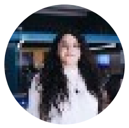


Pilar Figueroa Follow
Front-end Engineer at Crowdbotics
Oct 21, 2017 · 14 min read

How To Create Your Very Own Facebook Messenger Bot with Dialogflow and Node.js In Just One Day

Pilar is a software engineer at Crowdbotics specializing in chatbots, test automation, and frontend development.

So, you're here because you want to create your first chatbot. Or maybe you've already built one, and you want to know how you can improve your chatbot's responses with NLP.

Today, I'm going to show you exactly how to do that! We'll build a chatbot with rich NLP features using Dialogflow, Wit.ai, and Node.js. The best part? It'll take us just one day.

Let's review. What is a chatbot?

Chatbots are computer programs that mimic conversation with people through the use of artificial intelligence.

In a nutshell, when someone types in a message, the bot responds with the right reply.

Out of the thousands of chatbots that exist today, most are messenger apps designed to communicate with customers. They work with Natural Processing Language (NLP) systems which translate everyday language into a form that the bot can understand and learn new interactions from.

Through the help of that technology, bots now hold endless possibilities. You can use them to read everyday news, get the weather, transfer money, buy your favorite items, schedule a meeting, or even get service assistance. And all from the convenience of your favorite messenger app.

People are now spending more mobile screen time on messaging than on social media. A lot of companies have noticed the trend and are taking advantage of chatbots as a new channel to talk to us. Turns out that a whopping *60% of online adults in the US* use online messaging, voice or video chat services.

Even stranger? It's thought that by 2020, we'll have more conversations with bots than with our actual spouses.

Choose your tech: Dialogflow or wit.ai?

We're choosing between two popular platforms when building our chatbot: Dialogflow and wit.ai.

Dialogflow

Dialogflow (once known as Api.ai) is a service owned by Google that allows developers to build speech to text, natural language processing and artificially intelligent systems that you can train with your own custom functionality. This incredible tool uses machine learning to understand what users are saying and it's beyond simple to set up nonlinear bots quickly.

Advantages

1. **It's Useful**: It integrates NLP without much hassle.
2. **It's Easy:** Dialogflow contains lots of pre-built agents that are a breeze to activate.
3. **It's Integrated:** You can connect your favorite platforms such as Facebook, Twitter, Slack, Telegram, etc…
4. **It's Multilingual**: It recognizes more than 15 languages.
5. **It's Cheap**: You'll be glad to know it's totally free.

Disadvantages

1. **It's Not So Customizable:** If you want to create a customizable bot, you will need to implement a code flow, and these examples are not shown in any of its documentation.
2. **It's Not So Implementable**: It can be tricky to figure out how to perform platforms implementations as it is not documented.

Wit.ai

Wit.ai (owned by Facebook) works similarly to Dialogflow: it also processes human speech patterns and filters useful data like intent and context from it. Like Dialogflow, it provides a UI to help developers with creating intents, entities and agents.

What Are Its Pros?

1. **It's Also Useful:** Wit.ai also integrates NLP.
2. **It's Also Easy:** The Quickstart tutorial is very practical in order to get started.
3. **It's Also Integrated**: You'll also be able to integrate with several platforms, like Facebook, Twitter, Slack, Telegram, etc…
4. **It's Also Cheap:** Totally free!
5. **It's Adaptable:** You'll be able to build your bot with Node.js, Python or Ruby.
6. **It's Pragmatic**: Enjoy an easy to read "Recipes" section for common problems and how you can solve them.

What Are Its Cons?

1. **It's Not-So Fast**: Since the learning curve is steep, you'll need to invest time into figuring out how to implement it.
2. **It's Not-So Visual**: Given that there's no visual development environment, you'll need to be comfortable with code.

So which did we pick?

Given that Dialogflow provides junior developers with the best documentation and a great user experience for developing a bot without being an expert in the field, we've decided to give it a shot for this tutorial.

A step-by-step guide to building the chatbot

In this tutorial, we'll be using Node.js to build a simple bot, so please make sure it's installed on your computer.

So let's get started!

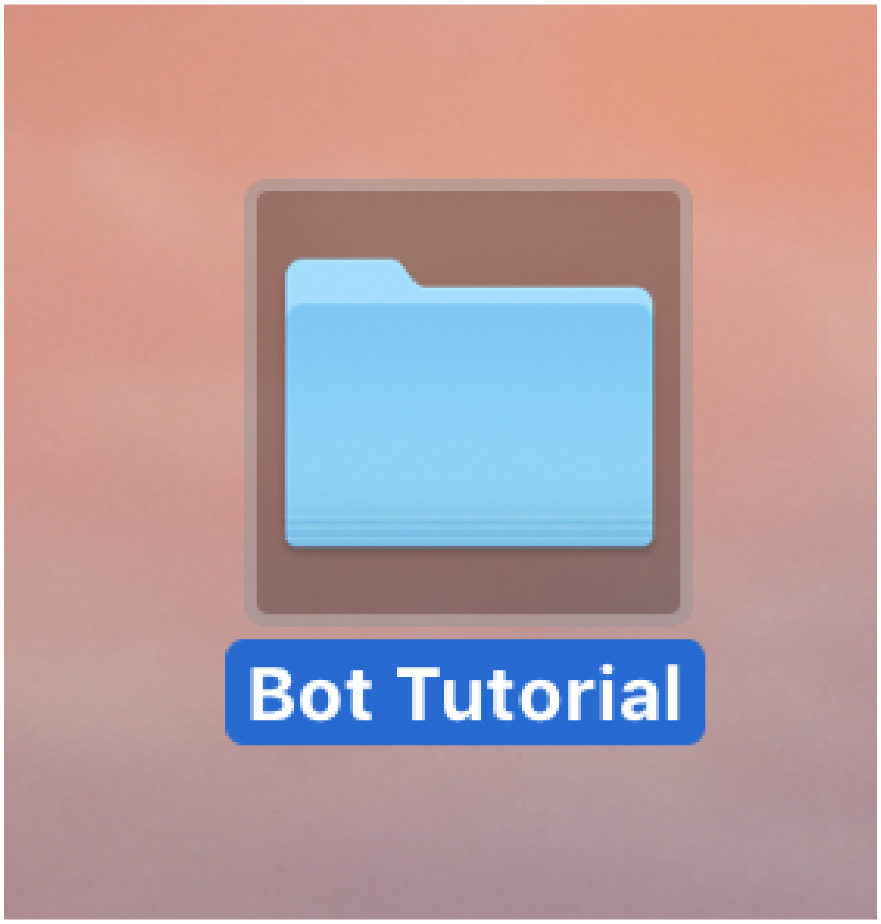
Step 1: Setting up your development environment

Let's create a simple webserver with one webhook endpoint. I'll use Express.js.

1. Writing webhook server with Express

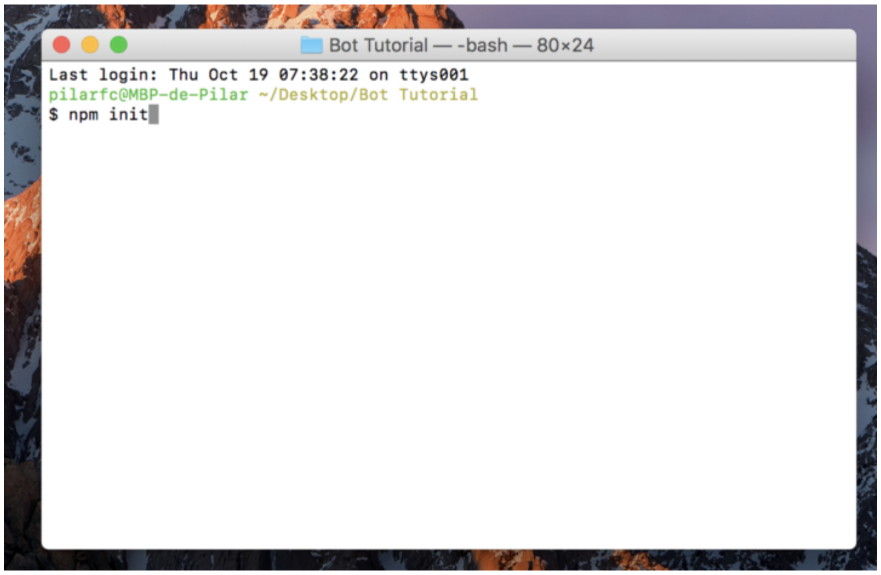
First of all, we need to know that a webhook (also called a web callback or HTTP push API) is a way for an app to provide other applications with real-time information. It delivers data to other applications as it happens, meaning you get data immediately, unlike typical APIs where you would need to poll for data very frequently in order to get it in real-time.

In order to get started, you will have to create a new directory, where you'll store your entire project. We are going to name it "Bot Tutorial".



