



Dialogflow



Telegram

Chatbot usando dialogflow, Node.js y Telegram

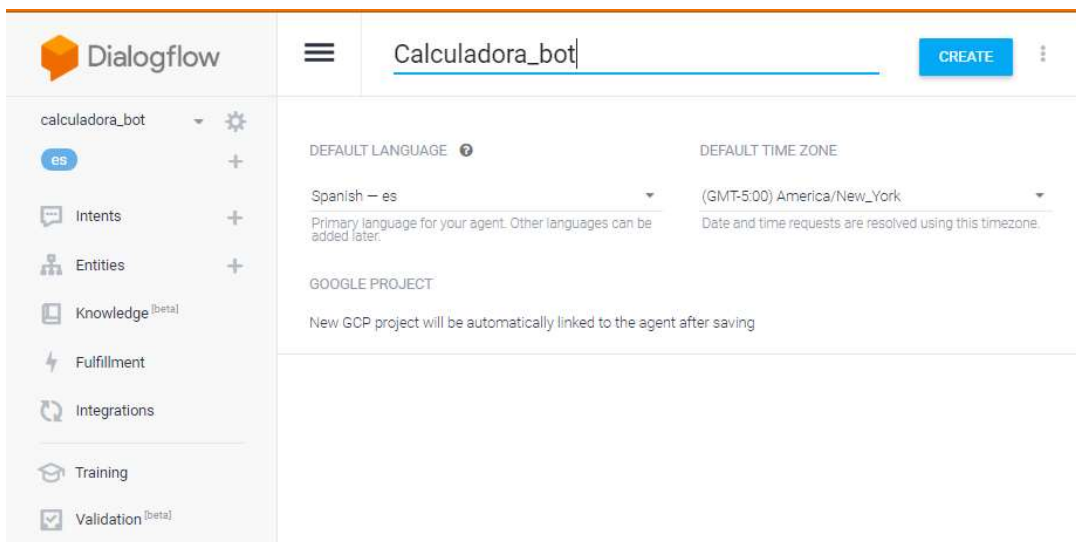
Tabla de contenido

Chatbot usando dialogflow, node js y Telegram	2
Creación del agente en dialogflow	2
Creación del servidor en Node.js	5
Configuración del servidor en dialogflow	7
Integración con Telegram	11
Referencias	13

Chatbot usando dialogflow, node js y Telegram

Creación del agente en dialogflow

Crear un nuevo Agente



The screenshot shows the Dialogflow console interface. On the left, there's a sidebar with the Dialogflow logo and a list of settings for the agent 'calculadora_bot', including 'es' (Spanish), 'Intents', 'Entities', 'Knowledge', 'Fulfillment', 'Integrations', 'Training', and 'Validation'. The main area displays the agent's configuration: 'DEFAULT LANGUAGE' is set to 'Spanish -- es', 'DEFAULT TIME ZONE' is '(GMT-5:00) America/New_York', and 'GOOGLE PROJECT' is set to 'New GCP project will be automatically linked to the agent after saving'. A 'CREATE' button is visible in the top right corner.

Crear intent para suma



The screenshot shows the Dialogflow console interface for creating a new intent. The top bar shows the agent name 'suma' and a 'SAVE' button. Below the bar, there are sections for 'Contexts', 'Events', and 'Training phrases'. The 'Training phrases' section is currently empty, with a search bar and an upward arrow icon.

Colocamos como frases de entrenamiento de la siguiente manera, tener en cuenta el tipo de dato.

Training phrases

Search training phrases  

” 2 + 2		
” 2 + 2		
PARAMETER NAME	ENTITY	RESOLVED VALUE
num1	@sys.number	2
num2	@sys.number	2

Los parámetros de acción deben quedar de la siguiente manera, no olvidar colocar el nombre, colocar como requeridos los dos parámetros

Action and parameters

suma					
REQUIRED 	PARAMETER NAME 	ENTITY 	VALUE	IS LIST 	PROMPTS 
<input checked="" type="checkbox"/>	num1	@sys.number	\$num1	<input type="checkbox"/>	Define prompts...
<input checked="" type="checkbox"/>	num2	@sys.number	\$num2	<input type="checkbox"/>	Define prompts...
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>	—

[+ New parameter](#)

Podemos agregar más frases

Training phrases

Search training phrases  

” Add user expression	
” a 2 sumale 2	
” cuanto es 2+2	
” 2 mas 2	
” 2+2	

En este ejemplo no tendremos respuestas en el “Response” sino que manejaremos cada petición por medio de un webhook que más adelante prepararemos, pero para esto debemos habilitar el “Fulfillment” en el intent, está al final solo clickeamos en “Enable Fulfillment” y luego en “Enable webhook call for this intent”.



