

# Implementação de Chatbots e a sua integração em plataformas online (redes sociais)

Seminário II - Mestrado em Computação Móvel - Instituto Politécnico da Guarda

16-07-2017

Daniel Fonseca nº 1009682

Definitivamente, 2016 foi um ano marcante para os Chatbots. Foi o ano em que grandes empresas (como Facebook, Microsoft, Google e até mesmo a Apple) decidiram abrir as suas plataformas para que milhares de desenvolvedores / designers / startups / empresas criassem chatbots.

## Tipos de chatbots

- **Chatbots baseados em regras**
  - São chatbots que funcionam através de comandos específicos (ou palavras chaves), ou seja, se você falar algo que o chatbot não conhece, ele não vai saber como agir—são limitados;
  - Geralmente eles obedecem fluxos de navegação bem definidos.
- **Chatbots baseados em inteligência artificial**
  - Tem a capacidade de entender o que você quer dizer através do que você escreve ou pergunta—ou seja, tem a capacidade de aprender e entender linguagem natural, não apenas comandos;
  - Aprende com o tempo e com outros serviços (dados). Quanto mais as pessoas usam, mais inteligente o chatbot fica.

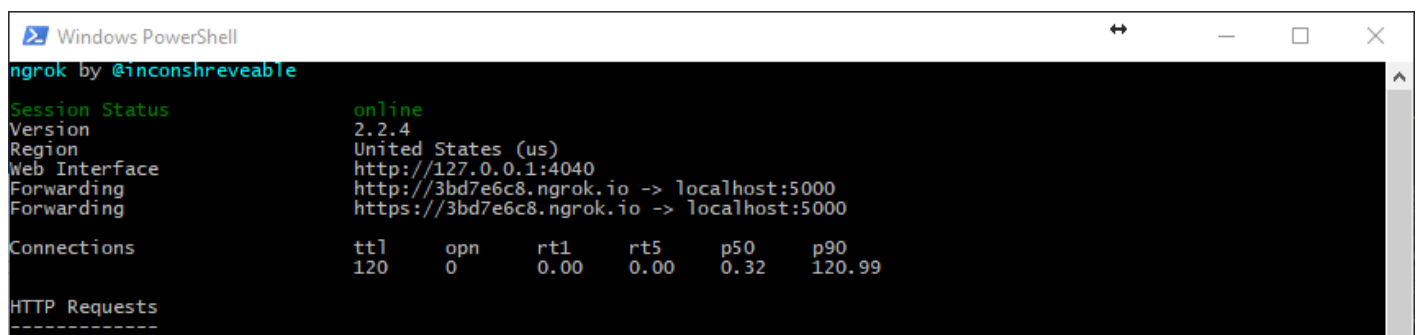
Para este tutorial vamos criar um chatbot baseado em regras, que vai ter uma pequena conversa e durante a conversa se a pessoa perguntar por animais o chatbot vai ao GETTY IMAGES e procurar por uma imagem do animal e publicar automaticamente na conversa. Para este tutorial vamos utilizar:

- [Facebook Messenger](#)
- [API.AI](#)
- [GETTY IMAGES API](#)
- [NodeJS](#)
- [ngrok](#)

O API.AI é a base para o chatbot que vai ser criado fazendo a ligação entre o Facebook e o Getty Images. Para este tutorial vamos basear o chatbot em regras e utilizar a opção do API.AI – “Prebuilt Agents” que permite realizar uma pequena conversa entre o utilizador e o chatbot e é baseado em inteligência artificial.

O primeiro passo a realizar é inicial o ngrok para podermos ter um link https, o Facebook só permite que as ligações chegam feitas em https por isso temos de utilizar o ngrok.

Comando: ngrok http 5000



```
Windows PowerShell
ngrok by @inconshreveable

Session Status      online
Version             2.2.4
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://3bd7e6c8.ngrok.io -> localhost:5000
                    https://3bd7e6c8.ngrok.io -> localhost:5000

Connections
  ttl    opn    rt1    rt5    p50    p90
   120    0     0.00   0.00   0.32   120.99

HTTP Requests
-----
```

Podemos verificar que o mesmo se encontra ligado e na porta 5000.

