación de Tailwind CSS desde CDN -->
    <script src="https://cdn.tailwindcss.com"></script>

    <!-- Estilos personalizados -->
    <style>
        /* Importación de fuente Google Fonts - Inter */
        @import
url('https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=swap');

        /* Estilos del cuerpo principal */
```

```
body {
    font-family: 'Inter', sans-serif;
    /* Fondo con imagen GIF animada (ruta relativa desde carpeta practicas) */
    background: url('../proyecto/assets/fondoIndex.gif') no-repeat center center fixed;
    background-size: cover;
    color: #8db7ed; /* Color de texto azul claro */
}

/* Clase para efecto de vidrio (glassmorphism) */
.glass-card {
    /* Fondo semi-transparente oscuro */
    background: rgba(20, 25, 40, 0.75);
    /* Borde azul semi-transparente */
    border: 1px solid rgba(8, 44, 247, 0.768);
    /* Efecto de desenfoque para glassmorphism */
    backdrop-filter: blur(8px);
    /* Transiciones suaves para animaciones hover */
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

/* Efectos hover para las tarjetas de vidrio */
.glass-card:hover {
    /* Elevación y escalado ligero al pasar el mouse */
    transform: translateY(-4px) scale(1.02);
    /* Sombra brillante azul en hover */
```

```
        box-shadow: 0 0 20px rgba(10, 21, 229, 0.777);
    }
  </style>
</head>

<body class="min-h-screen flex flex-col items-center">

  <!-- ========== SECCIÓN HEADER CON NAVEGACIÓN ========== -->
  <header class="w-full max-w-5xl p-6 rounded-xl glass-card mb-10 mt-6 flex justify-between items-center">
    <!-- Logo/Título principal (lado izquierdo) -->
    <h1 class="text-3xl font-bold text-cyan-400 drop-shadow-lg">RunnerJS</h1>

    <!-- Botón de regreso a inicio (lado derecho) -->
    <a href="../index.html"
      class="bg-cyan-400 text-gray-900 font-bold py-2 px-6 rounded-full shadow-lg hover:bg-cyan-300 transition-transform transform hover:scale-105">
      Regresar a Inicio
    </a>
  </header>

  <!-- ========== SECCIÓN MAIN - CONTENIDO PRINCIPAL ========== -->
  <main class="w-full max-w-5xl grid gap-6 md:grid-cols-2 lg:grid-cols-3 px-4">

    <!-- ========== SECCIÓN DE TODAS LAS PRÁCTICAS ========== -->
```

```
<section id="practicas-section" class="md:col-span-2 lg:col-span-3 grid gap-6 grid-cols-1 md:grid-cols-2 lg:grid-cols-3">

    <!-- ========== TARJETA PRÁCTICA 1 ========== -->
    <div class="p-6 rounded-xl glass-card">
      <!-- Título de la práctica -->
      <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 1</h2>
      <!-- Descripción de la práctica -->
      <p class="text-gray-300">Maquetación y estilos de pantalla de Login Fantástico</p>
      <!-- Enlace para acceder a la práctica -->
      <a href="Practica 1 Maquetacion y estilos/html/index.html"
        class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
    </div>

    <!-- ========== TARJETA PRÁCTICA 2 ========== -->
    <div class="p-6 rounded-xl glass-card">
      <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 2</h2>
      <p class="text-gray-300">Página web interactiva de "Tarjeta de Presentación"</p>
      <a href="Practica 1 y 2 Prácticas desarrollo Web/Practica 1/html/index.html"
        class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
    </div>

    <!-- ========== TARJETA PRÁCTICA 2.1 ========== -->
    <div class="p-6 rounded-xl glass-card">
```

```
        <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 2.1</h2>
        <p class="text-gray-300">Construcción de un Juego Interactivo Simple</p>
        <a href="Practica 1 y 2 Prácticas desarrollo Web/Practica 2/html/index.html"
          class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
      </div>

      <!-- ========== TARJETA PRÁCTICA 3 ========== -->
      <div class="p-6 rounded-xl glass-card">
        <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 3</h2>
        <p class="text-gray-300">Landing Page Interactiva con Validación de Formulario</p>
        <a href="Practica 3/html/index.html"
          class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
      </div>

      <!-- ========== TARJETA PRÁCTICA 4 ========== -->
      <div class="p-6 rounded-xl glass-card">
        <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 4</h2>
        <p class="text-gray-300">Juego de Memoria con Cartas Personalizadas</p>
        <a href="Practica 4/html/index.html"
          class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
      </div>

      <!-- ========== TARJETA PRÁCTICA 5 ========== -->
      <div class="p-6 rounded-xl glass-card">
        <h2 class="text-2xl font-semibold text-cyan-300 mb-2">Práctica 5</h2>
```

```
            <p class="text-gray-300">App de Clima Local con API y Geolocalización</p>
            <a href="Practica 5/html/index.html"
              class="mt-4 inline-block text-cyan-400 hover:underline">Ver Práctica</a>
          </div>
        </section>
    </main>
</body>
</html>
```

LEADERBOARD

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tabla de Puntuaciones</title>
    <link rel="stylesheet" href="game.css">
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-gray-100 font-sans">
    <div id="leaderboard-container" class="container-main">
        <a href="../index.html" class="self-start mb-4 bg-blue-600 text-white font-bold py-2 px-4 rounded-full shadow-lg hover:bg-blue-700 transition-colors duration-300">
            Regresar a Inicio
        </a>
        <div class="w-full max-w-md bg-white p-6 rounded-xl shadow-md">
```

```
        <h3 class="text-2xl font-bold mb-4 text-gray-800 text-center">Tabla de Mejores Puntuaciones</h3>
        <ul id="scores-list" class="space-y-2">
            <!-- Las puntuaciones se insertarán aquí dinámicamente -->
        </ul>
    </div>
</div>
<script type="module">
    import { api } from './api.js';

    document.addEventListener('DOMContentLoaded', () => {
        loadLeaderboard();
    });

    async function loadLeaderboard() {
        const scoresList = document.getElementById('scores-list');

        try {
            const scores = await api.getScores();
            scoresList.innerHTML = '';
            if (scores.length === 0) {
                scoresList.innerHTML = '<li class="text-center text-gray-500">No hay puntuaciones guardadas.</li>';
            } else {
                scores.forEach((s, index) => {
                    const li = document.createElement('li');
```

```
                li.classList.add('p-3', 'bg-gray-100', 'rounded-lg', 'flex', 'justify-between', 'items-center');
                li.innerHTML = `
                    <span class="font-bold text-gray-800">${index + 1}. ${s.name}</span>
                    <span class="text-gray-600">Nivel ${s.level}</span>
                    <span class="font-mono text-blue-600">${s.score}</span>
                `;
                scoresList.appendChild(li);
            });
        }
    } catch (error) {
        scoresList.innerHTML = '<li class="text-center text-red-500">Error al cargar la tabla de puntuaciones.</li>';
        console.error(error);
    }
}
</script>
</body>
</html>
```

