Manual Técnico del Proyecto FortunaSpin

1. Documentación de la Estructura HTML (`index.html`)

Este documento es la **página de aterrizaje (Landing Page)** del sitio web "FortunaSpin - Ruleta Magica". Su objetivo principal es dar la bienvenida a los usuarios, destacar los beneficios del juego y dirigir el tráfico a las páginas de registro e inicio de sesión.

1.1. Estructura y Metadatos

Elemento	Descripción	Notas de Desarrollo
`<!DOCTYPE html>`	Declaración de tipo de documento.	Asegura que el navegador renderice la página en modo estándar (HTML5).
`<html lang="es">`	Raíz del documento.	Define el idioma del contenido como **español** (`es`).
`<meta charset="UTF-8">`	Codificación de caracteres.	Esencial para soportar tildes, la letra "ñ" y otros caracteres especiales.
`<meta name="viewport">`	Configuración para Responsive Design.	`width=device-width, initial-scale=1.0` garantiza que la página se vea correctamente en dispositivos móviles.
`<title>`	Título de la página.	`FortunaSpin - Ruleta Magica`. Se muestra en la pestaña del navegador.
`<link rel="stylesheet">`	Enlace a la hoja de estilos.	Conecta el archivo `css/style.css`, crucial para la apariencia visual.

1.2. Componentes de la Cabecera (`<header>`)

El `<header>` contiene la identidad de la marca y la navegación principal.

Elemento	Clase CSS	Propósito
`<header>`	`header`	Contenedor principal de la cabecera.
`<h1>`	`logo`	Título principal de la marca. El `<span>` se usa probablemente para darle un estilo de color diferente a la palabra "Spin" (separación estilística).
`<nav>`	*Ninguna*	Contiene los enlaces de navegación.
`<a>`	`btn-login`	Botón/enlace de acceso directo para la **página de inicio de sesión** (`login.html`).

1.3. Secciones Principales de Contenido

El `<body>` está dividido en varias secciones temáticas que guían al usuario a través del mensaje de la marca.

A. Sección de Bienvenida (`<section class="Bienveida">`)

- **Función:** Presentación inicial de la aplicación.
- **Título (`<h2>`):** `BIENVENIDO A FORTUNASPIN!` (Clase: `titulo`).
- **Subtítulo (`<p>`):** `El mejor juego de ruleta del momento` (Clase: `sub_titilo`).
- **Llamada a la Acción (CTA):** Enlace principal para la **página de registro** (`registro.html`) (Clase: `btn-principal`).

B. Sección de Beneficios (`<section class="beneficios">`)

- **Función:** Detallar las características clave que diferencian a FortunaSpin.
- **Título (`<h3>`):** `¿Por que elegir FortunaSpin?` (Clase: `titulo-bene`).
- **Lista (`<ul>`):** Lista no ordenada de beneficios (Clase: `lista-beneficios vegas-style`). La clase `vegas-style` sugiere una tematización visual específica (probable color neón o estilo casino).

C. Sección de Soporte Técnico (Estilos en Línea)

- **Función:** Proporcionar información de contacto para ayuda y soporte.
- **Observación:** Esta sección utiliza **estilos en línea** (`style="text-align:center; margin-top:30px;"`), lo cual es una práctica que se recomienda **evitar** en favor de las hojas de estilo externas (`style.css`) para mantener el HTML limpio y la separación de preocupaciones.

1.4. Pie de Página (`<footer>`)

- **Clase CSS:** `footer`
- **Contenido:** Muestra el aviso de derechos de autor y el eslogan del sitio: `FortunaSpin © 2025 - RULETA DE LA SUERTE.`

2. Documentación de Estilos CSS (`style.css`)

El diseño de FortunaSpin está fuertemente influenciado por una **estética de casino/apuestas (Vegas Style)**, utilizando una paleta de colores centrada en el **rojo/naranja brillante (`#ff3d00`)** y el **oro/amarillo (`#ffdd44`)** con un fondo oscuro para crear un efecto de **neón y contraste dramático**.

2.1. Configuración Global y Tipografía

Regla	Propósito	Notas de Desarrollo
`@import url(...)`	**Tipografía Externa.**	Importa dos fuentes: **'Poppins'** (para el texto general) y **'Bebas Neue'** (para títulos y el logo, dándole un estilo de cabecera impactante y condensado).
`* { ... }`	**Reset Global.**	Establece `margin: 0` y `padding: 0` para eliminar los espacios predeterminados del navegador. Usa `box-sizing: border-box`, crucial para un layout predecible, ya que asegura que `padding` y `border` se incluyan dentro del `width` y `height` del elemento.
`body { ... }`	**Fondo y Fuente Base.**	Define una imagen de fondo (`Fondo.png`) que cubre toda la pantalla (`background-size: cover`), y establece **'Poppins'** como fuente principal y el color de texto base como **blanco** (`white`).

