Instrucciones de implementación externa:

- ***Agregar la referencia a google icon font en la sección <head> del sitio web:***

* Copie y pegue el siguiente código dentro de la sección <head> de su página para incluir los estilos necesarios del widget:

1 . <!--Import Google Icon Font-->

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

- ***Agregar el contenedor del widget en el <body> del sitio web:***

* Incluya el siguiente contenedor HTML en la sección <body> de la página, en el lugar donde desea que se muestre el widget:

2. <!-- Contenedor Widget -->

- **Añadir el script JavaScript para la funcionalidad del widget:**

***** Incluya el siguiente script al final de la sección <body> del sitio web, antes del cierre de la etiqueta </body>. Esto asegurará que el script necesario para el widget se cargue y funcione correctamente; configurar la URL que se les comparta reemplazando todo lo que diga APP_URL en el script que se les comparte en el punto 3:

3. Scripts

```
<!-- Import js y css -->
 <script>
  // ========================================================================
  // INTERCEPTOR DE FETCH PARA CAPTURAR idChatWeb
  // ========================================================================

  // Variable global para almacenar el idChatWeb capturado
  window.capturedIdChatWeb = null;

  // Guardar la función fetch original
  const originalFetch = window.fetch;

  // Sobrescribir fetch para interceptar peticiones
  window.fetch = function(...args) {
   const [url, options] = args;

   // Interceptar la petición de crear chat
   if (url && url.includes('/widget/chat/crear')) {

    // Intentar extraer el idChatWeb del body
    if (options && options.body) {
     try {
```

```
      const body = JSON.parse(options.body);
      if (body.idChatWeb) {
        window.capturedIdChatWeb = body.idChatWeb;

        // Notificar que tenemos el ID
        window.dispatchEvent(new CustomEvent('idChatWebCapturado', {
          detail: { idChatWeb: window.capturedIdChatWeb }
        }));
      }
    } catch (e) {
      console.error("❌ Error al parsear body de crear chat:", e);
    }
  }
 }

 // Ejecutar el fetch original y retornar la promesa
 return originalFetch.apply(this, args);
};

// ========================================================================
// CARGA DEL WIDGET
// ========================================================================

// Definición de constantes para las URLs
const URL_BASE = "APP_URL";
const URL_CSS = `${URL_BASE}/chatWeb.css`;
const URL_JS = `${URL_BASE}/chatWeb.js`;

// Función para cargar con timeout
```

```
function cargarConTimeout(cargarFn, src, timeout = 1500) {
  return new Promise((resolve, reject) => {
    let el;
    const timer = setTimeout(() => {
      if (el && el.parentNode) el.parentNode.removeChild(el);
      reject(new Error("Timeout al cargar: " + src));
    }, timeout);

    cargarFn(src)
      .then(() => {
        clearTimeout(timer);
        resolve();
      })
      .catch((err) => {
        clearTimeout(timer);
        reject(err);
      });

    // Guardar el elemento para poder removerlo si hay timeout
    if (cargarFn === cargarCSS) {
      el = document.querySelector(`link[href='${src}']`);
    }
    if (cargarFn === cargarJS) {
      el = document.querySelector(`script[src='${src}']`);
    }
  });
}

// Funcion para cargar el css
```

```
function cargarCSS(src) {
  return new Promise((resolve, reject) => {
