

# CONCEPTION D'UN CHATBOT

**Megane Tcheeko  
Julien Tanga  
Finance ING4 GR02**

# DÉFINITION

Un chatbot est **un agent conversationnel**. Littéralement, il s'agit d'un robot (“bot”) qui converse (“chat”). Le chatbot est donc en effet un programme conçu pour dialoguer avec les utilisateurs via une plateforme.

Le chatbot se différencie des bots car il allie le **dialogue** et l'**automatisation**. Il a pour but d'échanger avec l'utilisateur pour lui apporter des réponses de façon automatisée.

# DEUX TYPES DE CHATBOT

- Les chatbots qui fonctionnent selon le **principe des mots clés**. Ils possèdent une base de donnée et fournissent les réponses en fonction de celle-ci et des mots clés saisis par l'utilisateur.
- Les chatbots qui permettent des interactions plus complexes car ils utilisent le **machine learning**. Ils analysent de façon sémantique les demandes de l'utilisateur en passant par le contexte et les synonymes.

# AVANTAGES DU CHATBOT

- Le chatbot fait partis des outils incontournables du marketing qui a été rendu possible grâce au progrès de l'**intelligence artificielle**. Cette technique de marketing est appelée le **marketing conversationnel**.
- Le chatbot permet une **prise de contact personnel** avec l'utilisateur et possible à tout moment (**24h/24 - 7j/7**).
- Il apporte un **réduction des coûts** pour l'entreprise qui l'utilise grâce à l'automatisation des tâches à faible valeur ajoutée.

# ETAPE DE LA CONCEPTION D'UN CHATBOT

- Définir les **objectifs** du chatbot
- Identifier les **besoins** des utilisateurs
- Définir et modéliser les **scénarios de conversations**
- Modéliser le bot et paramétrer l'intelligence artificielle (**NLP et machine learning**)
- Faire des **tests**
- Faire des **améliorations**



# DES FACTEURS CLÉS POUR UN CHATBOT

- **Le langage naturel** : Le chatbot étant en communication avec un utilisateur humain, il doit être développé de telle sorte qu'il comprenne le langage.
- **Le rythme de conversation** : le temps de réponse ne doit pas être trop long.
- **Le champ de compétence**: Il doit être clairement défini et la base de connaissance doit pouvoir être accessible.
- **La personnalité du chatbot**: Important pour une expérience plus personnalisée avec l'utilisateur.

# QUELQUES FRAMEWORKS UTILES À LA CONCEPTION D'UN CHATBOT

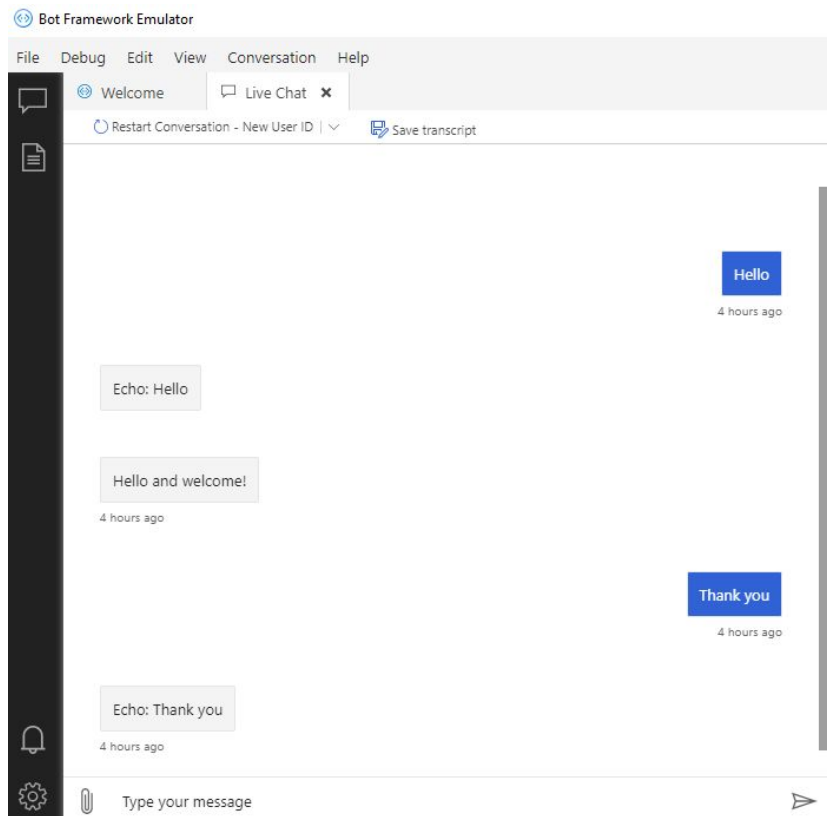
- **Microsoft Bot Framework** : va permettre de créer, connecter, publier et gérer les chatbots. Il existe des modèles prédéfinis pour interagir sur Skype ou Facebook Messenger par exemple.
- **Wit.ai** : permet de créer des bots mais aussi des interfaces vocales pour appareil mobile. Il est disponible en node.js, python et ruby.
- **Dialogflow** : permet de concevoir des chatbots et des assistants vocaux et est alimenté par l'apprentissage automatique de Google qui peut se connecter aux applications mobiles et aux sites web. Il est disponible en node.js, python, c#, java.

