```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
  <meta name="apple-mobile-web-app-capable" content="yes">
  <title>TCM Assist - Demo Funcional</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <style>
    * { font-family: -apple-system, BlinkMacSystemFont, sans-serif; -webkit-tap-highlight-color: transparent; }
    body { background: #0a0a0a; color: #fff; overflow: hidden; margin: 0; height: 100vh; }
    .glass { background: rgba(30,30,30,0.95); backdrop-filter: blur(20px); border: 1px solid rgba(255,255,255,0.1); }
    .agent { transition: all 0.3s; border-radius: 12px; padding: 12px; background: rgba(255,255,255,0.05); border: 1px solid rgba(255,255,255,0.1); }
    .agent.active { background: rgba(59,130,246,0.2); border-color: rgba(59,130,246,0.5); box-shadow: 0 0 20px rgba(59,130,246,0.3); }
    .agent.alert { background: rgba(239,68,68,0.2); border-color: rgba(239,68,68,0.5); animation: pulse 2s infinite; }
    @keyframes pulse { 0%,100% { box-shadow: 0 0 0 0 rgba(239,68,68,0.4); } 50% { box-shadow: 0 0 0 15px rgba(239,68,68,0); } }
    .live-word { display: inline-block; animation: popIn 0.3s; margin-right: 4px; }
    @keyframes popIn { from { opacity: 0; transform: translateY(10px) scale(0.8); } to { opacity: 1; transform: translateY(0) scale(1); } }
    .highlight-symptom { background: rgba(59,130,246,0.4); padding: 2px 6px; border-radius: 4px; font-weight: 600; color: #60a5fa; }
    .highlight-vague { background: rgba(245,158,11,0.4); padding: 2px 6px; border-radius: 4px; font-weight: 600; color: #fbbf24; }
    .btn { border-radius: 12px; padding: 12px 20px; font-weight: 600; border: none; cursor: pointer; transition: all 0.2s; }
    .btn:active { transform: scale(0.95); }
    .btn-blue { background: linear-gradient(135deg, #3b82f6, #7c3aed); color: white; }
    .btn-green { background: linear-gradient(135deg, #10b981, #059669); color: white; }
    .btn-red { background: linear-gradient(135deg, #ef4444, #dc2626); color: white; }
    .btn-amber { background: linear-gradient(135deg, #f59e0b, #d97706); color: white; }
    .typing::after { content: '|'; animation: blink 1s infinite; }
    @keyframes blink { 0%,100% { opacity: 1; } 50% { opacity: 0; } }
    .scroll-hide::-webkit-scrollbar { display: none; }
    .alert-box { animation: slideIn 0.4s; border-left: 3px solid; }
```

```
      @keyframes slideIn { from { transform: translateX(100px); opacity: 0; } to { transform: translateX(0); opacity: 1; } }
    </style>
</head>
<body class="flex flex-col h-screen">

    <!-- HEADER -->
    <div class="glass h-14 flex items-center justify-between px-4 border-b border-white/10">
        <div class="flex items-center gap-3">
            <div class="w-8 h-8 bg-gradient-to-br from-blue-500 to-purple-600 rounded-lg flex items-center justify-center text-lg">☯️</div>
            <div>
                <h1 class="font-bold text-sm">TCM Assist</h1>
                <p class="text-xs text-gray-400">Transcripción en vivo - DEMO</p>
            </div>
        </div>
        <div class="flex items-center gap-2">
            <span id="statusIndicator" class="px-2 py-1 rounded-full bg-gray-500/20 text-gray-400 text-xs">● Listo</span>
        </div>
    </div>

    <!-- MAIN -->
    <div class="flex-1 flex overflow-hidden">

        <!-- LEFT: CONTROLES DE SIMULACIÓN -->
        <div class="w-2/5 flex flex-col border-r border-white/10 bg-black/40">

            <!-- Simulador de Voz -->
            <div class="p-4 border-b border-white/10">
                <h3 class="text-xs font-semibold text-gray-400 uppercase mb-3">🎙️ Simular Entrada de Voz</h3>

                <div class="space-y-2">
                    <div class="flex gap-2">
                        <button onclick="simulateSpeech('doctor', 'Buenos días, ¿qué síntomas tiene?')" class="flex-1 btn btn-blue text-xs py-3">
                            👨‍⚕️ Dr. Saludo
                        </button>
                        <button onclick="simulateSpeech('patient', 'Me duele la cabeza un poco')" class="flex-1 btn btn-green text-xs py-3">
```

```
                    👴 Paciente 1
                  </button>
                </div>
                <div class="flex gap-2">
                  <button onclick="simulateSpeech('doctor', '¿Dónde exactamente y desde
cuándo?')" class="flex-1 btn btn-blue text-xs py-3">
                    👨‍⚕️ Dr. Profundiza
                  </button>
                  <button onclick="simulateSpeech('patient', 'En la frente, desde ayer y me
mareo')" class="flex-1 btn btn-green text-xs py-3">
                    👴 Paciente 2
                  </button>
                </div>
                <div class="flex gap-2">
                  <button onclick="simulateSpeech('doctor', '¿Náuseas? ¿Cómo duerme?')"