Once, you have created it, go to your terminal, access this directory and initialize your Node.js app with:

```
npm init
```



```
Last login: Thu Oct 19 07:38:22 on ttys001
pilarfc@MBP-de-Pilar ~/Desktop/Bot Tutorial
$ npm init
```

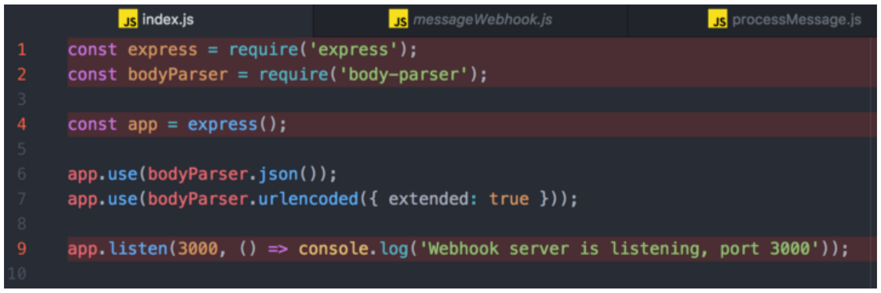
Press Enter.

After filling out all the needed info (if you don't know how to fill package.json, take a look here), your next step will be to install **Express** to setup a server and one middleware for it, called **body-parser** to parse incoming request bodies. So, in your terminal type:

```
npm install express body-parser --save
```

Once the installation is complete, go to your directory and create a file called **index.js** and start Express server listening to the port 3000 (you can take any you want).

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

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

app.listen(3000, () => console.log('Webhook server is
listening, port 3000'));
```



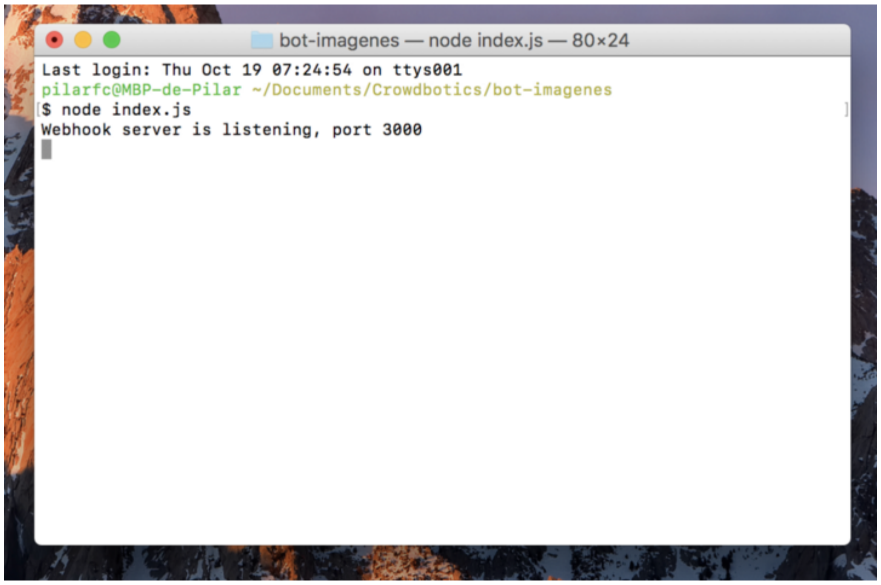
```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

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

app.listen(3000, () => console.log('Webhook server is listening, port 3000'));
```

Code example above

Save it and let's check if it's working. Run this command in your terminal:

```
node index.js
```

And if everything is working how it should be, you will receive the following message in your terminal:



```
Last login: Thu Oct 19 07:24:54 on ttys001
pilarfc@MBP-de-Pilar ~/Documents/Crowdbotics/bot-imagenes
$ node index.js
Webhook server is listening, port 3000
```

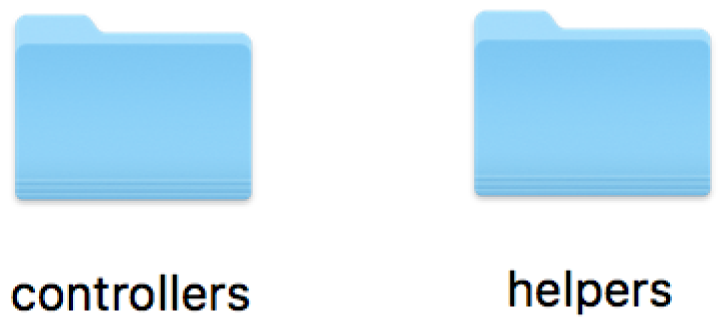
Great, our server is listening! Now we are going to create two endpoints:

1. One will be for Facebook initial verification. The reason why we need to do this is because when you're connecting a Facebook page with the webhook server you will also need have a token. Facebook will make a request and match this token to that one

webhook response to be sure that you're not connecting your page with a random webhook server.

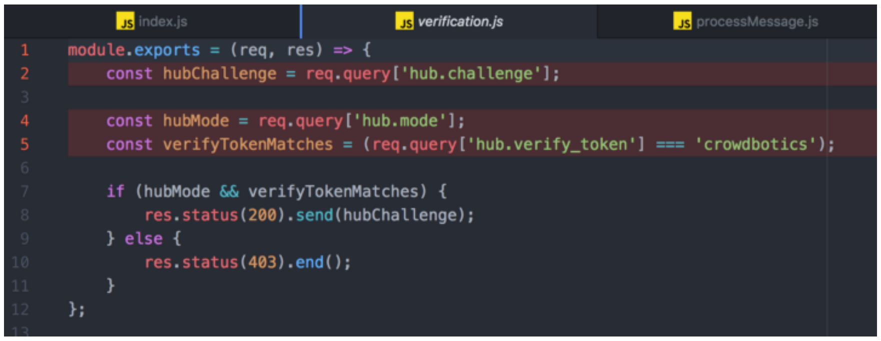
2. The second one will be responsible for the other messages from Facebook Messenger.

But before we go any further, we will organize our code in two separate folders: **controllers and helpers**. For each separate endpoint we'll create a function in a separate file in the controllers folder.



The first verification endpoint goes to controllers/verification.js.

```
module.exports = (req, res) => {
 const hubChallenge = req.query['hub.challenge'];


const hubMode = req.query['hub.mode'];
 const verifyTokenMatches = (req.query['hub.verify_token']
=== 'crowdbotics');


if (hubMode && verifyTokenMatches) {
 res.status(200).send(hubChallenge);
 } else {
 res.status(403).end();
 }
};
```



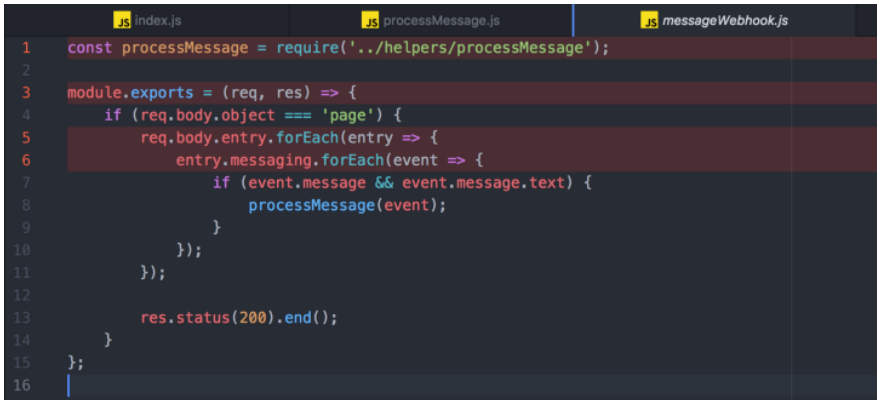
```
module.exports = (req, res) => {
    const hubChallenge = req.query['hub.challenge'];

    const hubMode = req.query['hub.mode'];
    const verifyTokenMatches = (req.query['hub.verify_token'] === 'crowdbotics');

    if (hubMode && verifyTokenMatches) {
        res.status(200).send(hubChallenge);
    } else {
        res.status(403).end();
    }
};
```

Code example above

Here you can see a string called “crowdbotics” . You can change it and choose any word or text string that you prefer. Make sure to make a note of it as you will need it later when setting up your Facebook app.

The second endpoint for handling all the Facebook bot messages will go to controllers/messageWebhook.js.