Se debe dar salvar después de realizar estos cambios, para que entrenar y podamos probar su funcionamiento más adelante.

Creación del servidor en Node.js

Después de esto se crearemos un servidor que maneje las solicitudes realizadas al chatbot, la programación la realizaremos por medio de Node.js

Se deberá crear una carpeta, donde se almacenarán los archivos y después la abrimos en Visual Studio Code.

Desde la terminal (View -> Terminal), ejecutar el siguiente comando, para crear un proyecto de node

```
npm init -y
```

Después de esto, instalamos unos componentes necesarios par montar el servidor.

```
npm i body-parser express ngrok -D
```

Nota: ngrok expone los servidores locales detrás de NAT y firewalls a la Internet pública a través de túneles seguros.

Se crea un archivo con el nombre de 'index.js' y le agregamos el siguiente contenido

```
var express = require("express");
var bodyParser = require("body-parser");
const ngrok = require('ngrok');
const linea = '\n-----\n'
var d = new Date()

var app = express();

var port = process.env.PORT || 3000;
var ip = process.env.IP || "127.0.0.1";

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.post('/', function (req, res) {
  if (req.body.queryResult.action == "suma") {
    let num1 = parseFloat(req.body.queryResult.parameters.num1);
    let num2 = parseFloat(req.body.queryResult.parameters.num2);
    let sum = num1 + num2;
    response = num1 + " + " + num2 + " es " + sum;
    res.json({
      "fulfillmentText": response
    })
  }
})
```

```

    });
  }
});

app.listen(port, ip);

(async function () {
  const url = await ngrok.connect(port);
  console.log(` ${linea} Servidor corriendo en la URL ${url} `);
  console.log(` Pegar esta URL en dialogFlow -> Fulfillment -> Webhook -> URL `);
  console.log(` inicia: ${d} ${linea} `);
})();

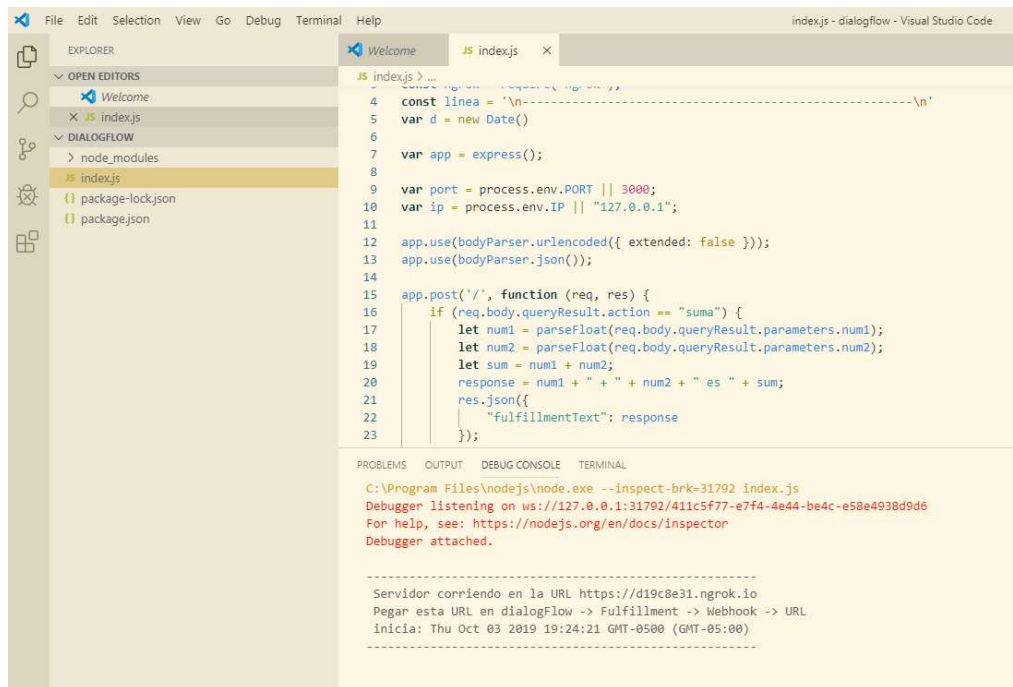
```

El código anterior prepara un servidor con express, esto es porque al habilitar el webhook en dialogflow lo que pasará cada vez que le hagamos una pregunta al bot el dirigirá la petición a la url que le proporcionemos del webhook para procesar la pregunta y dar una respuesta, es por eso por lo que instalamos ngrok.

