# JAVASCRIPT API

La API de JavaScript proporciona una colección de funciones y eventos que sustentan la mayor parte de la funcionalidad de ZampiBot. Para comenzar a usar la API de JavaScript, siga el tutorial a continuación.

---

# ⚙ Configuración y Uso

Asegúrese de cargar el chat incluyendo el siguiente script en la página donde desea utilizar las APIs e inclúyalo **antes** del script del chat. Asegúrese de que los enlaces sean correctos. Si está utilizando la versión de WordPress, este paso no es necesario.

Si está utilizando la **versión Cloud**, incluya solo jQuery y el código de inserción (embed) desde su panel de control. Vea el ejemplo a continuación (reemplace `12345678` con su propio ID).

HTML

```
<script src="https://chatbot.zampisoft.com/script/js/min/jquery.min.js"></script>
<script id="chat-init"
src="https://chatbot.zampisoft.com/account/js/init.js?id=12345678"></script>
```

Ingrese los fragmentos de código, funciones y métodos de esta documentación dentro de una de las funciones a continuación:

JavaScript

```
(function ($) {
    $(document).on("SBInit", function () {
        // Su código aquí
    });
}(jQuery));
```

Si el evento `SBInit` no se dispara, utilice el evento `SBReady` en su lugar.

---

# Parámetros de Función

Ingrese los parámetros de la función en el mismo orden de esta documentación, por ejemplo: `SBChat.sendMessage(user_id, message, attachments)`.

---

# Depuración (Debug)

Verifique la consola del navegador para ver errores e información de depuración.

---

# □□ Objetos

Los objetos de JavaScript utilizados por ZampiBot se enumeran a continuación. En algunos casos, necesitará usar estos objetos para utilizar una función.

## □□ SBUser

Representa a un usuario.

**Uso:** `new SBUser(settings, extra)`

- `settings`: Array con los detalles del usuario, ej. `{ first_name: "", last_name: "", profile_image: "", email: "", user_type: "" }`
- `extra`: Array con los detalles adicionales del usuario, ej. `{ phone: "", city: "", language: "", country: "", birthday: "" }`

**Métodos:**

- `id`: Devuelve el ID del usuario.
- `type`: Devuelve el tipo de usuario. Valores disponibles: `visitor`, `lead`, `user`, `agent`, `admin`.
- `name`: Devuelve el nombre completo del usuario (nombre y apellido).
- `nameBeautified`: Devuelve el nombre completo del usuario si está disponible; de lo contrario, devuelve el nombre de visitante predeterminado.
- `image`: Devuelve la imagen de perfil del usuario.
- `get(key)`: Devuelve el detalle del usuario de la clave dada si está disponible; de lo contrario, devuelve una cadena vacía. Ej. `get("email")`.
- `getExtra(key)`: Devuelve el detalle adicional del usuario de la clave dada si está disponible; de lo contrario, devuelve una cadena vacía. Ej. `get("phone")`.
- `set(key, value)`: Actualiza un detalle de usuario o añade uno nuevo. Ej. `set("phone", "(02) 123 456789")`.

- `setExtra(key, value)`: Actualiza un detalle adicional de usuario o añade uno nuevo.
- `update(function(){})`: Conecta a la base de datos y actualiza los detalles del usuario y los detalles adicionales con datos frescos. Opcionalmente ejecuta una función al finalizar. Este método es asíncrono y requiere el ID del usuario para funcionar.
- `getConversations(function(conversations){}, exclude_id)`: Conecta a la base de datos y obtiene las conversaciones del usuario; cada conversación incluye un extracto del último mensaje. Opcionalmente ejecuta una función al finalizar. El parámetro `exclude_id` puede ser cualquier ID de conversación y excluirá esa conversación del valor devuelto. Este método es asíncrono y requiere el ID del usuario para funcionar.
- `getConversationsCode(conversations)`: Devuelve el código HTML de la lista de conversaciones. Opcionalmente acepta un array de objetos `SBConversation`.
- `getFullConversation(conversation_id, function(conversation){})`: Conecta a la base de datos y devuelve una conversación completa, incluyendo todos los mensajes. Opcionalmente ejecuta una función al finalizar. Este método es asíncrono y requiere un ID de conversación para funcionar.
- `getConversationByID(conversation_id, index)`: Busca una conversación con el ID dado y la devuelve; de lo contrario, devuelve `false`. Establezca `index` en `true` para obtener la posición en el array de conversaciones.
- `addConversation(conversation)`: Añade una nueva conversación al objeto de usuario. El parámetro `conversation` debe ser un objeto `SBConversation`. Este método no actualiza la base de datos.
- `removeConversation(conversation_id)`: Elimina una conversación.
- `getLastConversation()`: Devuelve la última conversación si existe; de lo contrario, devuelve `false`.
- `isConversationsEmpty()`: Devuelve `true` si el usuario tiene al menos 1 conversación; de lo contrario, devuelve `false`.
- `isExtraEmpty()`: Devuelve `true` si los detalles adicionales del usuario ya han sido establecidos; de lo contrario, devuelve `false`. Este método devuelve `true` también si la lista de detalles adicionales está vacía, pero está establecida.
- `delete(function(){})`: Elimina al usuario de la base de datos y todas las conversaciones y mensajes vinculados permanentemente. Opcionalmente ejecuta una función al finalizar. Este método es asíncrono y requiere el ID del usuario para funcionar.
- `language()`: Devuelve el idioma del usuario.

**Variables:**

- `details`: Array con los detalles del usuario.
- `extra`: Array con los detalles adicionales del usuario.
- `conversations`: Array con las conversaciones del usuario. Cada conversación contiene solo el último mensaje.

---

# ✉ SBMessage

Representa un mensaje de una conversación.

**Uso:** `new SBMessage(details)`

- `details`: Array con los detalles del mensaje, ej. `{ "id": "2319", "user_id": "377", "message": "...", ... }`

**Métodos:**

- `id`: Devuelve el ID del mensaje.
- `attachments`: Devuelve el array de adjuntos.
- `message`: Devuelve el texto del mensaje.
- `get(key)`: Devuelve el contenido de la clave dada si está disponible; de lo contrario, devuelve una cadena vacía. Ej. `get("message")`.
- `set(key, value)`: Actualiza un detalle del mensaje o añade uno nuevo. Ej. `set("message", "Hello!")`.
- `payload(key, value)`: Establece u obtiene el payload. El payload es un array asociativo que contiene datos extra. Si el valor `value` se establece, el método añade o actualiza la clave con el valor dado.
- `getCode(translation)`: Devuelve el código HTML del mensaje, listo para ser insertado en el elemento DOM del chat. Este método procesa adjuntos, Mensajes Enriquecidos y formato de texto. Establezca `translation` en `true` para obtener el mensaje traducido si está disponible.
- `render(message)`: Renderiza el mensaje y le da formato añadiendo estilos como negrita y cursiva, y convirtiendo URLs en enlaces clicables. Opcionalmente acepta una cadena y la renderiza en su lugar.
- `strip(message)`: Elimina el formato de texto. Opcionalmente acepta una cadena y elimina el formato de texto de ella en su lugar.

**Variables:**

- `details`: Array con los detalles del mensaje.

---

# □□ SBConversation

Representa una conversación.

**Uso:** `new SBConversation(messages, details)`

- `messages`: Array de objetos `SBMessage`.
- `details`: (Opcional) Array con los detalles de la conversación, ej. `{ conversation_status_code: "2", ... }`

**Métodos:**

- `id`: Devuelve el ID de la conversación.

- `get(key)`: Devuelve el contenido de la clave dada si está disponible; de lo contrario, devuelve una cadena vacía.
- `set(key, value)`: Actualiza un detalle de la conversación o añade uno nuevo.
- `getMessage(ID)`: Devuelve el mensaje con el ID dado. Devuelve `false` si no se encuentra.
- `getLastMessage()`: Devuelve el último mensaje de la conversación. Devuelve `false` si no hay mensajes.
- `getLastUserMessage(index, agent)`: Devuelve el último mensaje enviado por el usuario (excluye bots, agentes, admins). Opcionalmente acepta un `index` para iniciar la búsqueda en orden inverso.
  - `agent`: `true` (último de agentes/admins), `bot` (último del bot), `no-bot` (usuario o agente excluyendo bot), `all` (agente o bot).
- `getNextMessage(message_id, user_type)`: Devuelve el mensaje siguiente al del ID proporcionado. `user_type` puede ser `user` o `agent`.
- `updateMessage(ID, message)`: Actualiza el mensaje con el ID dado. El parámetro `message` debe ser un objeto `SBMessage`.
- `addMessages(messages)`: Añade nuevos mensajes a la conversación. El parámetro `messages` debe ser un objeto `SBMessage` único o un array de ellos.
- `getCode(text_only)`: Devuelve el código HTML, o el texto formateado, del mensaje.
- `deleteMessage(ID)`: Elimina el mensaje con el ID dado del objeto de conversación. No actualiza la base de datos.
- `searchMessages(search, exact_match)`: Busca la cadena especificada en todos los mensajes y devuelve un array de coincidencias. `exact_match` en `true` para coincidencia exacta.
- `getUserMessages(user_type)`: Devuelve un array conteniendo solo los mensajes enviados por el usuario, el bot o los agentes. Valores: `user`, `agents`, `bot`. Por defecto: `user`.
- `getAttachments()`: Devuelve un array con todos los adjuntos de la conversación.
- `getLastConversationID()`: Devuelve el ID de la última conversación si existe; de lo contrario, devuelve `false`.
- `updateMessagesStatus(ids)`: Actualiza el código de estado de múltiples mensajes a leído. `ids` es un array opcional de IDs de mensaje; si se proporciona, se añade el icono de check pero no se actualiza en la base de datos.

**Variables:**

- `details`: Array con los detalles de la conversación.