    const link = document.createElement("link");
    link.rel = "stylesheet";
    link.href = src;
    link.onload = resolve;
    link.onerror = reject;
    document.head.appendChild(link);
    // Devuelve el elemento para poder removerlo si hay timeout
    resolve.el = link;
  });
}

// Funcion para cargar el js
function cargarJS(src) {
  return new Promise((resolve, reject) => {
    const script = document.createElement("script");
    script.src = src;
    script.onload = resolve;
    script.onerror = reject;
    document.head.appendChild(script);
  });
}

// Carga con timeout de 3 segundos
cargarConTimeout(cargarCSS, URL_CSS, 1500)
  .then(() => cargarConTimeout(cargarJS, URL_JS, 1500))
  .then(() => {
    if (
```

```
    window.WidgetChat &&
    typeof window.WidgetChat.init === "function" &&
    document.getElementById("contenedorWidget")
  ) {
    window.WidgetChat.init();
    console.log("✅ Widget cargado exitosamente");

    // Configurar sistema de comunicación con iframe
    setTimeout(configurarComunicacionIframe, 1000);
  } else {
    console.error("❌ El widget o el contenedor no están disponibles.");
  }
})
.catch((error) => {
  console.error(
    "❌ Error al cargar recursos del widget, por favor contactenos:",
    error
  );

  // Configuración de reintentos
  const intervaloReintento = 15000; // 15 segundos
  const tiempoMaximo = 300000; // 5 minutos
  const tiempoInicio = Date.now();

  // Funcion para intentar cargar nuevamente los recursos
  function intentarCarga() {
    const tiempoTranscurrido = Date.now() - tiempoInicio;

    if (tiempoTranscurrido >= tiempoMaximo) {
```

```
  console.error("❌ Tiempo máximo de reintentos alcanzado");
  return;
}

console.log("🔄 Reintentando cargar el widget...");

cargarConTimeout(cargarCSS, URL_CSS, 1500)
  .then(() => cargarConTimeout(cargarJS, URL_JS, 1500))
  .then(() => {
    if (
      window.WidgetChat &&
      typeof window.WidgetChat.init === "function" &&
      document.getElementById("contenedorWidget")
    ) {
      window.WidgetChat.init();
      console.log("✅ Widget cargado exitosamente");

      // Configurar sistema de comunicación con iframe
      setTimeout(configurarComunicacionIframe, 1000);
    } else {
      throw new Error(
        "El widget o el contenedor no están disponibles"
      );
    }
  })
  .catch((error) => {
    console.error("❌ Error en reintento:", error);
    setTimeout(intentarCarga, intervaloReintento);
  });
```

```
    }

    // Iniciar el primer reintento
    setTimeout(intentarCarga, intervaloReintento);
  });

// ========================================================================
// SISTEMA DE COMUNICACIÓN CON IFRAME
// ========================================================================

// Sistema para rastrear y limpiar timeouts pendientes
const timeoutsPendientes = new Set();
let idChatWebActual = null;

// Función para limpiar todos los timeouts pendientes
function limpiarTimeouts() {
  timeoutsPendientes.forEach(timeoutId => {
    clearTimeout(timeoutId);
  });
  timeoutsPendientes.clear();
}

// Función para agregar timeout con tracking
function agregarTimeout(callback, delay) {
  const timeoutId = setTimeout(() => {
    timeoutsPendientes.delete(timeoutId);
    callback();
  }, delay);
  timeoutsPendientes.add(timeoutId);
```

```
  return timeoutId;
}

// Función principal para configurar la comunicación con el iframe
