```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Royal Card Duel | Jogo Premium Completo</title>
                                                                                <link
href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@700;900&family=Mo
ntserrat:wght@400;600;800&display=swap" rel="stylesheet">
                                                                <link                rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
    <style>
        /* ===== VARIÁVEIS GLOBAIS ===== */
        :root {
            --primary-dark: #0a0618;
            --primary: #1a1433;
            --secondary: #2a2350;
            --accent-gold: #FFD700;
            --accent-red: #e63946;
            --accent-blue: #457b9d;
            --accent-green: #2a9d8f;
            --light: #f1faee;
            --dark: #121212;
            --card-bg: linear-gradient(145deg, #f8f9fa, #e9ecef);
            --card-shadow: 0 15px 30px rgba(0, 0, 0, 0.4);
            --transition: all 0.5s cubic-bezier(0.22, 0.61, 0.36, 1);
            --text-glow: 0 0 15px rgba(255, 215, 0, 0.7);
            --neon-glow: 0 0 10px var(--accent-gold), 0 0 20px var(--accent-gold);
            --player1-color: #ff6b6b;
            --player2-color: #48dbfb;
            --player3-color: #1dd1a1;
            --player4-color: #9b59b6;
            --player5-color: #f39c12;
            --modal-bg: rgba(10, 6, 24, 0.95);
        }

        /* ===== RESET E BASE ===== */
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Montserrat', sans-serif;
            background: linear-gradient(135deg, var(--primary-dark), var(--primary));
            color: var(--light);
            min-height: 100vh;
```

```
    overflow-x: hidden;
    line-height: 1.6;
}

/* ===== EFEITOS DE FUNDO ===== */
.luxury-bg {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background:
        radial-gradient(circle at 20% 30%, rgba(255, 215, 0, 0.05) 0%, transparent 25%),
        radial-gradient(circle at 80% 70%, rgba(230, 57, 70, 0.05) 0%, transparent 25%),
        url('https://www.transparenttextures.com/patterns/black-linen.png');
    z-index: -2;
}

.particles {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    z-index: -1;
    overflow: hidden;
}

.particle {
    position: absolute;
    background: rgba(255, 255, 255, 0.5);
    border-radius: 50%;
    animation: floatParticle linear infinite;
}

@keyframes floatParticle {
    0% { transform: translateY(0) translateX(0); opacity: 0; }
    10% { opacity: 1; }
    90% { opacity: 1; }
    100% { transform: translateY(-100vh) translateX(20px); opacity: 0; }
}

/* ===== CONTAINER PRINCIPAL ===== */
.game-container {
    width: 95%;
    max-width: 1400px;
    margin: 2rem auto;
    background: rgba(26, 20, 51, 0.9);
```

```
            backdrop-filter: blur(15px);
            border-radius: 25px;
            padding: 3rem;
            box-shadow: 0 25px 50px rgba(0, 0, 0, 0.5);
            border: 1px solid rgba(255, 215, 0, 0.2);
            position: relative;
            overflow: hidden;
        }

        .game-container::before {
            content: '';
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
                background: linear-gradient(135deg, transparent 65%, rgba(255, 215, 0, 0.05)
100%);
            pointer-events: none;
        }

        /* ===== CABEÇALHO ===== */
        header {
            text-align: center;
            margin-bottom: 3rem;
            position: relative;
        }

        .title-container {
            position: relative;
            display: inline-block;
            margin-bottom: 1rem;
        }

        h1 {
            font-family: 'Playfair Display', serif;
            font-size: 4.5rem;
            font-weight: 900;
                    background: linear-gradient(to right, var(--player1-color), var(--player2-color),
var(--player3-color));
            -webkit-background-clip: text;
            background-clip: text;
            color: transparent;
            text-shadow: var(--text-glow);
            position: relative;
            z-index: 2;
            letter-spacing: 2px;
        }
```

```
h1::after {
    content: 'ROYAL CARD DUEL';
    position: absolute;
    top: 5px;
    left: 5px;
    background: linear-gradient(to right, var(--accent-red), var(--accent-blue));
    -webkit-background-clip: text;
    background-clip: text;
    color: transparent;
    z-index: -1;
    filter: blur(3px);
}

.subtitle {
    font-size: 1.3rem;
    letter-spacing: 3px;
    color: rgba(255, 255, 255, 0.8);
    text-transform: uppercase;
    margin-bottom: 0.5rem;
}

.divider {
    width: 150px;
    height: 3px;
    background: linear-gradient(to right, transparent, var(--accent-gold), transparent);
    margin: 1rem auto;
    border: none;
}

/* ===== SEÇÕES DO JOGO ===== */
.section {
    display: none;
    animation: fadeIn 1s ease-out;
}

@keyframes fadeIn {
    from { opacity: 0; transform: translateY(30px); }
    to { opacity: 1; transform: translateY(0); }
}

.section.active {
    display: block;
}

/* ===== INTRODUÇÃO ===== */
#intro-section {
    max-width: 800px;
```

```
    margin: 0 auto;
    text-align: center;
}

.intro-text {
    font-size: 1.2rem;
    margin-bottom: 2rem;
    color: rgba(255, 255, 255, 0.9);
}

.highlight {
    color: var(--accent-gold);
    font-weight: 600;
}

/* ===== CONFIGURAÇÕES ===== */
#settings-section {
    max-width: 800px;
    margin: 0 auto;
    background: rgba(0, 0, 0, 0.3);
    padding: 2.5rem;
    border-radius: 20px;
    border: 1px solid rgba(255, 215, 0, 0.2);
}

.settings-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 2rem;
    margin-top: 2rem;
}

.setting-group {
    margin-bottom: 1.5rem;
}

.setting-group h3 {
    color: var(--accent-gold);
    margin-bottom: 1rem;
    font-size: 1.3rem;
    display: flex;
    align-items: center;
    gap: 0.8rem;
}

.setting-group i {
    font-size: 1.5rem;
}
```

```
.setting-option {
    display: flex;
    align-items: center;
    margin-bottom: 0.8rem;
}

.setting-option input {
    margin-right: 0.8rem;
}

.setting-option label {
    cursor: pointer;
}

.player-color-selector {
    display: flex;
    align-items: center;
    margin-bottom: 0.8rem;
}

.player-color-selector label {
    min-width: 100px;
}

.color-preview {
    width: 20px;
    height: 20px;
    border-radius: 50%;
    margin-left: 10px;
    border: 2px solid white;
}

/* ===== REGRAS ===== */
#rules-section {
    max-width: 900px;
    margin: 0 auto;
    background: rgba(0, 0, 0, 0.3);
    padding: 2.5rem;
    border-radius: 20px;
    border: 1px solid rgba(255, 215, 0, 0.2);
}

.rules-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 2rem;
    margin-top: 2rem;
```

```
}

.rule-card {
    background: rgba(42, 35, 80, 0.6);
    padding: 1.5rem;
    border-radius: 15px;
    border-left: 4px solid var(--accent-gold);
    transition: var(--transition);
}

.rule-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3);
}

.rule-card h3 {
    color: var(--accent-gold);
    margin-bottom: 1rem;
    font-size: 1.3rem;
    display: flex;
    align-items: center;
    gap: 0.8rem;
}

.rule-card i {
    font-size: 1.5rem;
}

/* ===== JOGO ===== */
#game-section {
    position: relative;