```
const processMessage = require('../helpers/processMessage');

module.exports = (req, res) => {
 if (req.body.object === 'page') {
 req.body.entry.forEach(entry => {
 entry.messaging.forEach(event => {
 if (event.message && event.message.text) {
 processMessage(event);
 }
 });
 });

res.status(200).end();
 }
};
```



```
const processMessage = require('../helpers/processMessage');

module.exports = (req, res) => {
    if (req.body.object === 'page') {
        req.body.entry.forEach(entry => {
            entry.messaging.forEach(event => {
                if (event.message && event.message.text) {
                    processMessage(event);
                }
            });
        });

        res.status(200).end();
    }
};
```

Code example above

Now, if you left your console running with the port 3000, stop it and rewrite this command in there:

```
node index.js
```

2. Setting up a proxy server with ngrok

Why do we need this step? Well, our local Express server url is not available for everyone on the Internet and also, it doesn't support HTTPS protocol, which is necessary for Facebook Messenger bots. We will therefore set up a grok server as a proxy.

Ngrok is a multiplatform tunnelling, reverse proxy software that establishes secure tunnels from a public endpoint such as the Internet to a locally running network service while capturing all traffic for detailed inspection and replay.

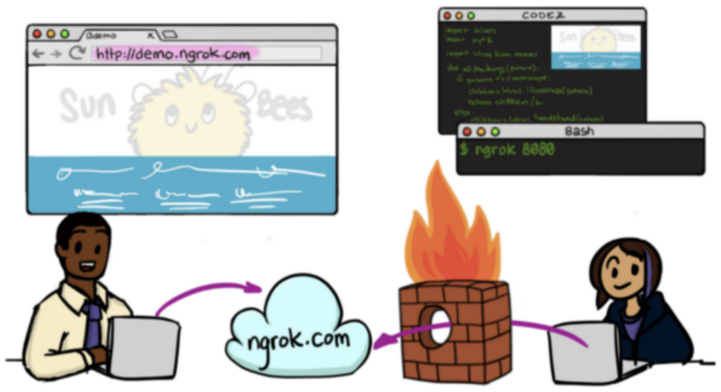


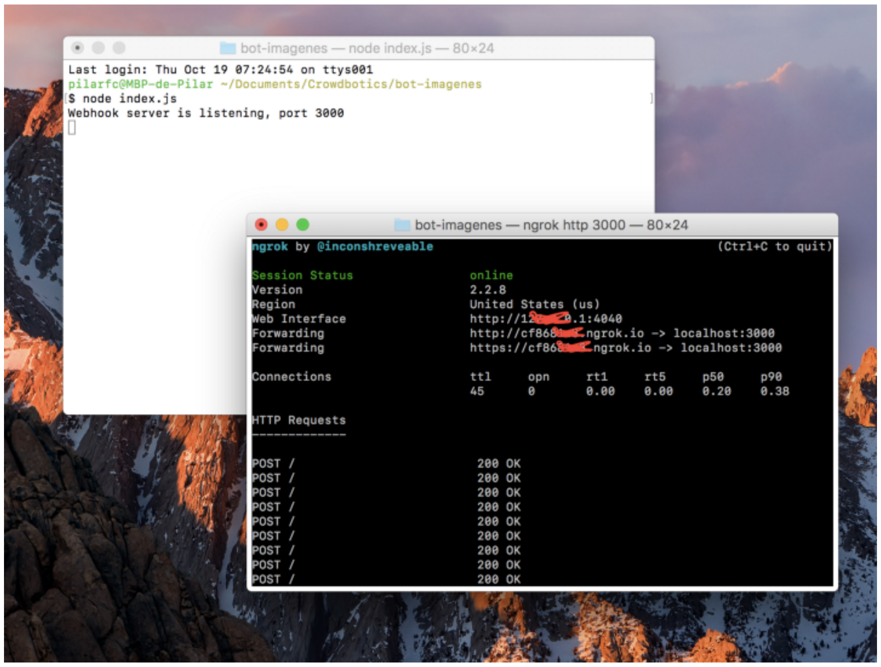
Image from https://ngrok.com/

But if this is not clear, you can read more about it here.

Once you finish reading and installing ngrok, open another terminal window and position it in the directory of your project (meanwhile your local server is running) and type the following command:

```
ngrok http 3000
```

"3000" is going to be the number you chose for your local server.


bot-imagenes — node index.js — 80×24
bot-imagenes — ngrok http 3000 — 80×24

```
Last login: Thu Oct 19 07:24:54 on ttys001
pilarfc@MBP-de-Pilar ~/Documents/Crowdbotics/bot-imagenes
$ node index.js
Webhook server is listening, port 3000
```

```
ngrok by @inconshreveable                                      (Ctrl+C to quit)

Session Status                online
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://12[redacted].1:4040
Forwarding                    http://cf868[redacted]ngrok.io -> localhost:3000
Forwarding                    https://cf86[redacted]ngrok.io -> localhost:3000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              45      0       0.00    0.00    0.20    0.38