function configurarComunicacionIframe() {

  // Escuchar el evento de captura de idChatWeb
  window.addEventListener('idChatWebCapturado', function(event) {

    // Limpiar todos los timeouts anteriores para evitar mensajes con IDs antiguos
    limpiarTimeouts();

    // Actualizar el ID actual
    idChatWebActual = event.detail.idChatWeb;

    // Actualizar la variable global
    window.capturedIdChatWeb = event.detail.idChatWeb;

    // Esperar un momento para que el iframe se cree
    agregarTimeout(() => {
      const iframe = document.getElementById('iframeChatWeb');
      if (iframe) {
        // Verificar que el ID no haya cambiado antes de enviar
        if (idChatWebActual === event.detail.idChatWeb) {
          enviarMensajeAlIframe(iframe, event.detail.idChatWeb);
        } else {
        }
      } else {
        // Observar para cuando se cree
```

```
      observarCreacionIframe(event.detail.idChatWeb);
    }
  }, 500);
});

// Escuchar mensajes del iframe para detectar cierre de chat
window.addEventListener('message', function(event) {
  // Permitir mensajes desde el origen correcto o desde file:// (para desarrollo local)
  if (event.origin === 'APP_URL' ||
      event.origin === 'null' ||
      event.origin === 'file://' ||
      event.origin.startsWith('file://')) {
    // Detectar cuando se cierra el chat desde chatWeb.js
    if (event.data && event.data.type === 'chatCerrado') {

      // Limpiar todos los timeouts pendientes
      limpiarTimeouts();

      // Limpiar el ID actual
      idChatWebActual = null;

      // Limpiar la variable global de idChatWeb capturado
      window.capturedIdChatWeb = null;

    }
  }
}, false);

// También observar cambios en el DOM para detectar el iframe
```

```
    observarCreacionIframe();


  }


  // Función para observar la creación del iframe
  function observarCreacionIframe(idChatWebConocido = null) {
    const observer = new MutationObserver(function(mutations) {
      mutations.forEach(function(mutation) {
        mutation.addedNodes.forEach(function(node) {
          // Verificar si el nodo es el iframe o lo contiene
          if (node.id === 'iframeChatWeb' || (node.querySelector &&
node.querySelector('#iframeChatWeb'))) {
            const iframe = document.getElementById('iframeChatWeb');
            if (iframe) {

              // Usar el ID capturado o intentar obtenerlo
              const idChat = idChatWebConocido || window.capturedIdChatWeb || '';

              if (idChat) {
                enviarMensajeAlIframe(iframe, idChat);
              } else {
                enviarMensajeAlIframe(iframe, '');
              }

              // Dejar de observar
              observer.disconnect();
            }
          }
        });
```

```
  });
 });

 // Observar el contenedor del widget
 const contenedor = document.getElementById('contenedorWidget');
 if (contenedor) {
  observer.observe(contenedor, {
   childList: true,
   subtree: true
  });
 }

 // Verificar si el iframe ya existe
 const iframeExistente = document.getElementById('iframeChatWeb');
 if (iframeExistente) {
  const idChat = idChatWebConocido || idChatWebActual || window.capturedIdChatWeb || '';
  if (idChat) {
   // Actualizar el ID actual si se está usando un ID conocido
   if (idChatWebConocido) {
    idChatWebActual = idChatWebConocido;
   }
   enviarMensajeAlIframe(iframeExistente, idChat);
  }
  observer.disconnect();
 }
}