}

.game-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 2rem;
    padding-bottom: 1rem;
    border-bottom: 1px solid rgba(255, 255, 255, 0.1);
}

.game-info {
    display: flex;
    gap: 2rem;
}

.info-item {
```

```
    display: flex;
    align-items: center;
    gap: 0.5rem;
    font-size: 1.1rem;
}

.info-item i {
    color: var(--accent-gold);
    font-size: 1.3rem;
}

/* ===== JOGADORES ===== */
.players-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 2rem;
    margin: 3rem 0;
}

.player {
    background: rgba(42, 35, 80, 0.6);
    border-radius: 20px;
    padding: 2rem;
    transition: var(--transition);
    position: relative;
    overflow: hidden;
    border: 1px solid rgba(255, 255, 255, 0.1);
}

.player-1 {
    border-top: 4px solid var(--player1-color);
}

.player-2 {
    border-top: 4px solid var(--player2-color);
}

.player-3 {
    border-top: 4px solid var(--player3-color);
}

.player-4 {
    border-top: 4px solid var(--player4-color);
}

.player-5 {
    border-top: 4px solid var(--player5-color);
}
```

```
.player.active {
    transform: translateY(-10px);
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.4);
    border-color: rgba(255, 215, 0, 0.3);
}

.player-winner {
    position: relative;
    overflow: visible;
}

.player-winner::after {
    content: '🏆';
    position: absolute;
    top: -20px;
    right: -20px;
    font-size: 3rem;
    filter: drop-shadow(0 0 10px var(--accent-gold));
    animation: pulse 2s infinite;
    z-index: 10;
}

@keyframes pulse {
    0% { transform: scale(1); }
    50% { transform: scale(1.2); }
    100% { transform: scale(1); }
}

.player-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 1.5rem;
}

.player-name {
    font-size: 1.5rem;
    font-weight: bold;
    display: flex;
    align-items: center;
    gap: 0.8rem;
}

.player-1 .player-name {
    color: var(--player1-color);
}
```

```
.player-2 .player-name {
    color: var(--player2-color);
}

.player-3 .player-name {
    color: var(--player3-color);
}

.player-4 .player-name {
    color: var(--player4-color);
}

.player-5 .player-name {
    color: var(--player5-color);
}

.player-score {
    background: rgba(0, 0, 0, 0.4);
    padding: 0.5rem 1.2rem;
    border-radius: 50px;
    font-weight: bold;
    border: 1px solid;
    font-size: 1.1rem;
}

.player-1 .player-score {
    color: var(--player1-color);
    border-color: var(--player1-color);
}

.player-2 .player-score {
    color: var(--player2-color);
    border-color: var(--player2-color);
}

.player-3 .player-score {
    color: var(--player3-color);
    border-color: var(--player3-color);
}

.player-4 .player-score {
    color: var(--player4-color);
    border-color: var(--player4-color);
}

.player-5 .player-score {
    color: var(--player5-color);
    border-color: var(--player5-color);
```

```
        }

        .player-cards {
            display: flex;
            justify-content: center;
            gap: 1rem;
            min-height: 180px;
        }

        /* ===== CARTAS ===== */
        .card {
            width: 120px;
            height: 180px;
            background: white;
            border-radius: 12px;
            display: flex;
            flex-direction: column;
            justify-content: space-between;
            padding: 1rem;
            font-weight: bold;
            box-shadow: var(--card-shadow);
            transition: var(--transition);
            position: relative;
            cursor: pointer;
            user-select: none;
            transform-style: preserve-3d;
            backface-visibility: hidden;
            background-size: cover;
            background-position: center;
        }

        .card::before {
            content: '';
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
              background: linear-gradient(135deg, rgba(255, 255, 255, 0.1), rgba(255, 255, 255,
0));
            border-radius: 12px;
            pointer-events: none;
        }

        .card:hover {
            transform: translateY(-15px) rotate(2deg);
            box-shadow: 0 25px 50px rgba(0, 0, 0, 0.5);
        }
```

```
.card.selected {
    transform: translateY(-25px) scale(1.1);
    box-shadow: var(--neon-glow);
    border: 2px solid var(--accent-gold);
    z-index: 5;
}

.card-value {
    font-size: 1.8rem;
    font-weight: bold;
    align-self: flex-start;
    z-index: 2;
}

.card-suit {
    font-size: 3.5rem;
    align-self: center;
    line-height: 1;
    z-index: 2;
}

.card-value-bottom {
    font-size: 1.8rem;
    font-weight: bold;
    align-self: flex-end;
    transform: rotate(180deg);
    z-index: 2;
}

/* Naipes coloridos */
.hearts, .diamonds {
    color: var(--accent-red);
}

.clubs, .spades {
    color: var(--dark);
}

/* Carta virada */
.card-back {
    width: 100%;
    height: 100%;
    background: linear-gradient(45deg, var(--primary), var(--secondary));
    color: white;
    display: flex;
    justify-content: center;
    align-items: center;
```

```
    font-size: 2.5rem;
    transform: rotateY(180deg);
    position: absolute;
    top: 0;
    left: 0;
    border-radius: 12px;
    background-image: url('https://deckofcardsapi.com/static/img/back.png');
    background-size: cover;
    background-position: center;
}

.card-back::before {
    content: '';
    position: absolute;
    top: 10px;
    left: 10px;
    right: 10px;
    bottom: 10px;
    border: 2px dashed rgba(255, 255, 255, 0.2);
    border-radius: 8px;
}

.card-front {
    transform: rotateY(0deg);
    position: absolute;
    width: 100%;
    height: 100%;
    backface-visibility: hidden;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    padding: 1rem;
    background-color: white;
    border-radius: 12px;
}

/* Animação de distribuição */
@keyframes dealAnimation {
    0% { transform: translateY(-100px) rotate(-15deg); opacity: 0; }
    100% { transform: translateY(0) rotate(0); opacity: 1; }
}

.card-dealing {
    animation: dealAnimation 0.8s cubic-bezier(0.175, 0.885, 0.32, 1.275) forwards;
}

/* Animação de virar carta */
@keyframes flipIn {
```

```
    0% { transform: rotateY(90deg); }
    100% { transform: rotateY(0deg); }
}

.flipping {
    animation: flipIn 0.6s ease forwards;
}

/* ===== CONTROLES DO JOGO ===== */
.game-controls {
    display: flex;
    justify-content: center;
    gap: 2rem;
    margin-top: 3rem;
    flex-wrap: wrap;
}

.btn {
    padding: 1.2rem 2.5rem;
    border: none;
    border-radius: 50px;
    font-size: 1.2rem;
    font-weight: bold;
    cursor: pointer;
    transition: var(--transition);
    position: relative;
    overflow: hidden;
    display: flex;
    align-items: center;
    gap: 1rem;
    z-index: 1;
    min-width: 220px;
    justify-content: center;
}

.btn::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: linear-gradient(45deg, var(--accent-gold), var(--accent-red));
    z-index: -1;
    transition: var(--transition);
}

.btn:hover::before {
```

```
    transform: scale(1.05);
    filter: brightness(1.1);
}

.btn:disabled {
    opacity: 0.7;
    cursor: not-allowed;
}

.btn:disabled::before {
    background: #666;
}

.btn-secondary {
    background: transparent;
    border: 2px solid var(--accent-gold);
    color: var(--accent-gold);
}