---

# □□ Variables Globales

Las variables en la lista a continuación son accesibles públicamente vía JavaScript.

| Variable | Descripción |
| --- | --- |
| `SB_ARTICLES_PAGE` | Establézcalo en `true` para mostrar la página de artículos en lugar del chat. |

| | |
|---|---|
| `SB_LOCAL_SETTINGS` | Sobrescribe las configuraciones predeterminadas del lado del cliente. El valor es un array de claves y valores (ej. `{ registration-required: false }`). Solo afecta configuraciones del cliente. |
| `SB_DISABLED` | Establézcalo en `true` e insértelo en una página para prevenir que el chat o el área de tickets carguen. |
| `SB_REGISTRATION_REQUIRED` | `true` para deshabilitar el registro obligatorio, `false` para deshabilitar el registro. |
| `SB_TICKETS` | Establézcalo en `true` para forzar la carga del área de tickets en lugar del chat. Requiere la App de Tickets. |
| `SB_DEFAULT_USER` | Establece los detalles del usuario por defecto para nuevos visitantes. Si un usuario registrado visita y su login difiere, se cierra sesión e inicia con los nuevos detalles. Incluya email y hash de contraseña para asegurar el funcionamiento. |
| `SB_DEFAULT_DEPARTMENT` | Asigne un **ID de departamento** para asignar automáticamente ese departamento a nuevas conversaciones creadas desde la página. |
| `SB_DEFAULT_AGENT` | Asigne un **ID de agente** para asignar automáticamente ese agente a nuevas conversaciones. Úselo con la opción "Ocultar conversaciones de otros agentes". |
| `SBChat.initialized` | `true` si el chat está inicializado. |
| `SBChat.conversation` | Devuelve la conversación activa. `false` si no hay ninguna. |
| `SBChat.is_busy` | `true` si el chat está en modo ocupado (no se pueden enviar mensajes). |
| `SBChat.chat_open` | `true` si el chat está abierto. |
| `SBChat.agent_id` | ID del agente activo en la conversación. `-1` si no hay. |
| `SBChat.agent_online` | `true` si el agente activo está en línea. |
| `SBChat.user_online` | `true` si el usuario está en línea. |
| `SBChat.timetable` | `true` si la hora actual está dentro del horario de oficina. |
| `SBChat.dashboard` | `true` si el panel de control está activo y visible. |

---

# ⚡ Funciones

Funciones para gestionar el chat, usuarios, conversaciones y mensajes.

## SBChat.submit()

Ejecuta el evento de clic del botón de envío del editor de chat, envía un mensaje con el contenido insertado por el usuario (mensaje y/o adjuntos) y limpia el editor.

---

# SBChat.sendMessage()

Añade un nuevo mensaje a la conversación activa.

**Parámetros**

- `user_id`: El ID del usuario que envía el mensaje. Utilice la función `SBF.setting("bot-id")` para obtener el ID del bot. **Por defecto:** -1 (ID del usuario activo).
- `message`: El texto del mensaje.
- `attachments`: Array de adjuntos. Sintaxis del array: `[["nombre", "enlace"], ["nombre", "enlace"], ...]`. Reemplace `nombre` con el nombre del adjunto y `enlace` con la URL completa del adjunto. **Por defecto:** `[]`.
- `onSuccess`: Función a ejecutar cuando la operación se completa. Sintaxis: `function(response) { ... }`. La respuesta es la misma que la del evento `SBMessageSent`. **Por defecto:** `false`.
- `payload`: Array asociativo conteniendo datos extra, ej. `{ "event": "delete-message" }`.
- `conversation_status_code`: El código de estado de la conversación, si se crea una nueva. Códigos de estado: activo (live) = `0`, esperando respuesta del usuario = `1`, esperando respuesta del agente = `2`, archivado = `3`, papelera = `4`. Establézcalo en `skip` para mantener el estado actual.

**Información**

- **Requisito:** El usuario debe estar activo.

---

# SBChat.updateMessage()

Cambia el texto de un mensaje existente.

**Parámetros**

- `message_id`: El ID del mensaje.
- `message`: El texto del mensaje.

**Información**

- **Requisito:** El usuario debe estar activo.
- Si el usuario activo es un agente o administrador, se pueden actualizar los mensajes de cualquier usuario. Si el usuario activo es un usuario final, solo puede actualizar sus

propios mensajes.

---

# SBChat.sendEmail()

Envía un correo electrónico a los usuarios o agentes devueltos por `getRecipientUserID()`.

**Parámetros**

- `message`: El texto del mensaje.
- `attachments`: Array de adjuntos. Sintaxis del array: `[["nombre", "enlace"], ["nombre", "enlace"], ...]`. Dialogflow puede leer este array.
- `send_to_active_user`: Establézcalo en `true` para enviar el correo al usuario activo. Establézcalo en un ID de usuario para enviarlo a ese usuario específico. **Por defecto:** `false`.
- `onSuccess`: Función a ejecutar cuando se completa. Sintaxis: `function(response) { ... }`. **Por defecto:** `false`.

---

# SBChat.sendSMS()

Envía un mensaje de texto (SMS) a los usuarios o agentes devueltos por `getRecipientUserID()`.

**Parámetros**

- `message`: El texto del mensaje.

---

# SBChat.desktopNotification()

Envía una notificación de escritorio (Notificación Web) al usuario, o al agente conectado si la notificación se envía desde el área de administración.

**Parámetros**

- `title` **Requerido**: El título de la notificación.
- `message` **Requerido**: El texto del mensaje.
- `icon` **Requerido**: El icono de la notificación.
- `conversation_id`: El ID de la conversación que se abrirá cuando el usuario haga clic en la notificación.

---

# SBChat.getRecipientUserID()

Determina el destinatario adecuado basándose en el contexto.

- Si el usuario activo es un **usuario**: devuelve el ID del último agente que respondió la conversación; de lo contrario, el ID del agente asignado; si no, el ID del departamento asignado; de lo contrario devuelve `agents` (significa todos los agentes).
- Si el usuario activo es un **admin o agente**: devuelve el ID del usuario activo.

---

# SBChat.initChat()

Inicializa el chat y muestra el botón de chat.

**Información**

- Use esta función en combinación con el ajuste **Inicialización manual** del área `Configuración > Chat`.
- Este método no debe insertarse en el evento `$(document).on("SBInit", function () { ... });`, use `$(document).on("SBReady", function () { ... });` en su lugar.

---

# SBChat.open()

Abre o cierra la ventana del chat. También puede añadir la clase o ID `sb-open-chat` a cualquier elemento (por ejemplo, un botón) para abrir o cerrar el chat cuando se haga clic en él.

**Parámetros**

- `open`: Establézcalo en `false` para cerrar el chat. **Por defecto:** `true`.

---

# SBChat.openConversation()

Abre una conversación y la muestra en la ventana de chat.

**Parámetros**

- `conversation_id` **Requerido**: El ID de la conversación a abrir. Solo se pueden abrir las conversaciones del usuario activo. Use la función `SBF.activeUser().conversations` para obtener la lista.

---

# SBChat.update()

Actualiza la conversación activa y comprueba si hay nuevos mensajes. Esta función se dispara automáticamente cada 1000ms.

**Información**

- **Requisito:** Debe haber una conversación activa.
- **Requisito:** El usuario debe estar activo.

---

# SBChat.populateConversations()

Puebla la lista de conversaciones del usuario en el panel (dashboard) con todas las conversaciones del usuario.

**Parámetros**

- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(conversations) { ... }`.

**Información**

- **Requisito:** El usuario debe estar activo.

---

# SBChat.updateConversations()

Actualiza la lista de conversaciones del usuario en el panel y comprueba si hay nuevas conversaciones. Esta función se dispara automáticamente cada 10000ms.

**Información**

- **Requisito:** El usuario debe estar activo.

---

# SBChat.newConversation()

Crea una nueva conversación y opcionalmente añade el primer mensaje a ella.

**Parámetros**

- `status_code`: El código de estado de la conversación. **Por defecto:** 0. (0=activo, 1=esperando usuario, 2=esperando agente, 3=archivo, 4=papelera).
- `user_id`: El ID del usuario vinculado a la nueva conversación. **Por defecto:** -1 (ID usuario activo).
- `message`: El texto del mensaje.
- `attachments`: Array de adjuntos.
- `department`: El ID de un departamento. Puede obtener los IDs en `Configuración > Misceláneas > Departamentos`. **Por defecto:** NULL.
- `agent_id`: El ID del agente asignado. **Por defecto:** NULL.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(conversation) { ... }`. **Por defecto:** `false`.

**Información**

- **Requisito:** El usuario debe estar activo.

---

# SBChat.setConversation()

Establece una conversación existente como la conversación activa.

**Parámetros**

- `conversation`: La conversación. Debe ser un objeto `SBConversation`.

**Información**

- **Requisito:** El usuario debe estar activo.

---

# SBChat.startRealTime()

Inicia la comprobación en tiempo real de nuevos mensajes para la conversación activa cada 1000ms.

---

# SBChat.stopRealTime()

Detiene la comprobación en tiempo real de nuevos mensajes.

---

# SBChat.busy()

Muestra u oculta el icono de carga y habilita o deshabilita el modo ocupado del chat. Cuando el chat está en modo ocupado, no se pueden enviar mensajes ni adjuntos.

**Parámetros**

- `value` **Requerido**: Establézcalo en `true` para mostrar el icono de carga, `false` para ocultarlo.

**Información**

- El icono de carga aparece solo en el área de conversación, no en el panel (dashboard) ni en otros paneles.
- **Requisito:** El usuario debe estar activo.

---

# SBChat.lastAgent()

Devuelve el último agente de la conversación activa.

**Parámetros**

- `bot`: Establézcalo en `false` para excluir al bot. **Por defecto:** `true`.

**Respuesta**

JSON

```
{
    "user_id": "123456",
    "full_name": "Don John",
    "profile_image": "https://zampisoft.com/agent.svg"
}
```

---

# SBChat.scrollBottom()

Desplaza el chat hacia el final (abajo).

**Parámetros**

- `top`: Establézcalo en `true` para desplazar hacia el principio (arriba). **Por defecto:** `false`.