GAME.JS

```
// ========== IMPORTACIONES ==========
import { api } from './api.js';

// ========== CONFIGURACIÓN INICIAL DEL JUEGO ==========
// Dimensiones del jugador (dinosaurio/caballero)
```

```
const DINO_WIDTH = 50;
const DINO_HEIGHT = 80;

// Física del juego
const GRAVITY = 0.7;        // Fuerza de gravedad que afecta al jugador
const JUMP_POWER = -15;     // Velocidad inicial del salto (negativa = hacia arriba)

// Configuración de obstáculos
const MIN_OBSTACLE_DISTANCE = 350;  // Distancia mínima entre obstáculos
const OBSTACLE_WIDTH = 50;          // Ancho de los obstáculos
const OBSTACLE_HEIGHT = 50;         // Alto de los obstáculos

// ========== VARIABLES GLOBALES DEL JUEGO ==========
// Variables del canvas y contexto de dibujo
let canvas, ctx;

// Variables de control de bucles del juego
let gameLoopInterval;    // Bucle principal del juego (60 FPS)
let scoreLoopInterval;   // Bucle de actualización de puntuación
let speedUpInterval;     // Bucle de aumento de velocidad/dificultad

// Variables de física y estado del jugador
let isJumping = false;   // Indica si el jugador está saltando
let yVelocity = 0;       // Velocidad vertical del jugador
```

```
// Variables de puntuación y progreso
let score = 0;           // Puntuación actual
let highScore = 0;       // Puntuación máxima guardada
let gameSpeed = 5;       // Velocidad de desplazamiento del juego
let level = 1;           // Nivel actual basado en la velocidad

// Variables de juego
let obstacles = [];      // Array que contiene todos los obstáculos
let isGameOver = false;  // Estado del juego

// ========== REFERENCIAS A ELEMENTOS DEL DOM ==========
const scoreDisplay = document.getElementById('score-display');
const highScoreDisplay = document.getElementById('high-score-display');
const levelDisplay = document.getElementById('level-display');
const gameOverScreen = document.getElementById('game-over-screen');
const finalScoreDisplay = document.getElementById('final-score');
const restartBtn = document.getElementById('restartGameBtn');
const saveScoreBtn = document.getElementById('saveScoreBtn');
const playerNameInput = document.getElementById('player-name');
const saveMessage = document.getElementById('save-message');

// ========== OBJETO JUGADOR ==========
// Estado inicial del jugador (dinosaurio/caballero)
let dino = {
    x: 50,              // Posición horizontal (fija)
```

```
    y: 350,            // Posición vertical (variable con saltos)
    width: DINO_WIDTH,  // Ancho del jugador
    height: DINO_HEIGHT // Alto del jugador
};

// ========== CARGA DE IMÁGENES ==========
// Imagen del jugador (caballero)
let dinoImage = new Image();
let isDinoImageLoaded = false;
dinoImage.src = 'assets/caballero.png';
dinoImage.onload = () => {
    isDinoImageLoaded = true;
    console.log("✅ Imagen del jugador cargada");
};

// Imagen del obstáculo (enemigo)
let obstacleImage = new Image();
let isObstacleImageLoaded = false;
obstacleImage.src = 'assets/enemigo.png';
obstacleImage.onload = () => {
    isObstacleImageLoaded = true;
    console.log("✅ Imagen del obstáculo cargada");
};

// Imagen del fondo
```

```
let backgroundImage = new Image();
let isBackgroundLoaded = false;
backgroundImage.src = 'assets/fondo.png';
backgroundImage.onload = () => {
    isBackgroundLoaded = true;
    console.log("✅ Imagen del fondo cargada");
};
backgroundImage.onerror = (e) => {
    console.error("❌ Error al cargar fondo:", backgroundImage.src, e);
};

// ========== FUNCIÓN PRINCIPAL DE INICIALIZACIÓN ==========
export function initializeGame() {
    // Resetear posición y física del jugador
    dino.y = 350;
    yVelocity = 0;
    isJumping = false;

    // Resetear variables de juego
    score = 0;
    gameSpeed = 5;
    level = 1;
    obstacles = [];
    isGameOver = false;
```

```
// Resetear interfaz de usuario
gameOverScreen.classList.add('hide');
playerNameInput.value = '';
saveMessage.textContent = '';

// Actualizar displays de puntuación y nivel
scoreDisplay.textContent = 'Puntuación: 0';
levelDisplay.textContent = 'Nivel: 1';

// Limpiar intervalos anteriores para evitar múltiples bucles
if (gameLoopInterval) clearInterval(gameLoopInterval);
if (scoreLoopInterval) clearInterval(scoreLoopInterval);
if (speedUpInterval) clearInterval(speedUpInterval);

// Iniciar bucles principales del juego
gameLoopInterval = setInterval(gameLoop, 1000 / 60); // 60 FPS
scoreLoopInterval = setInterval(updateScore, 100);   // Actualizar score cada 100ms

// Sistema de aumento progresivo de dificultad
setTimeout(() => {
    speedUpInterval = setInterval(() => {
        gameSpeed += 0.25;  // Incrementar velocidad
        // Calcular nivel basado en la velocidad actual
        level = Math.floor((gameSpeed - 5) / 0.25) + 1;
        levelDisplay.textContent = `Nivel: ${level}`;
```

```
        }, 2000); // Aumenta la velocidad cada 2 segundos
    }, 20000); // Comienza a aumentar después de 20 segundos

    // Agregar event listener para controles
    document.addEventListener('keydown', handleJump);
}

// ========== BUCLE PRINCIPAL DEL JUEGO ==========
function gameLoop() {
    if (isGameOver) return; // No ejecutar si el juego terminó

    update();         // Actualizar posiciones y lógica
    draw();           // Dibujar todo en el canvas
    checkCollision(); // Verificar colisiones
}

// ========== ACTUALIZACIÓN DE PUNTUACIÓN ==========
function updateScore() {
    if (isGameOver) return;
    score++; // Incrementar puntuación cada 100ms
    scoreDisplay.textContent = `Puntuación: ${score}`;
}

// ========== ACTUALIZACIÓN DE LÓGICA DEL JUEGO ==========
function update() {
```

```
    // ===== FÍSICA DEL JUGADOR =====
    yVelocity += GRAVITY;  // Aplicar gravedad
    dino.y += yVelocity;   // Actualizar posición vertical

    // Detectar cuando el jugador toca el suelo
    if (dino.y + dino.height > canvas.height) {
        dino.y = canvas.height - dino.height; // Posicionar en el suelo
        isJumping = false;                    // Ya no está saltando
    }

    // ===== GENERACIÓN DE OBSTÁCULOS =====