HTTP Requests
-------------

POST /                         200 OK
POST /                         200 OK
POST /                         200 OK
POST /                         200 OK
POST /                         200 OK
POST /                         200 OK
POST /                         200 OK
POST /                         200 OK
POST /                         200 OK
```

It'll show you a new screen with a couple of URLs, the URL that you'll need is the last one that starts with **https://cf86…**

3. Setting up Facebook Application

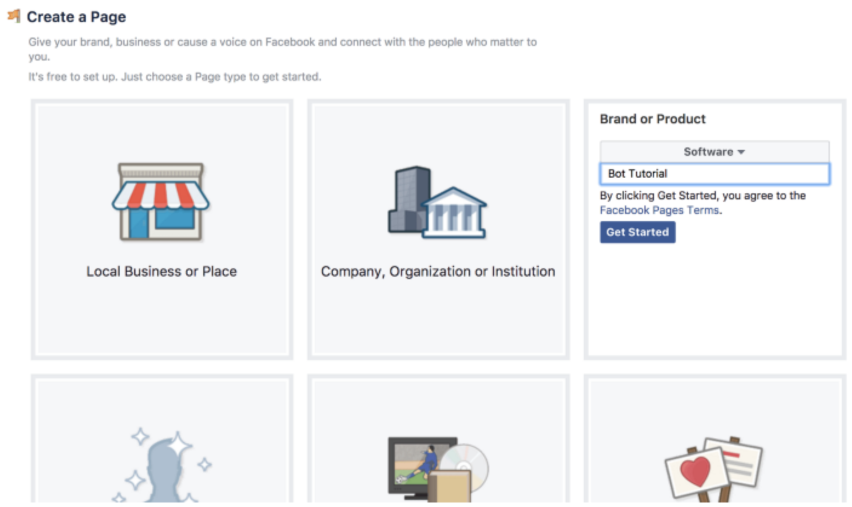
Before continuing to send requests to Facebook, let's create our Facebook app.

To create a Facebook bot, we need to two things:

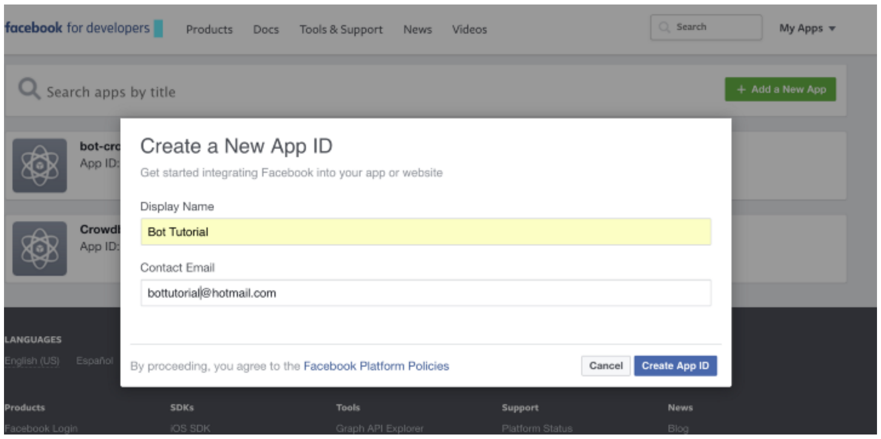
1. A Facebook public page, that you'll connect with your bot.

2. A Facebook Developer application, which will be connected to your webhook server and your public page, and work as a middleware between them.

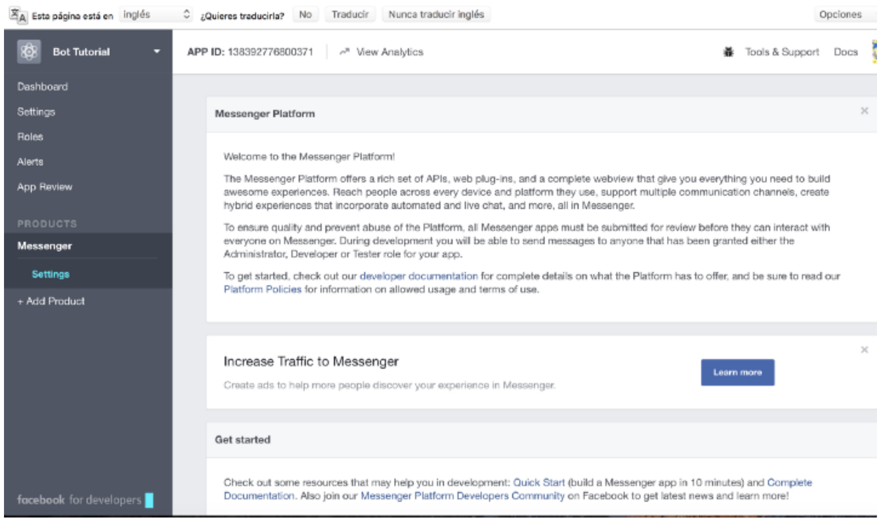
First you need to create a page. Go to facebook.com/pages/create. Choose a category, subcategory and click Get Started.


Create a Page
Give your brand, business or cause a voice on Facebook and connect with the people who matter to you.
It's free to set up. Just choose a Page type to get started.
Local Business or Place
Company, Organization or Institution
Brand or Product
Software
Bot Tutorial
By clicking Get Started, you agree to the Facebook Pages Terms.
Get Started

After that you'll need to create an app. Go to developers.facebook.com/quickstarts, give your Facebook app a name, type in your e-mail, and then click the "Create App ID" button.



After creating the App, you have to select a product. Click the"Messenger" icon and then click on the "Set Up" button. This will redirect you to the Messenger Platform.



Messenger Platform

Welcome to the Messenger Platform!

The Messenger Platform offers a rich set of APIs, web plug-ins, and a complete webview that give you everything you need to build awesome experiences. Reach people across every device and platform they use, support multiple communication channels, create hybrid experiences that incorporate automated and live chat, and more, all in Messenger.

To ensure quality and prevent abuse of the Platform, all Messenger apps must be submitted for review before they can interact with everyone on Messenger. During development you will be able to send messages to anyone that has been granted either the Administrator, Developer or Tester role for your app.

To get started, check out our developer documentation for complete details on what the Platform has to offer, and be sure to read our Platform Policies for information on allowed usage and terms of use.

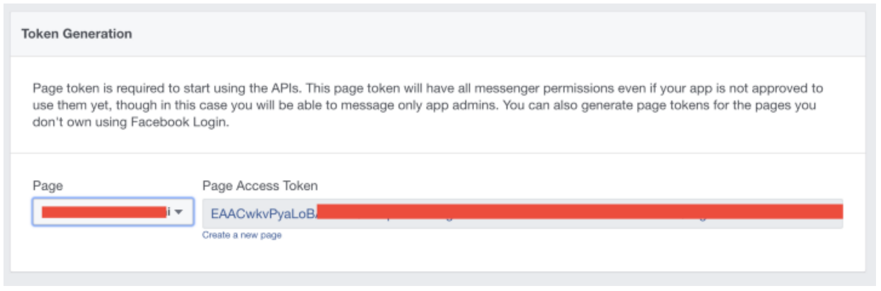
Increase Traffic to Messenger

Create ads to help more people discover your experience in Messenger.

Get started

Check out some resources that may help you in development: Quick Start (build a Messenger app in 10 minutes) and Complete Documentation. Also join our Messenger Platform Developers Community on Facebook to get latest news and learn more!

Once you're there, you must locate the "Token Generation" section. Select the page you already created, and it will give you a Page Access Token that we will use later.



Token Generation

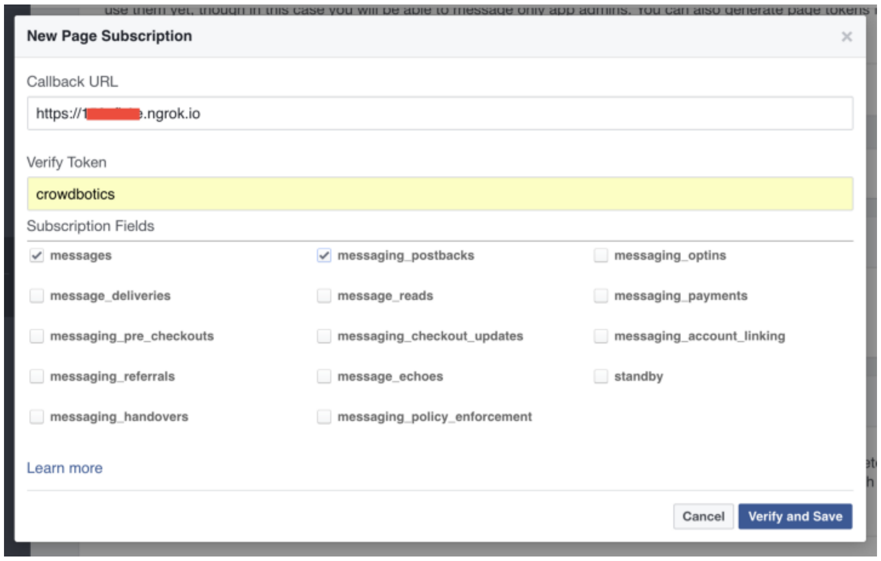
Page token is required to start using the APIs. This page token will have all messenger permissions even if your app is not approved to use them yet, though in this case you will be able to message only app admins. You can also generate page tokens for the pages you don't own using Facebook Login.

Page: [redacted]

Page Access Token: EAACwkvPyaLoB[redacted]

Below this section is the Webhooks section. Click on "Setup Webhooks" and it will show you a popup window, where you'll need to fill out the following:

- Callback URL: With your ngrok URL.
- Verify Token: The string for validation that you already chose from controller/verification.js.