# SBChat.isBottom()

Comprueba si el chat está desplazado hasta el final.

---

# SBChat.showDashboard()

Muestra el panel de control (dashboard).

---

# SBChat.hideDashboard()

Oculta el panel de control.

---

# SBChat.showPanel()

Muestra el área especificada dentro del widget de chat.

**Parámetros**

- `name`: El nombre del panel. Valores disponibles: `articles`.

---

# SBChat.hidePanel()

Oculta el panel activo dentro del widget de chat.

---

# SBChat.clear()

Limpia el área de conversación del widget de chat y deshabilita la conversación activa.

---

# SBChat.updateNotifications()

Actualiza el contador rojo de notificaciones del botón de chat que alerta al usuario de nuevos mensajes y conversaciones.

**Parámetros**

- `conversation_id` **Requerido**: El ID de la conversación vinculada a la actualización del contador.
- `message_id`: El ID del mensaje no leído, establézcalo para incrementar el contador en 1. Establézcalo en `false` para decrementar el contador en 1. **Por defecto:** `false`.

---

# SBChat.setConversationStatus()

Actualiza el código de estado de una conversación.

**Parámetros**

- `status_code` **Requerido**: El código de estado a asignar. (0=activo, 1=esperando usuario, 2=esperando agente, 3=archivo, 4=papelera).

**Información**

- **Requisito:** El usuario debe estar activo.

---

# SBChat.typing()

Gestiona la etiqueta "escribiendo..." del encabezado del chat.

**Parámetros**

- `user_id`: El ID del usuario activo (o del agente activo si está en el admin). **Por defecto:** -1.
- `action`: Valores disponibles:
  - `check`: Comprueba si el usuario con el ID dado está escribiendo y actualiza el encabezado.
  - `set`: Asigna el estado de escribiendo al usuario y actualiza el encabezado.
  - `start`: Muestra el estado de escribiendo en el encabezado.
  - `stop`: Oculta el estado de escribiendo del encabezado.

**Información**

- La etiqueta de escribiendo es visible solo cuando una conversación está abierta.

# SBChat.showArticles()

Muestra el área de artículos o un solo artículo.

**Parámetros**

- `id`: El ID de un artículo para mostrar. Use `SBChat.getArticles()` para obtener la lista. **Por defecto:** -1.

# SBChat.getArticles()

Devuelve la lista de artículos o el contenido de un solo artículo.

**Parámetros**

- `id`: El ID de un artículo a mostrar. Añada múltiples IDs separados por comas. **Por defecto:** `false`.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`. **Por defecto:** `false`.
- `category`: Devuelve solo los artículos de la categoría dada. Si es `true` devuelve también la lista de categorías. **Por defecto:** `true`.
- `count`: El número máximo de artículos a devolver. **Por defecto:** `true`.

**Respuesta**

JSON


```
[
    {
        "id": "6P2Oq",
        "title": "What's new with the API V2?p",
        "content": "The API V2 is the new iteration of our developer API ...",
        "link": "https://board.support",
        "categories": ["Nv9PG"]
    },
    {
        "title": "Which API version am I currently using?",
        "content": "The API version is configured separately for each ...",
        "link": "",
        "id": "IDkft",
        "categories": []
    },
```

```
    ...
]
```

Si es un solo artículo, el valor devuelto es el artículo:

```
{ "id" : "6P2Oq" , "title" : "¿Qué novedades trae la API V2?p" , "content" : "La API V2 es la nueva versión de nuestra API para desarrolladores. Esta nueva API integra Google Cloud Spe. La API V2 es la nueva versión de nuestra API para desarrolladores." , "link" : "https://zampisoft.com" }
```

**Información**

- Cada artículo de la lista contiene solo un extracto del contenido.

---

# SBChat.getArticleCategories()

Devuelve las categorías de artículos.

**Parámetros**

- `category_type`: Establézcalo en `parent` para devolver solo las categorías padres. Establézcalo en `normal` para devolver todas las categorías excepto las padres. **Por defecto:** `false`.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`. **Por defecto:** `false`.

**Respuesta**

```
[
  {
    "id": "audio",
    "titulo": "Audio",
    "descripcion": "Lorem ipsum dolor adipiscing elit.",
    "imagen": "https://example.com/image.png",
    "parent": true,
    "idiomas": {
      "es": {
        "titulo": "Audio",
        "descripcion": "Leorem ipsum dolor consectetur adipiscing elit."
      }
```

```
    }
  },
  {
    "id": "audio_digital",
    "titulo": "Audio digital",
    "descripcion": "Lorem ipsum dolor sit amet, adipiscing.",
    "imagen": "",
    "parent": false,
    "idiomas": {}
  }
]
```

---

# SBChat.searchArticles()

Muestra los artículos que coinciden con la búsqueda en la caja de artículos del panel (dashboard).

**Parámetros**

- `search` **Requerido**: Cadena con los términos de búsqueda. La función de búsqueda soporta el título y el contenido.
- `button` **Requerido**: El objeto del botón de búsqueda.
- `target` **Requerido**: El objeto HTML donde se insertarán los artículos.

**Información**

- **Requisito:** El panel (dashboard) debe estar activo.
- **Requisito:** Los artículos deben estar activos.

---

# SBChat.setArticleRating()

Establece la calificación de un artículo.

**Parámetros**

- `article_id` **Requerido**: El ID del artículo.
- `rating` **Requerido**: La calificación a añadir. Ingrese `1` para una calificación positiva o `0` para una negativa.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`. **Por defecto:** `false`.

---

# SBChat.categoryEmoji()

Selecciona una categoría de la caja de emojis.

**Parámetros**

- `category` **Requerido**: El nombre de la categoría. Valores disponibles: `Smileys`, `People`, `Animals`, `Food`, `Travel`, `Activities`, `Objects`, `Symbols`.

---

# SBChat.searchEmoji()

Busca emojis que coincidan con los términos de búsqueda y los muestra en la caja de emojis.

**Parámetros**

- `search` **Requerido**: El texto de búsqueda.

---

# SBChat.insertText()

Inserta una cadena en el editor de chat.

**Parámetros**

- `text`: La cadena a insertar en el editor.

**Información**

- **Requisito:** Debe haber una conversación activa y abierta.

---

# SBChat.privacy()

Muestra el mensaje de privacidad y obliga al usuario a aceptar los términos antes de iniciar el chat.

---

# SBChat.popup()

Muestra un mensaje emergente (pop-up) o lo cierra.

**Parámetros**

- `close`: Establézcalo en `true` para cerrar el pop-up. **Por defecto:** `false`.
- `content`: Array con el contenido del pop-up. Sintaxis del array: `{ image: "", title: "", message: "" }`. Establézcalo en `false` para usar el contenido de `Configuración > Chat > Mensaje emergente`. **Por defecto:** `false`.

**Información**

- **Requisito:** El chat debe estar cerrado.

---

# SBChat.slackMessage()

Envía un mensaje a Slack.

**Parámetros**

- `user_id` **Requerido**: El ID del usuario activo (o el ID del agente activo si está en el área de administración). Utilice la función `SBF.activeUser().id` para obtener el ID del usuario activo (utilice la variable `SB_ACTIVE_AGENT["id"]` para obtener el ID del agente activo si está en el área de administración). Utilice la variable `SBChat.agent_id` para obtener el ID del último agente de la conversación. **Por defecto:** -1.
- `full_name` **Requerido**: El nombre del remitente, aparecerá en Slack a la izquierda del mensaje. Debería ser el nombre de un agente si el mensaje proviene de un agente, o el nombre del usuario en caso contrario.
- `profile_image` **Requerido**: La imagen de perfil del remitente, aparecerá en Slack a la izquierda del mensaje. Debería ser la imagen de perfil de un agente si el mensaje proviene de un agente, o la imagen del usuario en caso contrario. Formatos soportados: `PNG` y `JPG`.
- `message` **Requerido**: El texto del mensaje.
- `attachments`: Array de adjuntos. Sintaxis del array: `[["nombre", "enlace"], ["nombre", "enlace"], ...]`. Reemplace `nombre` con el nombre del adjunto y `enlace` con la URL completa del adjunto.

**Información**

- **Requisito:** Debe haber una conversación activa.

---

# SBChat.deleteMessage()

Elimina un mensaje de la base de datos y de la conversación activa si está disponible.

**Parámetros**

- `message_id` **Requerido**: El ID del mensaje a eliminar. Utilice la función `SBChat.conversation.messages` para obtener la lista de mensajes de la conversación activa.

**Información**

- Solo se pueden eliminar los mensajes de las conversaciones del usuario activo.
- **Requisito:** Debe haber una conversación activa.

---

# SBChat.registration()

Muestra el área de registro o de inicio de sesión, o comprueba si el registro es obligatorio.

**Parámetros**

- `check`: Establézcalo en `true` para comprobar si el registro es obligatorio. **Por defecto:** `false`.
- `type`: Ingrese `registration` para mostrar el formulario de registro, inserte `login` para mostrar el formulario de inicio de sesión.

---

# SBChat.addUserAndLogin()

Registra un nuevo usuario como **visitante** y lo conecta (login) automáticamente después del registro. Opcionalmente ejecuta una función al completar.

**Parámetros**

- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`. **Por defecto:** `false`.

**Respuesta**


```
[
    {
        "id": "913",
        "profile_image": "https://zampisoft.com/user.svg",
        "first_name": "Don",
        "last_name": "John",
        "email": "hello@example.com",
        "user_type": "user",
```

```
        "token": "9b25351047ee758aa97ee4868d130cc1ceb8decf"
    },
    "YXNkWGNSeTdtRTdDYVkxVG8wckN4YWF6V2s0Tk1mczBSVHdQbHBpOWdmejVUTTdOUUxEUENhdUVoYmROWn..."
]
```

El último valor son los datos de inicio de sesión encriptados listos para ser almacenados en el Web Storage del navegador del usuario. Utilice la función `SBF.loginCookie(response[1]);` para almacenarlo.