.btn-secondary::before {
    background: transparent;
}

.btn-secondary:hover {
    background: var(--accent-gold);
    color: var(--primary-dark);
}

/* ===== RESULTADO ===== */
.game-result {
    text-align: center;
    margin-top: 2rem;
    padding: 3rem;
    background: rgba(0, 0, 0, 0.4);
    border-radius: 20px;
    border: 1px solid var(--accent-gold);
    display: none;
    animation: fadeIn 0.8s ease-out;
    position: relative;
    overflow: hidden;
}

.game-result::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
```

```
        height: 100%;
                background: linear-gradient(45deg, transparent, rgba(255, 215, 0, 0.05),
transparent);
    }

    .result-title {
        font-family: 'Playfair Display', serif;
        font-size: 3rem;
        color: var(--accent-gold);
        margin-bottom: 1.5rem;
        text-shadow: var(--text-glow);
    }

    .result-message {
        font-size: 1.8rem;
        margin-bottom: 2rem;
        line-height: 1.4;
    }

    .winner-name {
        font-weight: bold;
        text-transform: uppercase;
        letter-spacing: 2px;
        font-size: 2rem;
        display: inline-block;
        margin: 0 0.5rem;
    }

    .player-1-name {
        color: var(--player1-color);
    }

    .player-2-name {
        color: var(--player2-color);
    }

    .player-3-name {
        color: var(--player3-color);
    }

    .player-4-name {
        color: var(--player4-color);
    }

    .player-5-name {
        color: var(--player5-color);
    }
```

```
.hand-description {
    font-size: 1.3rem;
    color: rgba(255, 255, 255, 0.9);
    margin-top: 1rem;
    font-style: italic;
}

/* ===== NAVEGAÇÃO ===== */
.nav-buttons {
    display: flex;
    justify-content: center;
    gap: 1.5rem;
    margin-top: 3rem;
    flex-wrap: wrap;
}

/* ===== MODAL ===== */
.modal {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.8);
    z-index: 100;
    justify-content: center;
    align-items: center;
    animation: fadeIn 0.3s ease-out;
}

.modal-content {
    background: var(--modal-bg);
    padding: 3rem;
    border-radius: 20px;
    max-width: 800px;
    width: 90%;
    max-height: 90vh;
    overflow-y: auto;
    position: relative;
    border: 2px solid var(--accent-gold);
    box-shadow: 0 0 30px rgba(255, 215, 0, 0.3);
}

.close-modal {
    position: absolute;
    top: 1rem;
    right: 1rem;
```

```
    font-size: 1.5rem;
    color: var(--accent-gold);
    cursor: pointer;
    transition: var(--transition);
}

.close-modal:hover {
    transform: rotate(90deg);
    color: var(--accent-red);
}

/* ===== HISTÓRICO ===== */
.history-container {
    margin-top: 2rem;
    max-height: 300px;
    overflow-y: auto;
    padding-right: 1rem;
}

.history-item {
    background: rgba(42, 35, 80, 0.6);
    padding: 1rem;
    margin-bottom: 0.8rem;
    border-radius: 10px;
    border-left: 3px solid var(--accent-gold);
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.history-winner {
    font-weight: bold;
}

/* ===== BARRA DE PROGRESSO ===== */
.progress-container {
    width: 100%;
    background: rgba(255, 255, 255, 0.1);
    border-radius: 10px;
    margin: 1rem 0;
    height: 20px;
    overflow: hidden;
}

.progress-bar {
    height: 100%;
    background: linear-gradient(to right, var(--player1-color), var(--player2-color));
    border-radius: 10px;
```

```
    transition: width 0.5s ease;
}

/* ===== RESPONSIVIDADE ===== */
@media (max-width: 1200px) {
    .game-container {
        padding: 2rem;
    }

    h1 {
        font-size: 3.5rem;
    }
}

@media (max-width: 768px) {
    .game-container {
        padding: 1.5rem;
    }

    h1 {
        font-size: 2.5rem;
    }

    .subtitle {
        font-size: 1.1rem;
    }

    .players-container {
        grid-template-columns: 1fr;
    }

    .card {
        width: 90px;
        height: 140px;
    }

    .game-controls, .nav-buttons {
        flex-direction: column;
        align-items: center;
    }

    .btn {
        width: 100%;
        max-width: 300px;
    }
}

@media (max-width: 480px) {
```

```
            h1 {
                font-size: 2rem;
            }

            .card {
                width: 80px;
                height: 120px;
                padding: 0.8rem;
            }

            .card-suit {
                font-size: 2.5rem;
            }
        }
    </style>
</head>
<body>
    <!-- Efeitos de Fundo -->
    <div class="luxury-bg"></div>
    <div class="particles" id="particles"></div>

    <!-- Container Principal do Jogo -->
    <div class="game-container">
        <!-- Cabeçalho -->
        <header>
            <div class="title-container">
                <h1>ROYAL CARD DUEL</h1>
                <p class="subtitle">O DUELO DEFINITIVO ENTRE JOGADORES</p>
                <hr class="divider">
            </div>
        </header>

        <!-- Seção de Introdução -->
        <section id="intro-section" class="section active">
            <div class="intro-text">
                <p>Bem-vindo ao <span class="highlight">Royal Card Duel</span>, o jogo de cartas mais intenso para até 5 jogadores!</p>
                <p>Neste duelo estratégico, você competirá contra oponentes em batalhas emocionantes de pura habilidade.</p>
                <p>Cada jogador tem seu estilo único e cartas especiais - escolha sabiamente para dominar o jogo!</p>
            </div>

            <div class="nav-buttons">
                <button class="btn" id="startGameBtn">
                    <i class="fas fa-play"></i> Começar Jogo
                </button>
                <button class="btn btn-secondary" id="showSettingsBtn">
```

```
            <i class="fas fa-cog"></i> Configurações
        </button>
        <button class="btn btn-secondary" id="showRulesBtn">
            <i class="fas fa-book"></i> Ver Regras
        </button>
    </div>
</section>

<!-- Seção de Configurações -->
<section id="settings-section" class="section">
    <h2 class="section-title">Configurações do Jogo</h2>

    <div class="settings-container">
        <div class="settings-grid">
            <div class="setting-group">
                <h3><i class="fas fa-users"></i> Jogadores</h3>
                <div class="setting-option">
                    <input type="radio" id="players3" name="players" value="3" checked>
                    <label for="players3">3 Jogadores</label>
                </div>
                <div class="setting-option">
                    <input type="radio" id="players4" name="players" value="4">
                    <label for="players4">4 Jogadores</label>
                </div>
                <div class="setting-option">
                    <input type="radio" id="players5" name="players" value="5">
                    <label for="players5">5 Jogadores</label>
                </div>
            </div>

            <div class="setting-group">
                <h3><i class="fas fa-trophy"></i> Vitórias</h3>
                <div class="setting-option">
                    <input type="radio" id="wins3" name="wins" value="3" checked>
                    <label for="wins3">Melhor de 3</label>
                </div>
                <div class="setting-option">
                    <input type="radio" id="wins5" name="wins" value="5">