Todas estas solicitudes son enviadas a la raíz del servidor “/” es por esto que es importante colocar un nombre a la acción del intent, para saber cómo manejarla dentro del servidor, dialogflow nos envía una serie de datos en cada solicitud, entre estas el “queryResult.action” que es donde viene la acción ejecutada desde ese intent, también “queryResult.parameters” que es donde vienen los parámetros que el usuario proporcionó en el mensaje, más específicamente en este caso los dos números (num1, num2) .

Configuración del servidor en dialogflow

Después de poner a escuchar el servidor ejecutamos una función para poner en “línea” nuestro servidor con ngrok, la url que nos aparecerá en consola luego de hacer “node index.js” es la que colocaremos de webhook en dialogflow así



```
File Edit Selection View Go Debug Terminal Help
index.js - dialogflow - Visual Studio Code

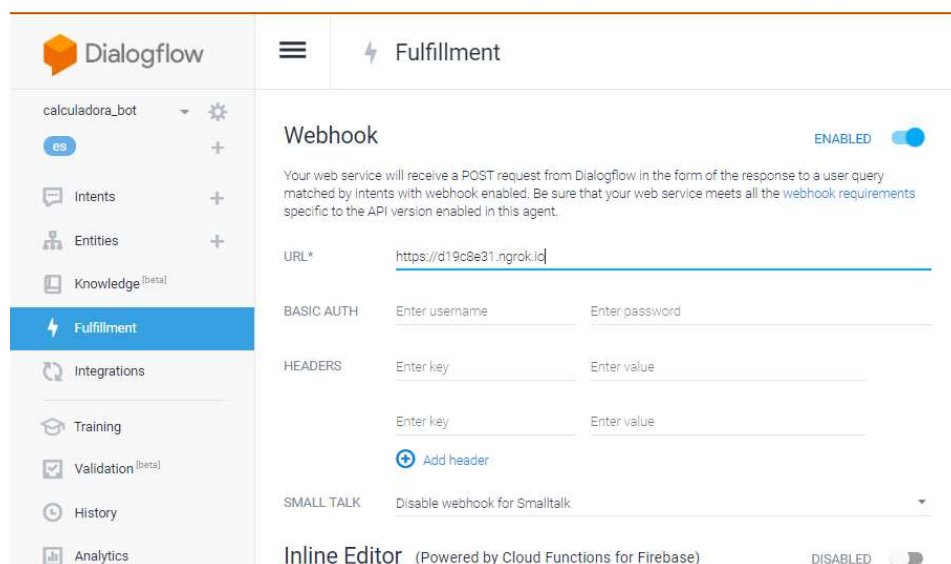
EXPLORER
  OPEN EDITORS
    Welcome
    JS index.js
  DIALOGFLOW
    node_modules
    JS index.js
    package-lock.json
    package.json

JS index.js
4 const línea = '\n-----\n'
5 var d = new Date()
6
7 var app = express();
8
9 var port = process.env.PORT || 3000;
10 var ip = process.env.IP || "127.0.0.1";
11
12 app.use(bodyParser.urlencoded({ extended: false }));
13 app.use(bodyParser.json());
14
15 app.post('/', function (req, res) {
16   if (req.body.queryResult.action == "suma") {
17     let num1 = parseFloat(req.body.queryResult.parameters.num1);
18     let num2 = parseFloat(req.body.queryResult.parameters.num2);
19     let sum = num1 + num2;
20     response = num1 + " + " + num2 + " es " + sum;
21     res.json({
22       "fulfillmentText": response
23     });
24   }
25 });

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
C:\Program Files\nodejs\node.exe --inspect-brk=31792 index.js
Debugger listening on ws://127.0.0.1:31792/411c5f77-e7f4-4e44-be4c-e58e4938d9d6
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.

-----
Servidor corriendo en la URL https://d19c8e31.ngrok.io
Pegar esta URL en dialogFlow -> Fulfillment -> Webhook -> URL
inicia: Thu Oct 03 2019 19:24:21 GMT-0500 (GMT-05:00)
-----
```

En dialogflow colocar la URL indicada en la consola



Dialogflow Fulfillment

calculadora_bot

Intents

Entities

Knowledge [beta]

Fulfillment

Integrations

Training

Validation [beta]

History

Analytics

Webhook

ENABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the [webhook requirements](#) specific to the API version enabled in this agent.