2.2. Diseño de la Cabecera (`.header`)

Esta sección implementa un diseño de navegación moderno y fijo visualmente.

Selector	Propiedad Clave	Explicación
`.header`	`display: flex; justify-content: space-between;`	Utiliza **Flexbox** para alinear el logo y el botón de inicio de sesión horizontalmente, separándolos a los extremos.

Selector	Propiedad Clave	Explicación
`.header`	`background: rgba(0, 0, 0, 0.6); backdrop-filter: blur(6px);`	Crea un **efecto semitransparente (vidrio esmerilado)** sobre el fondo, manteniendo la visibilidad del contenido debajo mientras el texto de la cabecera sigue siendo legible.
`.header`	`border-bottom: 3px solid #ff3d00;`	Añade un fuerte borde inferior de color rojo/naranja, reforzando la marca visual de "casino".
`.logo`	`font-family: 'Bebas Neue'; color: #ff3d00; text-shadow: ...`	Aplica la fuente estilizada, el color de acento y sombras de texto múltiples para crear un **efecto de brillo o neón**.
`.logo span`	`color: #ffdd44; text-shadow: ...`	Utiliza un color **amarillo/oro** diferente y su propia sombra para el "Spin", creando un alto contraste y un foco visual en el nombre.
`.btn-login`	`border: 2px solid #ff5500; transition: 0.3s;`	Define un botón con borde y asegura una **transición suave** para el efecto `:hover`.
`.btn-login:hover`	`background: #ff5500; box-shadow: ...`	Efecto visual al pasar el cursor: el botón se rellena con el color de acento y añade un fuerte brillo de sombra.

2.3. Sección de Bienvenida (`.Bienveida`)

Esta es la sección de mayor impacto visual de la página.

Selector	Propiedad Clave	Explicación
`.Bienveida`	`background: linear-gradient(45deg, ...);`	Utiliza un **degradado de tres colores** (rojo, naranja, amarillo) para darle una sensación vibrante y "caliente".

Selector	Propiedad Clave	Explicación
`.Bienveida`	`border-bottom-left-radius: 60px; border-bottom-right-radius: 60px;`	Crea un **diseño moderno y redondeado** en los bordes inferiores para separar visualmente esta sección del resto del contenido.
`.titulo` / `.sub_titilo`	`font-family: 'Bebas Neue'; text-shadow: ...`	Los títulos son impactantes, con la fuente Bebas Neue y sombras de texto para un **efecto neón** similar al logo.
`.btn-principal`	`background: linear-gradient(90deg, ...); border-radius: 35px;`	El botón principal de CTA tiene un **degradado horizontal** y un `border-radius` alto para un aspecto de "píldora", haciéndolo sobresalir.
`.btn-principal:hover`	`transform: scale(1.08);`	Efecto de **agrandamiento sutil** al pasar el cursor, indicando al usuario que este es el principal punto de interacción.

2.4. Estilos de Formularios (Login/Registro)

Los estilos para formularios están diseñados para ser utilizados en páginas secundarias (`login.html`, etc.), manteniendo la estética oscura y brillante.

Selector	Propiedad Clave	Explicación
`.main-login`	`max-width: 450px; background: rgba(0, 0, 0, 0.7);`	Define un **contenedor centrado** y semitransparente con un borde de acento y una sombra fuerte para destacarlo.
`.main-login::before`	`content: ""; background-image: url('LOGO.png');`	Utiliza el pseudoelemento `::before` para insertar dinámicamente una **imagen de logo** sobre el formulario, sin necesidad de modificar el HTML.

Selector	Propiedad Clave	Explicación
`.main-login input`	`background: #1a1a1a; border: 1px solid #ff6a00;`	Los campos de entrada tienen un **fondo muy oscuro** y un borde de color naranja/rojo para que coincidan con la temática.

2.5. Sección de Beneficios (`.beneficios`)

Esta sección presenta los beneficios de forma estructurada con fuerte impacto visual.

Selector	Propiedad Clave	Explicación
`.beneficios`	`background: rgba(255, 60, 0, 0.85); box-shadow: ...`	Usa un **fondo rojizo/naranja semitransparente** y fuerte sombra para encapsular los puntos de venta.
`.beneficios li`	`list-style: none; border-left: 5px solid #ffdd55;`	Elimina las viñetas predeterminadas y utiliza un **borde izquierdo de color dorado** como elemento de diseño llamativo, reforzando el estilo "Lucky Jackpot".
`.beneficios li:hover`	`transform: translateX(10px);`	Al pasar el cursor, el elemento se **desliza 10 píxeles a la derecha**, una animación sutil que llama la atención.