    let lastObstacle = obstacles[obstacles.length - 1];

    // Crear nuevo obstáculo si no hay ninguno o si hay suficiente distancia
    if (!lastObstacle || (canvas.width - lastObstacle.x) > (MIN_OBSTACLE_DISTANCE +
Math.random() * 200)) {
        obstacles.push({
            x: canvas.width,                    // Aparece desde el borde derecho
            y: canvas.height - OBSTACLE_HEIGHT,     // En el suelo
            width: OBSTACLE_WIDTH,
            height: OBSTACLE_HEIGHT
        });
    }

    // ===== MOVIMIENTO DE OBSTÁCULOS =====
    obstacles.forEach(obstacle => {
```

```
        obstacle.x -= gameSpeed; // Mover obstáculos hacia la izquierda
    });

    // Eliminar obstáculos que salieron de la pantalla (optimización)
    obstacles = obstacles.filter(obstacle => obstacle.x + obstacle.width > 0);
}

// ========== FUNCIÓN DE DIBUJO/RENDERIZADO ==========
function draw() {
    // Limpiar canvas para el nuevo frame
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // ===== DIBUJAR FONDO =====
    if (isBackgroundLoaded) {
        // Usar imagen de fondo si está cargada
        ctx.drawImage(backgroundImage, 0, 0, canvas.width, canvas.height);
    } else {
        // Color de respaldo mientras carga la imagen
        ctx.fillStyle = '#fbbf24'; // Amarillo
        ctx.fillRect(0, 0, canvas.width, canvas.height);
    }

    // ===== DIBUJAR SUELO =====
    ctx.fillStyle = '#4b5563'; // Color gris
    ctx.fillRect(0, canvas.height - 10, canvas.width, 10);
```

```
    // ===== DIBUJAR JUGADOR =====
    if (isDinoImageLoaded) {
        // Usar sprite del caballero si está cargado
        ctx.drawImage(dinoImage, dino.x, dino.y, dino.width, dino.height);
    } else {
        // Rectángulo de respaldo mientras carga la imagen
        ctx.fillStyle = '#1f2937'; // Color gris oscuro
        ctx.fillRect(dino.x, dino.y, dino.width, dino.height);
    }

    // ===== DIBUJAR OBSTÁCULOS =====
    obstacles.forEach(obstacle => {
        if (isObstacleImageLoaded) {
            // Usar sprite del enemigo si está cargado
            ctx.drawImage(obstacleImage, obstacle.x, obstacle.y, obstacle.width,
obstacle.height);
        } else {
            // Rectángulo de respaldo mientras carga la imagen
            ctx.fillStyle = '#864e4eff'; // Color marrón
            ctx.fillRect(obstacle.x, obstacle.y, obstacle.width, obstacle.height);
        }
    });
}

// ========== DETECCIÓN DE COLISIONES ==========
```

```
function checkCollision() {
    obstacles.forEach(obstacle => {
        // Algoritmo de detección de colisión por rectángulos (AABB)
        if (
            dino.x < obstacle.x + obstacle.width &&         // Borde izquierdo del jugador < borde derecho del obstáculo
            dino.x + dino.width > obstacle.x &&             // Borde derecho del jugador > borde izquierdo del obstáculo
            dino.y < obstacle.y + obstacle.height &&        // Borde superior del jugador < borde inferior del obstáculo
            dino.y + dino.height > obstacle.y           // Borde inferior del jugador > borde superior del obstáculo
        ) {
            gameOver(); // Si hay colisión, terminar el juego
        }
    });
}

// ========== CONTROL DE SALTO ==========
function handleJump(event) {
    // Saltar solo con la barra espaciadora, si no está saltando y el juego no terminó
    if (event.key === ' ' && !isJumping && !isGameOver) {
        isJumping = true;
        yVelocity = JUMP_POWER; // Aplicar velocidad inicial del salto
    }
}
```

```
// ========== FUNCIÓN DE GAME OVER ==========
function gameOver() {
    isGameOver = true;

    // Detener todos los bucles del juego
    clearInterval(gameLoopInterval);
    clearInterval(scoreLoopInterval);
    if (speedUpInterval) clearInterval(speedUpInterval);

    // Mostrar pantalla de game over
    gameOverScreen.classList.remove('hide');
    finalScoreDisplay.textContent = score;

    // Verificar si se estableció un nuevo récord
    if (score > highScore) {
        highScore = score;
        highScoreDisplay.textContent = `Máximo: ${highScore}`;
        // Guardar nuevo récord en localStorage
        localStorage.setItem('highScore', highScore);
    }

    // Remover event listener para evitar saltos después del game over
    document.removeEventListener('keydown', handleJump);
}
```

```
// ========== EVENT LISTENERS PARA BOTONES ==========
// Botón de reiniciar juego
restartBtn.addEventListener('click', () => {
    gameOverScreen.classList.add('hide');
    initializeGame(); // Reiniciar el juego completo
});

// Botón de guardar puntuación
saveScoreBtn.addEventListener('click', async () => {
    const playerName = playerNameInput.value.trim();

    // Validar que se ingresó un nombre
    if (!playerName) {
        saveMessage.textContent = 'Por favor, ingresa tu nombre.';
        return;
    }

    // Crear objeto con datos de la puntuación
    const newScore = {
        name: playerName,
        score: score,
        level: level
    };

    try {
```

```
        saveMessage.textContent = 'Guardando...';
        // Llamar a la API para guardar la puntuación
        await api.saveScore(newScore);
        saveMessage.textContent = 'Puntuación guardada con éxito.';
    } catch (error) {
        saveMessage.textContent = 'Error al guardar la puntuación. Intenta de nuevo.';
        console.error('Error al guardar puntuación:', error);
    }
});

// ========== FUNCIÓN DE CARGA DE PUNTUACIÓN MÁXIMA ==========
function loadHighScore() {
    // Recuperar puntuación máxima del localStorage
    const storedHighScore = localStorage.getItem('highScore');
    if (storedHighScore) {
        highScore = parseInt(storedHighScore);
        highScoreDisplay.textContent = `Máximo: ${highScore}`;
    }
}

// ========== INICIALIZACIÓN AL CARGAR LA PÁGINA ==========
window.onload = () => {
    // Obtener referencias del canvas y contexto 2D
    canvas = document.getElementById('gameCanvas');
    ctx = canvas.getContext('2d');
```

```
    loadHighScore();   // Cargar puntuación máxima guardada
    initializeGame();  // Inicializar y comenzar el juego
};
```