URL* https://d19c8e31.ngrok.io

BASIC AUTH Enter username Enter password

HEADERS Enter key Enter value

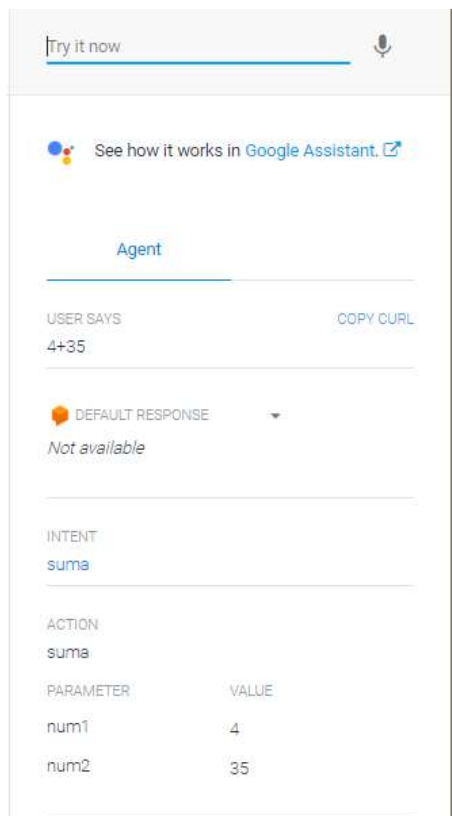
Enter key Enter value

+ Add header

SMALL TALK Disable webhook for Smalltalk

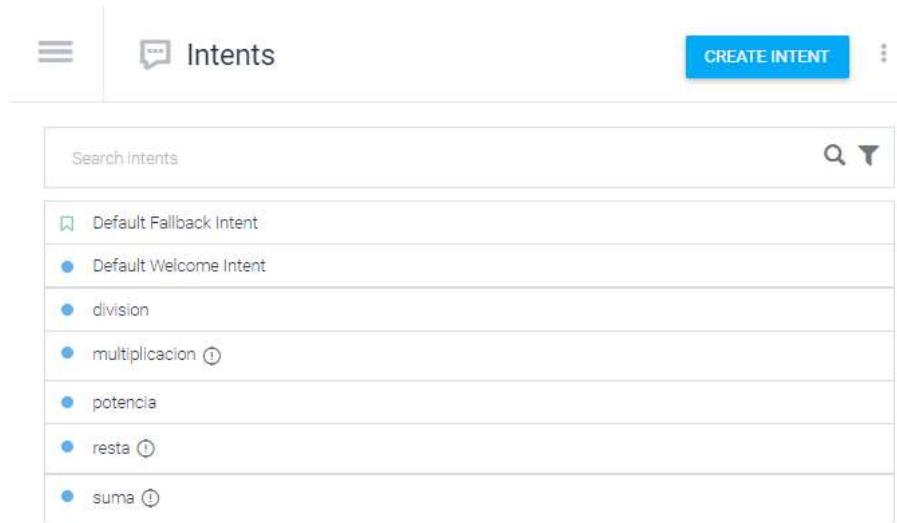
Inline Editor (Powered by Cloud Functions for Firebase) DISABLED

Salvamos y probamos en el funcionamiento



el bot está funcionando, en la anterior captura se puede ver de nuevo lo enviado por el usuario, la respuesta (que hemos enviado desde el servidor de node), el intent, la acción y los parámetros.

Ahora de igual modo crearemos los intent para restar, multiplicar y dividir y agregaremos su respectiva funcionalidad en el código del servidor.



Cambiar agregar la funcionalidades adicionales en el código

```
var express = require("express");
var bodyParser = require("body-parser");
const ngrok = require('ngrok');
const linea = '\n-----\n'
var d = new Date()

var app = express();

var port = process.env.PORT || 3000;
var ip = process.env.IP || "127.0.0.1";

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.post('/', function (req, res) {
  if (req.body.queryResult.action == "suma") {
    let num1 = parseFloat(req.body.queryResult.parameters.num1);
    let num2 = parseFloat(req.body.queryResult.parameters.num2);
    let sum = num1 + num2;
    response = num1 + " + " + num2 + " es " + sum;
    res.json({
      "fulfillmentText": response
    });
  } else if (req.body.queryResult.action == "resta") {
    let num1 = parseFloat(req.body.queryResult.parameters.num1);
    let num2 = parseFloat(req.body.queryResult.parameters.num2);
    let sum = num1 - num2;
    response = num1 + " - " + num2 + " es " + sum;
    res.json({
```

```

        "fulfillmentText": response
    });
} else if (req.body.queryResult.action == "multiplicacion") {
    let num1 = parseFloat(req.body.queryResult.parameters.num1);
    let num2 = parseFloat(req.body.queryResult.parameters.num2);
    let sum = num1 * num2;
    response = num1 + " * " + num2 + " es " + sum;
    res.json({
        "fulfillmentText": response
    });
} else if (req.body.queryResult.action == "division") {
    let num1 = parseFloat(req.body.queryResult.parameters.num1);
    let num2 = parseFloat(req.body.queryResult.parameters.num2);
    let sum = num1 / num2;
    response = num1 + " / " + num2 + " es " + sum;
    res.json({
        "fulfillmentText": response
    });
}
});

app.listen(port, ip);

(async function () {
    const url = await ngrok.connect(port);
    console.log(` ${linea} Servidor corriendo en la URL ${url} `);
    console.log(` Pegar esta URL en dialogFlow -> Fulfillment -> Webhook -> URL `);
    console.log(` inicia: ${d}  ${linea} `);
})();

