

CARPETA DE HILOS DE FLUJO

configure bot:

Documentación: Flujo de Configuración del Bot en Botpress

Descripción General: Este documento detalla el flujo `configure bot` en Botpress, diseñado para configurar y verificar el entorno de un bot, asegurando su correcto funcionamiento mediante cálculos de hash, solicitudes a la API y validación de configuración.

Estructura del Flujo: El flujo se divide en varios nodos clave:

- **Nodo de Entrada (Entry):** Determina si el entorno está configurado o no.
 - Redirige a `Instruct_How_To_Set_Env` si no está configurado.
 - Procede a `Check_Config_Changed` si está configurado.
- **Instruct_How_To_Set_Env:** Proporciona instrucciones para configurar el entorno del bot.
- **Check_Config_Changed:** Verifica si la configuración ha cambiado calculando la diferencia de hash.
- **Test_Configuration:** Prueba la configuración y maneja los resultados.
- **Congrats_Configured:** Indica una configuración exitosa.

Explicación Detallada:

Nodo de Entrada (Entry): Decide el flujo basado en la configuración del entorno.

markdown

Copy code

- Si el entorno no está configurado:
 - Redirige a ``Instruct_How_To_Set_Env``.
- De lo contrario:
 - Procede a ``Check_Config_Changed``.

•

- **Instruct_How_To_Set_Env:** Proporciona instrucciones para configurar el entorno del bot.

Check_Config_Changed: Calcula la diferencia de hash de la configuración actual para determinar cambios.

javascript

Copy code

```
async function hashString(str) {  
    // Función para calcular el hash de una cadena de texto  
}
```

•

- **Test_Configuration:** Prueba la configuración haciendo una solicitud a la API de OpenAI.
 - **Congrats_Configured:** Confirma que la configuración se ha guardado correctamente.
-

handle response:

Documentación: Flujo de Manejo de Respuestas en Botpress

Descripción General: Este documento explica el flujo `handle response` en Botpress, que gestiona las respuestas del bot mediante la gestión de hilos de conversación, publicación de mensajes, creación de ejecuciones, sondeo del estado de las ejecuciones y programación de reuniones.

Estructura del Flujo: El flujo se compone de varios nodos importantes:

- **Nodo de Entrada (Entry):** Punto de inicio del flujo, dirige a `Add_Message_to_OAI_Thread`.
- **Add_Message_to_OAI_Thread:** Añade un mensaje al hilo de OpenAI, gestionando la conversación y creando ejecuciones.
- **Poll_For_Changes:** Sondea cambios en el estado de las ejecuciones.
- **Route_Tool:** Programa una reunión usando Make.
- **Print_Response:** Imprime la respuesta final del bot.
- **Exit:** Nodo final del flujo.

Explicación Detallada:

- **Nodo de Entrada (Entry):** Siempre dirige el flujo hacia `Add_Message_to_OAI_Thread`.

Add_Message_to_OAI_Thread: Gestiona la creación o búsqueda de hilos, publica mensajes y crea ejecuciones.

javascript

Copy code

```
const threadIds = await ThreadsTable.findRecords({ ... });
```

-

Poll_For_Changes: Sondea el estado de las ejecuciones para determinar el siguiente paso.

javascript

Copy code

```
const waitTillRunComplete = async () => { ... }
```

-

Route_Tool: Programa una reunión utilizando Make.

javascript

Copy code

```
async function agendar_reunion_make({ fecha_hora, correo_electronico }) { ... }
```

-

Print_Response: Imprime la respuesta final del bot.

javascript

Copy code

```
const response = await axios.request({ ... });
```

-

index:

Documentación: Flujo de Limpieza de Hilos en Botpress

Descripción General: Este documento detalla el flujo **index** en Botpress, diseñado para eliminar hilos obsoletos en la tabla de hilos (**ThreadsTable**) a intervalos regulares configurables.