2.6. Pie de Página (`.footer`)

Selector	Propiedad Clave	Explicación
`.footer`	`background: rgba(255, 60, 0, 0.7);`	Mantiene el tema con un fondo rojizo semitransparente, asegurando que el contenido sea legible.

2.7. Adaptabilidad (Responsive Design)

La hoja de estilos incluye consultas de medios para asegurar que el diseño sea adaptable.

Media Query	Dispositivo/Propósito	Cambios Implementados
`@media (max-width: 768px)`	**Dispositivos Móviles (Teléfonos y Tabletas Pequeñas).**	Reduce el tamaño del logo (`.logo`) de 34px a **26px** para no desbordar en pantallas pequeñas. Aumenta el ancho de los elementos de la lista de beneficios (`.beneficios li`) al **95%**.
`@media (min-width: 769px)`	**Dispositivos de Escritorio (Tablet Grande en adelante).**	Utiliza **Flexbox** en la clase `.tipologin` (que se asume está en el HTML del login o registro) para colocar los botones o elementos de ese contenedor uno al lado del otro (`flex-direction: row;`), centrados y con un espacio mínimo (`gap: 10px`), lo que sugiere un diseño de botones múltiples en una fila.

- **-Sección de Login**

MANUAL TÉCNICO – LOGIN DE FORTUNASPIN

1. ¿Qué es este archivo?

Este es el archivo **HTML** de la pantalla de *Iniciar Sesión*.
Aquí se arma toda la estructura: título, botones, formulario y la parte de soporte.

2. Partes del código

◆ 2.1. Encabezado `<head>`

Aquí se define:

- El tipo de documento.
- El idioma (español).
- La codificación UTF-8 pa' que no salgan símbolos raros.
- Que sea adaptable a celulares.
- Se carga el archivo `style.css` donde van más estilos.

```
<title>Login | FortunaSpin</title>
<link rel="stylesheet" href="css/style.css">
```

◆ 2.2. Cuerpo `<body>`

Todo lo que ve el usuario.

A) Contenedor principal

```
<main class="main-login">
```

Aquí dentro va todo el login, y tiene un fondo oscuro con brillo dorado.

B) Título

```
<h2>Iniciar Sesión</h2>
```

C) Botones para elegir tipo de login

```
<section class="tipologin">
    <button id="btnUsuario">Usuario</button>
    <button id="btnAdmin">Administrador</button>
</section>
```

Aquí el usuario elige si entra como **usuario normal** o como **admin**.

D) Formulario de Login

Este formulario está oculto al principio (`display:none;`) y solo aparece cuando el usuario elige un tipo de acceso.

```
<form id="formLogin">
    <label>Usuario:</label>
    <input type="text" id="usuarioInput" required>

    <label>Contraseña:</label>
    <input type="password" id="passInput" required>

    <button type="submit">Ingresar</button>
</form>
```

Incluye:

- Campo para usuario
- Campo para contraseña
- Botón de ingreso
- Enlace para registrarse

E) Soporte Técnico

La parte donde se muestran los datos de ayuda.

```
<section>
    <h3>Soporte Tecnico</h3>
    <p>Email: soporte@fortunaspiin.com</p>
    <p>WhatsApp: +591 67136512</p>
</section>
```

F) Archivo JavaScript

```
<script src="js/login.js"></script>
```

Aquí se cargan las funciones que harán aparecer el login y verificar cosas.
(Ya me lo mandarás después para hacer otro manual).

3. Estilos del Login (CSS)

Estos estilos están al final del archivo.

◆ Fondo del login

```
.main-login{
    background: linear-gradient(145deg,#1a1a1a,#292202);
    border: 2px solid #ff9d00;
    box-shadow: 0 0 300px #ffe100f6;
}
```

- Fondo oscuro con degradado.
- Borde dorado.
- Brillo fuerte alrededor.

◆ Botones principales

```
.btn-principal{
    background:#d6b706;
}
.btn-principal:hover{
    background:#db290a;
}
```

- Amarillo dorado normal.
- Rojo al pasar el cursor.

◆ Botones modo usuario/admin

```
.tipo-login button{
    background:#ee9105;
}
.tipo-login button:hover{
    background:#db3b0a;
}
```

- Naranja fuerte.
- Rojo quemado al pasar el mouse.

Ojo: en el HTML dice `tipologin` pero aquí dice `tipo-login`.
Si quieres, lo puedo corregir para que funcione parejo.

4. Para qué sirve todo esto

Este archivo arma la pantalla donde el usuario:

- Elige si es **usuario** o **admin**,
- Mete su usuario y su contraseña,
- Puede ir a registrarse,
- Tiene los datos de soporte.

La lógica (mostrar formulario, validar, etc.) está en `login.js`.

• Login.js

1. Funcion principal
   Todo el codigo esta dentro de document.addEventListener("DOMContentLoaded").
   Esto sirve para que el script recien se ejecute cuando la pagina ya cargo.
   Asi no hay errores por elementos que aun no existen.