**Información**

- Si un usuario ya ha iniciado sesión, se requiere recargar la página para activar el nuevo usuario.

---

# SBChat.getDepartmentCode()

Devuelve un código HTML con los detalles del departamento dado o de todos los departamentos. Detalles del departamento: `id`, `color`, `image`, `name`.

**Parámetros**

- `department_id`: El ID del departamento. Si este parámetro no se establece o es nulo, la función devuelve el código de todos los departamentos.
- `onSuccess` **Requerido**: Función a ejecutar al completar. Sintaxis: `function(response) { ... }` .

**Respuesta**

HTML

```
<div data-color="red"><img src="sales.jpg" /><div>Sales<div></div>
```

---

# SBChat.offlineMessage()

Comprueba si el mensaje de fuera de línea puede enviarse y lo envía.

**Información**

- **Requisito:** Debe haber una conversación activa.

---

# SBChat.isInitDashboard()

Comprueba si el panel (dashboard) se muestra por defecto cuando se inicializa el widget de chat.

---

# SBChat.closeChat()

Archiva una conversación y la oculta del widget de chat.

**Parámetros**

- `update_conversation_status`: Establézcalo en `false` para solo ocultar la conversación del widget de chat sin establecer su estado como archivado. **Por defecto:** `true`.

---

# SBChat.flashNotification()

Inicia la notificación parpadeante (flash).

---

# SBChat.playSound()

Reproduce el sonido que suena cuando se recibe un nuevo mensaje.

**Parámetros**

- `repeat`: Establézcalo en `true` para repetir el sonido tantas veces como se haya configurado en los ajustes. **Por defecto:** `false`.

---

# SBChat.automations.runAll()

Comprueba todas las automatizaciones y las ejecuta si las condiciones de activación se validan.

---

# □□ JAVASCRIPT API: Apps

Lista de funciones de las aplicaciones de ZampiBot.

# SBApps.is()

Comprueba si una app está disponible.

**Parámetros**

- `name` **Requerido**: El nombre de la app, ej. `dialogflow`.

---

# SBApps.wordpress.ajax()

Realiza una llamada AJAX de WordPress.

**Parámetros**

- `action` **Requerido**: La acción. Valores disponibles: `wp-login`, `wp-registration`, `button-purchase`. Otros valores pueden estar disponibles, revise el archivo `main.js` para todos los valores.
- `data` **Requerido**: Los parámetros de la acción, revise el archivo `main.js` para más detalles.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`. **Por defecto:** `false`.

---

# SBApps.dialogflow.message()

Envía un mensaje a Dialogflow y añade la respuesta de Dialogflow a la conversación activa como un nuevo mensaje.

**Parámetros**

- `message`: El texto del mensaje.
- `attachments`: Array de adjuntos. Sintaxis del array: `[["nombre", "enlace"], ["nombre", "enlace"], ...]`. Reemplace `nombre` con el nombre del adjunto y `enlace` con la URL completa del adjunto.
- `delay`: Retraso de la respuesta del bot en milisegundos.
- `parameters`: Array de información opcional. Sintaxis del array: `{ "nombre": "valor", "nombre": "valor", ...}`.
- `audio`: URL o ruta de un archivo de audio para el reconocimiento de voz. **Por defecto:** `false`.

**Información**

- **Requisito:** La aplicación de Inteligencia Artificial es requerida y Dialogflow debe estar activo en el área de **Configuración**.
- **Requisito:** El usuario debe estar activo.
- Use `SBApps.dialogflow.project_id = AGENT_ID` para cambiar el agente de Dialogflow por defecto.

---

# SBApps.dialogflow.active()

Comprueba si Dialogflow y OpenAI están activos o los desactiva.

**Parámetros**

- `active`: Establézcalo en `false` para deshabilitar Dialogflow y OpenAI y detener el chatbot. Establézcalo en `activate` para activar el chatbot. **Por defecto:** `true`.
- `check_dialogflow`: Para verificar solo la activación de OpenAI, cámbielo a `false`. **Por defecto:** `true`.
- `check_open_ai`: Para verificar solo la activación de Dialogflow, cámbielo a `false`. **Por defecto:** `true`.

**Respuesta** Devuelve `true` si el bot de Dialogflow está activo; de lo contrario, devuelve `false`.

---

# SBApps.dialogflow.welcome()

Dispara la Intención de Bienvenida (Welcome Intent) de Dialogflow y muestra el mensaje de bienvenida de Dialogflow.

**Información**

- **Requisito:** La aplicación de Inteligencia Artificial es requerida y Dialogflow debe estar activo en el área de **Configuración**.

---

# SBApps.dialogflow.openAI()

Envía un mensaje a OpenAI (ChatGPT), devuelve la respuesta y opcionalmente añade la respuesta como un nuevo mensaje.

**Parámetros**

- `message` **Requerido**: El mensaje.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`.

- `audio`: URL o ruta de un archivo de audio para el reconocimiento de voz. **Por defecto:** `false`.

**Información**

- **Requisito:** La aplicación de Inteligencia Artificial es requerida y `Configuración > Inteligencia Artificial > OpenAI > Chatbot` debe estar activo.
- Si hay una conversación activa, la respuesta de OpenAI se añade automáticamente como un nuevo mensaje.

---

# SBApps.dialogflow.typing()

Inicia la animación de "escribiendo" del widget de chat.

---

# SBApps.dialogflow.humanTakeover()

Inicia la toma de control humana (human takeover) de Dialogflow configurada en `Configuración > Inteligencia Artificial > Toma de control humana`.

---

# SBApps.dialogflow.humanTakeoverActive()

Comprueba si la toma de control humana está activa para la conversación activa.

---

# SBApps.dialogflow.translate()

Traduce múltiples cadenas a través de Google Translate.

**Parámetros**

- `strings` **Requerido**: Array de las cadenas a traducir, ej. `["Hello world", "How are you?"]`.
- `language_code` **Requerido**: El código de idioma de dos letras al que desea traducir. Para Chino Tradicional use `zt`, para Chino Simplificado use `zh`, para Portugués Brasileño use `pt`.
- `onSuccess` **Requerido**: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`.

---

# □□ JAVASCRIPT API: Tickets

Lista de funciones de la App de Tickets.

## SBTickets.showPanel()

Muestra un panel o el área de conversación.

**Parámetros**

- `name`: El nombre del panel a mostrar. Nombres disponibles: `new-ticket`, `privacy`, `articles`, `edit-profile`, `login`, `registration`. Déjelo vacío para mostrar el área de conversación principal.
- `title`: El título a mostrar como nombre del panel.

---

## SBTickets.showSidePanels()

Muestra u oculta los paneles laterales.

**Parámetros**

- `show`: Establézcalo en `false` para ocultar los paneles.

---

## SBTickets.setAgent()

Obtiene los detalles del agente y puebla el área de perfil del agente del panel derecho.

**Parámetros**

- `agent_id`: El ID del agente.

---

## SBTickets.activateConversation()

Activa y muestra una conversación.

**Parámetros**

- `conversation`: La conversación a activar como un objeto `SBConversation`.

---

# SBTickets.selectConversation()

Establece el estilo de una conversación del panel izquierdo como la conversación activa.

**Parámetros**

- `conversation_id`: El ID de la conversación.

---

# SBTickets.getActiveConversation()

---

Devuelve el objeto HTML DOM de la conversación activa.

**Parámetros**

- `type`: Establézcalo en `ID` para obtener solo el ID de la conversación.

---

# □□ JAVASCRIPT API: Pusher

Lista de funciones para Pusher. Más detalles en `pusher.com`.

# SBPusher.init()

Inicializa Pusher y opcionalmente ejecuta una función al completar la inicialización.

**Parámetros**

- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`.

---

# SBPusher.start()

Inicia Pusher y las notificaciones Push. Ejecute esta función después de `SBPusher.init()` y después de que el usuario esté activo.

# SBPusher.subscribe()

Suscribe al usuario activo a un canal de Pusher.

**Parámetros**

- `channel_name` **Requerido**: El nombre del canal.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`.

**Información**

- Si Pusher no está inicializado, la función lo inicializa automáticamente.
- Puede acceder al canal desde `SBPusher.channels['NOMBRE DEL CANAL']`.

---

# SBPusher.event()

Suscribe al usuario activo a un evento de un canal de Pusher y ejecuta la función dada cuando se recibe el evento.

**Parámetros**

- `event` **Requerido**: El nombre del evento.
- `callback` **Requerido**: Función a ejecutar una vez que se recibe el evento. Sintaxis: `function(response) { ... }`.
- `channel`: El nombre del canal. **Por defecto:** `private-user-[ID usuario activo]`.

**Información**

- Si Pusher no está inicializado, la función lo inicializa automáticamente.
- Si el usuario no está suscrito al canal, la función suscribe al usuario automáticamente.

---

# SBPusher.trigger()

Dispara un evento en un canal de Pusher.

**Parámetros**

- `event` **Requerido**: El nombre del evento.
- `data`: Array de valores. Sintaxis: `{ "nombre": "valor" }`.
- `channel`: El nombre del canal. **Por defecto:** `private-user-[ID usuario activo]`.

# SBPusher.presence()

Suscribe al usuario activo al canal de presencia utilizado para el estado en línea de usuarios y administradores.

**Parámetros**

- `index`: El índice del canal de presencia. **Por defecto:** 1.
- `onSuccess`: Función a ejecutar al completar. Sintaxis: `function(response) { ... }`.

**Información**

- Si Pusher no está inicializado, la función lo inicializa automáticamente.

---

# SBPusher.presenceUnsubscribe()

Desuscribe al usuario activo del canal de presencia.

---

# SBF.serviceWorker.pushNotification()

Envía una notificación Push al último agente de la conversación, o a todos los agentes si ningún agente ha respondido aún.

**Parámetros**

- `message` **Requerido**: El texto del mensaje.
- `interests`: Los destinatarios. Puede ser un ID de usuario o agente, o las siguientes cadenas: `agents`, `department-123` (reemplace 123 con el ID del departamento). **Por defecto:** asignado automáticamente a los destinatarios correctos.