GAME.HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <!-- Configuración básica del documento -->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RunnerJS Game</title>

    <!-- Importación de estilos -->
    <link rel="stylesheet" href="game.css">          <!-- CSS personalizado del juego -->
    <script src="https://cdn.tailwindcss.com"></script> <!-- Framework CSS Tailwind desde CDN -->
</head>

<body class="bg-gray-100 font-sans">

    <!-- ========== CONTENEDOR PRINCIPAL DEL JUEGO ========== -->
    <div id="game-container" class="container-main">

        <!-- ===== BOTÓN DE NAVEGACIÓN ===== -->
        <!-- Botón para regresar a la página principal -->
```

```
<a href="../index.html"
  class="self-start mb-4 bg-blue-600 text-white font-bold py-2 px-4 rounded-full shadow-lg hover:bg-blue-700 transition-colors duration-300">
    Regresar a Inicio
</a>

<!-- ========== TABLERO PRINCIPAL DEL JUEGO ========== -->
<div id="game-board" class="flex flex-col items-center bg-white shadow-xl rounded-xl p-6">

    <!-- ===== TÍTULO DEL JUEGO ===== -->
    <h2 class="text-3xl font-bold mb-4 text-gray-800">RunnerJS</h2>

    <!-- ===== PANEL DE INFORMACIÓN ===== -->
    <!-- Barra superior con estadísticas del juego -->
    <div class="flex justify-between w-full mb-4 px-4 text-gray-600 font-bold">
        <!-- Puntuación actual -->
        <span id="score-display">Puntuación: 0</span>
        <!-- Puntuación máxima (récord personal) -->
        <span id="high-score-display">Máximo: 0</span>
        <!-- Nivel actual basado en velocidad -->
        <span id="level-display">Nivel: 1</span>
    </div>

    <!-- ===== CANVAS DEL JUEGO ===== -->
    <!-- Lienzo donde se renderiza todo el juego -->
```

```
<canvas id="gameCanvas"
        class="border-2 border-gray-300"
        width="800"
        height="400">
</canvas>

<!-- ========== PANTALLA DE GAME OVER ========== -->
<!-- Overlay que aparece cuando termina el juego -->
<div id="game-over-screen"
     class="hide absolute inset-0 flex flex-col items-center justify-center bg-gray-900 bg-opacity-75 rounded-xl">

    <!-- ===== TÍTULO DE GAME OVER ===== -->
    <h1 class="text-5xl font-bold text-white mb-4">Game Over</h1>

    <!-- ===== PUNTUACIÓN FINAL ===== -->
    <p class="text-white text-2xl mb-2">
        Puntuación: <span id="final-score">0</span>
    </p>

    <!-- ===== FORMULARIO PARA GUARDAR PUNTUACIÓN ===== -->
    <div id="score-form" class="mt-4 flex flex-col items-center">
        <!-- Campo de entrada para el nombre del jugador -->
        <input type="text"
               id="player-name"
               placeholder="Tu nombre"
```

```
                class="p-2 rounded-lg mb-2 text-center text-black">

            <!-- Botón para guardar la puntuación en la base de datos -->
            <button id="saveScoreBtn"
                class="bg-green-500 text-white font-bold py-2 px-4 rounded-lg shadow-md hover:bg-green-600 transition-colors duration-300">
                Guardar Puntuación
            </button>

            <!-- Mensaje de estado del guardado -->
            <p id="save-message" class="text-white mt-2"></p>
        </div>

        <!-- ===== BOTONES DE ACCIÓN ===== -->
        <div class="flex space-x-4 mt-6">
            <!-- Botón para reiniciar el juego -->
            <button id="restartGameBtn"
                class="bg-blue-600 text-white font-bold py-2 px-6 rounded-full shadow-lg hover:bg-blue-700 transition-colors duration-300">
                Reiniciar
            </button>

            <!-- Enlace para ver la tabla de puntuaciones -->
            <a href="leaderboard.html"
              class="bg-purple-600 text-white font-bold py-2 px-6 rounded-full shadow-lg hover:bg-purple-700 transition-colors duration-300">
```

```
                Ver tabla de puntajes
            </a>
          </div>
        </div>
      </div>
    </div>

    <!-- ========== CARGA DEL SCRIPT PRINCIPAL ========== -->
    <!-- Importar el archivo JavaScript del juego como módulo ES6 -->
    <script type="module" src="game.js"></script>
</body>
</html>
```

GAME.CSS

```
/* ========== ESTILOS GLOBALES ========== */

/* Configuración del cuerpo principal de la página */
body {
  font-family: 'Inter', sans-serif;                    /* Fuente personalizada Google Fonts */
  background: url('assets/fondo1.png') no-repeat center center fixed; /* Imagen de fondo fija y centrada */
  background-size: cover;                              /* Escalar imagen para cubrir toda la pantalla */
  color: #8db7ed;                                      /* Color de texto principal: azul claro */
}
```