                    <label for="wins5">Melhor de 5</label>
                </div>
                <div class="setting-option">
                    <input type="radio" id="wins7" name="wins" value="7">
                    <label for="wins7">Melhor de 7</label>
                </div>
            </div>

            <div class="setting-group">
                <h3><i class="fas fa-chess"></i> Dificuldade</h3>
```

```
                <div class="setting-option">
                    <input type="radio" id="easy" name="difficulty" value="easy" checked>
                    <label for="easy">Fácil</label>
                </div>
                <div class="setting-option">
                    <input type="radio" id="medium" name="difficulty" value="medium">
                    <label for="medium">Médio</label>
                </div>
                <div class="setting-option">
                    <input type="radio" id="hard" name="difficulty" value="hard">
                    <label for="hard">Difícil</label>
                </div>
            </div>

            <div class="setting-group">
                <h3><i class="fas fa-palette"></i> Cores dos Jogadores</h3>
                <div class="player-color-selector">
                    <label for="player1Color">Jogador 1:</label>
                    <input type="color" id="player1Color" value="#ff6b6b">
                    <div class="color-preview" style="background-color: #ff6b6b;"></div>
                </div>
                <div class="player-color-selector">
                    <label for="player2Color">Jogador 2:</label>
                    <input type="color" id="player2Color" value="#48dbfb">
                    <div class="color-preview" style="background-color: #48dbfb;"></div>
                </div>
                <div class="player-color-selector">
                    <label for="player3Color">Jogador 3:</label>
                    <input type="color" id="player3Color" value="#1dd1a1">
                    <div class="color-preview" style="background-color: #1dd1a1;"></div>
                </div>
                    <div class="player-color-selector" id="player4ColorSelector" style="display:
none;">
                    <label for="player4Color">Jogador 4:</label>
                    <input type="color" id="player4Color" value="#9b59b6">
                    <div class="color-preview" style="background-color: #9b59b6;"></div>
                </div>
                    <div class="player-color-selector" id="player5ColorSelector" style="display:
none;">
                    <label for="player5Color">Jogador 5:</label>
                    <input type="color" id="player5Color" value="#f39c12">
                    <div class="color-preview" style="background-color: #f39c12;"></div>
                </div>
            </div>
        </div>
    </div>

    <div class="nav-buttons">
```

```
            <button class="btn" id="saveSettingsBtn">
                <i class="fas fa-save"></i> Salvar Configurações
            </button>
            <button class="btn btn-secondary" id="backToIntroBtn">
                <i class="fas fa-arrow-left"></i> Voltar
            </button>
        </div>
    </section>

    <!-- Seção de Regras -->
    <section id="rules-section" class="section">
        <h2 class="section-title">Regras do Jogo</h2>

        <div class="rules-container">
            <p>Royal Card Duel é um jogo estratégico onde jogadores competem para ter a melhor combinação de cartas.</p>

            <div class="rules-grid">
                <div class="rule-card">
                    <h3><i class="fas fa-users"></i> Jogadores</h3>
                    <p>3-5 jogadores competem em cada rodada. Cada um tem um estilo único representado por cores distintas.</p>
                </div>

                <div class="rule-card">
                    <h3><i class="fas fa-chess"></i> Combinações</h3>
                    <p>Trinca (3 cartas iguais) > Sequência (3 em ordem) > Flush (mesmo naipe) > Par (2 cartas iguais) > Carta Alta.</p>
                </div>

                <div class="rule-card">
                    <h3><i class="fas fa-star"></i> Pontuação</h3>
                    <p>Ás (14), Rei (13), Dama (12), Valete (11), Números (2-10). Copas/Ouros valem +0.5 ponto extra.</p>
                </div>

                <div class="rule-card">
                    <h3><i class="fas fa-trophy"></i> Vitória</h3>
                    <p>O jogador com a combinação mais forte vence a rodada. O primeiro a atingir o número de vitórias configurado vence o jogo.</p>
                </div>

                <div class="rule-card">
                    <h3><i class="fas fa-random"></i> Cartas Especiais</h3>
                    <p>Algumas cartas têm efeitos especiais que podem mudar o curso do jogo quando jogadas.</p>
                </div>
```

```
            <div class="rule-card">
                <h3><i class="fas fa-clock"></i> Turnos</h3>
                    <p>Os jogadores jogam em turnos, selecionando uma carta por rodada. A ordem dos turnos muda a cada rodada.</p>
            </div>
        </div>
    </div>

    <div class="nav-buttons">
        <button class="btn" id="rulesToGameBtn">
            <i class="fas fa-play"></i> Ir para o Jogo
        </button>
        <button class="btn btn-secondary" id="backToIntroFromRulesBtn">
            <i class="fas fa-arrow-left"></i> Voltar
        </button>
    </div>
</section>

<!-- Seção do Jogo -->
<section id="game-section" class="section">
    <div class="game-header">
        <div class="game-info">
            <div class="info-item">
                <i class="fas fa-chess-board"></i>
                <span>Rodada: <span id="round-count">1</span></span>
            </div>
            <div class="info-item">
                <i class="fas fa-trophy"></i>
                <span>Vitórias: <span id="win-count">0/0/0</span></span>
            </div>
            <div class="info-item">
                <i class="fas fa-clock"></i>
                <span>Turno: <span id="current-turn">Jogador 1</span></span>
            </div>
        </div>

        <div class="game-actions">
            <button class="btn btn-secondary" id="gameToRulesBtn">
                <i class="fas fa-book"></i> Regras
            </button>
            <button class="btn btn-secondary" id="showHistoryBtn">
                <i class="fas fa-history"></i> Histórico
            </button>
        </div>
    </div>

    <div class="players-container" id="playersContainer">
        <!-- Jogadores serão inseridos dinamicamente via JavaScript -->
```

```
        </div>

        <!-- Resultado do Jogo -->
        <div class="game-result" id="gameResult">
            <h3 class="result-title">Resultado</h3>
                    <p class="result-message">O vencedor é: <span class="winner-name player-1-name" id="winnerName">Jogador 1</span>!</p>
            <p class="hand-description" id="resultDetails">Com uma trinca de Áses!</p>

            <div class="progress-container">
                <div class="progress-bar" id="progressBar" style="width: 33%"></div>
            </div>
        </div>

        <!-- Controles do Jogo -->