**Información**

- Esta función solo funciona si las notificaciones Push están activas en el área de configuración.
- Actualmente, las notificaciones Push solo son compatibles con dispositivos Mac, Windows y Android. Los iPhones y dispositivos iOS no son compatibles.

# JAVASCRIPT API: Más funciones

Lista de varias funciones que realizan diferentes tareas.

## SBF.translate()

Traduce una cadena al idioma del usuario activo.

**Parámetros**

- `string` **Requerido**: La cadena a traducir.

**Información**

- Devuelve la cadena traducida si está disponible; de lo contrario, devuelve la cadena original.
- Esta función funciona solo para las traducciones del **front-end**.
- Para más detalles sobre el idioma activo, consulte la documentación de traducciones.

---

## SBF.activeUser()

Devuelve el usuario activo como un objeto `SBUser`, devuelve `false` si no se encuentra el usuario activo.

**Representación JSON del usuario activo:**

JSON

```
{
    "details": {
        "id": "914",
        "profile_image": "https://zampisoft.com/user.svg",
        "first_name": "User",
        "last_name": "#23262",
        "email": null,
        "user_type": "visitor",
```

```
        "token": "bc308e274473fb685a729abe8a4bf82d3c49cd2f"
    },
    "extra": {},
    "conversations": []
}
```

**Información**

- Para los métodos del usuario y más detalles, consulte la documentación de `SBUser`.

---

# SBF.getActiveUser()

Activa el usuario conectado (logueado) y devuelve los detalles del usuario.

**Parámetros**

- `db`: Establézcalo en `true` para verificar que el usuario existe en la base de datos.
- `onSuccess`: Función a ejecutar cuando la función se completa. Sintaxis: `function(response) { ... }`.

**Respuesta**

JSON

```
{
    "id": "914",
    "profile_image": "https://zampisoft.com/user.svg",
    "first_name": "Don",
    "last_name": "John",
    "email": "hello@example.com",
    "user_type": "user",
    "token": "bc308e274473fb685a729abe8a4bf82d3c49cd2f"
}
```

**Información**

- Para los métodos del usuario y más detalles, consulte la documentación de `SBUser`.

---

# SBF.cors()

Ejecuta una petición HTTP POST o GET a una URL y devuelve la respuesta.

**Parámetros**

- `method`: Inserte `POST` o `GET`. **Por defecto:** `GET`.
- `url` **Requerido**: Ingrese la URL completa de la petición.
- `onSuccess` **Requerido**: Función a ejecutar cuando la función se completa. Sintaxis: `function(response) { ... }`.

---

# SBF.null()

Comprueba si una variable existente es nula o vacía.

**Parámetros**

- `variable` **Requerido**: La variable a comprobar.

**Información**

- Devuelve `true` si la variable es una cadena vacía, `null`, `'null'`, o `undefined`.

---

# SBF.deactivateAll()

Oculta todas las ventanas emergentes (pop-ups) y las ventanas. Esta función se utiliza principalmente en el área de administración.

---

# SBF.getURL()

Busca un parámetro en la URL y devuelve su valor, o devuelve un array con todos los parámetros.

**Parámetros**

- `name`: El parámetro a buscar. Devuelve `false` si el parámetro no se encuentra. Si no se proporciona el nombre del parámetro, se devuelve un array con todos los parámetros. **Por defecto:** `false`.
- `url`: La URL de la cual extraer los parámetros. **Por defecto:** URL actual.

---

# SBF.restoreJson()

Convierte una cadena codificada en JSON a texto normal.

**Parámetros**

- `value` **Requerido**: La cadena a convertir.

---

# SBF.stringToSlug()

Convierte una cadena a un slug eliminando todos los caracteres especiales, reemplazando todos los espacios con el carácter `-`, y convirtiendo la cadena a minúsculas.

**Parámetros**

- `value` **Requerido**: La cadena a convertir.

**Información**

- Utilice la función `slugToString(slug)` para convertir un slug de nuevo a una cadena.

---

# SBF.settingsStringToArray()

Convierte una cadena a un array de valores. Formato de la cadena: `nombre:valor,nombre:valor,...`.

**Parámetros**

- `value` **Requerido**: La cadena a convertir.

**Respuesta**

JSON

```
{
    "name": "value",
    "name": "value",
    ...
}
```

---

# SBF.random()

Devuelve una cadena alfanumérica aleatoria.

---

# SBF.isAgent()

Comprueba si un tipo de usuario es un agente. Devuelve `true` solo si el tipo de usuario es `agent`, `admin`, o `bot`.

**Parámetros**

- `user_type` **Requerido**: La cadena del tipo de usuario a comprobar.

---

# SBF.error()

Dispara el error de JavaScript personalizado de ZampiBot.

**Parámetros**

- `message` **Requerido**: Cadena con el mensaje de error o un objeto `Error`.

---

# SBF.errorValidation()

Comprueba si una respuesta de una llamada AJAX es un error de validación.

**Parámetros**

- `response` **Requerido**: La respuesta de la llamada AJAX.

# SBF.login()

Inicia sesión de un usuario o un agente. El inicio de sesión se puede completar de dos maneras: vía correo electrónico y contraseña, o vía ID de usuario y token.

**Parámetros**

- `email`: El correo electrónico del usuario para iniciar sesión. Si se establece este atributo, también debe establecer la contraseña. **Por defecto:** cadena vacía.
- `password`: La contraseña del usuario para iniciar sesión. Si se establece este atributo, también debe establecer el correo electrónico. **Por defecto:** cadena vacía.

- `user_id`: El ID del usuario para iniciar sesión. Si se establece este atributo, también debe establecer el token. **Por defecto:** cadena vacía.
- `token`: El token del usuario para iniciar sesión. Si se establece este atributo, también debe establecer el ID del usuario. Puede obtener el token desde el área de **Usuarios** abriendo el cuadro de perfil de un usuario, solo si usted es un administrador. **Por defecto:** cadena vacía.
- `onSuccess`: Función a ejecutar cuando la función se completa. Sintaxis: `function(response) { ... }`. **Por defecto:** `false`.

**Respuesta**

JSON

```
[
    {
        "id": "913",
        "profile_image": "https://board.support/user.svg",
        "first_name": "Don",
        "last_name": "John",
        "email": "example@domain.com",
        "user_type": "user",
        "token": "9b25351047ee758aa97ee4868d130cc1ceb8decf"
    },
    "YXNkWGNSeTdtRTdDYVkxVG8wckN4YWF6V2s0Tk1mczBSVHdQbHBpOWdmejVUTTdOUUxEUENhdUVoYmROWn..."
]
```

Devuelve `false` si el inicio de sesión no es exitoso.

---

# SBF.logout()

Cierra la sesión del usuario conectado y recarga la página.

---

# SBF.loginCookie()

Crea o actualiza la cookie de inicio de sesión.

**Parámetros**

- `value`: La cadena de inicio de sesión encriptada.

# SBF.reset()

Cierra la sesión del usuario, elimina todos los datos almacenados de ZampiBot y recarga la página.

---

# SBF.lightbox()

Muestra el contenido dado en una ventana modal (lightbox).

**Parámetros**

- `content`: El contenido a mostrar. Puede ser cualquier contenido HTML; para mostrar una imagen use el código `<img src="image.jpg" >`.

---

# SBF.storage()

Gestiona el almacenamiento local (Local Storage) de ZampiBot. El almacenamiento local es una tecnología que guarda datos en el navegador de forma permanente.

**Parámetros**

- `key` **Requerido**: El ID del valor a establecer o recuperar.
- `value`: El valor a guardar. Si no se establece, la función devuelve el valor de la clave dada.

---

# SBF.storageTime()

Guarda una clave y la fecha y hora actual en el almacenamiento local para comprobar en el futuro si está dentro del número de horas dado o si ha expirado.

**Parámetros**

- `key` **Requerido**: La clave a guardar o comprobar.
- `hours`: El número de horas para comparar con la fecha y hora guardada.

**Respuesta** Si el atributo `hours` está establecido, devuelve `true` si la suma de la fecha guardada y las horas dadas es menor que la fecha y hora actual; de lo contrario, devuelve `false`. Ejemplo: si la hora guardada es 5:00 pm, e inserta 3, la función devolverá `true` solo si la hora actual es 8:01 pm o más.

---

# SBF.setting()

Devuelve el valor de una configuración, o le asigna un valor.

**Parámetros**

- `key` **Requerido**: El nombre de la configuración.
- `value`: El valor de la configuración. **Por defecto:** -1.

**Configuraciones disponibles** Algunas de las configuraciones disponibles y sus valores se encuentran en la lista a continuación.

JSON


```
{
    "registration-required": "",
    "registration-timetable": false,
    "registration-offline": false,
    "registration-link": "",
    "visitors-registration": false,
    "privacy": false,
    "popup": true,
    "popup-mobile-hidden": true,
    "welcome": false,
    "welcome-trigger": "open",
    "welcome-delay": 0,
    "follow": false,
    "follow-delay": "1000",
    "chat-manual-init": false,
    "chat-login-init": false,
    "sound": ["code": "n", "volume": 0.5, "repeat": 5],
    "header-name": true,
    "desktop-notifications": "all",
    "flash-notifications": "all",
    "push-notifications": true,
    "notifications-icon": "",
    "bot-id": "377",
    "bot-name": "Bruce Peterson",
    "bot-image": "",
    "bot-delay": 0,
    "dialogflow-office-hours": false,
```

```
"dialogflow-active": true,
"dialogflow-human-takeover": false,
"slack-active": false,
"rich-messages": [
    "email",
    "registration",
    "login",
    "timetable",
    "articles",
    "immagine",
    "video"
],
"display-users-thumb": true,
"hide-agents-thumb": true,
"notify-user-email": true,
"notify-agent-email": false,
"translations": false,
"auto-open": false,
"office-hours": true,
"disable-office-hours": false,
"disable-offline": false,
"timetable": false,
"timetable-hide": [
    false,
    "checkbox"
],
"articles": true,
"articles-title": "",
"init-dashboard": false,
"wp": false,
"wp-users-system": "sb",
"queue": false,
"queue-message": "",
"queue-message-success": "",
"queue-response-time": "",
"routing": false,
"webhooks": true,
"agents-online": false,
"timetable-message": "",
"tickets-registration-required": true,
```

```
    "tickets-registration-redirect": "",
    "tickets-default-form": "login",
    ...
}
```

---

# SBF.shortcode()

Convierte un shortcode en un array que contiene el nombre del shortcode y la configuración del mismo.