2. Captura de elementos del HTML
   Se guardan las referencias a:

- El formulario: form
- Boton usuario: btnUsuario
- Boton admin: btnAdmin

Con esto podemos manipularlos despues.

3. Mostrar el formulario segun el tipo de login
   Cuando se hace click en btnUsuario:

- Se muestra el formulario (display block).
- Se guarda en form.dataset.tipo = "usuario" para saber que modo se eligio.

Cuando se hace click en btnAdmin:

- Igual se muestra el formulario.
- Se guarda form.dataset.tipo = "admin".

Esto permite que el mismo formulario funcione para dos tipos de acceso.

4. Evento submit del formulario
   Cuando el usuario presiona Ingresar, se captura el evento submit.
   Se hace e.preventDefault() para evitar que la pagina se recargue.

Se obtienen los datos del formulario:

- user = valor del campo usuario
- pass = valor del campo contraseña

5. Proceso de LOGIN para ADMIN
   Primero se revisa si el modo elegido es admin:
   form.dataset.tipo === "admin"

Las credenciales del admin estan fijas en el codigo:
usuario: "admin"
password: "1234"

Si coinciden:

- Se guarda en localStorage que la sesion esta activa como admin.
- Se redirige a admin.html.

Si no coinciden:

- Da un mensaje de error.

6. Proceso de LOGIN para USUARIO NORMAL
   Si el modo no es admin, entonces es usuario.

Los usuarios estan guardados en localStorage en la clave "usuarios".
Se cargan y se convierten a objeto con JSON.parse.

Se busca un usuario que coincida con:
usuario === user y password === pass

Si no existe coincidencia:

- Se muestra mensaje de error.

Si existe:

- Se guarda en localStorage el nombre del usuario que ingreso.
- Se guarda tambien el saldo del usuario.
- Se redirige a ruleta.html.

7. Resumen general del funcionamiento

   -El codigo espera que el usuario elija si es admin o usuario.

   -Segun lo que elija, se muestra el mismo formulario.

   -El admin usa datos fijos.

   -Los usuarios normales se leen desde localStorage.

   -Si las credenciales son correctas, se inicia sesion y se redirige a otra pagina.

- # Registro.html

1. Funcion del archivo
   Este archivo arma la primera parte del registro de un usuario.
   Aqui se piden datos personales basicos y el usuario luego pasa al siguiente paso del registro.

2. Estructura general
   El documento usa HTML basico:

- head: configuracion, titulo y enlace a los estilos
- body: encabezado, formulario, soporte y el script final

3. Encabezado
   En el header se muestra el nombre del sistema:

   <h1 class="logo">FortunaSpin</h1>

Sirve solo como presentacion.

4. Contenedor principal del formulario
   El formulario esta dentro de main con la clase main-login.
   La vista queda parecida al login, ya que usa el mismo estilo general.

   <h2>Registro - Paso 1</h2>

Este titulo indica en que parte del registro esta el usuario.

5. Formulario de registro
   El formulario tiene id="formPaso1".
   Aqui se piden los siguientes datos:

- pais
- nombre
- apellido
- usuario
- correo
- password
- celular

Todos los campos estan marcados como required, lo que significa que el navegador no dejara enviar el formulario si estan vacios.

El boton Siguiente manda la informacion al script registro.js para validar y continuar con el registro.

6. Seccion de soporte
   Al final hay un bloque con informacion de contacto.
   Es el mismo formato que en el login para mantener uniformidad.

7. Script
   Al final del body se carga:

   <script src="js/registro.js"></script>

Este archivo procesara los datos del formulario y guiara al usuario al paso 2 del registro.

8. Funcion final del archivo
   Este archivo muestra la primera pantalla donde el usuario llena sus datos.
   El siguiente paso se maneja desde el script registro.js y desde otra pagina del proceso de registro.

- # Ruleta.html

1. Funcion del archivo
   Este archivo muestra la pantalla principal donde el usuario juega a la ruleta.
   Incluye: menú, saldo, ruleta, panel de apuestas e historial.

2. Encabezado
   En el head se define el título, datos básicos del documento y se carga el archivo de estilos ruleta.css.
   Ese archivo controla el diseño visual de todo.

El título es: Ruleta FortunaSpin.

3. Barra superior (header)
   Esta parte muestra el nombre del sistema y un menú:

- Botón hamburguesa para abrir el menú en pantallas pequeñas.
- Enlaces: Depositar, Retirar y Salir.

Cada botón tiene su propia clase para darle estilo y el script ruleta.js controla lo que pasa al hacer clic.

4. Contenido principal
   El contenido central está dentro de main con clase Contenedor-Ruleta.
   Dentro de ese contenedor hay cuatro partes.