```
/* ========== CONTENEDOR PRINCIPAL ========== */

/* Layout principal que centra todo el contenido del juego */
.container-main {
    min-height: 100vh;        /* Altura mínima de toda la ventana del navegador */
    display: flex;            /* Usar flexbox para el layout */
    flex-direction: column;   /* Organizar elementos verticalmente */
    justify-content: center;  /* Centrar contenido verticalmente */
    align-items: center;      /* Centrar contenido horizontalmente */
    padding: 1rem;            /* Espaciado interno de 16px (1rem) en todos los lados */
}

/* ========== ESTILOS DEL CANVAS ========== */

/* Configuración visual del canvas donde se renderiza el juego */
canvas {
    background-color: #1455d6c4;           /* Fondo azul semi-transparente (rgba con opacidad) */
    border-radius: 1rem;                   /* Bordes redondeados de 16px */
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1); /* Sombra sutil: offset-y 4px, blur 12px, negro 10% opacidad */
}

/* ========== UTILIDADES ========== */

/* Clase utility para ocultar elementos completamente */
```

```
.hide {
   display: none !important;  /* Forzar ocultación con !important para override cualquier otro estilo */
}

/* ========== TABLERO DEL JUEGO ========== */

/* Contenedor principal del área de juego */
#game-board {
   background-color: rgb(184, 198, 240);  /* Fondo azul claro semi-transparente para contraste */
   /* Nota: Este color proporciona contraste con el fondo y mejora la legibilidad */
}
```

API.S

```
// URL base para las peticiones HTTP a la API de puntuaciones
// ☜ Funciona tanto en desarrollo local como en producción (Render)
const BASE_URL = "/api/scores";

// ========== MÓDULO API EXPORTADO ==========

// Objeto que contiene todos los métodos para interactuar con la API
export const api = {

  // ========== MÉTODO PARA OBTENER PUNTUACIONES ==========
```

```
  /**
   * Obtiene todas las puntuaciones desde el servidor
   * @returns {Promise<Array>} Array de objetos con las puntuaciones o array vacío si hay error
   */
  getScores: async () => {
    try {
      // Realizar petición GET al endpoint de puntuaciones
      const res = await fetch(BASE_URL);

      // Verificar si la respuesta es exitosa (status 200-299)
      if (!res.ok) throw new Error("Error al obtener puntajes");

      // Convertir la respuesta JSON a objeto JavaScript
      return await res.json();

    } catch (error) {
      // Manejar errores: log en consola y retornar array vacío como fallback
      console.error("❌ Error en getScores:", error);
      return []; // Retorna array vacío para evitar errores en la UI
    }
  },

  // ========== MÉTODO PARA GUARDAR PUNTUACIÓN ==========
  /**
```

```
   * Guarda una nueva puntuación en el servidor
   * @param {Object} newScore - Objeto con los datos de la nueva puntuación
   * @param {string} newScore.name - Nombre del jugador
   * @param {number} newScore.score - Puntuación obtenida
   * @param {number} newScore.level - Nivel alcanzado
   * @returns {Promise<Object>} Respuesta del servidor o objeto con success: false si hay error
   */
  saveScore: async (newScore) => {
    try {
      // Realizar petición POST al endpoint de puntuaciones
      const res = await fetch(BASE_URL, {
        method: "POST",                          // Método HTTP para crear recurso
        headers: {
          "Content-Type": "application/json"        // Especificar que enviamos JSON
        },
        body: JSON.stringify(newScore),             // Convertir objeto a JSON string
      });

      // Verificar si la respuesta es exitosa (status 200-299)
      if (!res.ok) throw new Error("Error al guardar puntaje");

      // Convertir la respuesta JSON a objeto JavaScript
      return await res.json();

    } catch (error) {
```

```
      // Manejar errores: log en consola y retornar objeto de error
      console.error("❌ Error en saveScore:", error);
      return { success: false }; // Retorna objeto con flag de error para manejo en UI
    }
  },
};
```

SERVER.JS

```
const express = require('express');
const path = require('path');
const scoresRouter = require('./scores.routes');

const app = express();
const PORT = process.env.PORT || 3000; // 👈 Render asigna el puerto

app.use(express.json());

// Sirve archivos estáticos desde la carpeta 'frontend'
app.use(express.static(path.join(__dirname, '../frontend')));

// Configura las rutas de la API (ahora en /api/scores)
app.use('/api/scores', scoresRouter);

// Fallback: si no encuentra ruta, devuelve index.html (SPA)
```

```
app.get('/users/*path', (req, res) => {
  // Ahora el parámetro tiene nombre
});

app.listen(PORT, () => {
  console.log(`Servidor escuchando en http://localhost:${PORT}`);
});
```

SCORES.ROUTES.JS

```
const express = require('express');
const fs = require('fs');
const path = require('path');

const router = express.Router();
const scoresFilePath = path.join(__dirname, 'data/scores.json');

// Leer puntuaciones desde archivo
function readScores() {
  try {
    if (!fs.existsSync(scoresFilePath)) {
      return [];
    }
    const data = fs.readFileSync(scoresFilePath, 'utf8');
    return JSON.parse(data || '[]');
  } catch (error) {
```

```
    console.error("Error leyendo scores:", error);
    return [];
  }
}

// Guardar puntuaciones en archivo
function saveScores(scores) {
  try {
    fs.writeFileSync(scoresFilePath, JSON.stringify(scores, null, 2));
  } catch (error) {
    console.error("Error guardando scores:", error);
  }
}

// GET /api/scores - Obtiene la lista de puntuaciones
router.get('/', (req, res) => {
  const scores = readScores();
  res.json(scores);
});