class="flex-1 btn btn-blue text-xs py-3">
                    👨‍⚕️ Dr. Evalúa
                  </button>
                  <button onclick="simulateSpeech('patient', 'Sí náuseas, duermo más o menos
regular')" class="flex-1 btn btn-green text-xs py-3">
                    👴 Paciente 3
                  </button>
                </div>
              </div>

              <div class="mt-4 pt-4 border-t border-white/10">
                <p class="text-xs text-gray-500 mb-2">O escribe manualmente:</p>
                <div class="flex gap-2">
                  <select id="speakerSelect" class="bg-black/30 border border-white/10 rounded-
lg px-3 py-2 text-xs outline-none">
                    <option value="patient">👴 Paciente</option>
                    <option value="doctor">👨‍⚕️ Dr.</option>
                  </select>
                  <input type="text" id="manualInput" placeholder="Escribe y presiona ↵"
                      class="flex-1 bg-black/30 border border-white/10 rounded-lg px-3 py-2 text-
xs outline-none"
                      onkeypress="if(event.key==='Enter')sendManual()">
                  <button onclick="sendManual()" class="btn btn-amber px-4 text-xs">➤</button>
                </div>
              </div>
            </div>
```

```
        <!-- Palabras Clave Rápidas -->
        <div class="p-4 flex-1 overflow-y-auto scroll-hide">
          <h3 class="text-xs font-semibold text-gray-400 uppercase mb-3">⚡ Insertar Síntomas</h3>
          <div class="flex flex-wrap gap-2">
            <button onclick="addWord('dolor de cabeza')" class="px-3 py-2 rounded-lg bg-blue-500/20 text-blue-400 text-xs border border-blue-500/30">dolor cabeza</button>
            <button onclick="addWord('mareo')" class="px-3 py-2 rounded-lg bg-blue-500/20 text-blue-400 text-xs border border-blue-500/30">mareo</button>
            <button onclick="addWord('náuseas')" class="px-3 py-2 rounded-lg bg-blue-500/20 text-blue-400 text-xs border border-blue-500/30">náuseas</button>
            <button onclick="addWord('vómito')" class="px-3 py-2 rounded-lg bg-blue-500/20 text-blue-400 text-xs border border-blue-500/30">vómito</button>
            <button onclick="addWord('insomnio')" class="px-3 py-2 rounded-lg bg-blue-500/20 text-blue-400 text-xs border border-blue-500/30">insomnio</button>
            <button onclick="addWord('fatiga')" class="px-3 py-2 rounded-lg bg-blue-500/20 text-blue-400 text-xs border border-blue-500/30">fatiga</button>
            <button onclick="addWord('un poco')" class="px-3 py-2 rounded-lg bg-amber-500/20 text-amber-400 text-xs border border-amber-500/30">un poco ⚠️</button>
            <button onclick="addWord('más o menos')" class="px-3 py-2 rounded-lg bg-amber-500/20 text-amber-400 text-xs border border-amber-500/30">más o menos ⚠️</button>
            <button onclick="addWord('regular')" class="px-3 py-2 rounded-lg bg-amber-500/20 text-amber-400 text-xs border border-amber-500/30">regular ⚠️</button>
          </div>

          <div class="mt-6 p-3 rounded-lg bg-white/5 border border-white/10">
            <p class="text-xs text-gray-400 mb-2">💡 En la versión con API:</p>
            <ul class="text-xs text-gray-500 space-y-1 list-disc list-inside">
              <li>El micrófono real transcribe automáticamente</li>
              <li>Whisper envía audio cada 2 segundos</li>
              <li>Los agentes analizan sin tocar botones</li>
            </ul>
          </div>
        </div>
      </div>

      <!-- RIGHT: TRANSCRIPCIÓN Y AGENTES -->
      <div class="w-3/5 flex flex-col">
```

```
        <!-- ÁREA DE TRANSCRIPCIÓN EN VIVO -->
        <div class="h-2/5 glass border-b border-white/10 p-4 flex flex-col">
          <div class="flex justify-between items-center mb-3">
            <h3 class="text-xs font-semibold text-gray-400 uppercase">📝 Transcripción en Vivo</h3>
            <button onclick="clearTranscript()" class="text-xs text-gray-500 hover:text-white">Limpiar</button>
          </div>

          <!-- Texto que se construye palabra por palabra -->
          <div id="liveArea" class="flex-1 bg-black/40 rounded-xl p-4 overflow-y-auto scroll-hide">
            <div id="currentLine" class="text-lg leading-relaxed min-h-[3rem] text-gray-400 italic text-sm">
              El texto aparecerá aquí palabra por palabra...