4.1. Panel del usuario
Sección que muestra:

- Nombre del usuario
- Saldo
- Historial de apuestas del propio usuario

Estos elementos se llenan desde ruleta.js usando el localStorage.

4.2. Cuadro del saldo
Un aside que solo muestra el saldo actual.
Se actualiza desde el script cada vez que hay una apuesta o un cambio.

4.3. Ruleta
Incluye:

- Un canvas donde se dibuja la ruleta.
- Espacio para mostrar el número ganador.
- Mensaje de resultado (ganado o perdido).

Todo el funcionamiento lo maneja ruleta.js.

4.4. Panel de apuestas
El usuario ingresa:

- Monto de la apuesta
- Número (0 a 36)
- Color (Rojo o Negro)
- Par o Impar

Solo debe escoger una de esas opciones por apuesta.
Luego presiona el botón Apostar y Girar para iniciar.

En esta misma sección hay un historial de los últimos resultados de la ruleta.

5. Sección de soporte
   Al final se repite el bloque de soporte técnico con correo y número de WhatsApp.
   Se mantiene igual en todas las páginas para uniformidad.

6. Script
   Al final del cuerpo se carga:

   <script src="js/ruleta.js"></script>

Este archivo contiene toda la lógica del juego:

- Dibujo de la ruleta
- Generación del número ganador
- Validación de las apuestas

- Actualización del saldo
- Guardado del historial
- Acción del botón salir
- Menú responsive

Cada punto depende del script.

7. Resumen final
   Este archivo es la interfaz completa del juego de ruleta.
   El HTML muestra la estructura, el CSS se encarga del diseño, y ruleta.js se ocupa de toda la lógica del juego, apuestas, saldo y movimientos.

- # Ruleta.css

1. Objetivo del archivo
   Este CSS define toda la apariencia visual de la página de la ruleta.
   Controla fondo, colores, paneles, menú, ruleta, historial, botones, sombras y el diseño responsivo para celulares.

2. Fondo general del sitio
   Selector: body

- Se elimina margen por defecto.
- Se define una fuente estándar (Arial).
- Se coloca una imagen de fondo fija, centrada y cubriendo toda la pantalla.
- El color del texto por defecto es blanco para generar contraste.

Función: establecer el ambiente visual general de la ruleta.

3. Encabezado (header)
   Selector: .header

- Usa un degradado horizontal en tonos rojos y naranjas.
- Tiene padding interno.
- Se usa flexbox para separar logo y menú.
- Agrega una sombra inferior para dar profundidad.

.logo span:

- Aplica color amarillo y negrilla al texto “Spin”.

Función: dar identidad visual y estructura al encabezado.

4. Menú superior
   Selectores: .menu, .menu-links

- El menú es un contenedor horizontal con separación entre elementos.
- Los enlaces y botones del menú tienen:
  - Padding estándar
  - Bordes redondeados
  - Fuente en negrita
  - Transición suave
  - Borde transparente por defecto
  - Se comportan como botones rectangulares centrados

Cada tipo de botón tiene su propio estilo:

4.1 Botón Depositar (.deposito-btn)

- Degradado naranja.
- Borde amarillo.
- Hover con aumento de tamaño y cambio a rojo.

4.2 Botón Salir (.salir-btn)

- Degradado rojo oscuro.
- Sombras externas e internas simulando brillo.
- Hover más claro y con mayor sombra.

4.3 Botón Retirar (.retirar-btn)

- Degradado rojo a naranja.
- Hover cambia el borde a celeste claro.

Función: menú llamativo y adaptado a acciones importantes (depósito, retiro, salir).

5. Botón menú hamburguesa
   Selector: .hamburger

- Oculto en escritorio.
- En pantallas pequeñas se muestra.
- Fondo transparente, letra blanca, tamaño grande.

Reglas del menú móvil (media query max-width: 768px):

- El menú se vuelve vertical.
- Se posiciona sobre el contenido (absolute).
- Tiene fondo oscuro semitransparente.
- Cada elemento toma todo el ancho.

- Se activa usando la clase .show.

Función: hacer que el menú sea utilizable en pantallas pequeñas.

6. Contenedor principal
   Selector: .Contenedor-Ruleta

- Usa flexbox para alinear ruleta, panel usuario y panel de apuestas.
- Permite salto de línea (wrap).
- Tiene separación amplia entre elementos.

Función: acomodar los elementos principales del juego de forma ordenada.

7. Ruleta
   Selector: .ruleta-area y #ruletaCanvas

- Contenedor cuadrado de 500x500 px.
- El canvas se convierte en círculo con border-radius 50%.
- Se aplican varias sombras (rojo, naranja, amarillo) para dar efecto de luz.

Función: presentar la ruleta como un elemento visual central.