// POST /api/scores - Guarda una nueva puntuación
router.post('/', (req, res) => {
  const newScore = req.body;

  if (!newScore.name || typeof newScore.score !== "number") {
```

```
    return res.status(400).json({ message: "Formato inválido" });
  }

  const scores = readScores();
  scores.push(newScore);
  scores.sort((a, b) => b.score - a.score);

  const topScores = scores.slice(0, 10); // Mantener solo el top 10
  saveScores(topScores);

  res.status(201).json({ message: "Puntuación guardada con éxito." });
});

module.exports = router;
```

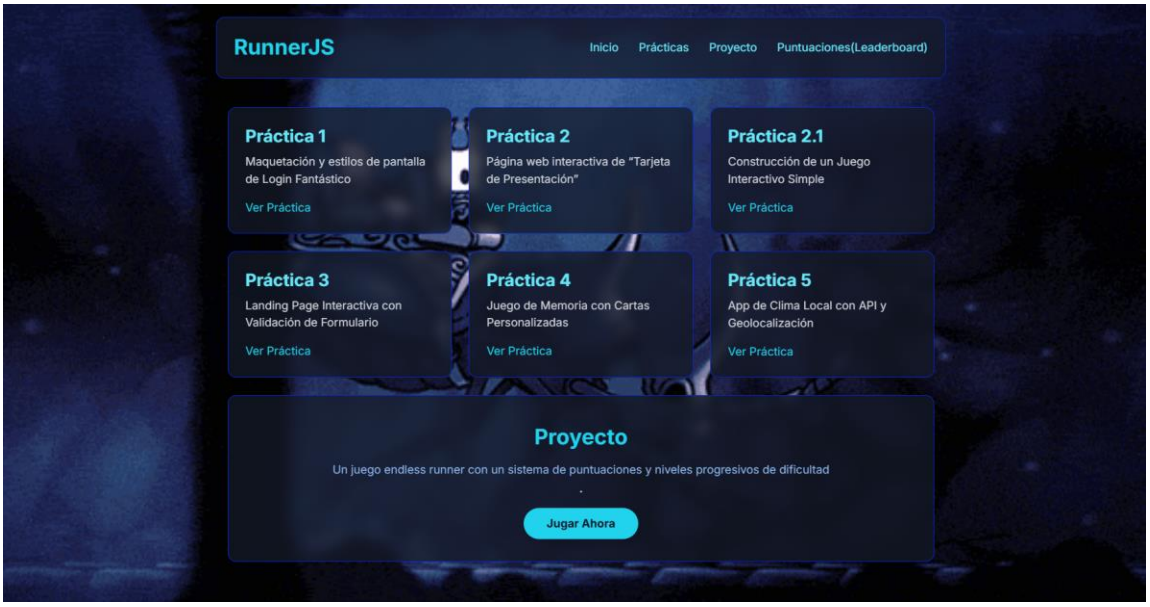
PACKEGE.JSON

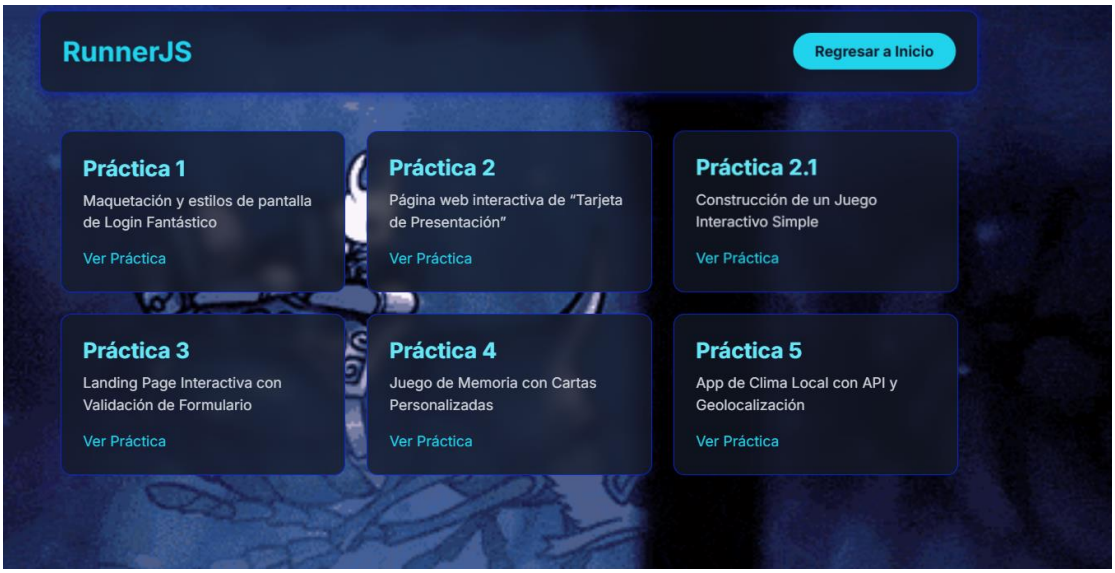
```
"name": "backend",              // Nombre del paquete/proyecto
  "version": "1.0.0",           // Versión actual del proyecto (SemVer: Major.Minor.Patch)
  "main": "server.js",          // Archivo principal de entrada de la aplicación

  // ========== SCRIPTS DE NPM ==========

  "scripts": {
    // Script de pruebas - actualmente no implementado
    "test": "echo \"Error: no test specified\" && exit 1",
```

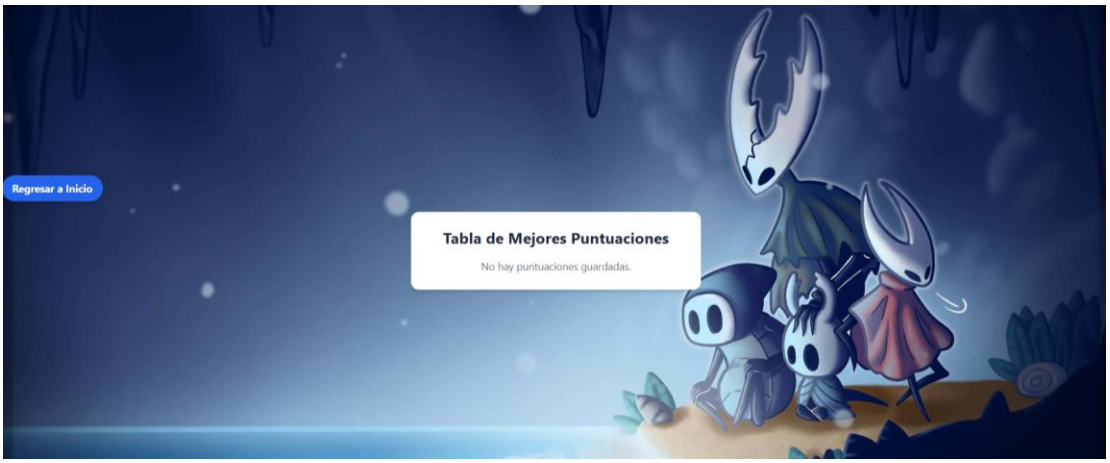
```
  // Script para iniciar el servidor en producción
  "start": "node server.js"