- Subscription Fields: Choose messages and messaging_postbacks. If you want to know more about webhook events read this information.



use them yet, though in this case you will be able to message only app admins. You can also generate page tokens

New Page Subscription

Callback URL

https://1[redacted].ngrok.io

Verify Token

crowdbotics

Subscription Fields

✓ messages	✓ messaging_postbacks	messaging_optins
message_deliveries	message_reads	messaging_payments
messaging_pre_checkouts	messaging_checkout_updates	messaging_account_linking
messaging_referrals	message_echoes	standby
messaging_handovers	messaging_policy_enforcement	

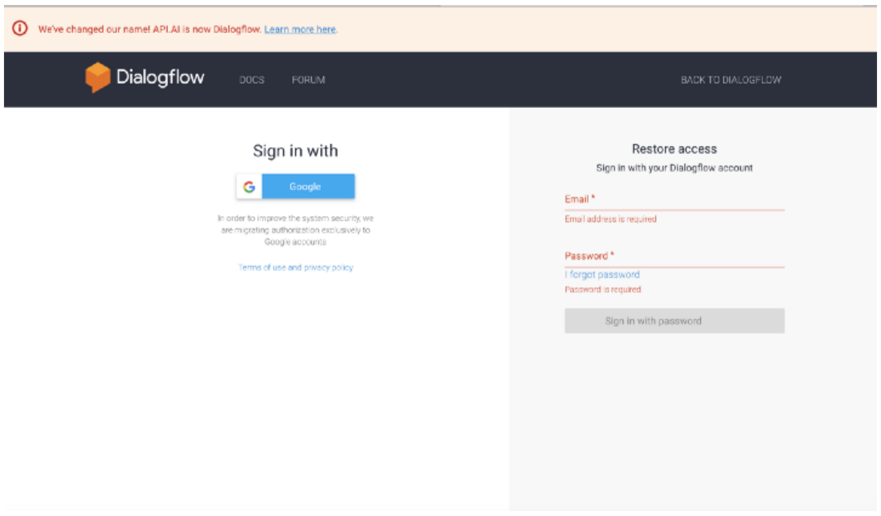
Click “Verify and Save” button.

Note: If the callback URL sent you a “502 Bad Gateway error”, it is because you aren’t running your local host server and the ngrok at the same time.

Until this moment our Facebook Application is well connected and working correctly, but we aren’t quite finished yet.

Step 2: Dialogflow integration

To get started we’ll head to the Dialogflow website and click the “Sign up for free” button. We’re then taken to a registration page, where you can log in with your Google Account.



We've changed our name! API.AI is now Dialogflow. Learn more here.

In order to improve the system security, we are migrating authorization exclusively to Google accounts

Restore access

Sign in with your Dialogflow account

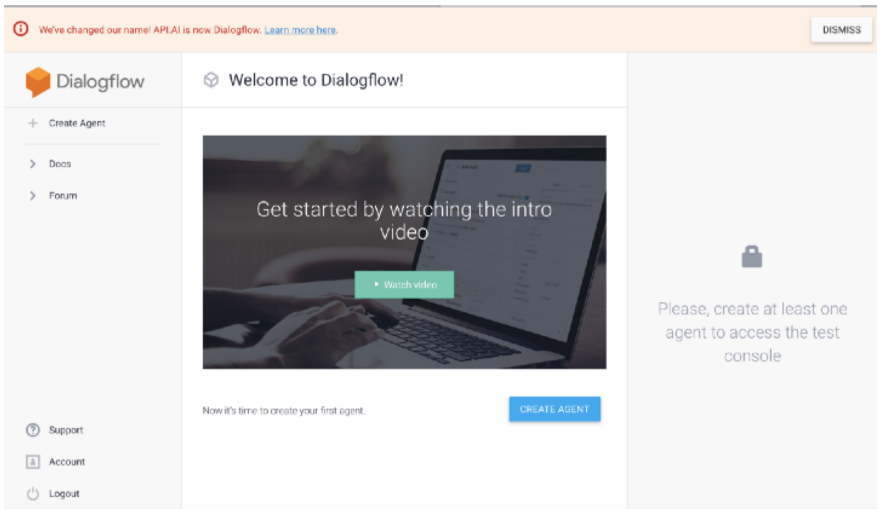
Email *

Email address is required

Password *

Password is required

Once you're in, you need to click **Allow** to grant Dialogflow access to your Google account in the screen that follows, and accept terms of service. And done! Now you're in a Dialogflow interface!



We've changed our name! API.AI is now Dialogflow. Learn more here.

Welcome to Dialogflow!

Get started by watching the intro video

Now it's time to create your first agent.

Please, create at least one agent to access the test console

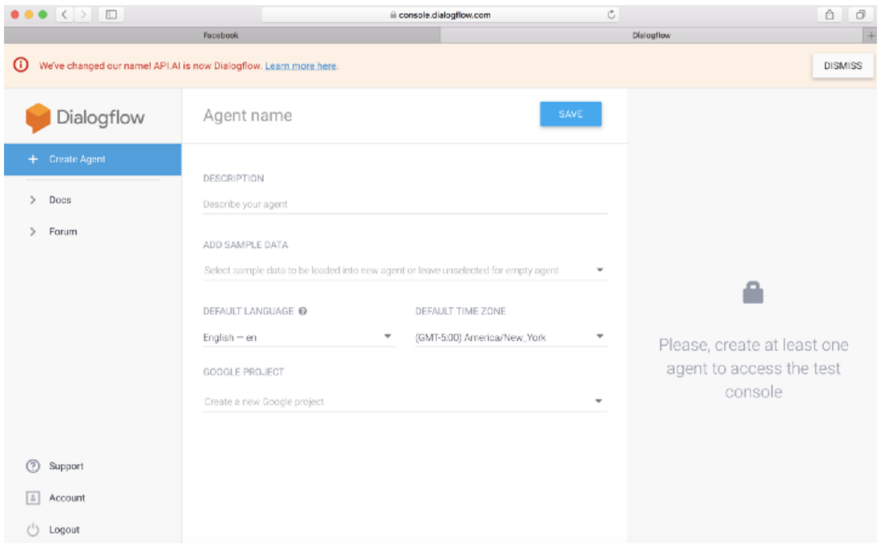
It's important that you watch the "Get Started' video. Here, the Dialogflow team explains in general terms how the platform works.

After this point, you can start prepping your virtual AI assistant. An agent is an assistant that you create and teach specific skills to. So, to begin with it, click on the "Create Agent" button. You may need to authorize Dialogflow again to have additional permissions for your Google account. This is normal, so click "Authorize".

On the next screen, we have our agent's details:

- Agent Name: This is for your own reference so that you can differentiate agents on your interface. You can choose any name you want.
- Description: A readable description, so you can remember what the agent does. This is optional.
- Language: The language which the agent works in. You need to choose right, because this cannot be changed. For this tutorial we're going to work with English.
- Time zone: The time zone you want your agent to be in.

Then just click the "Save" button.



We've changed our name! API.AI is now Dialogflow. Learn more here.

Agent name

DESCRIPTION
Describe your agent

ADD SAMPLE DATA
Select sample data to be loaded into new agent or leave unselected for empty agent

DEFAULT LANGUAGE: English — en

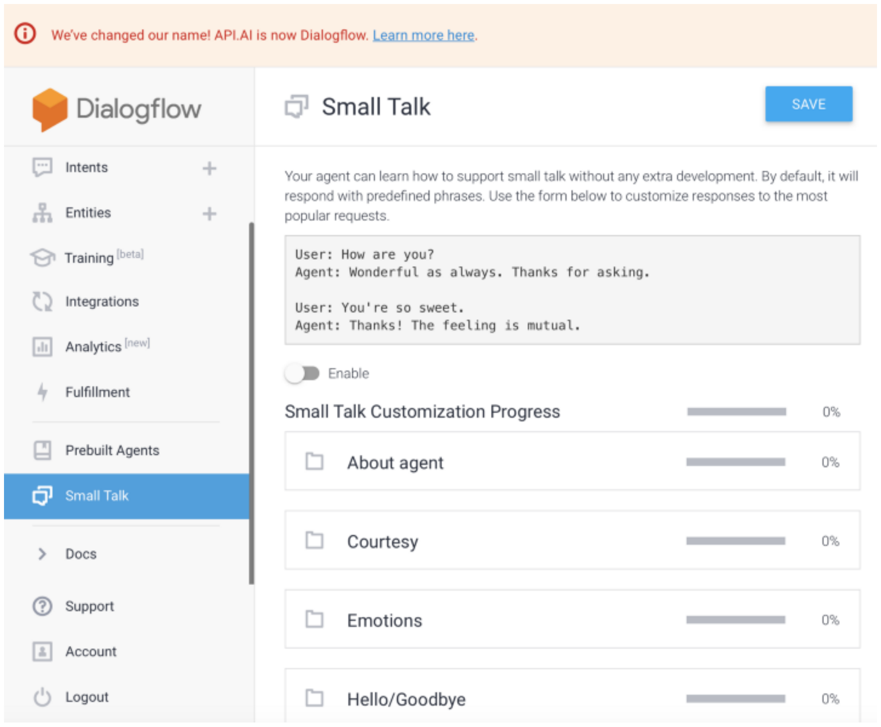
DEFAULT TIME ZONE: (GMT-5:00) America/New_York

GOOGLE PROJECT: Create a new Google project

Please, create at least one agent to access the test console

Step 3: Integrating Small Talk Agent To Your Bot

After you saved the main requirements, Dialogflow will redirect you to your bot main page. Here, go to the section called “Small Talk”.



We've changed our name! API.AI is now Dialogflow. Learn more here.

Small Talk

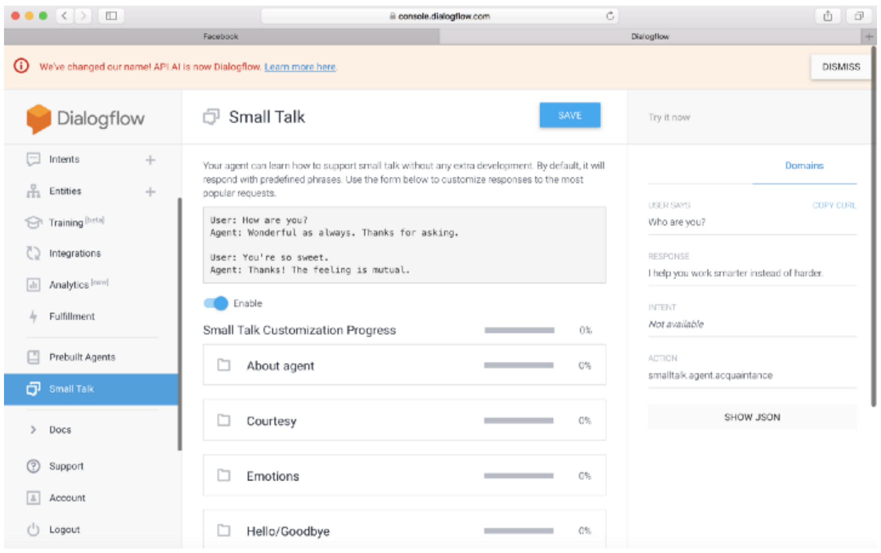
Your agent can learn how to support small talk without any extra development. By default, it will respond with predefined phrases. Use the form below to customize responses to the most popular requests.