        <div class="game-controls">
            <button class="btn" id="dealBtn">
                <i class="fas fa-hand-cards"></i> Distribuir Cartas
            </button>
            <button class="btn" id="playCardBtn" disabled>
                <i class="fas fa-play-circle"></i> Jogar Carta Selecionada
            </button>
            <button class="btn btn-secondary" id="newRoundBtn" disabled>
                <i class="fas fa-redo"></i> Nova Rodada
            </button>
        </div>
    </section>
</div>

<!-- Modal de Histórico -->
<div class="modal" id="historyModal">
    <div class="modal-content">
        <span class="close-modal" id="closeHistoryModal">&times;</span>
        <h2><i class="fas fa-history"></i> Histórico do Jogo</h2>
        <div class="history-container" id="historyContainer">
            <!-- Itens de histórico serão inseridos aqui -->
        </div>
    </div>
</div>

<script>
    document.addEventListener('DOMContentLoaded', () => {
        // ===== EFEITOS VISUAIS =====
        // Criar partículas flutuantes
        const particlesContainer = document.getElementById('particles');
        for (let i = 0; i < 50; i++) {
            const particle = document.createElement('div');
            particle.classList.add('particle');
```

```
    const size = Math.random() * 2 + 1;
    particle.style.width = `${size}px`;
    particle.style.height = `${size}px`;

    particle.style.left = `${Math.random() * 100}%`;
    particle.style.top = `${Math.random() * 100}%`;

    const duration = Math.random() * 30 + 20;
    particle.style.animationDuration = `${duration}s`;
    particle.style.animationDelay = `${Math.random() * 10}s`;

    particlesContainer.appendChild(particle);
}

// ===== GERENCIAMENTO DE SEÇÕES =====
const sections = {
    intro: document.getElementById('intro-section'),
    settings: document.getElementById('settings-section'),
    rules: document.getElementById('rules-section'),
    game: document.getElementById('game-section')
};

const navButtons = {
    startGame: document.getElementById('startGameBtn'),
    showSettings: document.getElementById('showSettingsBtn'),
    showRules: document.getElementById('showRulesBtn'),
    saveSettings: document.getElementById('saveSettingsBtn'),
    backToIntro: document.getElementById('backToIntroBtn'),
    backToIntroFromRules: document.getElementById('backToIntroFromRulesBtn'),
    rulesToGame: document.getElementById('rulesToGameBtn'),
    gameToRules: document.getElementById('gameToRulesBtn'),
    showHistory: document.getElementById('showHistoryBtn')
};

function showSection(section) {
    Object.values(sections).forEach(s => s.classList.remove('active'));
    sections[section].classList.add('active');
    window.scrollTo({ top: 0, behavior: 'smooth' });
}

navButtons.startGame.addEventListener('click', () => {
    showSection('game');
    initGame();
});

navButtons.showSettings.addEventListener('click', () => showSection('settings'));
navButtons.showRules.addEventListener('click', () => showSection('rules'));
```

```
        navButtons.saveSettings.addEventListener('click', saveSettings);
        navButtons.backToIntro.addEventListener('click', () => showSection('intro'));
                        navButtons.backToIntroFromRules.addEventListener('click',  ()  =>
showSection('intro'));
        navButtons.rulesToGame.addEventListener('click', () => {
            showSection('game');
            initGame();
        });
        navButtons.gameToRules.addEventListener('click', () => showSection('rules'));
        navButtons.showHistory.addEventListener('click', showHistoryModal);

        // ===== CONFIGURAÇÕES DO JOGO =====
        let gameSettings = {
            playerCount: 3,
            winsNeeded: 3,
            difficulty: 'easy',
            playerColors: ['#ff6b6b', '#48dbfb', '#1dd1a1', '#9b59b6', '#f39c12']
        };

        // Elementos de configuração
        const playerCountInputs = document.querySelectorAll('input[name="players"]');
        const winsNeededInputs = document.querySelectorAll('input[name="wins"]');
        const difficultyInputs = document.querySelectorAll('input[name="difficulty"]');
        const playerColorInputs = [
            document.getElementById('player1Color'),
            document.getElementById('player2Color'),
            document.getElementById('player3Color'),
            document.getElementById('player4Color'),
            document.getElementById('player5Color')
        ];
        const playerColorPreviews = document.querySelectorAll('.color-preview');
        const player4ColorSelector = document.getElementById('player4ColorSelector');
        const player5ColorSelector = document.getElementById('player5ColorSelector');

        // Atualizar visibilidade dos seletores de cor
        playerCountInputs.forEach(input => {
            input.addEventListener('change', () => {
                const count = parseInt(input.value);
                player4ColorSelector.style.display = count >= 4 ? 'flex' : 'none';
                player5ColorSelector.style.display = count >= 5 ? 'flex' : 'none';
            });
        });

        // Atualizar visualização das cores
        playerColorInputs.forEach((input, index) => {
            input.addEventListener('input', () => {
                playerColorPreviews[index].style.backgroundColor = input.value;
            });
```

```
        });

        // Carregar configurações salvas
        function loadSettings() {
            const savedSettings = localStorage.getItem('royalCardDuelSettings');
            if (savedSettings) {
                gameSettings = JSON.parse(savedSettings);

                // Atualizar UI com as configurações salvas

document.querySelector(`input[name="players"][value="${gameSettings.playerCount}"]`).che
cked = true;

document.querySelector(`input[name="wins"][value="${gameSettings.winsNeeded}"]`).check
ed = true;

document.querySelector(`input[name="difficulty"][value="${gameSettings.difficulty}"]`).check
ed = true;

                playerColorInputs.forEach((input, index) => {
                    input.value = gameSettings.playerColors[index];
                                            playerColorPreviews[index].style.backgroundColor =
gameSettings.playerColors[index];
                });

                // Mostrar/ocultar seletores de cor conforme necessário
                    player4ColorSelector.style.display = gameSettings.playerCount >= 4 ? 'flex' :
'none';
                    player5ColorSelector.style.display = gameSettings.playerCount >= 5 ? 'flex' :
'none';
            }
        }

        // Salvar configurações
        function saveSettings() {
                                                                gameSettings.playerCount =
parseInt(document.querySelector('input[name="players"]:checked').value);
                                                              gameSettings.winsNeeded =
parseInt(document.querySelector('input[name="wins"]:checked').value);
                                                                  gameSettings.difficulty =
document.querySelector('input[name="difficulty"]:checked').value;

            playerColorInputs.forEach((input, index) => {