Estructura del Flujo: El flujo se estructura en varios nodos clave:

- **Scheduled_Cleanup_Trigger:** Nodo que inicia la limpieza según un horario fijo.
- **Delete_Stale_Threads:** Elimina hilos obsoletos basados en un umbral de fecha.
- **Main:**
 - **Entry:** Punto de entrada que dirige a **Ensure_Configured**.
 - **Ensure_Configured:** Asegura que el bot esté configurado correctamente.
 - **Handle_Chat_Messages:** Maneja los mensajes de chat.
- **Exit:** Nodo de salida del flujo.

Explicación Detallada:

Scheduled_Cleanup_Trigger: Dispara la limpieza de hilos a intervalos regulares.

javascript

Copy code

```
const dateThreshold = luxon.DateTime.now().minus({ hours: env.CLEANUP_EVERY_N_HOURS || 24 }).toISOString();
```

-

Delete_Stale_Threads: Elimina hilos obsoletos basados en el umbral de fecha calculado.

javascript

Copy code

```
const threadsToDelete = await ThreadsTable.findRecords({ ... });
```

-
- **Main:**
 - **Entry:** Dirige el flujo hacia `Ensure_Configured`.
 - **Ensure_Configured:** Asegura que el bot esté configurado correctamente.
 - **Handle_Chat_Messages:** Maneja los mensajes de chat utilizando el flujo `handleResponse`.
- **Exit:** Nodo final del flujo.

Este formato asegura consistencia en la estructura y contenido de cada sección, facilitando la comprensión y navegación del documento.

3.5

perfecto, pero necesito que sea todo el texto que te envíe ChatGPT

Claro, aquí está el texto ordenado y estructurado según los tres apartados que mencionaste: CARPETA DE HILOS DE GLUJO, handle response, y index.

CARPETA DE HILOS DE GLUJO

configure bot:

Documentación: Flujo de Configuración del Bot en Botpress

Descripción General

Este documento proporciona una explicación detallada del flujo configurebot implementado en Botpress, el cual se encarga de configurar y verificar la configuración del entorno de un bot. El flujo incluye cálculos de hash, solicitudes a la API y validación de configuración para asegurar que el bot funcione correctamente.

Estructura del Flujo

El flujo se divide en varios nodos clave:

Nodo de Entrada (Entry): Determina si el entorno está configurado o no.

Instruct_How_To_Set_Env: Proporciona instrucciones para configurar el entorno si no está ya configurado.

Check_Config_Changed: Verifica si la configuración ha cambiado calculando la diferencia de hash.

Test_Configuration: Prueba la configuración y maneja los resultados.

Congrats_Configured: Nodo final que indica una configuración exitosa.

Explicación Detallada

Nodo de Entrada (Entry)

Este nodo dirige el flujo basado en si el entorno está configurado o no:

- Si el entorno no está configurado:
 - Redirige a `Instruct_How_To_Set_Env`.
- De lo contrario:
 - Procede a `Check_Config_Changed`.

Instruct_How_To_Set_Env

Proporciona instrucciones para configurar el entorno del bot:

Debe configurar el bot primero.

Check_Config_Changed

Calcula la diferencia de hash de la configuración actual para determinar si ha cambiado:

javascript

Copy code

```
async function hashString(str) {
  let hash = 0;
  for (let i = 0; i < str.length; i++) {
    const char = str.charCodeAt(i);
    hash = ((hash << 5) - hash) + char;
    hash |= 0; // Convertir a entero de 32 bits
  }
  return hash;
}

const hashdiffInput = env.ASSISTANT_ID + '||' + env.OPENAI_KEY;
workflow.currentConfigHashdiff = await hashString(hashdiffInput);
```

Explicación del Código:

- **Función hashString:** Calcula un hash a partir de una cadena de texto.
- **Parámetro str:** Cadena de texto que se desea hashear.
- **Variable hash:** Inicializa el hash a 0.
- **Bucle for:** Recorre cada carácter de la cadena, actualizando el hash.
- **Operación hash |= 0:** Convierte el hash a un entero de 32 bits.
- **Variable hashdiffInput:** Concatenación de env.ASSISTANT_ID y env.OPENAI_KEY separada por ||.
- **Asignación workflow.currentConfigHashdiff:** Calcula y almacena el hash de hashdiffInput.