```
User: How are you?
Agent: Wonderful as always. Thanks for asking.

User: You're so sweet.
Agent: Thanks! The feeling is mutual.
```

Enable

Small Talk Customization Progress	0%
About agent	0%
Courtesy	0%
Emotions	0%
Hello/Goodbye	0%

Click on “Enable” button. There, you can try your bot in the Console over on the right side. Type whatever you want like “Hello, how are you?”, “ Who are you?”, etc… and you will see a response.



Your agent can learn how to support small talk without any extra development. By default, it will respond with predefined phrases. Use the form below to customize responses to the most popular requests.

```
User: How are you?
Agent: Wonderful as always. Thanks for asking.

User: You're so sweet.
Agent: Thanks! The feeling is mutual.
```

Enable

Small Talk Customization Progress 0%

About agent 0%

Courtesy 0%

Emotions 0%

Hello/Goodbye 0%

USER SAYS: Who are you?

RESPONSE: I help you work smarter instead of harder.

INTENT: *Not available*

ACTION: smalltalk.agent.acquaintance

It works! You're talking to a bot! Now the next step is so embed this conversation into our code base.

Before that, you have to install two different packages on your terminal:

1. First, we have to install a request node package to be able to send requests to Facebook:

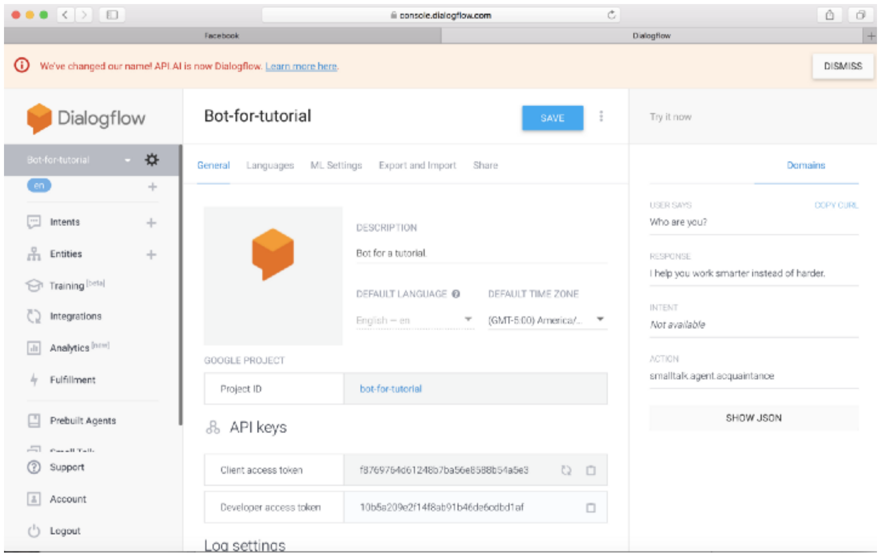
```
npm install —- save request
```

1. Second, we have to install Dialogflow node.js package, but until now the Api.ai package is still working:

```
npm install —- save apiai
```

Once you are at this stage, implement the processMessage function. To do that, create a file at helpers directory and name it **processMessage.js.**

First, we have to initialize apiaiClient with the API key, to retrieve it, click the Configuration icon (gear) at the left menu and copy "Client Access Token".



We've changed our name! API.AI is now Dialogflow. Learn more here.

Bot-for-tutorial

DESCRIPTION
Bot for a tutorial.

DEFAULT LANGUAGE: English — en
DEFAULT TIME ZONE: (GMT-5:00) America/...

GOOGLE PROJECT

Project ID	bot-for-tutorial

API keys

Client access token	f876976 4d61248b7ba56e8588b54a5e3
Developer access token	10b5a209e2f14f8ab91b46de6cdbd1af

Log settings

USER SAYS
Who are you?

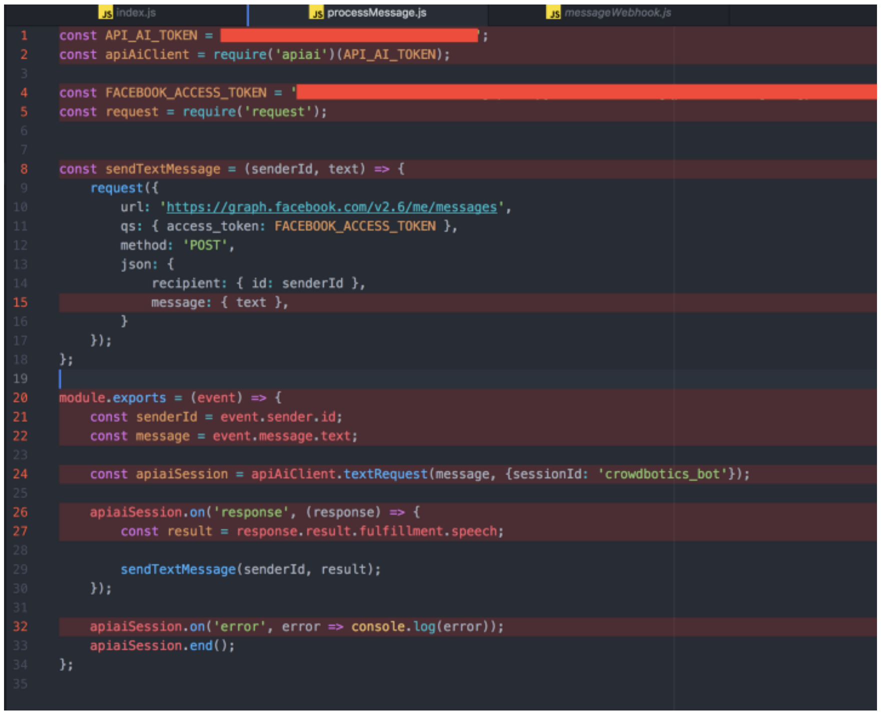
RESPONSE
I help you work smarter instead of harder.

INTENT
Not available

ACTION
smalltalk.agent.acquaintance

Now we have to implement processMessage function. Add your Client Access Token in const API_AI_TOKEN and your Facebook Page Access Token in const FACEBOOK_ACCESS_TOKEN.

```
const API_AI_TOKEN = 'your Dialogflow Client Access Token';
const apiAiClient = require('apiai')(API_AI_TOKEN);


const FACEBOOK_ACCESS_TOKEN = 'your Facebook Page Access
Token';
const request = require('request');


const sendTextMessage = (senderId, text) => {
 request({
 url: 'https://graph.facebook.com/v2.6/me/messages',
 qs: { access_token: FACEBOOK_ACCESS_TOKEN },
 method: 'POST',
 json: {
 recipient: { id: senderId },
 message: { text },
 }
 });
};


module.exports = (event) => {
 const senderId = event.sender.id;
 const message = event.message.text;


const apiaiSession = apiAiClient.textRequest(message,
{sessionId: 'crowdbotics_bot'});


apiaiSession.on('response', (response) => {
 const result = response.result.fulfillment.speech;
sendTextMessage(senderId, result);
 });

apiaiSession.on('error', error => console.log(error));
 apiaiSession.end();
};
```

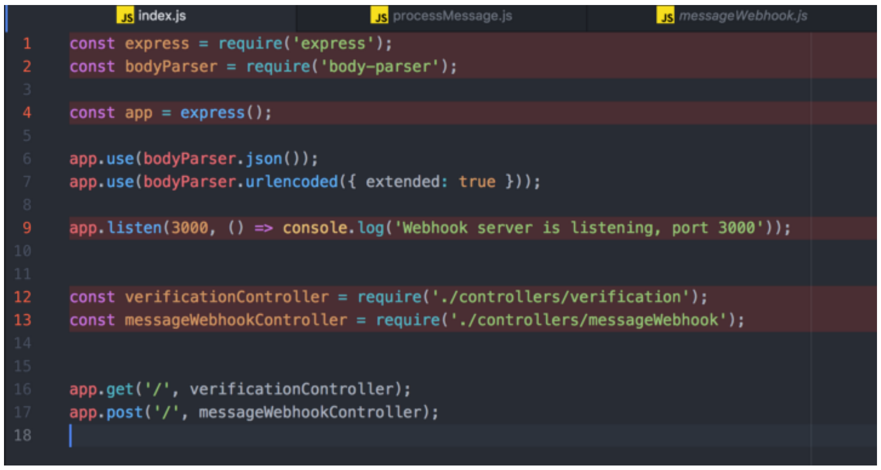


```
const API_AI_TOKEN = [redacted]';
const apiAiClient = require('apiai')(API_AI_TOKEN);

const FACEBOOK_ACCESS_TOKEN = '[redacted]
const request = require('request');


const sendTextMessage = (senderId, text) => {
    request({
        url: 'https://graph.facebook.com/v2.6/me/messages',
        qs: { access_token: FACEBOOK_ACCESS_TOKEN },
        method: 'POST',
        json: {
            recipient: { id: senderId },
            message: { text },
        }
    });
};

module.exports = (event) => {
    const senderId = event.sender.id;
    const message = event.message.text;

    const apiaiSession = apiAiClient.textRequest(message, {sessionId: 'crowdbotics_bot'});

    apiaiSession.on('response', (response) => {
        const result = response.result.fulfillment.speech;

        sendTextMessage(senderId, result);
    });

    apiaiSession.on('error', error => console.log(error));
    apiaiSession.end();
};
```

Code example above

And, to make it work you also need to add a couple of lines to your index.js to set up the endpoints.