# QUELQUES FRAMEWORKS UTILES À LA CONCEPTION D'UN CHATBOT

- **Pandorabots** : Une plateforme d'intelligence artificielle. Il peut être intégré à des applications de messagerie et des sites web. Son SDK est disponible en Python, PHP, Java, node.js.
- **Chatterbot** : crée une bibliothèque python et est indépendant du langage. Il s'adapte en permanence car il traite chaque information reçu en entrée.
- **RASA** : il s'agit d'un framework open-source qui est basé sur l'apprentissage automatique.



# SIMULATION



Notre 1ère simulation était la création du bot “echo” qui réagissait en renvoyant ce qu’on lui avait dit.

# SIMULATION

## Sample HTTP Request

Postman

Curl

```
POST /knowledgebases/6045cdea-582f-4da8-b515-ea4ff0f2a6cb/generateAnswer
Host: https://bases-a143.azurewebsites.net/qnamaker
Authorization: EndpointKey b7498c2c-8de0-4529-9e53-c08e6542bfaa
Content-Type: application/json
{"question": "<Your question>"}
```

Copy

Close

Nous avons utilisé Azure de Microsoft pour la création d'une base de connaissance qui stockera les informations, réponses et questions des bots.

# SIMULATION

Home >

## Bases

Resource group

Search (Ctrl+F)

Overview

Activity log

Access control (IAM)

Tags

Events

Settings

Deployments

Security

Policies

Properties

Locks

Cost Management

+ Add Edit columns Delete resource group Refresh Export to CSV Open query Feedback Open in mobile Assign tags

Subscription ID : 076b0e41-95af-45a1-9405-eb2bd37b5c0b Location : east US

Tags (change) : Click here to add tags

Filter for any field... Type == all Location == all Add filter

Showing 1 to 5 of 5 records. Show hidden types No grouping List view

Name ↑	Type ↑	Location ↑	
Bases	Cognitive Services	West US	...
Bases	App Service plan	East US	...
Bases-b143	App Service	East US	...
Bases-ai	Application Insights	East US	...
basesa143-asacdfjgg6x56	Search service	East US	...

< Previous Page 1 of 1 Next >

# SIMULATION

 To export related resources, select the resources from the Resource Group view then select the "Export template" option from the tool bar.

☒ Include parameters ⓘ

Template


Parameters

Scripts

>  Parameters (2)

 Variables (0)

▼  Resources (3)

 [parameters('sites\_Bases\_a143\_name')]  
(Microsoft.Web/sites)

 [concat(parameters('sites\_Bases\_a143\_name'), '/web')]  
(Microsoft.Web/sites/config)

<<

1

2

3

4

5

6

7

8

9

10

{

"\$schema": "https://schema.management.azure.com/schemas/2019-04-01/  
deploymentTemplate.json#",

"contentVersion": "1.0.0.0",

"parameters": {

  "sites\_Bases\_a143\_name": {

    "defaultValue": "Bases-a143",

    "type": "String"

  },

  "serverfarms\_Bases\_externalid": {

    "defaultValue": "/subscriptions/0f6cb0e4-958f-45a1-9405-eb26d37be6cb/"

Follow link (ctrl + click)

# SIMULATION

Template Parameters Scripts

> Parameters (2)

Variables (0)

Resources (3)