                gameSettings.playerColors[index] = input.value;
            });

            localStorage.setItem('royalCardDuelSettings', JSON.stringify(gameSettings));
            showSection('intro');
```

```
            // Atualizar variáveis CSS com as novas cores
                                document.documentElement.style.setProperty('--player1-color',
gameSettings.playerColors[0]);
                                document.documentElement.style.setProperty('--player2-color',
gameSettings.playerColors[1]);
                                document.documentElement.style.setProperty('--player3-color',
gameSettings.playerColors[2]);
                                document.documentElement.style.setProperty('--player4-color',
gameSettings.playerColors[3]);
                                document.documentElement.style.setProperty('--player5-color',
gameSettings.playerColors[4]);
        }

        // Carregar configurações ao iniciar
        loadSettings();

        // ===== MODAL DE HISTÓRICO =====
        const historyModal = document.getElementById('historyModal');
        const closeHistoryModal = document.getElementById('closeHistoryModal');
        const historyContainer = document.getElementById('historyContainer');

        function showHistoryModal() {
            historyModal.style.display = 'flex';
        }

        function closeModal() {
            historyModal.style.display = 'none';
        }

        closeHistoryModal.addEventListener('click', closeModal);
        window.addEventListener('click', (e) => {
            if (e.target === historyModal) {
                closeModal();
            }
        });

        // ===== LÓGICA DO JOGO =====
        const dealBtn = document.getElementById('dealBtn');
        const playCardBtn = document.getElementById('playCardBtn');
        const newRoundBtn = document.getElementById('newRoundBtn');
        const gameResult = document.getElementById('gameResult');
        const winnerName = document.getElementById('winnerName');
        const resultDetails = document.getElementById('resultDetails');
        const roundCount = document.getElementById('round-count');
        const winCount = document.getElementById('win-count');
        const currentTurn = document.getElementById('current-turn');
        const progressBar = document.getElementById('progressBar');
```

```
const playersContainer = document.getElementById('playersContainer');

let players = [];
let activePlayers = [];

const suits = ['hearts', 'diamonds', 'clubs', 'spades'];
const values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A'];
const valueMap = {
    '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, '10': 10,
    'J': 11, 'Q': 12, 'K': 13, 'A': 14
};
const valueNames = {
    '2': 'Dois', '3': 'Três', '4': 'Quatro', '5': 'Cinco',
    '6': 'Seis', '7': 'Sete', '8': 'Oito', '9': 'Nove',
    '10': 'Dez', 'J': 'Valete', 'Q': 'Dama', 'K': 'Rei', 'A': 'Ás'
};
const suitNames = {
    'hearts': 'Copas', 'diamonds': 'Ouros',
    'clubs': 'Paus', 'spades': 'Espadas'
};
const suitSymbols = {
    'hearts': '♥', 'diamonds': '♦',
    'clubs': '♣', 'spades': '♠'
};

let deck = [];
let currentRound = 1;
let gamePhase = 'waiting'; // waiting, dealing, selecting, revealing, results
let currentPlayerIndex = 0;
let gameHistory = [];

// Inicializar jogadores baseado nas configurações
function initializePlayers() {
    players = [];
    activePlayers = [];
    playersContainer.innerHTML = '';

    for (let i = 0; i < gameSettings.playerCount; i++) {
        const playerClass = `player-${i+1}`;
        const colorClass = `player-${i+1}-name`;

        const playerElement = document.createElement('div');
        playerElement.className = `player ${playerClass}`;
        playerElement.id = `player${i+1}`;
        playerElement.innerHTML = `
            <div class="player-header">
                <div class="player-name">
                    <i class="fas fa-user"></i> Jogador ${i+1}
```

```
            </div>
            <div class="player-score" id="player${i+1}-score">0 pts</div>
        </div>
        <div class="player-cards" id="player${i+1}-cards">
            <!-- Cartas serão inseridas aqui via JavaScript -->
        </div>
    `;

    playersContainer.appendChild(playerElement);

    players.push({
        id: `player${i+1}`,
        element: playerElement,
        cardsEl: playerElement.querySelector(`#player${i+1}-cards`),
        scoreEl: playerElement.querySelector(`#player${i+1}-score`),
        name: `Jogador ${i+1}`,
        colorClass: colorClass,
        score: 0,
        cards: [],
        selectedCard: null,
        wins: 0,
        isActive: false,
        isHuman: true // Todos os jogadores são humanos agora
    });

    activePlayers.push(i);
  }
}

function initGame() {
  initializePlayers();
  gamePhase = 'waiting';
  gameResult.style.display = 'none';
  currentPlayerIndex = 0;
  currentRound = 1;
  gameHistory = [];

  players.forEach(player => {
    player.cards = [];
    player.selectedCard = null;
    player.cardsEl.innerHTML = '';
    player.element.classList.remove('player-winner', 'active');
    player.isActive = false;
    player.score = 0;
    player.wins = 0;
    player.scoreEl.textContent = '0 pts';
  });
```

```
            deck = createDeck();
            shuffleDeck(deck);

            roundCount.textContent = currentRound;
            updateWinCount();
            updateProgressBar();

            dealBtn.disabled = false;
            playCardBtn.disabled = true;
            newRoundBtn.disabled = true;
        }

        function updateWinCount() {
            const winText = players.map(p => p.wins).join('/');
            winCount.textContent = `${winText} vitórias`;
        }

        function updateProgressBar() {
            const maxWins = gameSettings.winsNeeded;
            const leadingPlayer = players.reduce((prev, current) =>
                (prev.wins > current.wins) ? prev : current
            );

            if (leadingPlayer.wins > 0) {
                const percentage = (leadingPlayer.wins / maxWins) * 100;
                progressBar.style.width = `${percentage}%`;

                // Definir cor da barra de progresso baseado no jogador líder
                const playerIndex = players.indexOf(leadingPlayer);
                const nextPlayerIndex = (playerIndex + 1) % players.length;

                if (players.length > 1) {
                    const color1 = gameSettings.playerColors[playerIndex];
                    const color2 = gameSettings.playerColors[nextPlayerIndex];
                                progressBar.style.background = `linear-gradient(to right, ${color1},
${color2})`;
                } else {
                    progressBar.style.background = gameSettings.playerColors[playerIndex];
                }
            } else {
                progressBar.style.width = '0%';
            }
        }

        function createDeck() {
            const newDeck = [];
            for (let suit of suits) {
                for (let value of values) {
```

```
                newDeck.push({
                    suit,
                    value,
                    numericValue: valueMap[value],
                    displayValue: value,