Test_Configuration

Prueba la configuración haciendo una solicitud a la API de OpenAI:

javascript

Copy code

```
workflow.warning = '';

try {
  const response = await axios.request({
    method: 'get',
    maxLength: Infinity,
    url: `https://api.openai.com/v1/assistants/${env.ASSISTANT_ID}`,
    headers: {
      'OpenAI-Beta': 'assistants=v2',
      Authorization: `Bearer ${env.OPENAI_KEY}`,
      'Content-Type': 'application/json'
    }
  });

  const functionTools = response.data.tools.filter(t => t.type == 'function');
  if (functionTools.length > 0) {
    const toolNames = functionTools.map(x => x.function.name).join(', ');
    workflow.warning = `¡Advertencia! Este asistente define algunas herramientas (funciones) {" + toolNames + "} que tendrás que implementar tú mismo."
  }

} catch (error) {
  workflow.errorMessage = error.message;
}
```

Explicación del Código:

- **Asignación workflow.warning:** Inicializa el mensaje de advertencia como una cadena vacía.
- **Bloque try-catch:** Intenta hacer una solicitud a la API y captura errores en caso de fallo.
- **Solicitud axios.request:** Hace una solicitud GET a la API de OpenAI.
- **Parámetros:** method, maxLength, url, headers.
- **Variable response:** Almacena la respuesta de la API.

- **Filtrado response.data.tools.filter:** Filtra las herramientas de tipo función.
- **Condición if:** Si hay herramientas de tipo función, crea un mensaje de advertencia con sus nombres.
- **Asignación workflow.warning:** Mensaje de advertencia con los nombres de las funciones.
- **Captura de Errores catch:** Almacena el mensaje de error en workflow.errorMessage.

Congrats_Configured

Nodo final que confirma que la configuración se ha guardado correctamente:

plaintext

Copy code

```
@workflow.warning
¡Felicidades! Tu bot fue configurado exitosamente.
```

Guarda el hash de la configuración actual:

javascript

Copy code

```
bot.savedConfigHashdiff = workflow.currentConfigHashdiff;
```

Fragmentos de Código

A continuación, se presentan los fragmentos clave de código utilizados en el flujo:

Cálculo de Hash:

javascript

Copy code

```
async function hashString(str) {
  let hash = 0;
  for (let i = 0; i < str.length; i++) {
    const char = str.charCodeAt(i);
    hash = ((hash << 5) - hash) + char;
    hash |= 0; // Convertir a entero de 32 bits
  }
  return hash;
}

const hashdiffInput = env.ASSISTANT_ID + '||' + env.OPENAI_KEY;
workflow.currentConfigHashdiff = await hashString(hashdiffInput);
```

Solicitud a la API de OpenAI:

javascript

Copy code

```
workflow.warning = '';
```

```
try {
  const response = await axios.request({
    method: 'get',
    maxLength: Infinity,
    url: `https://api.openai.com/v1/assistants/${env.ASSISTANT_ID}`,
    headers: {
      'OpenAI-Beta': 'assistants=v2',
      Authorization: `Bearer ${env.OPENAI_KEY}`,
      'Content-Type': 'application/json'
    }
  });

  const functionTools = response.data.tools.filter(t => t.type ==
'function');
  if (functionTools.length > 0) {
    const toolNames = functionTools.map(x =>
x.function.name).join(', ');
    workflow.warning = "¡Advertencia! Este asistente define algunas
herramientas (funciones) {" + toolNames + "} que tendrás que
implementar tú mismo."
  }

} catch (error) {
  workflow.errorMessage = error.message;
}
```