8. Panel del usuario
   Selector: #panelUsuario

- Fondo rojo oscuro semitransparente.
- Sombras externas.
- Texto amarillo y blanco con brillo.
- Diseño vertical.
- Espaciado interno suficiente.

Función: mostrar datos del usuario (nombre, saldo, historial personal).

9. Panel de apuestas
   Selector: .panel-apuesta

- Fondo con degradado rojo–naranja–amarillo.
- Bordes redondeados.
- Sombras luminosas.
- Estructura vertical con espacios entre elementos.

Inputs:

- Ocupan todo el ancho.
- Bordes redondeados, sin borde visible.
- Tamaño de texto legible.

Función: espacio dedicado exclusivamente a ingresar apuestas.

10. Botón principal (Apostar)
    Selector: .btn-principal

- Botón ancho, alto y llamativo.
- Degradado rojo a naranja.
- Sombras fuertes para destacarlo.
- Hover con cambio de colores.

Función: llamar la atención al botón de acción principal del juego.

11. Saldo flotante
    Selector: .saldo-box

- Posición absoluta arriba a la derecha.
- Color amarillo con sombra.
- No depende del diseño principal, siempre visible.

Función: mostrar saldo restante de forma inmediata.

12. Historial de resultados
    Selector: #historialLista

- Lista horizontal flexible, elementos pequeños.
- Cada resultado es una caja con degradado rojo–naranja–amarillo.
- Texto negro con brillo.

Función: mostrar los últimos números que salieron en la ruleta.

13. Historial del usuario
    Selector: #historialUsuario

- Lista vertical dentro del panel del usuario.
- Scroll personalizado.
- Dos estilos según resultado:
    - GANASTE: verde
    - PERDISTE: rojo

Función: almacenar y mostrar la experiencia del usuario con cada apuesta.

14. Footer
    Selector: .footer

- Fondo naranja rojizo semitransparente.
- Texto claro y con sombra.
- Borde superior rojo.

Función: cierre visual del sitio.

15. Diseño responsivo (pantallas menores a 900px)

- El contenedor principal se vuelve vertical.
- La ruleta ajusta tamaño al ancho disponible.
- El panel de apuestas se hace más ancho.
- El diseño se reorganiza para ser legible en celulares.

Función: permitir jugar desde móviles sin perder funcionalidad.

- # Ruleta.js

Inicialización al cargar la página

```
document.addEventListener("DOMContentLoaded", () => {
    const usuario = JSON.parse(localStorage.getItem("sesionActiva")) ||
"Invitado";
    let saldoInicial =
parseFloat(localStorage.getItem("fortunaSpinSaldo"));
    if (isNaN(saldoInicial) || saldoInicial < 0) saldoInicial = 0;
    localStorage.setItem("fortunaSpinSaldo", saldoInicial.toFixed(2));

    document.getElementById("mostrarUsuario").innerHTML = `🎮 Usuario:
<b>${usuario}</b>`;
    document.getElementById("mostrarSaldo").innerHTML = `💰 Saldo Inicial:
<b>Bs ${saldoInicial.toFixed(2)}</b>`;

    actualizarUI();
    dibujarRuleta();
    actualizarHistorialUsuario();
});
```

- **`DOMContentLoaded`**: Se asegura que el código se ejecute solo cuando el DOM esté completamente cargado.

- **`localStorage.getItem("sesionActiva")`**: Recupera el usuario activo o asigna `"Invitado"`.
- **`saldoInicial`**: Recupera el saldo guardado en localStorage, verifica si es un número válido, si no se inicializa en 0.
- **`mostrarUsuario` y `mostrarSaldo`**: Actualiza los elementos del HTML con los datos del usuario y saldo.
- **`actualizarUI()`, `dibujarRuleta()`, `actualizarHistorialUsuario()`**: Inicializa la interfaz visual, la ruleta y el historial de apuestas.

2. Variables globales

```
const canvas = document.getElementById("ruletaCanvas");
const ctx = canvas?.getContext("2d");

const btnApostar = document.getElementById("btnApostar");
const montoInput = document.getElementById("montoInput");
const numeroInput = document.getElementById("numeroInput");
const colorInput = document.getElementById("colorInput");
const parImparInput = document.getElementById("parImparInput");
const saldoText = document.getElementById("saldoText");
const historialLista = document.getElementById("historialLista");
const resultadoMensaje = document.getElementById("resultadoMensaje");
const historialUsuarioList = document.getElementById("historialUsuario");

const centro = 250;
const radio = 250;
const totalSectores = 37;
const anguloSector = (2 * Math.PI) / totalSectores;

let anguloActual = 0;
let velocidad = 0;
let ballAngle = 0;
let ballVelocity = 0;

let girando = false;
let frenando = false;
let targetRotation = 0;
let numeroGanadorFinal = null;

let saldo = parseFloat(localStorage.getItem("fortunaSpinSaldo")) || 0;
let nombreUsuarioActivo = JSON.parse(localStorage.getItem("sesionActiva"))
|| "Invitado";

let apuestaActiva = null;
let animationFrameId = null;
let spinTimeoutId = null;
let giroMaxTimeoutId = null;

let historialUsuario =
JSON.parse(localStorage.getItem(`historial_${nombreUsuarioActivo}`)) || [];
```

- **Variables de canvas**: `canvas` y `ctx` para dibujar la ruleta.
- **Inputs y botones**: elementos HTML de la apuesta.
- **Propiedades de la ruleta**:
    - `centro`, `radio`: centro y radio de la ruleta.