  // 💡 Scripts adicionales que podrías agregar:
  // "dev": "nodemon server.js",          // Para desarrollo con auto-reload
  // "build": "echo \"No build needed\"",  // Para builds si fuera necesario
  // "lint": "eslint .",                  // Para linting de código
},

// ========== METADATOS DEL PROYECTO ==========

"keywords": [],              // Palabras clave para búsqueda en npm (vacío actualmente)
"author": "",                // Autor del proyecto (vacío actualmente)
"license": "ISC",            // Licencia del proyecto (ISC es permisiva como MIT)
"description": "",           // Descripción del proyecto (vacío actualmente)

// ========== DEPENDENCIAS DE PRODUCCIÓN ==========

"dependencies": {
  // Framework web para Node.js - versión 5.1.0 o superior compatible
  "express": "^5.1.0"

  // 📝 Nota: El símbolo ^ permite actualizaciones menores y parches
  // ^5.1.0 = acepta 5.1.0, 5.2.0, 5.9.0, pero NO 6.0.0
```

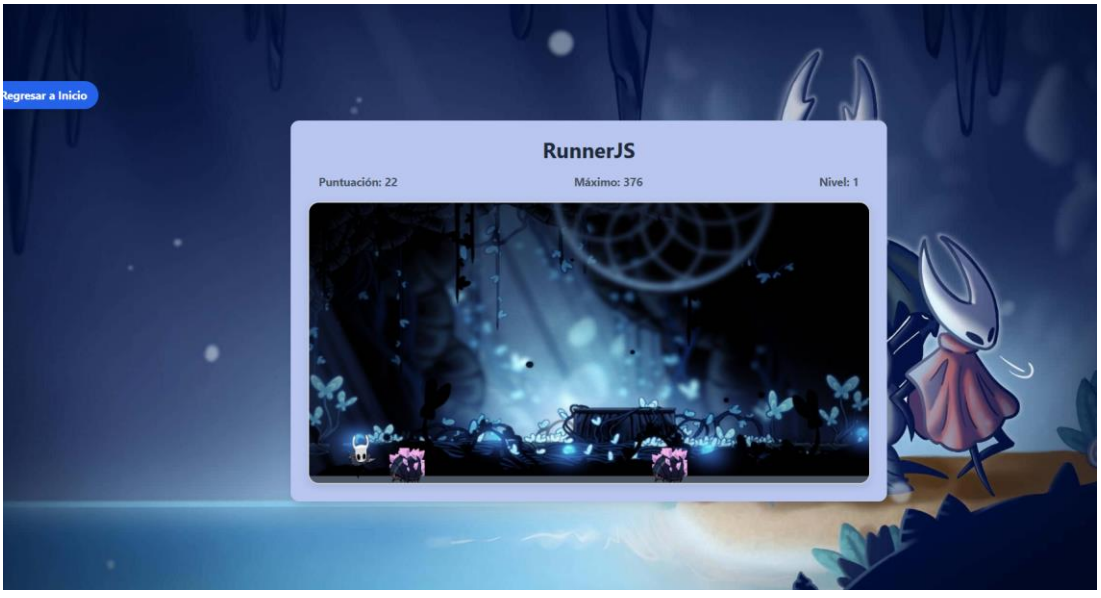
```
  },

 // ========== DEPENDENCIAS DE DESARROLLO ==========

 "devDependencies": {
   // Herramienta para reiniciar automáticamente la aplicación al detectar cambios
   "nodemon": "^3.1.10"