                    suitSymbol: suitSymbols[suit],
                    imageUrl: getCardImageUrl(value, suit)
                });
            }
        }

        // Adicionar cartas especiais para modos avançados
        if (gameSettings.difficulty === 'hard') {
            for (let i = 0; i < 4; i++) {
                newDeck.push({
                    suit: 'special',
                    value: 'W',
                    numericValue: 15,
                    displayValue: 'W',
                    suitSymbol: '★',
                    isWildcard: true,
                    imageUrl: 'https://deckofcardsapi.com/static/img/back.png'
                });
            }
        }

        return newDeck;
    }

    function getCardImageUrl(value, suit) {
        // Usando a API Deck of Cards para imagens reais
        const valueMap = {
            '2': '2', '3': '3', '4': '4', '5': '5', '6': '6', '7': '7',
            '8': '8', '9': '9', '10': '10', 'J': 'JACK', 'Q': 'QUEEN',
            'K': 'KING', 'A': 'ACE'
        };

        const suitMap = {
            'hearts': 'HEARTS',
            'diamonds': 'DIAMONDS',
            'clubs': 'CLUBS',
            'spades': 'SPADES'
        };

                                                                    return
`https://deckofcardsapi.com/static/img/${valueMap[value]}${suitMap[suit]}.png`;
    }
```

```
function shuffleDeck(deck) {
    for (let i = deck.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [deck[i], deck[j]] = [deck[j], deck[i]];
    }
    return deck;
}

function dealCards() {
    if (gamePhase !== 'waiting') return;

    gamePhase = 'dealing';
    dealBtn.disabled = true;

    players.forEach(player => {
        player.cards = [];
        player.selectedCard = null;
        player.cardsEl.innerHTML = '';
        player.element.classList.remove('player-winner', 'active');
        player.isActive = false;
    });

    // Distribuir cartas para cada jogador
    players.forEach((player, playerIndex) => {
        for (let i = 0; i < 3; i++) {
            const cardIndex = playerIndex * 3 + i;
            const card = deck[cardIndex];
            player.cards.push(card);

            setTimeout(() => {
                const cardEl = createCardElement(card, false);

                  // Adiciona evento de clique para todos os jogadores (todos são humanos
agora)
                cardEl.addEventListener('click', () => {
                    if (player.isActive && !player.selectedCard) {
                        selectCard(player, card, cardEl);
                    }
                });

                player.cardsEl.appendChild(cardEl);

                setTimeout(() => {
                    cardEl.classList.add('card-dealing');
                }, 50);
            }, i * 300 + playerIndex * 100);
        }
    });
```

```
    setTimeout(() => {
        gamePhase = 'selecting';
        startPlayerTurn();
    }, 1500);
}

function createCardElement(card, isFaceUp = false) {
    const cardEl = document.createElement('div');
    cardEl.className = 'card';

    const cardBack = document.createElement('div');
    cardBack.className = 'card-back';

    const cardFront = document.createElement('div');
    cardFront.className = `card-front ${card.suit}`;

    if (card.imageUrl && card.suit !== 'special') {
        cardFront.style.backgroundImage = `url(${card.imageUrl})`;
        cardFront.style.backgroundSize = 'cover';
        cardFront.style.backgroundPosition = 'center';
        cardFront.innerHTML = `
            <div class="card-value">${card.displayValue}</div>
            <div class="card-suit">${card.suitSymbol}</div>
            <div class="card-value-bottom">${card.displayValue}</div>
        `;
    } else {
        cardFront.innerHTML = `
            <div class="card-value">${card.displayValue}</div>
            <div class="card-suit">${card.suitSymbol}</div>
            <div class="card-value-bottom">${card.displayValue}</div>
        `;
    }

    cardFront.style.display = isFaceUp ? 'flex' : 'none';

    cardEl.appendChild(cardBack);
    cardEl.appendChild(cardFront);
    cardEl.dataset.value = card.numericValue;
    cardEl.dataset.suit = card.suit;
    cardEl.dataset.faceUp = isFaceUp;

    return cardEl;
}

function startPlayerTurn() {
    players.forEach(player => {
        player.element.classList.remove('active');
```

```
        player.isActive = false;
    });

    const currentPlayer = players[currentPlayerIndex];
    currentPlayer.element.classList.add('active');
    currentPlayer.isActive = true;
    currentTurn.textContent = currentPlayer.name;

    // Habilitar o botão de jogar carta apenas para o último jogador
    playCardBtn.disabled = currentPlayerIndex !== players.length - 1;

    // Habilitar seleção de cartas apenas para o jogador atual
    players.forEach(player => {
        const cards = player.cardsEl.querySelectorAll('.card');
        cards.forEach(card => {
            card.style.pointerEvents = player.isActive ? 'auto' : 'none';
        });
    });
}

function selectCard(player, card, cardEl) {
    if (gamePhase !== 'selecting' || !player.isActive || player.selectedCard) return;

    // Revelar a carta selecionada
    const cardFront = cardEl.querySelector('.card-front');
    cardFront.style.display = 'flex';
    cardFront.classList.add('flipping');

    player.selectedCard = card;
    cardEl.classList.add('selected');

    // Desativar outras cartas do jogador
    const otherCards = player.cardsEl.querySelectorAll('.card');
    otherCards.forEach(c => {
        if (c !== cardEl) {
            c.style.pointerEvents = 'none';
        }
    });

    // Avançar para o próximo jogador
    setTimeout(() => {
        currentPlayerIndex = (currentPlayerIndex + 1) % players.length;

        if (currentPlayerIndex === 0 && players.every(p => p.selectedCard)) {
            // Todos jogaram, revelar todas as cartas
            revealAllCards();
        } else {
            // Próximo jogador
```

```
                startPlayerTurn();
            }
        }, 800);
    }

    function revealAllCards() {
        gamePhase = 'revealing';

        // Revelar todas as cartas de todos os jogadores
        players.forEach(player => {
            player.cards.forEach(card => {
                                                                const cardEl =
player.cardsEl.querySelector(`[data-value="${card.numericValue}"][data-suit="${card.suit}"]`);
                if (cardEl) {
                    const cardFront = cardEl.querySelector('.card-front');
                    cardFront.style.display = 'flex';
                    cardFront.classList.add('flipping');
                }
            });
        });

        setTimeout(() => {
            determineRoundWinner();
            newRoundBtn.disabled = false;
        }, 1000);
    }

    function determineRoundWinner() {
        players.forEach(player => {
            if (player.selectedCard) {
                player.currentPlayValue = evaluateCard(player.selectedCard);
            } else {
                player.currentPlayValue = 0;
            }
        });

                    const sortedPlayers = [...players].sort((a, b) => b.currentPlayValue -
a.currentPlayValue);