```
const verificationController =
require('./controllers/verification');
const messageWebhookController =
require('./controllers/messageWebhook');

app.get('/', verificationController);
app.post('/', messageWebhookController);
```



```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

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

app.listen(3000, () => console.log('Webhook server is listening, port 3000'));


const verificationController = require('./controllers/verification');
const messageWebhookController = require('./controllers/messageWebhook');


app.get('/', verificationController);
app.post('/', messageWebhookController);

```

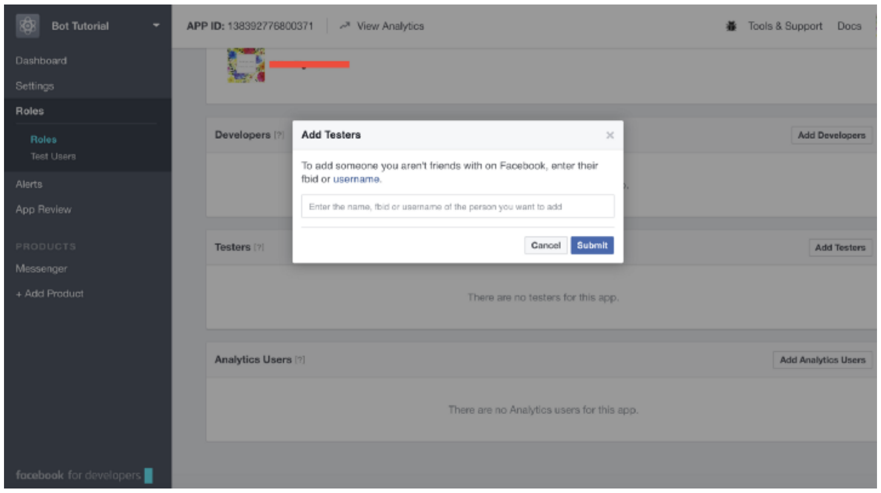
Code example above

What happened?

If you look at the code, your webhook server received the information from a user via Facebook Messenger, then passed the text content to Dialogflow. Once it responded, the response event was triggered, and the result was sent back to Facebook Messenger.

Before testing our bot, it is very important that you know when your app is in Development Mode. Plugin and API functionality will only work for admins, developers and testers of the app. After your app is approved and public, it will work for the general public.

To add a tester, you just need to go back to your Facebook App, find the “Roles” section and click on it. Then in the “Testers” section, click “Add Testers” button, it will throw you a popup window where you can add the Facebook ID or username for the person you want test your bot. Finally, click “Submit” button and the person can send a message to your bot throw your Facebook page.



[redacted]

Add Testers

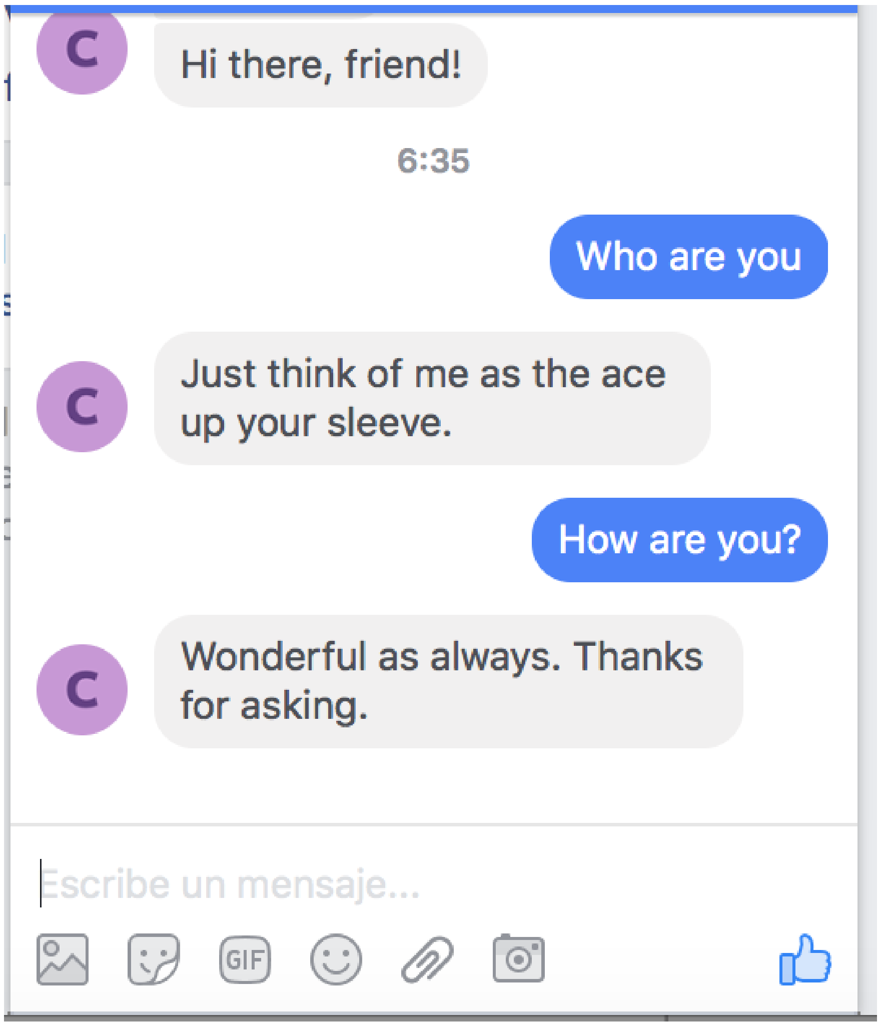
To add someone you aren't friends with on Facebook, enter their fbid or username.

Enter the name, fbid or username of the person you want to add

There are no testers for this app.

There are no Analytics users for this app.

Now, all that's left is to test our bot. If your server was running, turn it off and run index.js again. If all is well with your bot, you should get replies from the bot for some simple questions like "How are you?", "Who are you?", etc…



Hi there, friend!

6:35

Who are you

Just think of me as the ace up your sleeve.

How are you?

Wonderful as always. Thanks for asking.

Escribe un mensaje...

Awesome!

But guess what—this is just the beginning. There are so many other things that you can do with your bot.

As we mentioned before, an agent (your bot) is essentially the container or project and it contains intents, entities, and the responses that you want to deliver to your user.

Intents are the mechanisms that pick up what your user is requesting (using entities) and direct the agent to respond accordingly.

Step 4: Creating an Intent

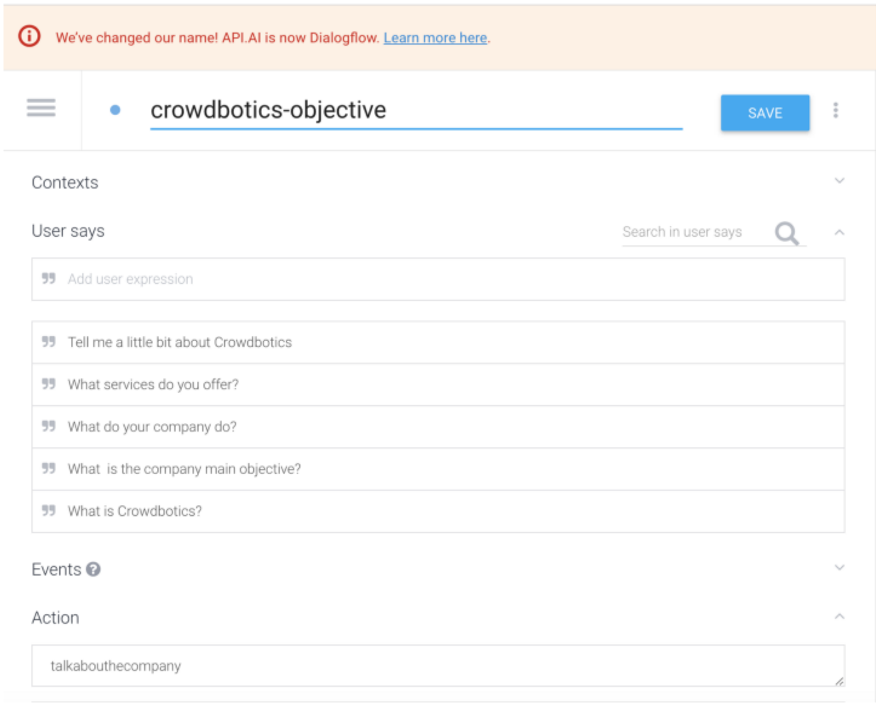
In order to create an intent, log into the agent you'd like to add the new functionality to. Go to the Dialogflow console, locate the section of intents, and click on the "+" button.

The sample intent for this agent is to teach it to answer simple questions. In this case we're going ask him about Crowdbotics.

To start, give a name to your intent. It is recommendable that you choose a name that describes what you want your intent do. In this case we will name it the "crowdbotics-objective".

Then in the "User says" section, add your first trigger sentence. As the name suggests, this is going to be how a user can ask your bot about something and why you need to add several examples for the bot to learn from.

To add a sentence, you just need to type it and the hit "Enter".


We've changed our name! API.AI is now Dialogflow. Learn more here.

crowdbotics-objective

Tell me a little bit about Crowdbotics