            </div>
            <div id="completedLines" class="mt-4 space-y-2"></div>
          </div>

          <!-- Indicador de hablante -->
          <div class="flex gap-2 mt-3">
            <div id="indicatorDoctor" class="flex-1 py-2 rounded-lg bg-blue-500/10 text-blue-400 text-xs text-center border border-blue-500/20 opacity-30">
              👨‍⚕️ Escuchando al Dr...
            </div>
            <div id="indicatorPatient" class="flex-1 py-2 rounded-lg bg-green-500/10 text-green-400 text-xs text-center border border-green-500/20 opacity-30">
              👨 Escuchando al Paciente...
            </div>
          </div>
        </div>

        <!-- AGENTES ANALIZANDO EN VIVO -->
        <div class="h-1/3 glass border-b border-white/10 p-3">
          <div class="flex justify-between items-center mb-2">
            <h3 class="text-xs font-semibold text-gray-400 uppercase">🤖 Agentes Analizando</h3>
            <span id="agentStatus" class="text-xs text-gray-500">Inactivos</span>
          </div>

          <div class="grid grid-cols-2 gap-2 h-full pb-6">
```

```
        <!-- Agente 1 -->
        <div id="agent1" class="agent">
          <div class="flex items-center gap-2 mb-1">
            <span class="text-blue-400">🔍</span>
            <span class="text-xs font-medium">Keywords TCM</span>
          </div>
          <p id="agent1text" class="text-xs text-gray-500">Esperando...</p>
          <div id="agent1tags" class="flex flex-wrap gap-1 mt-1"></div>
        </div>

        <!-- Agente 2 -->
        <div id="agent2" class="agent">
          <div class="flex items-center gap-2 mb-1">
            <span class="text-amber-400">⚠️</span>
            <span class="text-xs font-medium">Profundización</span>
          </div>
          <p id="agent2text" class="text-xs text-gray-500">Esperando...</p>
          <p id="agent2suggestion" class="text-xs text-amber-400 mt-1 hidden"></p>
        </div>

        <!-- Agente 3 -->
        <div id="agent3" class="agent">
          <div class="flex items-center gap-2 mb-1">
            <span class="text-purple-400">☯️</span>
            <span class="text-xs font-medium">Patrones TCM</span>
          </div>
          <p id="agent3text" class="text-xs text-gray-500">Base lista</p>
          <div id="agent3patterns" class="flex flex-wrap gap-1 mt-1"></div>
        </div>

        <!-- Agente 4 -->
        <div id="agent4" class="agent">
          <div class="flex items-center gap-2 mb-1">
            <span class="text-green-400">✓</span>
            <span class="text-xs font-medium">Completitud</span>
          </div>
          <p id="agent4text" class="text-xs text-gray-500">Faltan: sueño, digestión...</p>
          <div id="agent4checks" class="flex flex-wrap gap-1 mt-1"></div>
        </div>
      </div>
    </div>
```

```
        <!-- ALERTAS -->
        <div id="alertsArea" class="flex-1 overflow-y-auto p-3 scroll-hide bg-black/20">
          <div class="text-center text-gray-600 mt-8 text-sm">
            <div class="text-2xl mb-2">📊</div>
            <p>Las alertas aparecerán aquí</p>
            <p class="text-xs mt-1 text-gray-500">Los agentes reaccionan en tiempo real</p>
          </div>
        </div>
      </div>
    </div>

    <script>
      // Estado
      let currentSpeaker = 'patient';
      let currentText = '';
      let wordQueue = [];
      let isTyping = false;
      let detectedSymptoms = new Set();
      let detectedPatterns = new Set();
      let missingFields = ['sueño', 'digestión', 'heces', 'lengua', 'pulso'];
      let hasVagueTerm = false;

      // Base de datos TCM
      const symptomsDB = {
        'dolor de cabeza': { pattern: 'Viento-Calor', points: 'GB20, LI4' },
