```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Herramienta de Tampón de Clonar</title>
  <style>
    /* Tema oscuro profesional */
    body {
      background-color: #1e1e1e;
      color: #d4d4d4;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      display: flex;
      margin: 0;
      height: 100vh;
      overflow: hidden;
    }

    /* Barra lateral izquierda */
    #sidebar {
      width: 320px;
      background-color: #252526;
      border-right: 1px solid #3e3e42;
      padding: 20px;
      box-sizing: border-box;
      display: flex;
```

```
    flex-direction: column;
    gap: 20px;
    overflow-y: auto;
}

h2 {
    margin: 0 0 10px 0;
    font-size: 1.2rem;
    font-weight: 600;
    color: #ffffff;
}

/* Controles y botones */
.control-group {
    display: flex;
    flex-direction: column;
    gap: 8px;
}

label {
    font-size: 0.9rem;
    display: flex;
    justify-content: space-between;
}

input[type="range"] {
```

```
    width: 100%;
    cursor: pointer;
}

button, .file-upload-btn {
    background-color: #0e639c;
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 4px;
    cursor: pointer;
    font-size: 0.9rem;
    text-align: center;
    transition: background-color 0.2s;
}

button:hover, .file-upload-btn:hover {
    background-color: #1177bb;
}

button:active {
    background-color: #094771;
}

input[type="file"] {
    display: none;
```

```
}

/* Área de trabajo y Canvas */
#workspace {
    flex: 1;
    background-color: #121212;
    display: flex;
    justify-content: center;
    align-items: center;
    overflow: auto;
    position: relative;
}

canvas {
    background-color: transparent;
    box-shadow: 0 0 20px rgba(0,0,0,0.5);
    cursor: crosshair;
}

/* Instrucciones */
.instructions {
    background-color: #2d2d30;
    padding: 15px;
    border-radius: 6px;
    font-size: 0.85rem;
    line-height: 1.5;
```

```
        border-left: 4px solid #0e639c;
      }

      .instructions code {
        background: #1e1e1e;
        padding: 2px 5px;
        border-radius: 3px;
        color: #4fc1ff;
      }
    </style>
</head>
<body>

    <div id="sidebar">
      <h2>Herramientas de Edición</h2>

      <div class="control-group">
        <label for="imageLoader" class="file-upload-btn">Subir Imagen</label>
        <input type="file" id="imageLoader" accept="image/*">
      </div>

      <div class="control-group">
        <label>Tamaño del Pincel: <span id="sizeVal">20px</span></label>
        <input type="range" id="brushSize" min="1" max="100" value="20">
      </div>
```

```
    <div class="control-group">
      <label>Opacidad: <span id="opacityVal">100%</span></label>
      <input type="range" id="brushOpacity" min="1" max="100" value="100">
    </div>

    <div class="control-group">
      <label>Dureza: <span id="hardnessVal">50%</span></label>
      <input type="range" id="brushHardness" min="0" max="100" value="50">
    </div>

    <div class="control-group" style="margin-top: 10px;">
      <button id="undoBtn">Deshacer Última Acción</button>
      <button id="downloadBtn">Descargar Imagen</button>
    </div>

    <div class="instructions">
      <strong>Cómo usar el Tampón de Clonar:</strong><br><br>
      1. Sube una imagen.<br>
      2. Mantén presionado <code>Alt</code> + <code>Clic izquierdo</code> en un área limpia para establecer el punto de origen (ancla).<br>