Próximo passo é irmos ao Facebook Developers: <https://developers.facebook.com/>

## Create a New App ID

Get started integrating Facebook into your app or website

Display Name

Chatbot

Contact Email

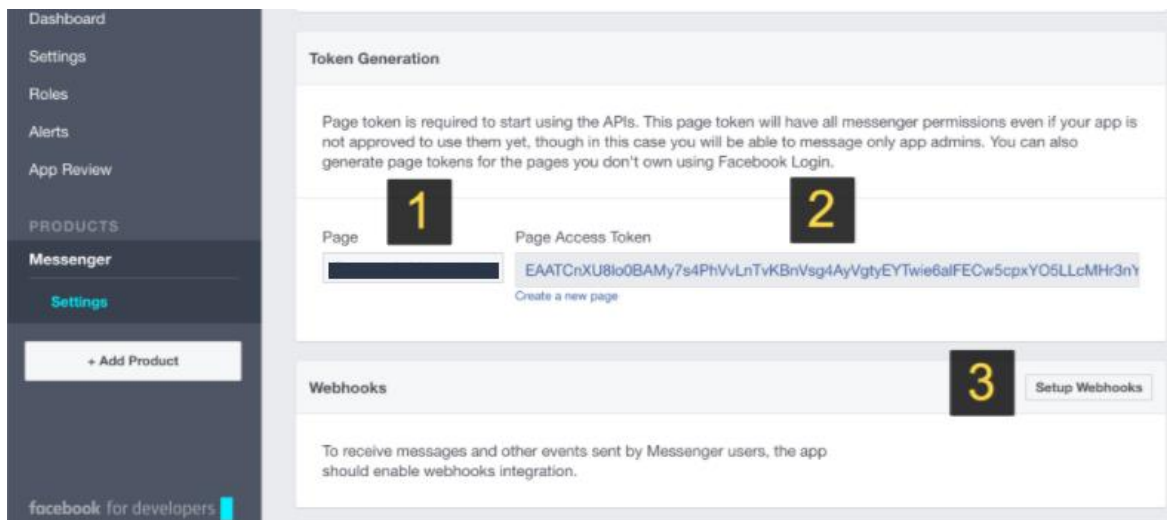
Used for important communication about your app

By proceeding, you agree to the [Facebook Platform Policies](#)

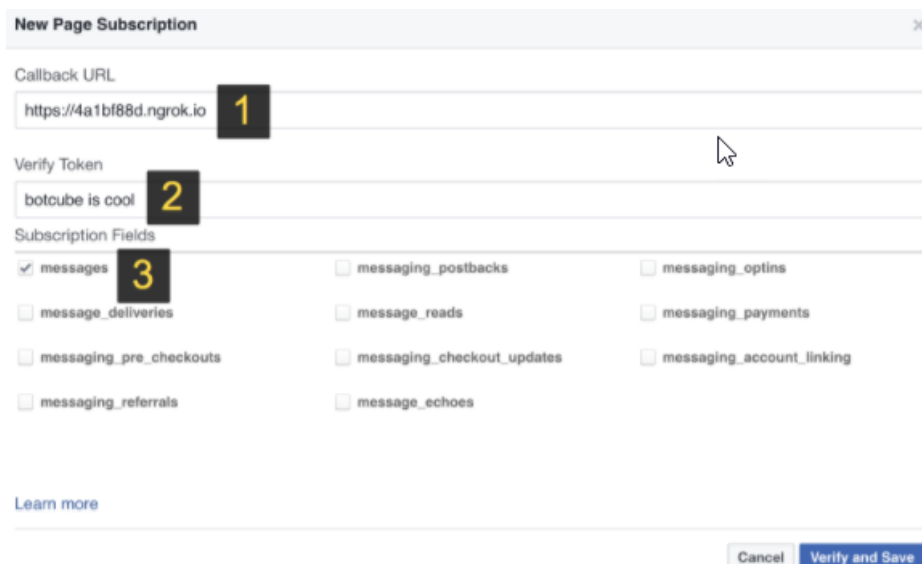
Cancel

Create App ID

Vamos adicionar um novo produto e seleccionar Messenger, no Token Generation vamos seleccionar a pagina que vamos utilizar. É nos apresentado um token que vamos guardar para utilizar futuramente.