        'dolor cabeza': { pattern: 'Viento-Calor', points: 'GB20, LI4' },
        'mareo': { pattern: 'Subida Yang', points: 'LV3, GB20' },
        'mareos': { pattern: 'Subida Yang', points: 'LV3, GB20' },
        'náuseas': { pattern: 'Rebeldía Qi', points: 'E36, P6' },
        'nauseas': { pattern: 'Rebeldía Qi', points: 'E36, P6' },
        'vómito': { pattern: 'Estómago rebelde', points: 'E44, R12' },
        'vomito': { pattern: 'Estómago rebelde', points: 'E44, R12' },
        'insomnio': { pattern: 'Shen inestable', points: 'H7, C6' },
        'fatiga': { pattern: 'Qi Xu', points: 'E36, B6' },
        'palpitaciones': { pattern: 'Shen Xu', points: 'H7, C7' }
      };

      const vagueTerms = ['un poco', 'mas o menos', 'regular', 'normal', 'aveces', 'tal vez', 'no se',
'asi asi'];
      const questions = {
```

```
            'dolor': '¿Dónde exactamente? ¿Irradia a algún lado?',
            'mareo': '¿Al levantarse? ¿Con movimientos rápidos?',
            'insomnio': '¿Dificultad para dormir o para mantener el sueño?',
            'fatiga': '¿Mejora con reposo? ¿Desde cuándo?'
        };

        // SIMULAR ENTRADA DE VOZ (palabra por palabra)
        function simulateSpeech(speaker, text) {
            currentSpeaker = speaker;
            wordQueue = text.split(' ');
            currentText = '';

            // Indicador visual
            document.getElementById('indicatorDoctor').style.opacity = speaker === 'doctor' ? '1' :
'0.3';
            document.getElementById('indicatorPatient').style.opacity = speaker === 'patient' ? '1' :
'0.3';
            document.getElementById('statusIndicator').textContent = speaker === 'doctor' ? '● Dr.
hablando' : '● Paciente hablando';
            document.getElementById('statusIndicator').className = speaker === 'doctor'
                ? 'px-2 py-1 rounded-full bg-blue-500/20 text-blue-400 text-xs'
                : 'px-2 py-1 rounded-full bg-green-500/20 text-green-400 text-xs';

            // Iniciar "transcripción" palabra por palabra
            if(!isTyping) {
                isTyping = true;
                typeNextWord();
            }
        }

        function typeNextWord() {
            if(wordQueue.length === 0) {
                // Terminó esta línea
                finalizeLine();
                isTyping = false;
                return;
            }

            const word = wordQueue.shift();
            currentText += (currentText ? ' ' : '') + word;
```

```
    // Renderizar con análisis en vivo
    renderLiveText();

    // Analizar esta palabra inmediatamente
    analyzeWord(word);

    // Continuar con siguiente palabra (velocidad natural)
    setTimeout(typeNextWord, 200 + Math.random() * 300);
}

function addWord(word) {
    // Agregar palabra individual
    if(!isTyping) {
        currentText = currentText ? currentText + ' ' + word : word;
        renderLiveText();
        analyzeWord(word);
    }
}

function sendManual() {
    const input = document.getElementById('manualInput');
    const text = input.value.trim();
    const speaker = document.getElementById('speakerSelect').value;
    if(text) {
        simulateSpeech(speaker, text);
        input.value = '';
    }
}

// RENDERIZAR TEXTO EN VIVO CON RESALTADO
function renderLiveText() {
    const container = document.getElementById('currentLine');

    // Dividir en palabras y resaltar
    const words = currentText.split(' ');
    const html = words.map((word, i) => {
        const lower = word.toLowerCase();
        let className = 'live-word';

        // Verificar si es síntoma
        const isSymptom = Object.keys(symptomsDB).some(s => lower.includes(s) ||
```

```
currentText.toLowerCase().includes(s));
            if(isSymptom && Object.keys(symptomsDB).some(s =>