 // Función para enviar mensaje al iframe
 function enviarMensajeAlIframe(iframe, idChatWeb) {
```

```
// Validar que el idChatWeb no esté vacío antes de enviar
if (!idChatWeb || idChatWeb === '') {
  return;
}

// Verificar que el ID coincida con el actual (evitar enviar IDs antiguos)
if (idChatWebActual && idChatWeb !== idChatWebActual) {
  return;
}

// Función para enviar el mensaje
const enviar = function() {
  // Verificar nuevamente que el ID sigue siendo válido antes de enviar
  if (idChatWebActual && idChatWeb !== idChatWebActual) {
    return;
  }

  if (iframe.contentWindow) {
    try {
      // Verificar que el iframe esté cargado y no sea about:blank
      if (iframe.src === 'about:blank' || !iframe.src.includes('widget/chat/web')) {
        return;
      }

      const mensaje = {
        chatWeb: 'Crear',
        idWidgetChatWeb: idChatWeb
      };
```

```
  // Determinar el targetOrigin para postMessage
  // Usar el origen del iframe si está disponible, o '*' como fallback
  let targetOrigin = 'APP_URL';
  try {
    // Intentar obtener el origin del iframe si es posible
    const iframeOrigin = iframe.contentWindow.location.origin;
    if (iframeOrigin && iframeOrigin !== 'null' && iframeOrigin !== 'about:blank') {
      targetOrigin = iframeOrigin;
    } else {
      // Si el origin es null, usar '*' para permitir cualquier origen
      targetOrigin = '*';
    }
  } catch (e) {
    // Si hay error de same-origin, usar '*' como fallback
    targetOrigin = '*';
  }

  iframe.contentWindow.postMessage(
    mensaje,
    targetOrigin
  );

} catch (e) {
  console.error("❌ Error al enviar mensaje al iframe:", e);
  // Si falla con origen específico, intentar con '*'
  try {
    iframe.contentWindow.postMessage(
      { chatWeb: 'Crear', idWidgetChatWeb: idChatWeb },
```

```
            '*'
          );
        } catch (e2) {
          console.error("❌ Error al enviar mensaje incluso con '*':", e2);
        }
      }
    } else {
      console.error("❌ contentWindow no disponible");
    }
  };

  // Enviar cuando el iframe cargue
  const onLoadHandler = function() {
    agregarTimeout(enviar, 300);
  };

  // Remover listener anterior si existe para evitar duplicados
  iframe.removeEventListener('load', onLoadHandler);
  iframe.addEventListener('load', onLoadHandler);

  // También intentar enviar en múltiples momentos solo si el iframe está cargado
  // Usar agregarTimeout para poder rastrear y limpiar estos timeouts
  agregarTimeout(() => {
    if (iframe.src && iframe.src.includes('widget/chat/web') && idChatWeb ===
idChatWebActual) {
      enviar();
    }
  }, 500);
```

```
      agregarTimeout(() => {
        if (iframe.src && iframe.src.includes('widget/chat/web') && idChatWeb === idChatWebActual) {
          enviar();
        }
      }, 1000);

      agregarTimeout(() => {
        if (iframe.src && iframe.src.includes('widget/chat/web') && idChatWeb === idChatWebActual) {
          enviar();
        }
      }, 2000);
    }

  // Exponer funciones globalmente para debugging
  window.configurarComunicacionIframe = configurarComunicacionIframe;
  window.enviarMensajeAlIframe = enviarMensajeAlIframe;
 </script>

 <!-- Script de debugging -->
 <script>
  // Detectar errores globales
  window.addEventListener('error', function(e) {
    if (e.target !== window) {
      console.error(" Error al cargar recurso:", e.target.src || e.target.href);
    }
  }, true);
 </script>
```