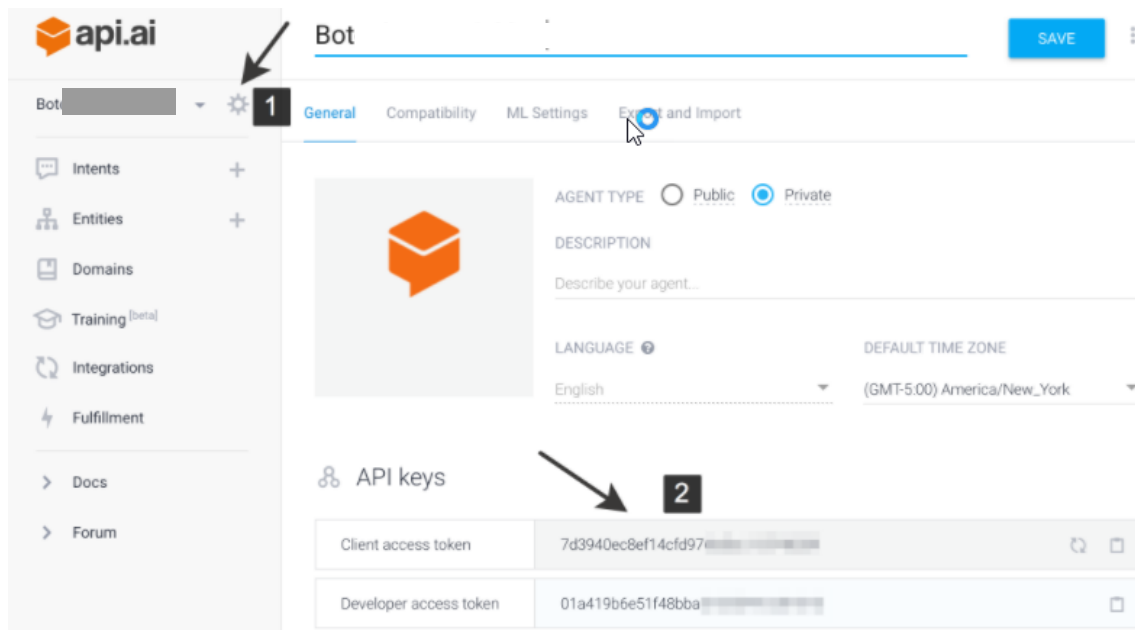


Carregando no Setup Webhooks é aberta uma janela onde vamos colocar o link que nos foi atribuído quando ligamos o ngrok, de seguida colocamos um token que vai servir como token de verificação entre o Facebook e o nosso chatbo, por ultimo seleccionamos que só vamos utilizar a opção de mensagens.