currentText.toLowerCase().includes(s))) {
                className += ' highlight-symptom';
            }

            // Verificar si es término vago
            if(vagueTerms.some(v => lower.includes(v) || currentText.toLowerCase().includes(v))) {
                className += ' highlight-vague';
            }

            return `<span class="${className}">${word}</span>`;
        }).join('');

        container.innerHTML = html + '<span class="typing"></span>';
        container.className = 'text-lg leading-relaxed min-h-[3rem] text-white';

        // Auto-scroll
        container.scrollIntoView({ behavior: 'smooth', block: 'end' });
    }

    // ANÁLISIS EN VIVO DE CADA PALABRA
    function analyzeWord(word) {
        const lower = word.toLowerCase();
        const fullText = currentText.toLowerCase();

        // Agente 1: Keywords (inmediato)
        Object.keys(symptomsDB).forEach(symptom => {
            if(fullText.includes(symptom) && !detectedSymptoms.has(symptom)) {
                detectedSymptoms.add(symptom);
                activateAgent('agent1');
                document.getElementById('agent1text').textContent = `Detectado: ${symptom}`;
                addTag('agent1tags', symptom, 'blue');
                updateAgentStatus();

                // Alerta inmediata
                showAlert('keyword', '🔍 Síntoma Detectado', `"${symptom}" - $
{symptomsDB[symptom].pattern}`, 'blue');
            }
        });
```

```
        // Agente 2: Profundización (inmediato)
        vagueTerms.forEach(term => {
            if(fullText.includes(term) && !hasVagueTerm) {
                hasVagueTerm = true;
                activateAgent('agent2', 'alert');
                document.getElementById('agent2text').textContent = `Vago: "${term}"`;

                // Buscar sugerencia
                let suggestion = '¿Podría especificar más detalles?';
                Object.keys(questions).forEach(q => {
                    if(fullText.includes(q)) suggestion = questions[q];
                });

                document.getElementById('agent2suggestion').textContent = `💡 ${suggestion}`;
                document.getElementById('agent2suggestion').classList.remove('hidden');
                updateAgentStatus();

                showAlert('depth', '⚠️ Respuesta Vaga', `"${term}" detectado. ${suggestion}`,
'amber');
            }
        });

        // Agente 3: Patrones (cuando hay suficiente contexto)
        if(detectedSymptoms.size >= 2) {
            checkComplexPatterns(fullText);
        }

        // Agente 4: Completitud
        const fields = ['sueño', 'dormir', 'digestión', 'comer', 'heces', 'evacuación', 'lengua',
'pulso'];
        fields.forEach(field => {
            if(fullText.includes(field)) {
                const idx = missingFields.indexOf(field);
                if(idx > -1) {
                    missingFields.splice(idx, 1);
                    updateCompleteness();
                }
            }
        });
    }
```

```
    function checkComplexPatterns(text) {
        // Detectar combinaciones
        if(text.includes('mareo') && text.includes('nauseas') && !
detectedPatterns.has('Shaoyang')) {
            detectedPatterns.add('Shaoyang');
            activateAgent('agent3');
            document.getElementById('agent3text').textContent = 'Patrón complejo detectado';
            addTag('agent3patterns', 'Shaoyang', 'purple');
            updateAgentStatus();

            showAlert('pattern', '☯️ Patrón Shaoyang',
                'Síndrome de medio: mareo + náuseas<br>Tratamiento: Xiao Chai Hu
Tang<br>Puntos: GB20, TB5, LV3', 'purple');
        }

        if(text.includes('dolor de cabeza') && text.includes('mareo') && !