- **Consideraciones Adicionales:**

* Asegúrese de que los archivos referenciados sean accesibles desde la red donde está alojada la página oficial en la cual se desea integrar el widget.

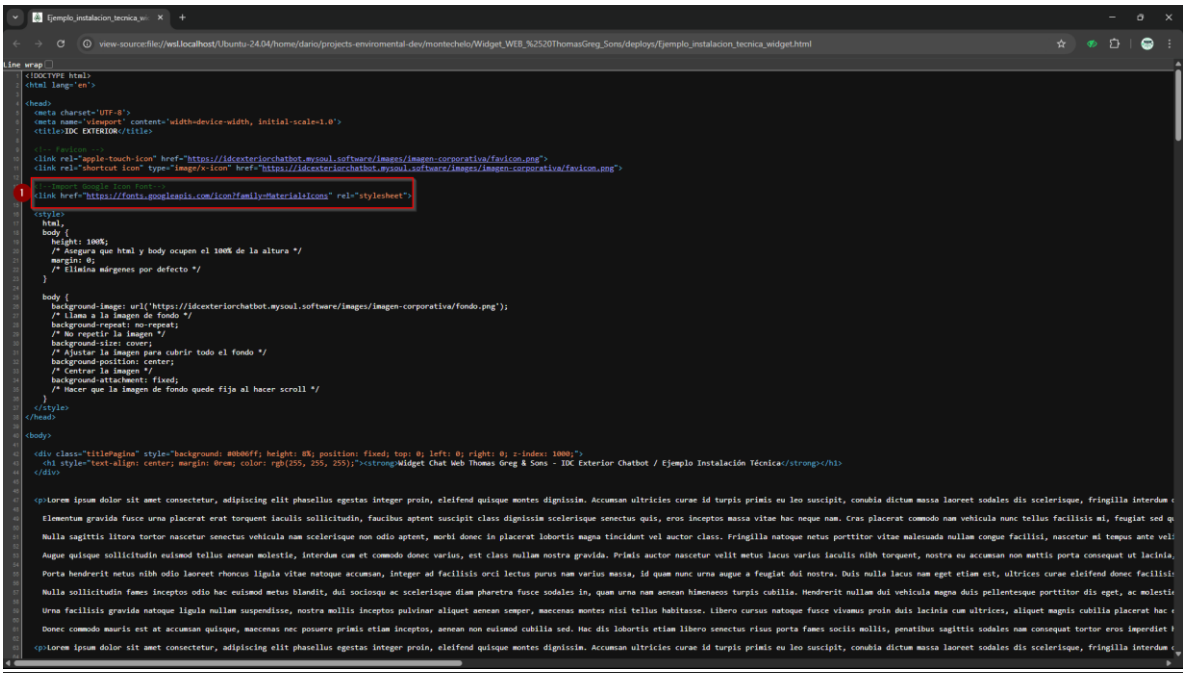
* Verifique que no existan conflictos con otros estilos o scripts en el sitio web que puedan afectar el diseño o el funcionamiento del widget.

* Configuración política de CORS y CSP por lo que es necesario garantizar configuraciones tanto de nuestra parte los orígenes como de la parte cliente en cuanto a restricciones para garantizar la comunicación de los servicios.

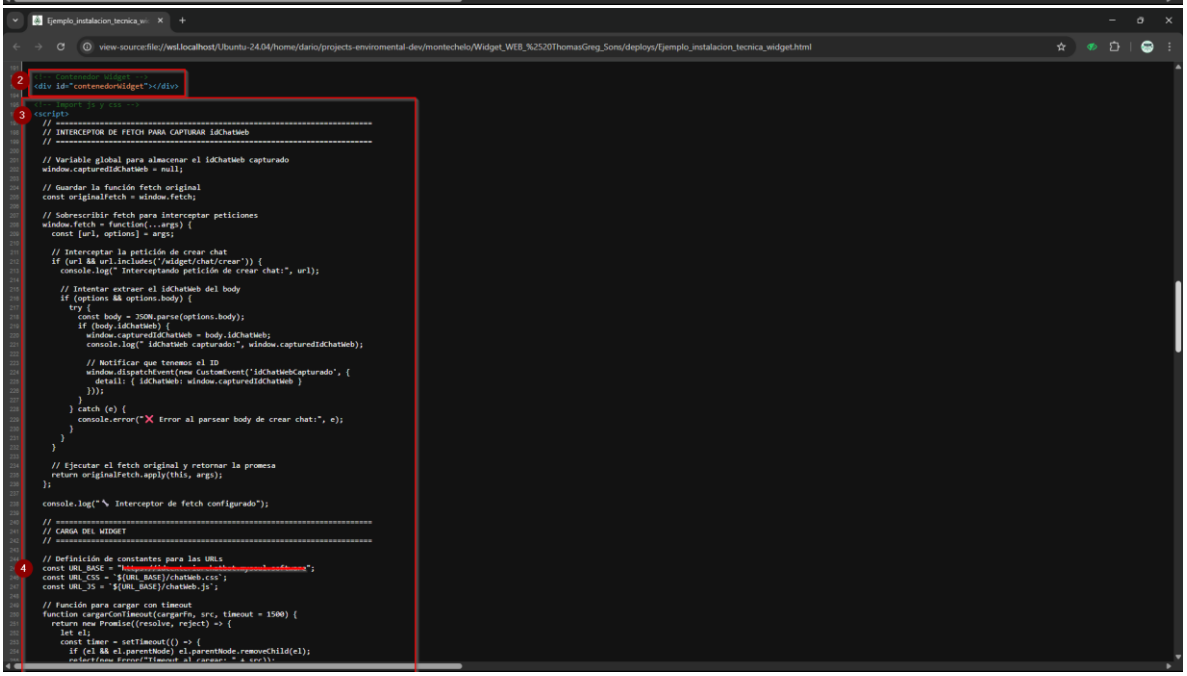
* Se recomienda realizar pruebas en un ambiente de desarrollo antes de implementar los cambios en el sitio de producción.