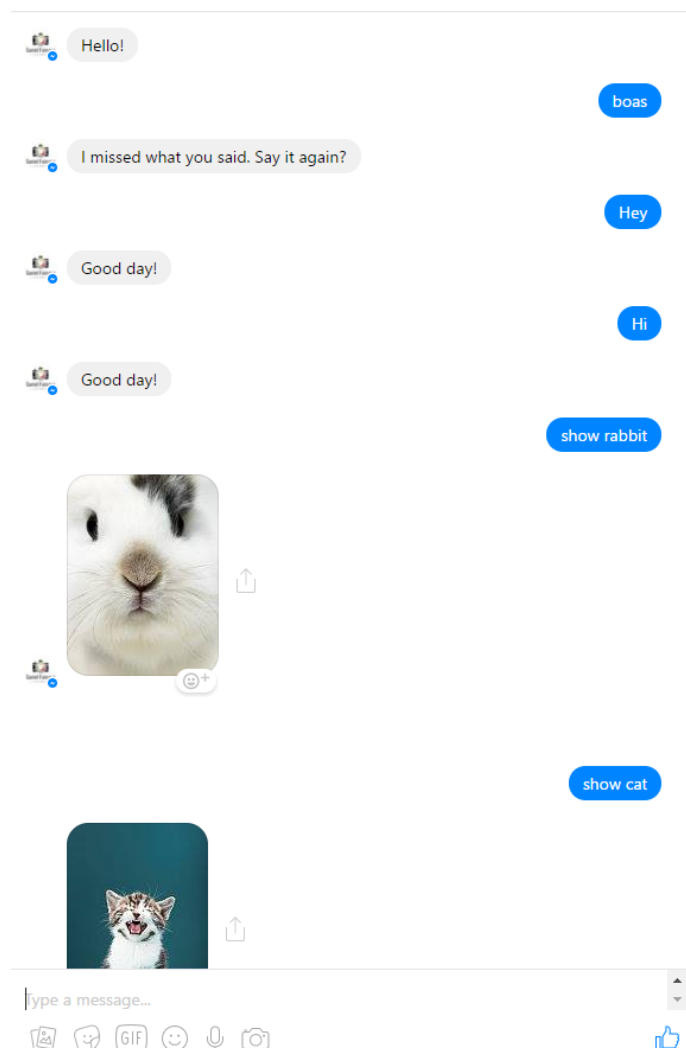


Do lado do Facebook developer encontra-se preparado para receber o nosso chatbot, vamos agora para o API.AI para criamos a nossa base.

Para fazer login no facebook é preciso ter uma conta da Google, realizado o login vamos criar um novo Agent



Criado o agente vamos passar para o node JS e criar o nosso projeto, em que vai receber do API.AI o nome do animal que o utilizador perguntou, tendo o nome do animal vai ao Getty Images e procurar a imagem mais popular relativo a esse animal, tendo a imagem vai fazer um POST com a imagem no Messenger.



O primeiro ficheiro a criar é o “index.js” onde vamos iniciar o servidor

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

const app = express();

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

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

app.listen(5000, () => console.log('Webhook server is listening at port 5000'));

app.get('/', verificationController);
app.post('/', messageWebhookController);
app.post('/image-search', imageSearchController);
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

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

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

app.listen(5000, () => console.log('Webhook server is listening at port 5000'));

app.get('/', verificationController);
app.post('/', messageWebhookController);
app.post('/image-search', imageSearchController);
```

O ficheiro seguinte a ser criado é o “processMessage.js”, onde vamos colocar o token do API.AI e o token do Facebook Developers

```
const API_AI_TOKEN = 'COLOCA O TOKEN DO TEU AGENTE DO API.AI';
const FACEBOOK_ACCESS_TOKEN = 'COLOCA O TEU TOKEN DO FACEBOOK DEVELOPERS';

const request = require('request');

const apiAiClient = require('apiai')(API_AI_TOKEN);

const sendImage = (senderId, imageUri) => {
  return request({
    url: 'https://graph.facebook.com/v2.9/me/messages',
    qs: { access_token: FACEBOOK_ACCESS_TOKEN },
    method: 'POST',
    json: {
      recipient: { id: senderId },
```

```

        message: {
          attachment: {
            type: 'image',
            payload: { url: imageUrl }
          }
        }
      }
    });
  });
};

const sendTextMessage = (senderId, text) => {
  request({
    url: 'https://graph.facebook.com/v2.9/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: 'CHRONO_ENG'});

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

    if (response.result.metadata.intentName === 'images.search') {
      sendImage(senderId, result);
    } else {
      sendTextMessage(senderId, result);
    }
  });

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

```

O próximo a ser criado é “imageSearch.js”, vai ser o que vai pesquisar no Getty Images a imagem, vamos ao [GETTY IMAGES API](#) e feito o login vai ser criada uma chave para utilizarmos no nosso projeto.

Application	Danielg
Key:	xa5g5hc
Secret:	ept7Szl
Status:	active
Created:	14 hours ago
<b>Key Rate Limits</b>	
	3 Calls per second
	1,000 Calls per day