What services do you offer?

What do your company do?

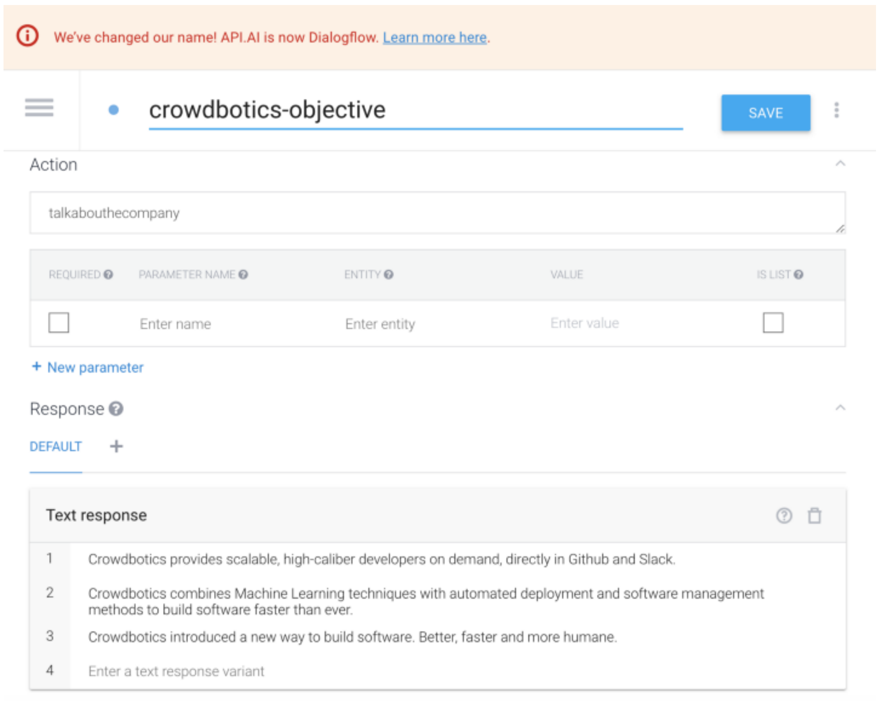
What is the company main objective?

What is Crowdbotics?

talkabouthecompany

Now there's a range of sentences the agent should understand, but you haven't told it what action is expected when it hears them. To do so, create an "action". It will need to be all lowercase with no spaces.

After your user has told the agent what it is expecting from it, you need to add some responses. To do that, go to the "Response" section and there you can add some different responses for your users questions.


We've changed our name! API.AI is now Dialogflow. Learn more here.

crowdbotics-objective

Action

talkaboutthecompany

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	Enter name	Enter entity	Enter value	☐

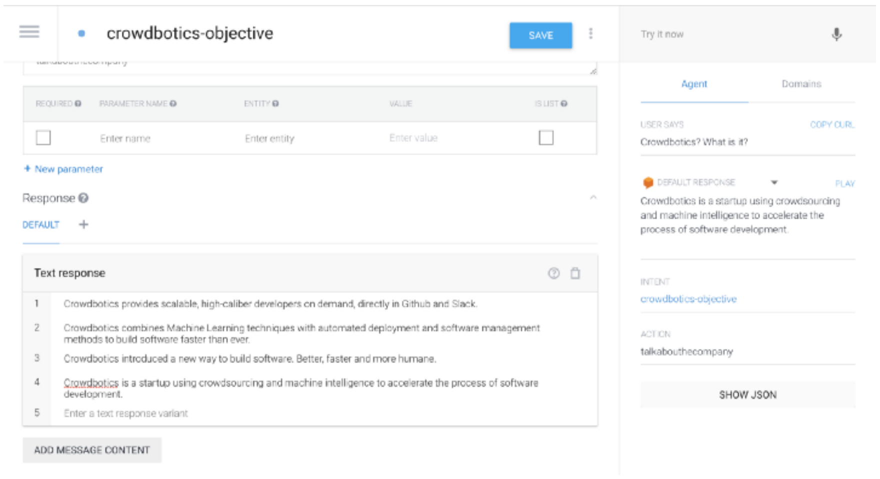
Text response

1. Crowdbotics provides scalable, high-caliber developers on demand, directly in Github and Slack.
2. Crowdbotics combines Machine Learning techniques with automated deployment and software management methods to build software faster than ever.
3. Crowdbotics introduced a new way to build software. Better, faster and more humane.
4. Enter a text response variant

Finally click the "Save" button next to your intent name to save the entire progress.

So now, one thing is missing—you have to try out your agent!

You can test your new intent by typing a test statement into the Console on the right. Type a question similar to one you already added the in "User says" section and watch the response.


crowdbotics-objective

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	Enter name	Enter entity	Enter value	☐

Text response

1. Crowdbotics provides scalable, high-caliber developers on demand, directly in Github and Slack.
2. Crowdbotics combines Machine Learning techniques with automated deployment and software management methods to build software faster than ever.
3. Crowdbotics introduced a new way to build software. Better, faster and more humane.
4. Crowdbotics is a startup using crowdsourcing and machine intelligence to accelerate the process of software development.
5. Enter a text response variant

USER SAYS: Crowdbotics? What is it?

DEFAULT RESPONSE: Crowdbotics is a startup using crowdsourcing and machine intelligence to accelerate the process of software development.

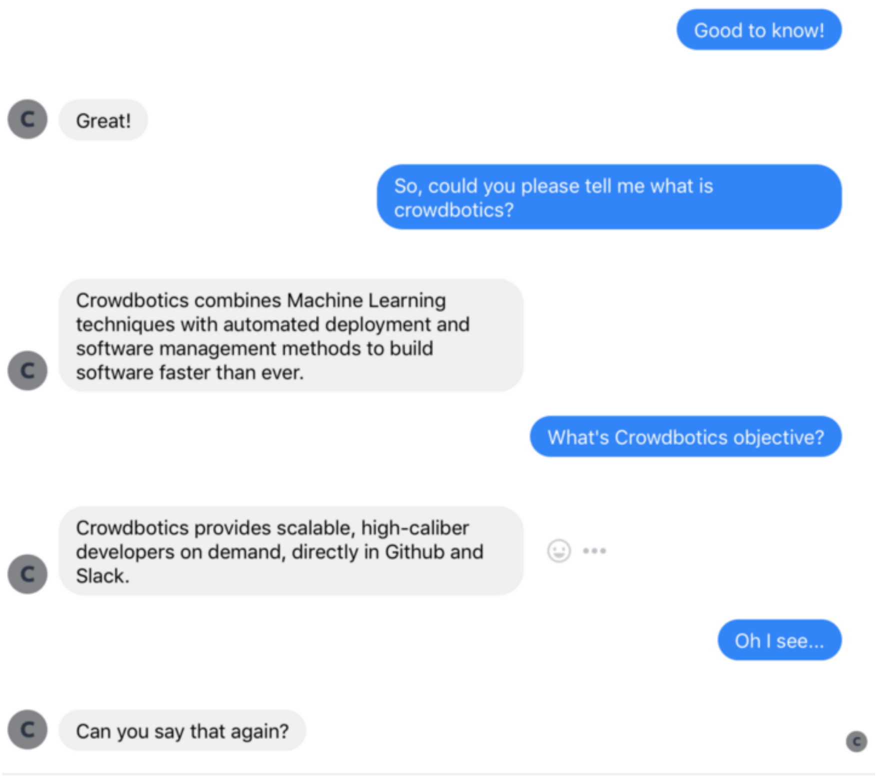
INTENT: crowdbotics-objective

ACTION: talkaboutthecompany

As you can see, our agent responds back with one of your trained responses. You can see the power of machine learning in action since even if you enter a question that you didn't define, the agent knows how to interpret it and return a response.

Remember, the more statements you add, the better your agent will be able to respond.

Finally, let's going try on Facebook Messenger.



Good to know!

Great!

So, could you please tell me what is crowdbotics?

Crowdbotics combines Machine Learning techniques with automated deployment and software management methods to build software faster than ever.

What's Crowdbotics objective?

Crowdbotics provides scalable, high-caliber developers on demand, directly in Github and Slack.

Oh I see...

Can you say that again?

And it is working! So, now you can create a simple but very efficient chatbot. You can add as much interactivity as you want. You can have your bot work with entities, make API calls, voice recognition, etc…

Thanks for reading! As you can see, creating a chatbot doesn't have to be as daunting as it seems.

Pilar works on new technology development with Crowdbotics. If you'd like to have Pilar work with you to help implement chatbots (or any other technology), please get in touch with us at Crowdbotics.