### Parámetros

- `shortcode` **Requerido**: La cadena del shortcode, ej. `[rating title="Califique su conversación" message="Cuéntenos su experiencia." success="¡Gracias!"]`.

### Respuesta

JSON

```
[
    "rating",
    {
        "title": "Califique su conversación",
        "message": "Cuéntenos su experiencia.",
        "success": "¡Gracias!"
    }
]
```

---

# SBF.openWindow()

Abre una página web en una nueva ventana.

### Parámetros

- `link` **Requerido**: El enlace a abrir.
- `width`: El ancho de la ventana en px.
- `height`: La altura de la ventana en px.

---

# SBF.loadResource()

Incluye un archivo .js o .css en el área `<head>` de la página.

**Parámetros**

- `url` **Requerido**: La URL del archivo a cargar.
- `script`: Establézcalo en `true` para archivos .js. **Por defecto:** archivos .css.

---

# SBF.debounce()

Ejecuta la función dada solo una vez por el tiempo en milisegundos dado; el temporizador se reinicia en cada llamada.

**Parámetros**

- `bounceFunction` **Requerido**: Función a ejecutar cuando la función se completa. Sintaxis: `function() { ... }`.
- `id`: Inserte una cadena única y aleatoria nunca usada por otras instancias de debounce. Las cadenas #1, #2, #3, #4, ... están reservadas y no pueden usarse.
- `interval`: El tiempo mínimo en milisegundos entre la nueva ejecución y la anterior. **Por defecto:** 500.

---

# SBF.translate()

Traduce una cadena utilizando las traducciones de ZampiBot.

**Parámetros**

- `string` **Requerido**: La cadena a traducir.

---

# SBF.escape()

Escapa una cadena (sanitización).

**Parámetros**

- `string` **Requerido**: La cadena a escapar.

---

# SBF.convertUTCDateToLocalDate()

Convierte una fecha a la hora local.

**Parámetros**

- `datetime` **Requerido**: La cadena de fecha y hora. Formato: Y/m/d H:i:s.
- `UTCoffset`: Un desplazamiento UTC. **Por defecto:** 0.

**Respuesta** `Mon Jan 30 2023 10:45:00 GMT+0000 (Greenwich Mean Time)`

---

# SBF.getLocationTimeString()

Devuelve una cadena que contiene la ubicación y la hora local actual de la zona horaria dada.

**Parámetros**

- `details` **Requerido**: Array con los detalles de la ubicación, ej. `{ "timezone" : "", "country" : "", "city" : "" }`. Esta función acepta el método `extra` de cualquier objeto `SBUser`, ej. `SBF.getLocationTimeString(activeUser().extra, function(){})`.
- `onSuccess` **Requerido**: Función a ejecutar cuando la función se completa. Sintaxis: `function(response) { ... }`.

**Respuesta** `05:15 AM in London, United Kingdom`

---

# SBF.beautifyTime()

Convierte una fecha al formato local y realiza otras optimizaciones para hacer la fecha más amigable.

**Parámetros**

- `date` **Requerido**: Fecha y hora en el siguiente formato: YYYY-MM-DD HH:MM:SS. Ej. 2020-05-13 13:35:59. Puede eliminar la hora y dejar solo la fecha. Las fechas almacenadas en la base de datos están en UTC+0.
- `extended`: Establézcalo en `true` para incluir minutos y segundos. **Por defecto:** `false`.
- `future`: Establézcalo en `true` si la fecha es una fecha futura. **Por defecto:** `false`.

**Respuesta** `<span>30/01/2023</span>`

---

# SBF.dateDB()

Convierte un formato de fecha al formato de fecha de la base de datos y establece el UTC en +0.

**Parámetros**

- `date` **Requerido**: Fecha y hora en el siguiente formato: YYYY-MM-DD HH:MM:SS. Ej. 2020-05-13 13:35:59. Puede eliminar la hora y dejar solo la fecha. Ingrese `now` para obtener la fecha y hora actual. Las fechas almacenadas en la base de datos están en UTC+0.

---

# SBF.UTC()

Devuelve la fecha y hora especificada en formato unix utilizando el desplazamiento UTC establecido en `Configuración > Misceláneas > Zona horaria`.

**Parámetros**

- `datetime` **Requerido**: La cadena de fecha y hora.

**Respuesta** `1675075500000`

---

# SBF.unix()

Obtiene el valor de la marca de tiempo (timestamp) unix de una cadena de fecha y hora con formato `yyyy-mm-dd hh:mm:ss`.

**Parámetros**

- `datetime` **Requerido**: La cadena de fecha y hora.

**Respuesta** `1675075500000`

---

# ⚡ JAVASCRIPT API: Eventos

Los eventos le permiten interceptar las acciones de ZampiBot en tiempo real y ejecutar código JavaScript personalizado cuando se dispara una acción.

# Uso

Utilice el código a continuación y reemplace `EVENT-NAME` con el nombre del evento. La `response` (respuesta) representa el valor devuelto si solo hay un atributo; de lo contrario, es un array de valores. Se requiere **JQuery**.

JavaScript

```
$(document).on("EVENT-NAME", function (e, response) {
    // Su código aquí
});
```

**Ejemplo:**

JavaScript

```
$(document).on("SBMessageSent", function (e, response) {
    console.log(response["user_id"]);
    console.log(response["conversation_id"]);
    console.log(response["message"]);
});
```

---

# SBReady

Evento disparado al cargar la página después de que el script del chat `main.js` se haya cargado. Este evento también se dispara en el área de administración.

---

# SBInit

Evento disparado al cargar la página cuando el chat ha completado la inicialización.

---

# SBTicketsInit

Evento disparado al cargar la página cuando el área de tickets ha completado la inicialización. Este evento está disponible solo cuando la App de Tickets está activa.

---

# SBLogout

Evento disparado cuando el usuario activo cierra sesión.

---

# SBError

Evento disparado cuando ocurre un error en ZampiBot.

**Respuesta**

- `message`: El mensaje de error.
- `function_name`: El nombre de la función que generó el error.

---

# SBDoubleLoginError

Evento disparado en la inicialización del chat si un agente o un administrador ya ha iniciado sesión.

---

# SBUserDeleted

Evento disparado cuando se elimina un usuario.

**Respuesta** El ID del usuario que ha sido eliminado.

---

# SBMessageSent

Evento disparado cuando se ha enviado un nuevo mensaje.

**Respuesta**

- `user_id`: El ID del usuario que envió el mensaje.
- `user`: El objeto de usuario.
- `conversation_id`: El ID de la conversación vinculada al mensaje.
- `conversation_status`: El código de estado de la conversación vinculada al mensaje. (0=activo, 1=esperando usuario, 2=esperando agente, 3=archivo, 4=papelera).
- `message_id`: El ID del mensaje.
- `message`: El texto del mensaje.
- `attachments`: Los adjuntos del mensaje.

- `conversation_source`: La fuente de la conversación. Fuentes disponibles: `em` (Email), `tk` (Ticket), `wa` (WhatsApp), `fb` (Facebook Messenger), `ig` (Instagram), `tw` (Twitter), `wc` (WeChat), `tx` (Mensaje de texto), `gb` (Google Business Messages), `ln` (LINE), `vb` (Viber).

---

# SBBotMessage

Evento disparado cuando el chatbot responde a un mensaje.

**Respuesta**

- `response`: El array de respuesta en formato JSON de Dialogflow.
- `message`: El mensaje de entrada del usuario.

---

# SBSlackMessageSent

Evento disparado cuando se envía un mensaje a Slack.

**Respuesta**

- `message`: El texto del mensaje.
- `conversation_id`: El ID de la conversación vinculada al mensaje.

---

# SBEmailSent

Evento disparado cuando se envía un correo electrónico de notificación a un usuario o un agente.

**Respuesta**

- `recipent_id`: El ID del usuario que recibirá el correo.
- `message`: El mensaje del correo.
- `attachments`: Los adjuntos del correo.

---

# SBNotificationsUpdate

Evento disparado cuando se actualiza el contador rojo de notificaciones del botón de chat que alerta al usuario de nuevos mensajes y conversaciones.

**Respuesta**

- `conversation_id`: La acción puede ser `add` si el contador aumenta en 1, o `remove` si disminuye en 1.
- `message_id`: El ID del mensaje no leído; si se establece, el contador aumenta en 1. Si es `false`, el contador disminuye en 1.

---

# SBConversationOpen

Evento disparado cuando una conversación está completamente cargada y se abre en el chat.

**Respuesta**

- `response`: La conversación como un objeto `SBConversation`.

---

# SBNewMessagesReceived

Evento disparado cuando hay nuevos mensajes en la conversación activa.

**Respuesta**

- `messages`: Array de objetos `SBMessage`.
- `conversation_id`: El ID de conversación vinculado a los mensajes.

---

# SBMessageDeleted

Evento disparado cuando se elimina un mensaje.

**Respuesta** El ID del mensaje que ha sido eliminado.

---

# SBNewConversationReceived

Evento disparado cuando se recibe una nueva conversación.

**Respuesta** La nueva conversación como un objeto `SBConversation`.

---

# SBNewConversationCreated

Evento disparado cuando se crea una nueva conversación.

**Respuesta** La nueva conversación como un objeto `SBConversation`.

---

# SBActiveConversationChanged

Evento disparado cuando se cambia la conversación activa.