    - `totalSectores`, `anguloSector`: número de sectores (0-36) y ángulo de cada sector.
- **Estados del giro**:
    - `anguloActual`, `velocidad`: ángulo de la ruleta y velocidad angular.
    - `ballAngle`, `ballVelocity`: ángulo y velocidad de la bola.
    - `girando`, `frenando`: estado de la animación.
    - `targetRotation`: rotación final calculada para detener la ruleta.
- **Saldo y usuario**: variables para manejar el dinero y usuario activo.
- **Historial**: se recupera el historial desde `localStorage`.

3. Actualizar historial del usuario

```
function actualizarHistorialUsuario() {
    if (!historialUsuarioList) return;
    historialUsuarioList.innerHTML = "";
    historialUsuario.slice(-10).reverse().forEach(h => {
        const li = document.createElement("li");
        li.textContent = `${h.tipo.toUpperCase()} (${h.valor}) - Bs
${h.monto.toFixed(2)} - ${h.resultado}`;
        li.style.background = h.resultado === "GANASTE" ? "#05b887" :
"#f30317";
        li.style.color = "#fff";
        li.style.padding = "3px 6px";
        li.style.margin = "2px 0";
        li.style.borderRadius = "4px";
        historialUsuarioList.appendChild(li);
    });
}
```

- Limpia y vuelve a renderizar los últimos 10 registros de apuestas del usuario.
- Diferencia colores según si ganó o perdió.

4. Información de los números

```
const numerosInfo = [
  { num: 0, color: "G" }, { num: 32, color: "R" }, ...
];
```

- Define todos los números de la ruleta, indicando el color:
    - `"G"` = verde (0)
    - `"R"` = rojo
    - `"B"` = negro

5. Dibujar la ruleta

```
function dibujarRuleta() {
    if (!ctx) return;
```

```
    ctx.clearRect(0, 0, 500, 500);
    ctx.save();
    ctx.translate(centro, centro);
    ctx.rotate(anguloActual);

    for (let i = 0; i < totalSectores; i++) {
        const info = numerosInfo[i];
        const color = info.color === "G" ? "green" : info.color === "R" ?
"#e63946" : "black";

        ctx.beginPath();
        ctx.fillStyle = color;
        const start = i * anguloSector;
        ctx.moveTo(0, 0);
        ctx.arc(0, 0, radio * 0.95, start, start + anguloSector);
        ctx.fill();

        // Número ganador resaltado
        if (numeroGanadorFinal !== null && info.num === numeroGanadorFinal
&& !girando) {
            ctx.strokeStyle = "#ffd700";
            ctx.lineWidth = 5;
            ctx.beginPath();
            ctx.arc(0, 0, radio * 0.95, start, start + anguloSector);
            ctx.stroke();
        }

        ctx.save();
        ctx.rotate(start + anguloSector / 2);
        ctx.fillStyle = "white";
        ctx.font = "bold 18px Arial";
        ctx.fillText(info.num, radio * 0.7, 5);
        ctx.restore();
    }
    ctx.restore();

    // Centro de la ruleta
    ctx.beginPath();
    ctx.fillStyle = "#1e1e1e";
    ctx.arc(centro, centro, radio * 0.15, 0, 2 * Math.PI);
    ctx.fill();

    // Dibujar bola
    const ballR = Math.max(6, radio * 0.03);
    const pos = radio * 0.90;
    const ballX = centro + Math.cos(ballAngle) * pos;
    const ballY = centro + Math.sin(ballAngle) * pos;
    ctx.beginPath();
    ctx.fillStyle = "#ffc107";
    ctx.arc(ballX, ballY, ballR, 0, 2 * Math.PI);
    ctx.fill();
    ctx.strokeStyle = "rgba(0,0,0,0.5)";
    ctx.stroke();
}
```

- Dibuja cada sector según el color y número.
- Marca el número ganador si la ruleta ya terminó.
- Dibuja la bola girando alrededor del borde de la ruleta.

6. Obtener número ganador

```
async function obtenerNumeroGanador() {
    try {
        const resp = await
fetch("https://fortunaapi.online/generar_numero.php");
        const j = await resp.json();
        if (j && typeof j.numero === "number") return j.numero;
    } catch (e) {}
    return Math.floor(Math.random() * 37);
}
```

- Intenta obtener el número de la API.
- Si falla, genera un número aleatorio entre 0 y 36.