                                const isTie = sortedPlayers[0].currentPlayValue ===
sortedPlayers[1].currentPlayValue;

        gameResult.style.display = 'block';

        // Registrar no histórico
        const historyEntry = {
            round: currentRound,
            players: players.map(p => ({
                name: p.name,
```

```
                card: p.selectedCard ? getCardName(p.selectedCard) : 'Nenhuma',
                value: p.currentPlayValue
            })),
            winners: []
        };

        if (isTie) {
                        const winners = sortedPlayers.filter(p => p.currentPlayValue ===
sortedPlayers[0].currentPlayValue);
            winnerName.textContent = winners.map(w => w.name).join(' e ');
            winnerName.className = 'winner-name';
            winners.forEach(w => winnerName.classList.add(w.colorClass));

            const cardDesc = getCardName(sortedPlayers[0].selectedCard);
            resultDetails.textContent = `Empate com ${cardDesc}`;

            winners.forEach(winner => {
                winner.score += 5;
                winner.scoreEl.textContent = `${winner.score} pts`;
                historyEntry.winners.push(winner.name);
            });
        } else {
            const winner = sortedPlayers[0];
            winnerName.textContent = winner.name;
            winnerName.className = 'winner-name';
            winnerName.classList.add(winner.colorClass);

            const cardDesc = getCardName(winner.selectedCard);
            resultDetails.textContent = `Com ${cardDesc}`;

            winner.score += 10;
            winner.wins += 1;
            winner.scoreEl.textContent = `${winner.score} pts`;
            winner.element.classList.add('player-winner');

            historyEntry.winners.push(winner.name);

            updateWinCount();
            updateProgressBar();

            // Verificar se alguém ganhou o jogo
            if (winner.wins >= gameSettings.winsNeeded) {
                resultDetails.textContent += ` - ${winner.name} venceu o jogo!`;
                newRoundBtn.textContent = 'Novo Jogo';
                gamePhase = 'game-over';

                // Adicionar animação de confete
                createConfetti(winner.element);
```

```
        }
    }

    // Adicionar ao histórico
    gameHistory.push(historyEntry);
    updateHistoryDisplay();
}

function evaluateCard(card) {
    if (!card) return 0;

    let value = card.numericValue;

    // Cartas especiais têm valores diferentes
    if (card.isWildcard) {
        value = 15; // Valor alto para curingas
    } else if (card.suit === 'hearts' || card.suit === 'diamonds') {
        value += 0.5; // Bonus para cartas vermelhas
    }

    return value;
}

function getCardName(card) {
    if (!card) return 'Nenhuma carta';

    if (card.isWildcard) {
        return 'Curinga';
    }

    return `${valueNames[card.value]} de ${suitNames[card.suit]}`;
}

function updateHistoryDisplay() {
    historyContainer.innerHTML = '';

    gameHistory.forEach((entry, index) => {
        const historyItem = document.createElement('div');
        historyItem.className = 'history-item';

        const roundInfo = document.createElement('div');
        roundInfo.innerHTML = `<strong>Rodada ${entry.round}:</strong>`;

        const playersInfo = document.createElement('div');
        playersInfo.innerHTML = entry.players.map(p =>
            `${p.name}: ${p.card} (${p.value.toFixed(1)})`
        ).join('<br>');
```

```
            const winnerInfo = document.createElement('div');
            winnerInfo.className = 'history-winner';
            winnerInfo.textContent = `Vencedor: ${entry.winners.join(' e ')}`;

            historyItem.appendChild(roundInfo);
            historyItem.appendChild(playersInfo);
            historyItem.appendChild(winnerInfo);

            historyContainer.appendChild(historyItem);
        });

        // Rolagem automática para o final
        historyContainer.scrollTop = historyContainer.scrollHeight;
    }

    function createConfetti(element) {
        const colors = [
            gameSettings.playerColors[0],
            gameSettings.playerColors[1],
            gameSettings.playerColors[2],
            '#FFD700', '#FFFFFF'
        ];

        for (let i = 0; i < 150; i++) {
            const confetti = document.createElement('div');
            confetti.style.position = 'absolute';
            confetti.style.width = '10px';
            confetti.style.height = '10px';
                    confetti.style.backgroundColor = colors[Math.floor(Math.random() *
colors.length)];
            confetti.style.borderRadius = '50%';
            confetti.style.left = `${Math.random() * 100}%`;
            confetti.style.top = '0';
            confetti.style.transform = 'translateY(-100%)';
            confetti.style.zIndex = '1000';
                confetti.style.animation = `confetti-fall ${Math.random() * 3 + 2}s linear
forwards`;

            document.body.appendChild(confetti);

            // Definir posição inicial baseada no elemento vencedor
            const rect = element.getBoundingClientRect();
            const startX = rect.left + rect.width / 2;
            const startY = rect.top;

            confetti.style.left = `${startX + (Math.random() - 0.5) * 200}px`;
            confetti.style.top = `${startY}px`;
```

```
        // Animação de queda
        const keyframes = `
            @keyframes confetti-fall {
                0% { transform: translateY(0) rotate(0deg); opacity: 1; }
                100% { transform: translateY(100vh) rotate(360deg); opacity: 0; }
            }
        `;

        const style = document.createElement('style');
        style.innerHTML = keyframes;
        document.head.appendChild(style);

        // Remover após animação
        setTimeout(() => {
            confetti.remove();
            style.remove();
        }, 5000);
    }
}

// Event listeners
dealBtn.addEventListener('click', dealCards);
playCardBtn.addEventListener('click', () => {
    // Jogar carta selecionada (para o último jogador)
    const currentPlayer = players[currentPlayerIndex];
    if (currentPlayer.selectedCard) {
        revealAllCards();
    }
});
newRoundBtn.addEventListener('click', () => {
    const gameWinner = players.find(p => p.wins >= gameSettings.winsNeeded);
    if (gameWinner) {
        // Resetar para um novo jogo
        initGame();
    } else {
        // Nova rodada no mesmo jogo
        currentRound += 1;
        roundCount.textContent = currentRound;

        gamePhase = 'waiting';
        gameResult.style.display = 'none';
        currentPlayerIndex = 0;

        players.forEach(player => {
            player.cards = [];
            player.selectedCard = null;
            player.cardsEl.innerHTML = '';
            player.element.classList.remove('player-winner', 'active');
```

```
                player.isActive = false;
            });

            deck = createDeck();
            shuffleDeck(deck);

            dealBtn.disabled = false;
            playCardBtn.disabled = true;
            newRoundBtn.disabled = true;
        }
    });

    // Inicializar o jogo
    initGame();
  });
 </script>
</body>
</html>
```