```

Integración con Telegram

Se necesita un token que nos permita la conexión con desde dialogflow hacia telegram, para conseguir este token debemos crear el bot en telegram, ingresamos a <https://web.telegram.org/#/im?p=@BotFather>, estando en el chat con el BotFather en telegram le escribimos:

/newbot

Se le deba asignar un nombre al bot y un username para el bot, después de esto nos mostrara un numero el cual colocamos en dialogflow



BotFather

4:04

Done! Congratulations on your new bot. You will find it at t.me/ejem_calculadora_bot. You can now add a description, about section and profile picture for your bot, see [/help](#) for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:

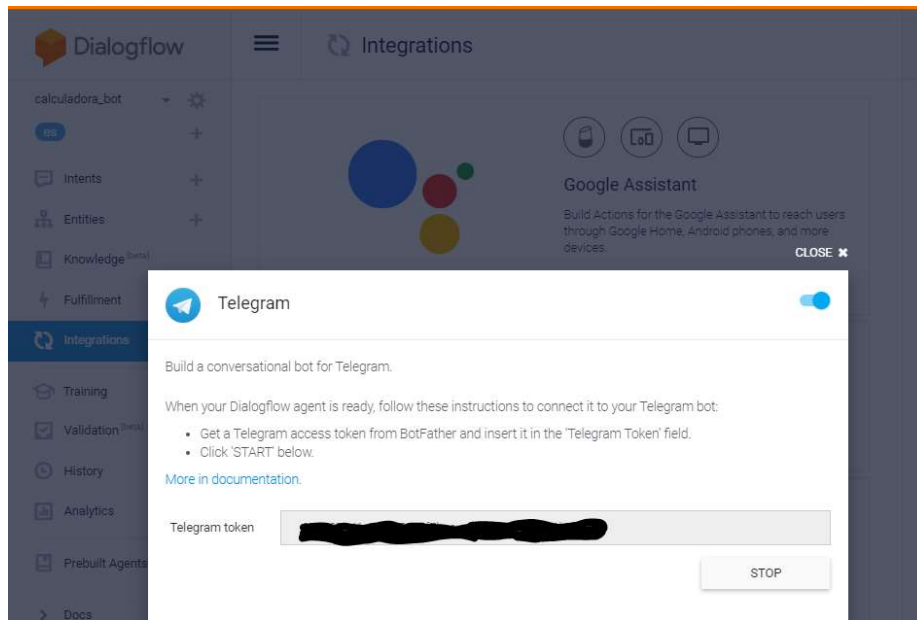
[REDACTED]

Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page:

<https://core.telegram.org/bots/api>

En dialogflow realizamos



En el chat del usuario del chatbot podemos validar el funcionamiento

CA	Cristian cuanto es 3 menos 6	4:06:32 PM
CB	calculadora_bot 3 - 6 es -3	4:06:36 PM
CA	Cristian 40 +2789	6:25:06 PM
CB	calculadora_bot 40 + 2789 es 2829	6:25:08 PM

Referencias

<https://planetachatbot.com/creando-un-chatbot-para-reservar-entradas-de-cine-2-52c7c9e2823d-52c7c9e2823d>

<https://joralmogithubio20181107/Como-crear-un-chatbot-Node-js-y-DialogFlow/>

<https://blog.crowdbotics.com/build-chatbot-dialogflow-nodejs-webhooks/>

<https://medium.com/byteridge/building-a-voice-enabled-chat-bot-for-a-website-using-dialogflow-firebase-jquery-3a10a3a36e2>