detectedPatterns.has('Yang Ascendente')) {
            detectedPatterns.add('Yang Ascendente');
            addTag('agent3patterns', 'Yang Asc.', 'purple');
            showAlert('pattern', '☯️ Yang Ascendente',
                'Subida de Yang al cabeza<br>Tian Ma Gou Teng Yin<br>Puntos: LV3, GB20, LI4',
'purple');
        }
    }

    function updateCompleteness() {
        activateAgent('agent4');
        const remaining = missingFields.slice(0, 2).join(', ');
        document.getElementById('agent4text').textContent = missingFields.length === 0
            ? '✓ Historia completa'
            : `Faltan: ${remaining}...`;

        // Mostrar checks
        const checks = document.getElementById('agent4checks');
        checks.innerHTML = '';
        ['sueño', 'digestión', 'heces', 'lengua', 'pulso'].forEach(field => {
            if(!missingFields.includes(field)) {
                addTag('agent4checks', '✓' + field, 'green');
            }
        });
```

```
        updateAgentStatus();
    }

    // FINALIZAR LÍNEA
    function finalizeLine() {
        // Mover a completados
        const completed = document.getElementById('completedLines');
        const div = document.createElement('div');
        div.className = 'p-3 rounded-lg text-sm ' + (currentSpeaker === 'doctor' ? 'bg-blue-500/10 border-l-2 border-blue-500' : 'bg-green-500/10 border-l-2 border-green-500');
        div.innerHTML = `
            <div class="flex justify-between text-xs opacity-60 mb-1">
                <span>${currentSpeaker === 'doctor' ? '👨‍⚕️ Dr.' : '👴 Paciente'}</span>
                <span>${new Date().toLocaleTimeString()}</span>
            </div>
            <p>${currentText}</p>
        `;
        completed.prepend(div);

        // Limpiar actual
        document.getElementById('currentLine').innerHTML = '<span class="text-gray-500 italic text-sm">Esperando siguiente...</span>';
        currentText = '';
        hasVagueTerm = false;

        // Resetear agentes de turno
        document.getElementById('agent2suggestion').classList.add('hidden');
        deactivateAgent('agent2');
    }

    // UTILIDADES
    function activateAgent(id, type = 'active') {
        const el = document.getElementById(id);
        el.classList.remove('active', 'alert');
        el.classList.add(type);
    }

    function deactivateAgent(id) {
        document.getElementById(id).classList.remove('active', 'alert');
    }
```

```
    function addTag(containerId, text, color) {
      const container = document.getElementById(containerId);
      const tag = document.createElement('span');
      tag.className = `px-2 py-0.5 rounded-full bg-${color}-500/20 text-${color}-400 text-xs
animate-pulse`;
      tag.textContent = text;
      container.appendChild(tag);
      setTimeout(() => tag.classList.remove('animate-pulse'), 1000);
    }

    function updateAgentStatus() {
      const active = document.querySelectorAll('.agent.active, .agent.alert').length;
      document.getElementById('agentStatus').textContent = active === 0 ? 'Inactivos' : `$
{active} analizando`;
    }

    function showAlert(type, title, message, color) {
      const colors = {
        blue: 'border-blue-500 bg-blue-500/10 text-blue-400',
        amber: 'border-amber-500 bg-amber-500/10 text-amber-400',
        purple: 'border-purple-500 bg-purple-500/10 text-purple-400',
        green: 'border-green-500 bg-green-500/10 text-green-400',
        red: 'border-red-500 bg-red-500/10 text-red-400'
      };

      const area = document.getElementById('alertsArea');
      if(area.children[0]?.classList.contains('text-center')) area.innerHTML = '';

      const alert = document.createElement('div');
      alert.className = `alert-box p-3 rounded-lg mb-2 ${colors[color]}`;
      alert.innerHTML = `
        <div class="flex justify-between items-start">
          <div>
            <div class="font-bold text-sm mb-0.5">${title}</div>
            <div class="text-xs opacity-90 leading-snug">${message}</div>
          </div>
          <button onclick="this.parentElement.parentElement.remove()" class="text-xs
opacity-50 ml-2">X</button>
        </div>
      `;
      area.prepend(alert);
```

```
            if(navigator.vibrate) navigator.vibrate(50);
        }

        function clearTranscript() {
            document.getElementById('currentLine').innerHTML = '<span class="text-gray-500 italic
text-sm">Esperando...</span>';
            document.getElementById('completedLines').innerHTML = '';
            document.getElementById('alertsArea').innerHTML = '<div class="text-center text-
gray-600 mt-8 text-sm"><div class="text-2xl mb-2">📊</div><p>Las alertas aparecerán aquí</
p></div>';

            // Resetear estado
            currentText = '';
            detectedSymptoms.clear();
            detectedPatterns.clear();
            missingFields = ['sueño', 'digestión', 'heces', 'lengua', 'pulso'];
            hasVagueTerm = false;

            ['agent1', 'agent2', 'agent3', 'agent4'].forEach(id => {
                document.getElementById(id).classList.remove('active', 'alert');
                document.getElementById(id + 'tags') && (document.getElementById(id +
'tags').innerHTML = '');
                document.getElementById(id + 'patterns') && (document.getElementById(id +
'patterns').innerHTML = '');
                document.getElementById(id + 'checks') && (document.getElementById(id +
'checks').innerHTML = '');
            });

            document.getElementById('agent1text').textContent = 'Esperando...';
            document.getElementById('agent2text').textContent = 'Esperando...';
            document.getElementById('agent3text').textContent = 'Base lista';
            document.getElementById('agent4text').textContent = 'Faltan: sueño, digestión...';
            document.getElementById('agent2suggestion').classList.add('hidden');
        }
    </script>
</body>
</html>
```