```

const GETTY_IMAGES_API_KEY = 'COLOCAR KEY do GETTY IMAGES';

const request = require('request');

module.exports = (req, res) => {
  if (req.body.result.action === 'image') {
    const imageName = req.body.result.parameters['image_name'];
    const apiUrl =
'https://api.gettyimages.com/v3/search/images?fields=id,title,thumb,referral_destinations&
sort_order=most_popular&phrase=' + imageName;

    request({
      uri: apiUrl,
      method: 'GET',
      headers: {'Api-Key': GETTY_IMAGES_API_KEY}
    }, (err, response, body) => {
      const imageUrl = JSON.parse(body).images[0].display_sizes[0].uri;

      return res.json({
        speech: imageUrl,
        displayText: imageUrl,
        source: 'image_name'
      });
    });
  }
}

```

Criado o nosso projeto vamos voltar para o API.AI onde vamos criar um INTENT para o API.AI poder saber quando o utilizador esta a fazer um conversa em que momento o mesmo quer ver imagens de animais.

Para isso vamos criar um Intent com o nome de

The screenshot shows the API.AI console interface. On the left is a sidebar with navigation options: CHRONO\_ENG, Intents, Entities, Training, Integrations, Analytics, Fulfillment, Prebuilt Agents, Small Talk, Docs, and Forum. The main area is titled 'images.search' and contains several sections: 'Contexts', 'User says' (with a search bar), 'Events', 'Action' (with a text input), a table for parameters, 'Response' (with a 'Text response' section), and 'Fulfillment'.

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>

Below the table, there is a 'New parameter' link and a 'Response' section with a 'Text response' tab. The 'Text response' section has a list with one item: '1 Enter a text response'. There is also an 'ADD MESSAGE CONTENT' button and a 'Fulfillment' section at the bottom.