7. Iniciar el giro

```
async function iniciarGiro() {
    if (girando || !apuestaActiva || apuestaActiva.monto > saldo) return;

    saldo -= apuestaActiva.monto;
    actualizarUI();

    girando = true;
    frenando = false;
    btnApostar.disabled = true;
    resultadoMensaje.textContent = "Girando...";
    resultadoMensaje.style.background = "red";

    velocidad = 0.4;
    ballVelocity = -0.8;
    ballAngle = Math.random() * Math.PI * 2;
    anguloActual = Math.random() * Math.PI * 2;

    animar();

    numeroGanadorFinal = await obtenerNumeroGanador();

    spinTimeoutId = setTimeout(() => { frenando = true; calcularTarget();
}, 8000 + Math.random() * 4000);
    giroMaxTimeoutId = setTimeout(() => { if (girando && !frenando) {
frenando = true; calcularTarget(); } }, 15000);
}
```

- Verifica que no haya giro activo.
- Resta el monto de la apuesta al saldo.
- Inicializa variables de animación.
- Llama a `animar()` para iniciar la animación de la ruleta.
- Calcula el giro máximo y cuándo frenar.

8. Animación de giro

```
function animar() {
    if (!girando) return;

    if (!frenando) velocidad *= 0.998;
    else {
        const diff = targetRotation - anguloActual;
        if (Math.abs(diff) > 0.001) { velocidad += diff * 0.001; velocidad
*= 0.985; }
        else { velocidad = 0; anguloActual = targetRotation; }
    }

    anguloActual += velocidad;
    ballVelocity *= 0.985;
    ballAngle += ballVelocity;

    dibujarRuleta();
    animationFrameId = requestAnimationFrame(animar);

    if (frenando && Math.abs(velocidad) < 0.0005 && Math.abs(ballVelocity)
< 0.001) finalizarGiro();
}
```

- Actualiza la posición de la ruleta y la bola.
- Si está frenando, ajusta suavemente la velocidad hasta detenerse.
- Llama recursivamente a sí misma con `requestAnimationFrame`.

9. Finalizar giro

```
function finalizarGiro() {
    cancelAnimationFrame(animationFrameId);
    clearTimeout(spinTimeoutId);
    clearTimeout(giroMaxTimeoutId);

    girando = false;
    frenando = false;
    btnApostar.disabled = false;

    dibujarRuleta();
    mostrarResultado();
    actualizarHistorial(numeroGanadorFinal);

    apuestaActiva = null;
}
```

- Cancela la animación y los timeouts.
- Muestra resultado y actualiza el historial.

10. Procesar apuesta

```
function procesarApuesta() {
    if (girando) return;

    const monto = parseFloat(montoInput.value);
```

```
    if (isNaN(monto) || monto <= 0) { alert("Monto inválido"); return; }

    let tipo = null, valor = null, count = 0;
    const num = numeroInput.value.trim();
    const col = colorInput.value.trim().toLowerCase();
    const parImpar = parImparInput.value.trim().toLowerCase();

    if (num !== "") { const n = parseInt(num); if (n>=0 &&
n<=36){tipo="numero";valor=n;count++;} }
    if (col==="rojo"||col==="negro"){tipo="color";valor=col;count++; }
    if
(parImpar==="par"||parImpar==="impar"){tipo="parimpar";valor=parImpar;count
++; }

    if(count!==1){alert("Ingrese solo una apuesta."); return;}
    if(monto>saldo){alert("No tienes suficiente saldo."); return;}

    apuestaActiva = {monto, tipo, valor};
    iniciarGiro();
}
```

- Valida el monto y tipo de apuesta.
- Solo permite una opción activa por apuesta.
- Llama a `iniciarGiro()`.

11. Cerrar sesión

```
function cerrarSesion() {
    if (!confirm("¿Seguro que deseas salir de la página?")) return;
    localStorage.removeItem("sesionActiva");
    localStorage.removeItem("fortunaSpinSaldo");
    window.location.href = "index.html";
}
```

- Elimina datos de usuario y redirige al inicio.

12. Menú hamburguesa

```
hamburger.addEventListener("click", () => {
    menuLinks.classList.toggle("show");
});
```

- Alterna la visibilidad del menú en móvil.

13. Botón salir

```
document.getElementById("salirBtn").addEventListener("click", () => {
    localStorage.removeItem("sesionActiva");
    window.location.href = "index.html";
```

```
});
```

- Cierra la sesión instantáneamente.

14. Evento apostar

```
btnApostar?.addEventListener("click", procesarApuesta);
```

- Vincula el botón de apostar con la función `procesarApuesta`.