      3. Haz <code>Clic normal</code> y arrastra sobre el área que deseas cubrir para clonar los píxeles.
    </div>
  </div>

  <div id="workspace">
    <canvas id="canvas"></canvas>
```

```
</div>

<script>
  const canvas = document.getElementById('canvas');
  const ctx = canvas.getContext('2d', { willReadFrequently: true });

  // Elementos de la interfaz
  const imageLoader = document.getElementById('imageLoader');
  const brushSize = document.getElementById('brushSize');
  const brushOpacity = document.getElementById('brushOpacity');
  const brushHardness = document.getElementById('brushHardness');
  const undoBtn = document.getElementById('undoBtn');
  const downloadBtn = document.getElementById('downloadBtn');

  // Valores de visualización
  const sizeVal = document.getElementById('sizeVal');
  const opacityVal = document.getElementById('opacityVal');
  const hardnessVal = document.getElementById('hardnessVal');

  // Variables de estado
  let img = new Image();
  let history = [];
  let isDrawing = false;
  let isSourceSet = false;
  let sourceX = 0;
  let sourceY = 0;
```

```
    let currentX = 0;
    let currentY = 0;
    let offsetX = 0;
    let offsetY = 0;

    // Actualizar etiquetas de los controles deslizantes
    brushSize.addEventListener('input', (e) => sizeVal.textContent = `${e.target.value}px`);
    brushOpacity.addEventListener('input', (e) => opacityVal.textContent = `${e.target.value}%`);
    brushHardness.addEventListener('input', (e) => hardnessVal.textContent = `${e.target.value}%`);

    // Cargar imagen
    imageLoader.addEventListener('change', handleImage);

    function handleImage(e) {
      const reader = new FileReader();
      reader.onload = function(event) {
        img.onload = function() {
          canvas.width = img.width;
          canvas.height = img.height;
          ctx.drawImage(img, 0, 0);
          saveState(); // Guardar estado inicial
        }
        img.src = event.target.result;
      }
```

```
    if(e.target.files[0]) {
        reader.readAsDataURL(e.target.files[0]);
    }
}

// Gestión de estado para Deshacer
function saveState() {
    if (history.length > 20) history.shift(); // Limitar historial a 20 pasos
    history.push(canvas.toDataURL());
}

undoBtn.addEventListener('click', () => {
    if (history.length > 1) {
        history.pop(); // Eliminar estado actual
        const imgData = new Image();
        imgData.onload = () => {
            ctx.clearRect(0, 0, canvas.width, canvas.height);
            ctx.drawImage(imgData, 0, 0);
        };
        imgData.src = history[history.length - 1];
    }
});

// Lógica del Tampón de Clonar
canvas.addEventListener('mousedown', (e) => {
    const rect = canvas.getBoundingClientRect();
```

```
const x = (e.clientX - rect.left) * (canvas.width / rect.width);
const y = (e.clientY - rect.top) * (canvas.height / rect.height);

if (e.altKey) {
  // Establecer punto de origen
  sourceX = x;
  sourceY = y;
  isSourceSet = true;

  // Feedback visual del ancla
  ctx.beginPath();
  ctx.arc(x, y, 3, 0, Math.PI * 2);
  ctx.fillStyle = 'red';
  ctx.fill();
  setTimeout(() => {
    // Restaurar píxeles borrando el ancla roja (usando el historial)
    const imgData = new Image();
    imgData.onload = () => { ctx.drawImage(imgData, 0, 0); };
    imgData.src = history[history.length - 1];
  }, 200);

} else if (isSourceSet) {
  // Iniciar clonación
  isDrawing = true;
  offsetX = x - sourceX;
  offsetY = y - sourceY;
```

```
        saveState();
        cloneStamp(x, y);
    }
});

canvas.addEventListener('mousemove', (e) => {
    if (!isDrawing) return;
    const rect = canvas.getBoundingClientRect();
    const x = (e.clientX - rect.left) * (canvas.width / rect.width);
    const y = (e.clientY - rect.top) * (canvas.height / rect.height);
    cloneStamp(x, y);
});

canvas.addEventListener('mouseup', () => {
    isDrawing = false;
});

canvas.addEventListener('mouseleave', () => {
    isDrawing = false;
});

// Función principal de clonación
function cloneStamp(targetX, targetY) {
    const size = parseInt(brushSize.value);
    const opacity = parseInt(brushOpacity.value) / 100;
    const hardness = parseInt(brushHardness.value) / 100;
```

```
        // Calcular de dónde estamos leyendo basándonos en el desplazamiento inicial
        const srcX = targetX - offsetX;
        const srcY = targetY - offsetY;

        // Crear un canvas temporal para extraer el patrón de origen con suavizado
        const tempCanvas = document.createElement('canvas');
        const tempCtx = tempCanvas.getContext('2d');
        tempCanvas.width = canvas.width;
        tempCanvas.height = canvas.height;

        // Dibujar la imagen actual en el canvas temporal
        tempCtx.drawImage(canvas, 0, 0);

        // Preparar el contexto principal para la clonación
        ctx.save();
        ctx.globalAlpha = opacity;

        // Crear un degradado radial para la dureza/suavidad del pincel
        const radGrad = ctx.createRadialGradient(targetX, targetY, size * hardness,
targetX, targetY, size);
        radGrad.addColorStop(0, 'rgba(0, 0, 0, 1)');
        radGrad.addColorStop(1, 'rgba(0, 0, 0, 0)');

        // Usar composición para crear el borde suave
        ctx.globalCompositeOperation = 'source-over';
```

```
    ctx.beginPath();
    ctx.arc(targetX, targetY, size, 0, Math.PI * 2);
    ctx.closePath();

    // Recortar al círculo del pincel
    ctx.clip();

    // Dibujar desde la posición fuente relativa al objetivo actual
    ctx.drawImage(tempCanvas, targetX - srcX, targetY - srcY);

    // Si la dureza no es del 100%, aplicar el difuminado de borde
    if (hardness < 1) {
      ctx.globalCompositeOperation = 'destination-in';
      ctx.fillStyle = radGrad;
      ctx.fillRect(targetX - size, targetY - size, size * 2, size * 2);
    }

    ctx.restore();
}

// Descargar la imagen
downloadBtn.addEventListener('click', () => {
    if (history.length === 0) return;
    const link = document.createElement('a');
    link.download = 'imagen_editada.png';
```

```
        link.href = canvas.toDataURL('image/png');
        link.click();
      });
    </script>
</body>
</html>
```