Criado o Intent vamos introduzir varias frases que o utilizador pode dizer para o chatbot lhe mostrar imagens de animais

User says

” Add user expression

” please show **leon**

” show **tiger**

” hey, show me please **tiger**

” hey, show me please **fish**

” hey, show me please **lion**

” hey, show me please **dog**

” hey, show me please **cat**

” hey, show me please **crocodile**

” hey, show me please **rabbit**

” hey, show me please **cows**

No Action vamos criar um nome parâmetro com o nome image\_name e com a Entity - @image\_name e com Value de \$image\_name e selecionar em todas as frases anteriores o nome do animal.

” hey, show me please **cows**

PARAMETER NAME	ENTITY	RESOLVED VALUE
image_name	@image_name	cows

1 OF 2

Events ?

Action

image


REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?
<input type="checkbox"/>	image_name	@image_name	\$image_name	<input type="checkbox"/>

Ultimo passo é selecionar no Fulfillment a opção de usar Webhook

Fulfillment

☒ Use webhook ☐ Use webhook for slot-filling

Salvamos o Intent e vamos para o Fulfillment onde vamos colocar o nosso link do ngrok e no final do link colocar /image-search



CHRONO\_ENG

- Intents
- Entities
- Training (beta)
- Integrations
- Analytics (new)
- Fulfillment**
- Prebuilt Agents
- Small Talk
- Docs

**Fulfillment**

**Webhook**

Your web service will receive a POST request from API.AI 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](#).

[Webhook example](#)

ENABLED ☒

URL\*

BASIC AUTH

HEADERS

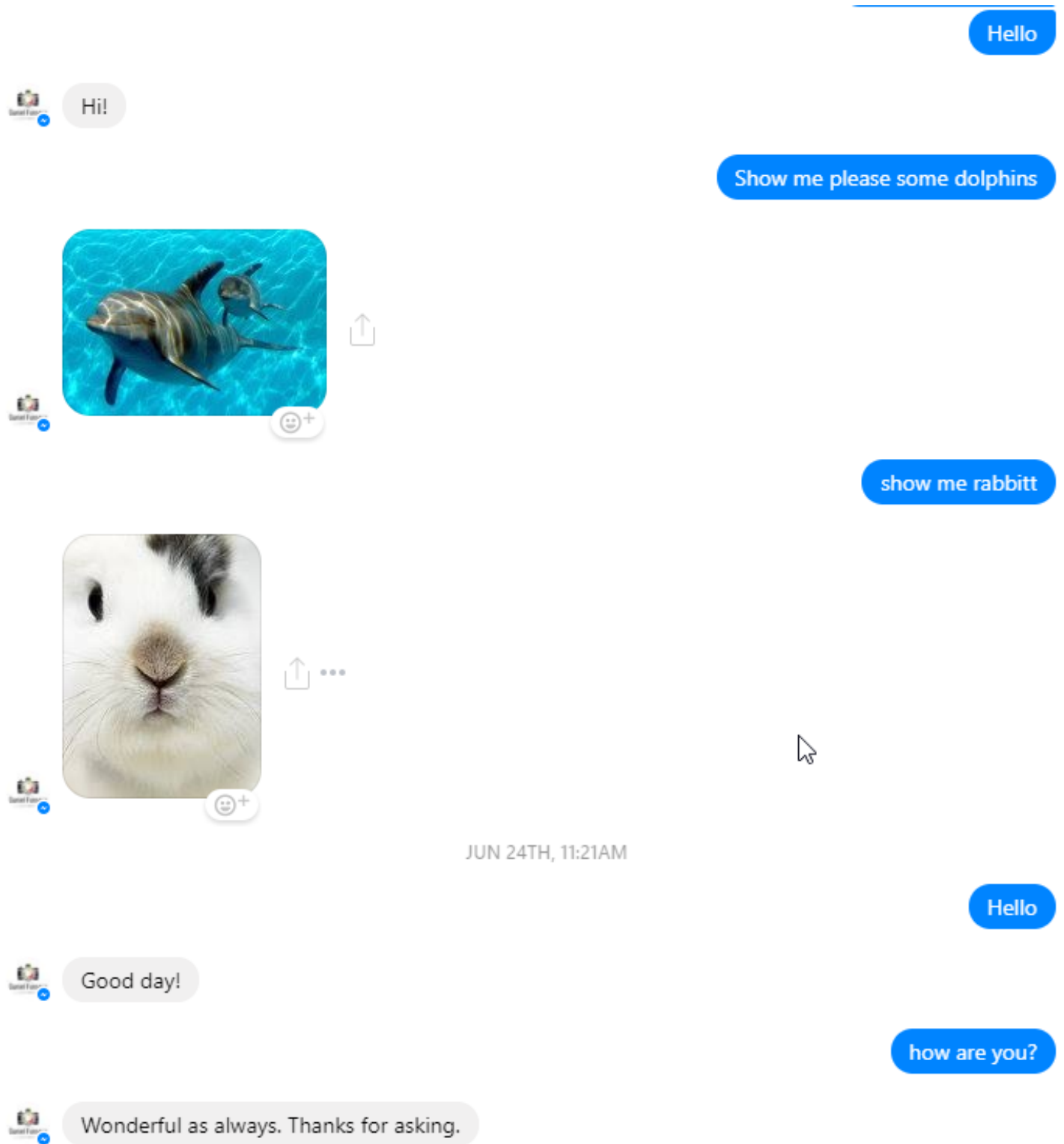
[Add header](#)

DOMAINS

[SAVE](#)