   // 💡 Se usa solo en desarrollo, no se instala en producción
   // Comando típico: npm run dev (con script "dev": "nodemon server.js")
 }
}
```

Capturas de pantalla de funcionamiento.




RunnerJS
Regresar a Inicio
Práctica 1
Maquetación y estilos de pantalla de Login Fantástico
Ver Práctica
Práctica 2
Página web interactiva de "Tarjeta de Presentación"
Ver Práctica
Práctica 2.1
Construcción de un Juego Interactivo Simple
Ver Práctica
Práctica 3
Landing Page Interactiva con Validación de Formulario
Ver Práctica
Práctica 4
Juego de Memoria con Cartas Personalizadas
Ver Práctica
Práctica 5
App de Clima Local con API y Geolocalización
Ver Práctica


Regresar a Inicio
Tabla de Mejores Puntuaciones
No hay puntuaciones guardadas.


Regresar a Inicio
RunnerJS
Puntuación: 22
Máximo: 376
Nivel: 1

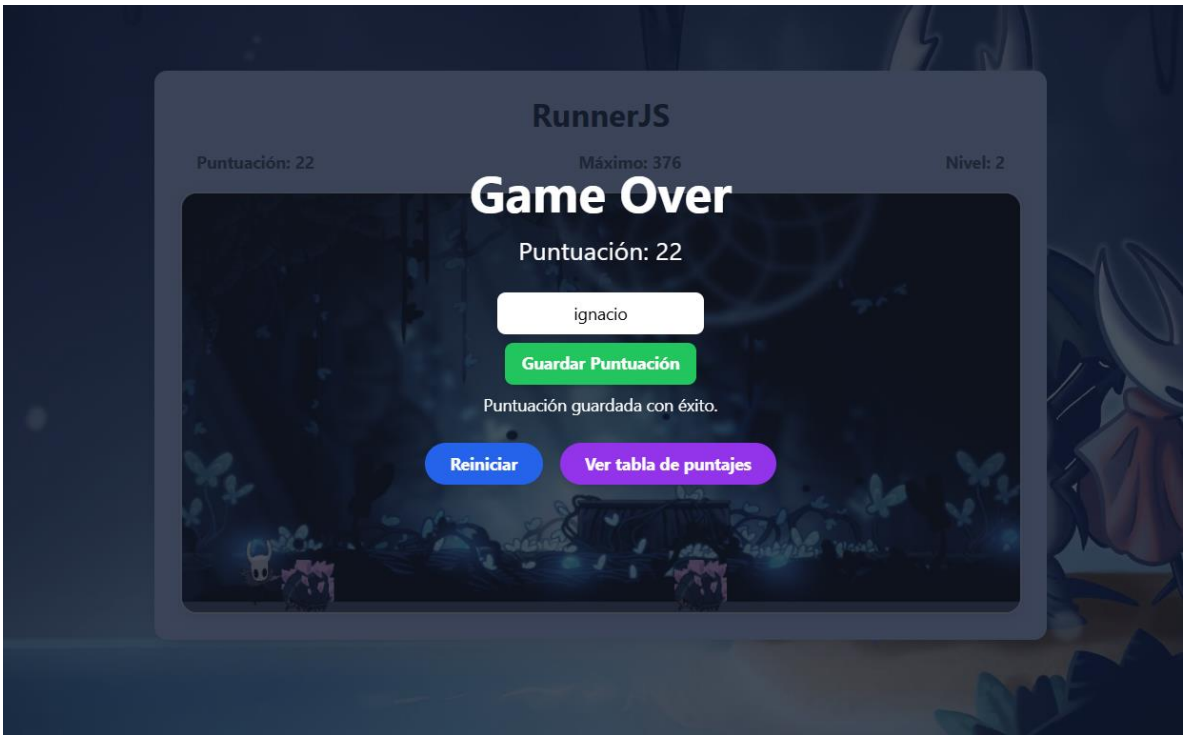

RunnerJS
Puntuación: 22
Máximo: 376
Nivel: 2
Game Over
Puntuación: 22
ignacio
Guardar Puntuación
Puntuación guardada con éxito.
Reiniciar
Ver tabla de puntajes


Tabla de Mejores Puntuaciones
1. ignacio
Nivel 1
22

Link al repositorio y al sitio publica

Repositorio:

https://github.com/JoseIgnacioSalinasLopez/Proyecto-1er-Departamental.git

Sitio publico:

https://api-node-jrsy.onrender.com

https://joseignaciosalinaslopez.github.io/Proyecto-1er-Departamental/frontend/index.html

Respuestas a las 10 preguntas de reflexión.

1. ¿Qué ventajas y limitaciones tiene usar Canvas para este juego?

Ventajas: El Canvas permite controlar cada píxel, lo que te da total libertad para crear gráficos y animaciones personalizados, es ideal para juegos 2D, ya que es muy eficiente para dibujar formas, imágenes y texto directamente.

Limitaciones: Requiere que dibujes cada elemento manualmente, no hay una estructura de elementos que puedas inspeccionar o manipular, lo que hace que la detección de interacciones y la depuración sean más complejas.

2. ¿Cómo se diseñó la progresión de los niveles y la dificultad?

La dificultad se diseñó de forma progresiva y gradual, en lugar de un aumento repentino, la velocidad del juego permanece constante durante los primeros 20 segundos, después de ese tiempo, la velocidad aumenta en un pequeño valor (0.25) cada 2 segundos. Esto hace que el jugador tenga tiempo para acostumbrarse y luego enfrente un desafío creciente de manera más justa.

3. ¿Qué método de detección de colisiones se usó y por qué?

Se utilizó el método de "detección de colisiones por caja" este método compara las coordenadas y dimensiones del jugador y los obstáculos para ver si sus cajas imaginarias se superponen, es un método simple, rápido y muy eficiente para juegos 2D, lo que lo hace perfecto para este tipo de proyecto.

4. ¿Qué mejoras harías en la accesibilidad del juego?

-Añadiría sonidos (saltos, colisiones) para los jugadores con discapacidades visuales.

-Ofrecería alternativas al teclado, como controles en pantalla para dispositivos táctiles.

-Añadiría una pausa automática al cambiar de pestaña en el navegador para que el juego no se pierda.

5. ¿Qué validaciones aplicaste en el Leaderboard?

En el Leaderboard, se aplicó una validación simple para asegurarse de que el nombre del jugador no esté vacío antes de guardar la puntuación.

6. ¿Cómo asegurarías la integridad de los scores guardados en el servidor?

Para asegurar la integridad, aplicaría validaciones del lado del servidor, el servidor validaría que la puntuación enviada sea un número y que no sea imposible de alcanzar en el tiempo de juego, para evitar que los jugadores envíen puntuaciones falsas.

7. ¿Qué diferencias notaste entre guardar datos en localStorage y en la API?

La principal diferencia es la persistencia. localStorage solo guarda los datos en el navegador del usuario, lo que significa que las puntuaciones son privadas para esa persona y no se pueden compartir, en cambio, una API y un servidor guardan los datos de forma centralizada, lo que permite crear una tabla de puntuaciones global a la que todos pueden acceder.

8. ¿Cómo modularizaste el código del juego para que fuera más fácil de mantener?

El código se modularizó al separar el proyecto en directorios y archivos específicos para cada función:

HTML, CSS y JavaScript se dividieron para la interfaz, los estilos y la lógica.

La API del cliente (api.js) se separó para manejar la comunicación con el servidor.

El backend se dividió en un servidor principal (server.js) y un archivo de rutas (scores.routes.js), lo que organiza la lógica de la API en un solo lugar.

9. ¿Qué pruebas hiciste para validar que las colisiones y puntuaciones funcionaran correctamente?

Las pruebas se hicieron de manera manual. Para las colisiones, se jugaron múltiples partidas para observar que el juego terminara correctamente al chocar con un obstáculo. Para las puntuaciones, se revisó que el contador aumentara de manera constante y que la puntuación máxima se actualizara en el navegador.

10. ¿Qué mejora de alto impacto implementarías si tuvieras más tiempo?

Implementaría una autenticación de usuarios. Esto permitiría a los jugadores crear perfiles únicos, lo que haría la tabla de puntuaciones más personal y competitiva. Además, permitiría guardar el progreso de los jugadores, como los niveles desbloqueados, en sus perfiles.