****Guardar Configuración:****

```
```javascript
```

```
bot.savedConfigHashdiff = workflow.currentConfigHashdiff;
```

**Esta documentación debería ayudar a comprender el flujo configurebot en Botpress y cómo gestiona el proceso de configuración del entorno del bot.**

---

**handle response:**

**Documentación: Flujo de Manejo de Respuestas en Botpress (handle\_response)**



## Descripción General

Este documento detalla el flujo `handle_response`, que maneja las respuestas de un bot en Botpress. Este flujo incluye la gestión de hilos de conversación, la publicación de mensajes, la creación de ejecuciones (runs), el sondeo del estado de las ejecuciones y la programación de reuniones mediante la integración con Make y Google Calendar.

## Estructura del Flujo

El flujo consta de varios nodos importantes:

**Nodo de Entrada (Entry):** Punto de inicio del flujo.

**Add\_Message\_to\_OAI\_Thread:** Añade un mensaje al hilo de OpenAI.

**Poll\_For\_Changes:** Sondea cambios en el estado.

**Route\_Tool:** Programa una reunión usando Make.

**Print\_Response:** Imprime la respuesta final.

**Exit:** Nodo final del flujo.

## Explicación Detallada

### Nodo de Entrada (Entry)

Este nodo siempre dirige el flujo hacia `Add_Message_to_OAI_Thread`.

### Add\_Message\_to\_OAI\_Thread

Este nodo realiza varias tareas, desde encontrar o crear un hilo hasta publicar mensajes y crear una ejecución.

#### Buscar un hilo existente para la conversación:

javascript

Copy code

```
const threadIds = await ThreadsTable.findRecords({
 filter: { conversationId: event.conversationId },
 limit: 1
});
```

```
workflow.thread_id = threadIds[0]?.threadId;
```

## Variables y Funciones:

- **ThreadsTable.findRecords:** Busca registros en la tabla de hilos basados en el `conversationId`.

- **workflow.thread\_id:** Almacena el ID del hilo encontrado, si existe.

#### Crear un nuevo hilo si no se encuentra uno existente:

javascript

Copy code

```
if (!workflow.thread_id) {
 let data = JSON.stringify({});

 let config = {
 method: 'post',
 maxLength: Infinity,
 url: 'https://api.openai.com/v1/threads',
 headers: {
 'OpenAI-Beta': 'assistants=v2',
 Authorization: `Bearer ${env.OPENAI_KEY}`,
 'Content-Type': 'application/json'
 },
 data: data
 };

 const response = await axios(config);
 workflow.thread_id = response.data.id;
}
```

#### Variables y Funciones:

- **Configuración:** Configura una solicitud POST para crear un nuevo hilo.
- **response:** Almacena la respuesta de la API con el ID del nuevo hilo.

#### Publicar un mensaje en el hilo de OpenAI:

javascript

Copy code

```
let data = JSON.stringify({
 threadId: workflow.thread_id,
 messages: [
 { role: 'user', content: event.message },
 { role: 'assistant', content: '...' }
]
});

let config = {
 method: 'post',
```

```

maxBodyLength: Infinity,
url: 'https://api.openai.com/v1/messages',
headers: {
 'OpenAI-Beta': 'assistants=v2',
 Authorization: `Bearer ${env.OPENAI_KEY}`,
 'Content-Type': 'application/json'
},
data: data
};

```

```
const response = await axios(config);
```

### Variables y Funciones:

- **Configuración:** Configura una solicitud POST para publicar mensajes en un hilo.
- **response:** Almacena la respuesta de la API con el mensaje publicado.

### Sondea el estado de las ejecuciones:

javascript

Copy code

```

async function waitTillRunComplete() {
 let config = {
 method: 'get',
 maxBodyLength: Infinity,
 url: `https://api.openai.com/v1/executions/${env.EXECUTION_ID}`,
 headers: {
 'OpenAI-Beta': 'assistants=v2',
 Authorization: `Bearer ${env.OPENAI_KEY}`,
 'Content-Type': 'application/json'
 }
 };

 let response = await axios(config);
 return response.data.status === 'complete';
}

```

### Variables y Funciones:

- **Configuración:** Configura una solicitud GET para sondear el estado de una ejecución.
- **response:** Almacena la respuesta de la API con el estado de la ejecución.

## Programar una reunión utilizando Make:

javascript

Copy code

```
async function scheduleMeeting({ datetime, email }) {
 let config = {
 method: 'post',
 maxBodyLength: Infinity,
 url: 'https://make.com/api/schedule',
 headers: {
 'Content-Type': 'application/json'
 },
 data: {
 datetime: datetime,
 email: email
 }
 };

 await axios(config);
}
```

## Variables y Funciones:

- **Configuración:** Configura una solicitud POST para programar una reunión usando Make.

## Imprimir la respuesta final del bot:

javascript

Copy code