* Por favor, no duden en contactarnos si necesitan mayor información o soporte técnico durante el proceso de integración.

Ejemplo visual de implementación



```
view-source:file://wsl.localhost/Ubuntu-24.04/home/dario/projects-enviromental-dev/montechelo/Widget_WEB_%2520ThomasGreg_Sons/deploys/Ejemplo_instalacion_tecnica_widget.html

Line wrap

<!DOCTYPE html>
<html lang='en'>

<head>
  <meta charset='UTF-8'>
  <meta name='viewport' content='width=device-width, initial-scale=1.0'>
  <title>IDC EXTERIOR</title>

  <!-- Favicon -->
  <link rel="apple-touch-icon" href="https://idcexteriorchatbot.mysoul.software/images/imagen-corporativa/favicon.png">
  <link rel="shortcut icon" type="image/x-icon" href="https://idcexteriorchatbot.mysoul.software/images/imagen-corporativa/favicon.png">

  <!--Import Google Icon Font-->
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  <style>
    html,
    body {
      height: 100%;
      /* Asegura que html y body ocupen el 100% de la altura */
      margin: 0;
      /* Elimina márgenes por defecto */
    }

    body {
      background-image: url('https://idcexteriorchatbot.mysoul.software/images/imagen-corporativa/fondo.png');
      /* Llama a la imagen de fondo */
      background-repeat: no-repeat;
      /* No repetir la imagen */
      background-size: cover;
      /* Ajustar la imagen para cubrir todo el fondo */
      background-position: center;
      /* Centrar la imagen */
      background-attachment: fixed;
      /* Hacer que la imagen de fondo quede fija al hacer scroll */
    }
  </style>
</head>

<body>

  <div class="titlePagina" style="background: #0b06ff; height: 8%; position: fixed; top: 0; left: 0; right: 0; z-index: 1000;">
    <h1 style="text-align: center; margin: 0rem; color: rgb(255, 255, 255);"><strong>Widget Chat Web Thomas Greg & Sons - IDC Exterior Chatbot / Ejemplo Instalación Técnica</strong></h1>
  </div>

  <p>Lorem ipsum dolor sit amet consectetur, adipiscing elit phasellus egestas integer proin, eleifend quisque montes dignissim. Accumsan ultricies curae id turpis primis eu leo suscipit, conubia dictum massa laoreet sodales dis scelerisque, fringilla interdum
  Elementum gravida fusce urna placerat erat torquent iaculis sollicitudin, faucibus aptent suscipit class dignissim scelerisque senectus quis, eros inceptos massa vitae hac neque nam. Cras placerat commodo nam vehicula nunc tellus facilisis mi, feugiat sed q
  Nulla sagittis litora tortor nascetur senectus vehicula nam scelerisque non odio aptent, morbi donec in placerat lobortis magna tincidunt vel auctor class. Fringilla natoque netus porttitor vitae malesuada nullam congue facilisi, nascetur mi tempus ante vel
  Augue quisque sollicitudin euismod tellus aenean molestie, interdum cum et commodo donec varius, est class nullam nostra gravida. Primis auctor nascetur velit metus lacus varius iaculis nibh torquent, nostra eu accumsan non mattis porta consequat ut lacinia,
  Porta hendrerit netus nibh odio laoreet rhoncus ligula vitae natoque accumsan, integer ad facilisis orci lectus purus nam varius massa, id quam nunc urna augue a feugiat dui nostra. Duis nulla lacus nam eget etiam est, ultrices curae eleifend donec facilisis
  Nulla sollicitudin fames inceptos odio hac euismod metus blandit, dui sociosqu ac scelerisque diam pharetra fusce sodales in, quam urna nam aenean himenaeos turpis cubilia. Hendrerit nullam dui vehicula magna duis pellentesque porttitor dis eget, ac molestie