Neste momento já podemos iniciar o nosso servidor Node JS realizando o comando: node index.js

Vamos a pagina de Facebook e vamos conversar com a mesma para ver o chatbot a funcionar.



Como utilizamos 3 ferramentas diferentes para o nosso chatbot as mesmas proporcionam variações analises que podem ser verificadas a seguir.

O Facebook Developers permiti-nos verificar quantos utilizados estiveram a falar com o chabot o seu género e idade, permite também verificar quantas chamadas foram efetuadas, quantas falharam e também em media o tempo de resposta.

Do lado do API.AI temos a possibilidade de verificar quantas sessões tiveram os Intents e quantos pedidos por ultimo tempo medio de resposta.

Por ultimo o Getty Images permite verificar quantas chamadas foram efetuadas a sua API e media de latência quantos pesquisas de imagem tiveram sucesso e quantas falharam.



Facebook Developer



API.AI

Intents ?

Intent	Sessions	Count	Exit %	Agent response time
images.search	4	52	23.08%	1.40s 1.89s
Welcome	4	28	7.14%	0.01s 0.02s
Default Welcome Intent	1	1	0%	0.00s 0.09s

api.ai

CHRONO\_BNS

Intents

Entities

Training

Integrations

Analytics

Pullment

Prebuilt Agents

Small Talk

Docs

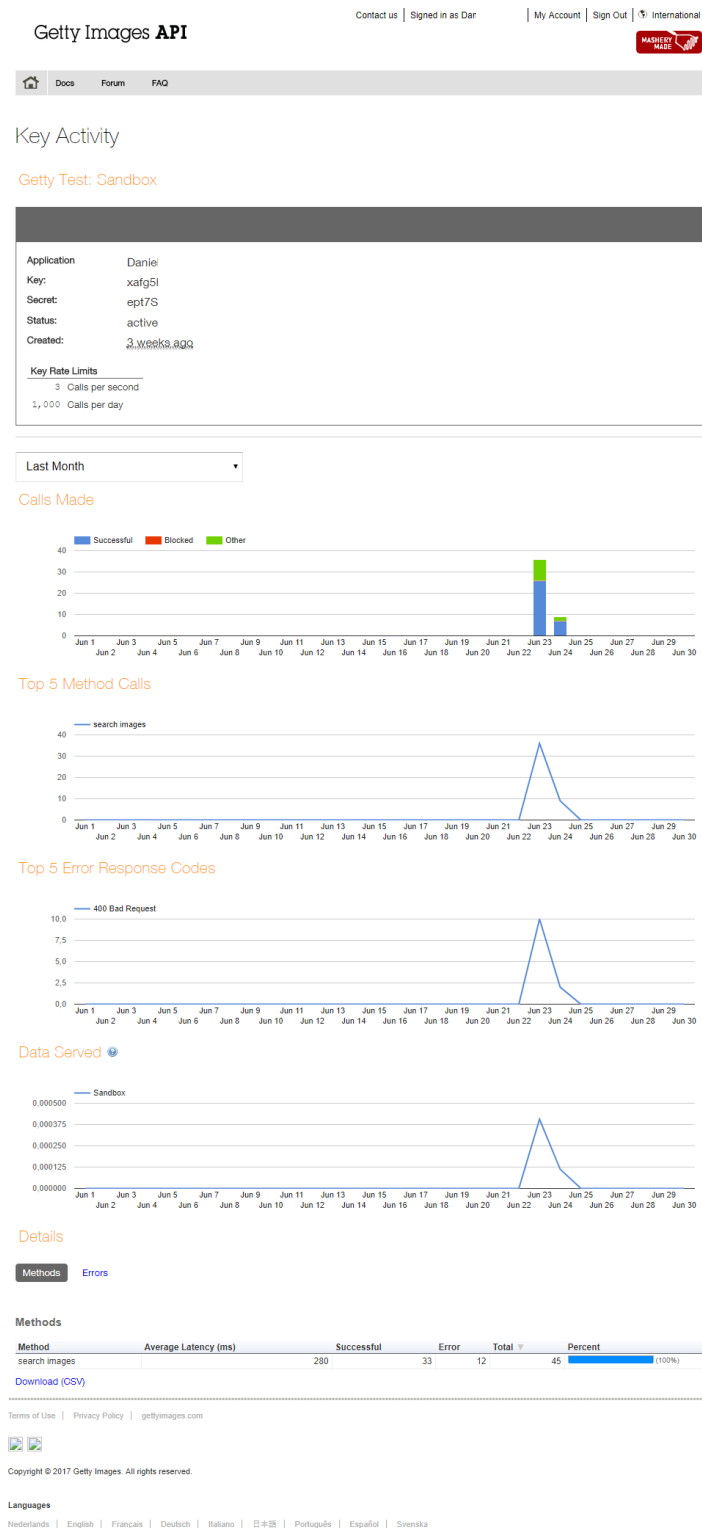
Forum

Support

Account

Logout





"Trocar mensagens são umas das poucas coisas que as pessoas fazem mais do que usar as redes sociais."

— Mark Zuckerberg, [The Verge](#)

O Chatbot estão para ficar e podemos verificar que eles não são difíceis de personalizar e utilizar. Para uma melhor experiencia só precisam de ser treinamos para poderem começando a desenvolver a capacidade de se adaptar ao meio para que foram criados.

<https://github.com/Danielgfonseca/SeminarioII>