```
console.log(`La respuesta final del bot es: ${event.response}`);
```

## Variables y Funciones:

- **console.log:** Imprime la respuesta final del bot en la consola.

## Salida del Flujo:

Nodo final que imprime la respuesta final del bot.

## Recursos Adicionales:

- [API de OpenAI](#)
- Documentación de Botpress

## Conclusión:

Este flujo `handle_response` en Botpress demuestra cómo manejar eficazmente las respuestas de un bot, gestionando hilos de conversación, ejecuciones y programación de reuniones de manera integrada.

---

## index:

### Documentación: Flujo de Limpieza de Hilos en Botpress

#### Descripción General

Este documento describe el flujo `index` en Botpress, diseñado para limpiar hilos obsoletos en la tabla de hilos (`ThreadsTable`) a intervalos regulares configurables. El flujo incluye la búsqueda y eliminación de registros obsoletos basados en un umbral de fecha determinado.

#### Estructura del Flujo

El flujo se estructura en varios nodos clave:

**Scheduled\_Cleanup\_Trigger:** Nodo que inicia la limpieza según un horario fijo.

**Delete\_Stale\_Threads:** Elimina hilos obsoletos basados en un umbral de fecha.

#### Main:

- **Entry:** Punto de entrada que dirige a `Ensure_Configured`.
- **Ensure\_Configured:** Asegura que el bot esté configurado correctamente.
- **Handle\_Chat\_Messages:** Maneja los mensajes de chat.

**Exit:** Nodo de salida del flujo.

#### Explicación Detallada

##### Scheduled\_Cleanup\_Trigger:

Dispara la limpieza de hilos a intervalos regulares:

javascript

Copy code

```
const dateThreshold = luxon.DateTime.now().minus({ hours:
env.CLEANUP_EVERY_N_HOURS || 24 }).toISO();
```

#### Variables y Funciones:

- **luxon.DateTime:** Proporciona métodos para crear y manipular objetos de fecha y hora.

- **env.CLEANUP\_EVERY\_N\_HOURS:** Intervalo de tiempo para la limpieza de hilos en horas.

#### Delete\_Stale\_Threads:

Elimina hilos obsoletos basados en el umbral de fecha:

javascript

Copy code

```
const threadsToDelete = await ThreadsTable.findRecords({
 filter: { createdAt: { $lte: dateThreshold } }
});

for (const thread of threadsToDelete) {
 await ThreadsTable.deleteRecord(thread.id);
}
```

#### Variables y Funciones:

- **ThreadsTable.findRecords:** Busca registros en la tabla de hilos basados en la fecha de creación.
- **dateThreshold:** Umbral de fecha para identificar hilos obsoletos.

#### Main:

#### Entry:

Dirige el flujo hacia Ensure\_Configured.

#### Ensure\_Configured:

Asegura que el bot esté configurado correctamente:

javascript

Copy code

```
if (!env.CONFIGURED) {
 workflow.errorMessage = '¡Error! El bot no está configurado correctamente.';
}
```

#### Variables y Funciones:

- **env.CONFIGURED:** Variable que indica si el bot está configurado.
- **workflow.errorMessage:** Mensaje de error en caso de configuración incorrecta.

#### Handle\_Chat\_Messages:

Maneja los mensajes de chat utilizando el flujo `handle_response`:

javascript

Copy code

```
await bp.dialogEngine.handleEvent('msgReceived', event);
```

### **Variables y Funciones:**

- **bp.dialogEngine.handleEvent:** Maneja eventos del motor de diálogo de Botpress.

### **Exit:**

Nodo final del flujo.

### **Recursos Adicionales:**

- Documentación de Botpress
- ThreadsTable API

### **Conclusión:**

Este flujo index en Botpress proporciona una solución eficiente para limpiar hilos obsoletos, asegurando un rendimiento óptimo y una gestión eficaz de recursos.