**Respuesta** La nueva conversación activa como un objeto `SBConversation`.

---

# SBActiveConversationStatusUpdated

Evento disparado cuando se actualiza el código de estado de la conversación activa.

**Respuesta**

- `conversation_id`: El ID de la conversación.
- `status_code`: El nuevo código de estado de la conversación. (0=activo, 1=esperando usuario, 2=esperando agente, 3=archivo, 4=papelera).

---

# SBPopulateConversations

Evento disparado después de que se han obtenido todas las conversaciones del usuario.

**Respuesta**

- `conversations`: Array de conversaciones como objetos `SBConversation`.

---

# SBChatOpen

Evento disparado cuando se abre el chat.

---

# SBChatClose

Evento disparado cuando se cierra el chat.

# SBQueueUpdate

Evento disparado cuando se inicia la cola y cada vez que la cola se actualiza.

**Respuesta** La posición del usuario en la cola. Si la posición es 0, la cola ha terminado y el usuario puede iniciar el chat.

---

# SBBusy

Evento disparado cuando cambia el estado de "ocupado" del chat. El chat está ocupado si está operando, como enviando un mensaje. Cuando el chat está ocupado, algunas funciones no se pueden disparar, como enviar un nuevo mensaje.

**Respuesta** Devuelve `true` si el chat está ocupado, de lo contrario `false`.

---

# SBDashboard

Evento disparado cuando se muestra el panel de control (dashboard).

---

# SBDashboardClosed

Evento disparado cuando se cierra el panel de control y se muestra una conversación en su lugar.

---

# SBTyping

Evento disparado cuando un usuario o un agente está escribiendo en el editor.

**Respuesta** Devuelve `true` si el usuario o agente está escribiendo, de lo contrario devuelve `false`.

---

# SBArticles

Evento disparado cuando se muestra el panel de artículos o cuando se abre un solo artículo.

**Respuesta**

- `id`: El ID del artículo. Este valor es `-1` si se muestra el panel de artículos.
- `articles`: Puede ser el array con los detalles del artículo individual o el array con la lista de todos los artículos.

---

# SBPrivacy

Evento disparado en la inicialización del chat si se muestra el formulario de privacidad.

---

# SBPopup

Evento disparado cuando se muestra un mensaje emergente (pop-up).

**Respuesta** El array con el contenido del pop-up: `{ image: "", title: "", message: "" }`.

---

# SBFollowUp

Evento disparado cuando se envía el mensaje de seguimiento.

---

# SBWelcomeMessage

Evento disparado cuando se envía el mensaje de bienvenida.

---

# SBLoginForm

Evento disparado cuando el usuario inicia sesión correctamente desde el formulario de inicio de sesión del chat. Este evento se dispara solo si el inicio de sesión es exitoso.

**Respuesta** El usuario como un objeto `SBUser`.

---

# SBRegistrationForm

Evento disparado cuando el usuario se registra correctamente desde el formulario de registro del chat. Este evento se dispara solo si el registro es exitoso. También se dispara si el registro se actualiza a través del formulario de Mensaje Enriquecido.

**Respuesta**

- `user`: Array con los detalles del usuario.
- `extra`: Array con los detalles adicionales del usuario.

---

# SBRichMessageShown

Evento disparado cuando se muestra un Mensaje Enriquecido (Rich Message) cargado asíncronamente.

**Respuesta**

- `name`: El nombre del Mensaje Enriquecido.
- `settings`: La configuración del Mensaje Enriquecido.
- `response`: El código HTML del Mensaje Enriquecido.

---

# SBRichMessageSubmit

Evento disparado cuando se recibe la respuesta de un Mensaje Enriquecido.

**Respuesta**

- `result`: La respuesta del Mensaje Enriquecido.
- `data`: Los detalles del Mensaje Enriquecido y los datos enviados por el usuario.
- `id`: El ID del Mensaje Enriquecido.

---

# SBAttachments

Evento disparado cuando el usuario adjunta un archivo.

---

# SBNewEmailAddress

Evento disparado cuando un usuario registra su correo electrónico a través del mensaje de seguimiento o el formulario de registro.

**Respuesta**

- `name`: El nombre completo del usuario.
- `email`: El correo electrónico del usuario.
- `ID`: El ID del mensaje enriquecido.

---

# SBPanelActive

Evento disparado cuando se abre un panel del área de Tickets. Este evento está disponible solo cuando la App de Tickets está activa.

**Respuesta**

- `name`: El nombre del panel activo. Valores: `new-ticket`, `privacy`, `articles`, `edit-profile`, `login`, `registration`. Si el valor está vacío, el área de conversación principal está activa.
- `email`: El correo electrónico del usuario.

---

# SBBotPayload

Evento disparado cuando un mensaje de Dialogflow contiene un Payload de ZampiBot. Ejemplos de payload: `human-takeover`, `redirect`, `woocommerce-update-cart`, `woocommerce-checkout`.

**Respuesta** El nombre del payload.

---

# SBBotAction

Evento disparado cuando un mensaje de Dialogflow contiene una acción de Dialogflow. Actualmente solo está disponible la acción `end` (Finalizar conversación).

**Respuesta** El nombre de la acción.

---

# SBSMSSent

Evento disparado cuando se envía una notificación por mensaje de texto a un usuario o un agente.

**Respuesta**

- `recipent_id`: El ID del usuario que recibirá el mensaje de texto.

- `message`: El mensaje de texto.
- `response`: La respuesta de Twilio.

---

# SBActiveUserLoaded

Evento disparado cuando el usuario activo ha sido cargado.

**Respuesta**

- `recipent_id`: El ID del usuario que recibirá el mensaje de texto.
- `message`: El mensaje de texto.
- `response`: La respuesta de Twilio.

---

# SBOpenAIMessage

Evento disparado cuando OpenAI (ChatGPT) devuelve una respuesta.

**Respuesta**

- `response`: La respuesta de OpenAI.
- `message`: El mensaje de texto.
- `response`: La respuesta de Twilio.

---

# SBGetUser

Evento disparado cuando los detalles de un usuario son obtenidos de la base de datos.

**Respuesta** El usuario como un objeto `SBUser`.

---

# SBSettingsSaved

Evento disparado cuando se guardan las configuraciones del área de administración.

**Respuesta**

- `settings`: Las configuraciones.
- `external_settings`: Configuraciones guardadas en una entrada dedicada de la base de datos.

- `external_settings_translations`: Traducciones de configuraciones externas.

---

# □□ MISCELÁNEOS: AJAX

## Funciones AJAX

Lista de funciones AJAX. Las funciones AJAX son similares a la WEB API: utilizan el mismo nombre de función, los mismos parámetros y devuelven las mismas respuestas.

Utilice la función a continuación para iniciar una llamada AJAX:

JavaScript

```
SBF.ajax({
    function: 'NOMBRE-DE-LA-FUNCION',
    parameter: value,
    parameter: value,
    ...
}, (response) => {
   // Su código va aquí
});
```

Reemplace `NOMBRE-DE-LA-FUNCION` con el nombre de una de las funciones a continuación. Reemplace la lista de `parameter: value` con los parámetros de la función. Los parámetros y respuestas son los mismos que en la WEB API. La respuesta está en formato JSON.

**¡Advertencia!** No incluya el token en los parámetros. El token debe mantenerse siempre en secreto.

---

## Lista de Funciones Disponibles

*Las funciones marcadas como "Función de Admin" requieren permisos especiales.*

- **upload-path**: Función de Admin (requiere admin).
- **update-bot**: Función de Admin (requiere admin).
- **get-last-message**: Función de Admin (requiere admin).
- **delete-attachments**: Función de Admin (requiere admin).

- **execute-bot-message**: Si es usuario final, solo conversaciones propias. Si es agente/admin, cualquier conversación.
- **logs**: Función de Admin (requiere agente o admin).
- **newsletter**: Función de Admin (requiere agente o admin).
- **get-last-agent-in-conversation**: Función de Admin (requiere agente o admin).
- **messaging-platforms-send-message**: Función de Admin (requiere agente o admin).
- **aws-s3**: Si es usuario final, solo su ID. Si es agente/admin, cualquier usuario.
- **is-allowed-extension**: Si es usuario final, solo su ID. Si es agente/admin, cualquier usuario.
- **translate-string**: Si es usuario final, solo su ID. Si es agente/admin, cualquier usuario.
- **saved-replies**
- **get-settings**: Función de Admin (requiere admin).
- **save-settings**: Función de Admin (requiere admin).
- **get-multi-setting**: Función de Admin (requiere admin).
- **save-external-setting**: Función de Admin (requiere admin).
- **get-external-setting**: Función de Admin (requiere admin).
- **export-settings**: Función de Admin (requiere admin).
- **import-settings**: Función de Admin (requiere admin).
- **is-online**: Función de Admin (requiere agente o admin).
- **add-user**: Los atributos principales van en `settings` y los extra en `extra_settings`.
- **add-user-and-login**
- **get-user**: Restringido al propio usuario si no es admin/agente.
- **get-user-by**: Función de Admin (requiere admin).
- **get-users**: Función de Admin (requiere agente o admin).
- **get-new-users**: Función de Admin (requiere agente o admin).
- **get-user-extra**: Restringido al propio usuario si no es admin/agente.
- **get-user-language**: Función de Admin (requiere agente o admin).
- **get-online-users**: Función de Admin (requiere agente o admin).
- **get-user-from-conversation**: Función de Admin (requiere agente o admin).
- **agents-online**
- **search-users**: Función de Admin (requiere agente o admin).
- **delete-user**: Función de Admin (requiere admin).
- **delete-users**: Función de Admin (requiere admin).
- **update-user**: Los atributos principales van en `settings` y los extra en `extra_settings`. Restringido al propio ID si no es admin/agente.
- **update-user-to-lead**: Restringido al propio ID si no es admin/agente.
- **count-users**: Función de Admin (requiere agente o admin).
- **get-conversations**: Función de Admin (requiere agente o admin). Argumento `routing` disponible.
- **get-new-conversations**: Función de Admin (requiere agente o admin). Argumento `queue` disponible.
- **get-conversation**: Restringido a conversaciones propias si no es admin/agente.
- **get-user-conversations**: Restringido a propias si no es admin/agente.
- **get-new-user-conversations**: Restringido a propias si no es admin/agente.
- **search-conversations**: Función de Admin (requiere agente o admin).