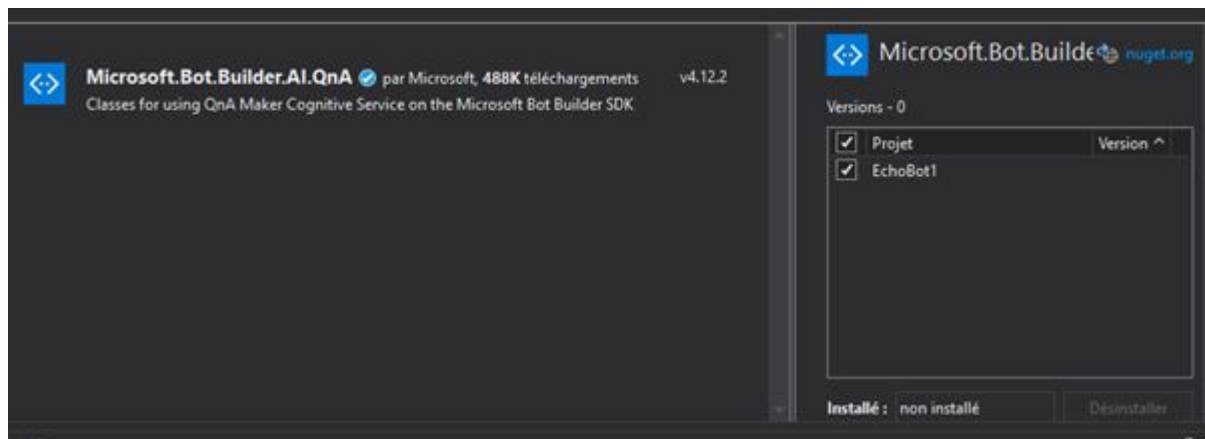
[parameters('sites\_Bases\_a143\_name')]  
(Microsoft.Web/sites)

[concat(parameters('sites\_Bases\_a143\_name'),  
'/web')]  
(Microsoft.Web/sites/config)

```
<< 27
28
29 {
30     "name": "bases-a143.azurewebsites.net",
31     "sslState": "Disabled",
32     "hostType": "Standard"
33 },
34 {
35     "name": "bases-a143.scm.azurewebsites.net",
36     "sslState": "Disabled",
37     "hostType": "Repository"
38 }
```

# SIMULATION

La bibliothèque  
Microsoft.Bot.Builder.AI.QnA ne  
s'intégrait pas  
dans notre code  
donc il nous était  
impossible  
d'importer la base  
de connaissance  
qu'on a créé sur  
Azure



# SIMULATION

```
using EchoBot1.Bots;  
using Microsoft.AspNetCore.Builder;  
using Microsoft.AspNetCore.Hosting;  
using Microsoft.Bot.Builder;  
using Microsoft.Bot.Builder.Integration.AspNet.Core;  
using Microsoft.Extensions.Configuration;  
using Microsoft.Extensions.DependencyInjection;  
using Microsoft.Extensions.Hosting;  
using Microsoft.Bot.Builder.AI.QnA;  
  
namespace EchoBot1  
{  
    2 références  
    public class Startup  
    {  
        0 références  
        public Startup(IConfiguration configuration)  
        {  
            Configuration = configuration;  
        }  
  
        4 références  
        public IConfiguration Configuration { get; }  
    }  
}
```

Les instructions suivantes servent à connecter notre bot Python à notre base de connaissances.

# SIMULATION

0 references

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers().AddNewtonsoftJson();

    // Create the Bot Framework Adapter with error handling enabled.
    services.AddSingleton<IBotFrameworkHttpAdapter, AdapterWithErrorHandler>();
    // Create QnA Maker endpoint as a singleton
    services.AddSingleton(new QnAMakerEndpoint
    {
        KnowledgeBaseId = Configuration.GetValue<string>("QnAKnowledgebaseId"),
        EndpointKey = Configuration.GetValue<string>("QnAAuthKey"),
        Host = Configuration.GetValue<string>("QnAEndpointHostName")
    });
    // Create the bot as a transient. In this case the ASP Controller is expecting an IBot.
    services.AddTransient<IBot, EchoBot>();
}
```

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.



# SIMULATION

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;

namespace EchoBot1
{
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        1 reference
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

# SIMULATION

En conclusion, le code est fonctionnel pour un bot simple qui renvoie le message de l'utilisateur. Cependant, à cause des problèmes d'importation dans les librairies NuGet de Microsoft.Bot.Builder.AI.QnA nous n'avons pas pu intégrer la base de connaissance que nous avons créée sur Azure. Nous avons tout de même rajouter les bouts de codes nécessaires.

# BIBLIOGRAPHIE

- <https://cloud.google.com/dialogflow/es/docs/tutorials/build-an-agent>
- <https://docs.microsoft.com/fr-fr/azure/bot-service/index-bf-sdk?view=azure-bot-service-4.0>
- <https://fr.wikipedia.org/wiki/Chatbot>
- <https://blog.hubspot.fr/marketing/chatbot-definition>
- <https://blog.hubspot.fr/marketing/guide-creer-premier-chatbot>
- <https://www.conseilsmarketing.com/chatbot-et-callbot/projet-de-chatbot-le-guide-de-a-a-z-pour-creer-son-premier-chatbot/>
- <https://geekflare.com/fr/chatbot-development-frameworks/#anchor-rasa-stack>
-