  Urna facilisis gravida natoque ligula nullam suspendisse, nostra mollis inceptos pulvinar aliquet aenean semper, maecenas montes nisi tellus habitasse. Libero cursus natoque fusce vivamus proin duis lacinia cum ultrices, aliquet magnis cubilia placerat hac
  Donec commodo mauris est at accumsan quisque, maecenas nec posuere primis etiam inceptos, aenean non euismod cubilia sed. Hac dis lobortis etiam libero senectus risus porta fames sociis mollis, penatibus sagittis sodales nam consequat tortor eros imperdiet
  <p>Lorem ipsum dolor sit amet consectetur, adipiscing elit phasellus egestas integer proin, eleifend quisque montes dignissim. Accumsan ultricies curae id turpis primis eu leo suscipit, conubia dictum massa laoreet sodales dis scelerisque, fringilla interdum
```



```
view-source:file://wsl.localhost/Ubuntu-24.04/home/dario/projects-enviromental-dev/montechelo/Widget_WEB_%2520ThomasGreg_Sons/deploys/Ejemplo_instalacion_tecnica_widget.html

<!-- Contenedor Widget -->
<div id="contenedorWidget"></div>

<!-- Import js y css -->
<script>
  // ================================================================
  // INTERCEPTOR DE FETCH PARA CAPTURAR idChatWeb
  // ================================================================

  // Variable global para almacenar el idChatWeb capturado
  window.capturedIdChatWeb = null;

  // Guardar la función fetch original
  const originalFetch = window.fetch;

  // Sobrescribir fetch para interceptar peticiones
  window.fetch = function(...args) {
    const [url, options] = args;

    // Interceptar la petición de crear chat
    if (url && url.includes('/widget/chat/crear')) {
      console.log(" Interceptando petición de crear chat:", url);

      // Intentar extraer el idChatWeb del body
      if (options && options.body) {
        try {
          const body = JSON.parse(options.body);
          if (body.idChatWeb) {
            window.capturedIdChatWeb = body.idChatWeb;
            console.log(" idChatWeb capturado:", window.capturedIdChatWeb);

            // Notificar que tenemos el ID
            window.dispatchEvent(new CustomEvent('idChatWebCapturado', {
              detail: { idChatWeb: window.capturedIdChatWeb }
            }));
          }
        } catch (e) {
          console.error("❌ Error al parsear body de crear chat:", e);
        }
      }
    }

    // Ejecutar el fetch original y retornar la promesa
    return originalFetch.apply(this, args);
  };

  console.log("🔧 Interceptor de fetch configurado");

  // ================================================================
  // CARGA DEL WIDGET
  // ================================================================

  // Definición de constantes para las URLs
  const URL_BASE = "https://idcexteriorchatbot.mysoul.software";
  const URL_CSS = `${URL_BASE}/chatWeb.css`;
  const URL_JS = `${URL_BASE}/chatWeb.js`;

  // Función para cargar con timeout
  function cargarConTimeout(cargarFn, src, timeout = 1500) {
    return new Promise((resolve, reject) => {
      let el;
      const timer = setTimeout(() => {
        if (el && el.parentNode) el.parentNode.removeChild(el);
        reject(new Error(`Timeout al cargar: ${src}`));
```

1
2
3
4

Ejemplo de visualización UI