- **search-user-conversations**: Restringido a propias si no es admin/agente. `user_id` no requerido (por defecto usuario activo).
- **new-conversation**: Restringido a propias si no es admin/agente.
- **update-conversation-status**
- **update-conversation-department**: Restringido a propias si no es admin/agente.
- **update-conversation-agent**: Restringido a propias si no es admin/agente.
- **get-new-messages**: Restringido a propias si no es admin/agente.
- **send-message**: Restringido a propias si no es admin/agente.
- **send-email**: Si es usuario final, solo a agentes/admins. Si es agente/admin, a cualquiera.
- **send-slack-message**: Restringido a propias si no es admin/agente.
- **update-message**: Restringido a mensajes propios si no es admin/agente.
- **update-messages-status**: Restringido a mensajes propios si no es admin/agente.
- **delete-message**: Restringido a mensajes propios si no es admin/agente.
- **slack-users**: Función de Admin (requiere agente o admin).
- **archive-slack-channels**: Función de Admin (requiere agente o admin).
- **slack-channels**: Función de Admin (requiere admin).
- **current-url**
- **set-rating**: Restringido a conversaciones propias si no es admin/agente.
- **get-rating**: Función de Admin (requiere agente o admin).
- **get-articles**
- **save-article**: Función de Admin (requiere agente o admin).
- **search-articles**
- **get-articles-categories**: Función de Admin (requiere agente o admin).
- **save-articles-categories**: Función de Admin (requiere agente o admin).
- **article-ratings**
- **get-versions**
- **update**: Función de Admin (requiere agente o admin).
- **app-get-key**: Función de Admin (requiere admin).
- **app-activation**: Función de Admin (requiere admin).
- **csv-users**: Función de Admin (requiere admin o agente).
- **transcript**: Restringido a conversaciones propias si no es admin/agente.
- **is-agent-typing**: Función de Admin (requiere admin o agente).
- **dialogflow-message**: Funciona solo para el usuario activo.
- **dialogflow-get-intents**: Función de Admin (requiere admin).
- **dialogflow-create-intent**: Función de Admin (requiere admin).
- **dialogflow-update-intent**: Función de Admin (requiere admin).
- **dialogflow-entity**: Función de Admin (requiere admin).
- **dialogflow-get-entity**: Función de Admin (requiere admin o agente).
- **google-get-token**
- **dialogflow-get-agent**: Función de Admin (requiere admin o agente).
- **dialogflow-set-active-context**: Restringido a usuario activo si no es admin/agente.
- **dialogflow-curl**: Función de Admin (requiere admin).
- **dialogflow-human-takeover**: Restringido a usuario activo si no es admin/agente.
- **dialogflow-smart-reply**: Función de Admin (requiere admin o agente).
- **cron-jobs**
- **woocommerce-get-customer**: Función de Admin (requiere admin o agente).

- **woocommerce-get-user-orders**: Función de Admin (requiere admin o agente).
- **woocommerce-get-order**: Función de Admin (requiere admin o agente).
- **woocommerce-get-product**: Función de Admin (requiere admin o agente).
- **woocommerce-get-products**: Función de Admin (requiere admin o agente).
- **woocommerce-search-products**: Función de Admin (requiere admin o agente).
- **woocommerce-get-taxonomies**: Función de Admin (requiere admin).
- **woocommerce-get-attributes**: Función de Admin (requiere admin).
- **woocommerce-get-product-id-by-name**: Función de Admin (requiere admin o agente).
- **woocommerce-get-product-images**: Función de Admin (requiere admin o agente).
- **woocommerce-get-product-taxonomies**: Función de Admin (requiere admin o agente).
- **woocommerce-get-attribute-by-term**: Función de Admin (requiere admin o agente).
- **woocommerce-get-attribute-by-name**: Función de Admin (requiere admin o agente).
- **woocommerce-is-in-stock**: Función de Admin (requiere admin o agente).
- **woocommerce-coupon**: Función de Admin (requiere admin o agente).
- **woocommerce-coupon-check**: Función de Admin (requiere admin o agente).
- **woocommerce-coupon-delete-expired**: Función de Admin (requiere admin o agente).
- **woocommerce-get-url**: Función de Admin (requiere admin o agente).
- **woocommerce-get-session**: Función de Admin (requiere admin).
- **woocommerce-get-session-key**: Función de Admin (requiere admin o agente).
- **woocommerce-payment-methods**: Función de Admin (requiere admin o agente).
- **woocommerce-shipping-locations**: Función de Admin (requiere admin o agente).
- **chat-css**
- **get-avatar**: Restringido a usuario activo si no es admin/agente.
- **get-agents-ids**: Función de Admin (requiere admin o agente).
- **get-users-with-details**: Función de Admin (requiere admin o agente).
- **send-custom-email**: Función de Admin (requiere admin o agente).
- **send-sms**: Restringido a usuario activo si no es admin/agente.
- **email-piping**
- **get-notes**: Función de Admin (requiere admin o agente).
- **add-note**: Función de Admin (requiere admin o agente).
- **delete-note**: Función de Admin (requiere admin o agente).
- **automations-get**
- **automations-save**: Función de Admin (requiere admin).
- **automations-validate**
- **automations-run-all**
- **automations-run**
- **automations-is-sent**: Función de Admin (requiere agente o admin).
- **get-agents-in-conversation**: Restringido a conversaciones propias si no es admin/agente.
- **get-departments**
- **login**
- **logout**
- **update-login**: Restringido a conversaciones propias si no es admin/agente.
- **is-typing**
- **set-typing**

- **clean-data**: Función de Admin (requiere admin o agente).
- **get-translation**
- **get-translations**
- **save-translations**: Función de Admin (requiere agente o admin).
- **delete-leads**: Función de Admin (requiere admin).
- **reports**: Función de Admin (requiere admin).
- **reports-update**: Función de Admin (requiere admin).
- **direct-message**: Función de Admin (requiere agente o admin).
- **count-conversations**: Función de Admin (requiere agente o admin).
- **updates-available**: Función de Admin (requiere admin).
- **google-translate**: Función de Admin (requiere admin o agente).
- **google-language-detection-update-user**: Restringido a conversaciones propias si no es admin/agente.
- **check-conversations-assignment**: Función de Admin (requiere admin o agente).

---

# Más funciones AJAX

Las siguientes funciones están disponibles solo a través de AJAX.

# close-message

Envía el mensaje de cierre a una conversación. Puede establecer el mensaje de cierre en el área `Configuración > Chat`.

**Requisitos** Esta es una función de administración y funciona solo si el usuario activo es un **agente**.

**Parámetros**

- `function` **Requerido**: Inserte `close-message`.
- `bot_id` **Requerido**: El ID del bot. Use la función `SBF.setting("bot-id")` para obtener el ID del bot. Si está en el área de administración use `SB_ADMIN_SETTINGS['bot-id']` en su lugar.
- `conversation_id` **Requerido**: El ID de la conversación a la cual enviar el mensaje.

**Respuesta**

JSON

```
{
    "success": true,
    "response": {
```

```
        "status": "success",
        "message-id": 123456,
        "queue": false
    }
}
```

---

# update-user-and-message

Actualiza los detalles de un usuario y el contenido de un mensaje.

**Requisitos** Si el usuario activo es un **usuario**, solo se puede actualizar el usuario activo y solo los mensajes vinculados a las conversaciones del usuario. Si el usuario activo es un **agente** o **admin**, la función funciona para cualquier usuario y cualquier mensaje.

**Parámetros**

- `function` **Requerido**: Ingrese `update-user-and-message`.
- `user_id` **Requerido**: El ID del usuario a actualizar.
- `settings`: Array con los detalles del usuario. Ej: `{ first_name: "Don", last_name: "John", profile_image: "image.jpg", email: "example@gmail.com", user_type: "admin" }`.
- `settings_extra`: Array con detalles adicionales del usuario. Sintaxis del array: `{ "ID": ["valor", "Nombre"], ...}`.
- `message_id`: El ID del mensaje a actualizar.
- `message`: El texto del mensaje.
- `payload`: Array asociativo conteniendo datos extra, ej. `{"rich-messages":{"123":{"type":"buttons","result":"Premium Plan"}}}`.

**Respuesta** `true`

---

# get-agent

Devuelve los detalles de un agente, administrador o bot.

**Parámetros**

- `agent_id` **Requerido**: El ID del agente.

**Respuesta**

JSON

```
{
    "id": "2",
```

```
    "first_name": "Don",
    "last_name": "John",
    "department": null,
    "flag": "gb.png",
    "country_code": "GB",
    "details": [
        {
            "slug": "city",
            "name": "City",
            "value": "London"
        },
        {
            "slug": "country",
            "name": "Country",
            "value": "United Kingdom"
        },
        {
            "slug": "sport",
            "name": "Sport",
            "value": "email@example.com"
        },
        {
            "slug": "timezone",
            "name": "Timezone",
            "value": "Europe/London"
        }
        ...
    ]
}
```


---

# user-autodata

Obtiene los siguientes detalles sobre el usuario activo y actualiza los detalles del usuario: IP, ciudad, ubicación, país, zona horaria, moneda, navegador, idioma del navegador, sistema operativo.

**Parámetros**

- `user_id` **Requerido**: El ID del usuario.

**Respuesta** `true`

# get-agent-department

Devuelve el departamento del agente o administrador activo.

**Respuesta** Devuelve el ID del departamento si está establecido; de lo contrario, devuelve `false`.

Created 2026-01-11 09:29:06 UTC by ZampiSoft
Updated 2026-01-11 10:45:32 UTC by ZampiSoft