

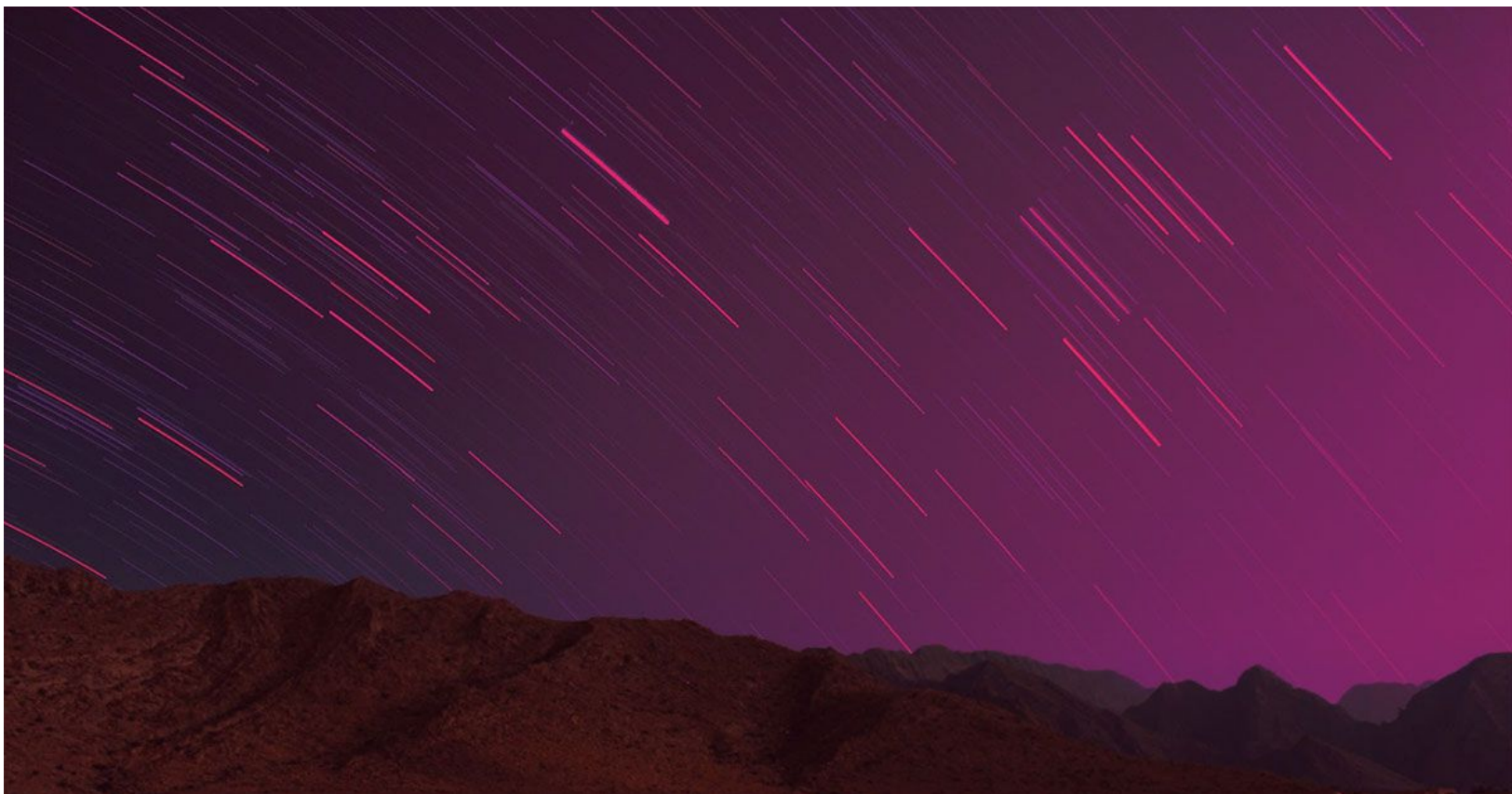


Workshop Google Assistant

Steven Klinger - Consultant IT - Vorthil.me

klinger.steven.contact@gmail.com





Introduction

- - - - x

Dialogflow est une interface permettant de créer simplement votre chatbot sans même rentrer une ligne de code si vous le souhaitez. Intégrant l'API [Cloud Naturel Language](#), par le biais d'intentions, le système va reconnaître (ou non) les requêtes de l'utilisateur et lui répondre de manière la plus correcte possible.

Il est désormais possible de créer des applications pour Google Home par ce biais, mais aussi pour d'autres supports, car nous allons créer une application pour Google Assistant.



Intents

- - - - x

Un “intent” représente une **corrélation** entre ce qu’un utilisateur va dire et l’action que notre plateforme va effectuer en conséquence, en somme, une intention.

Par défaut, il existe deux types d’intent déjà présents :

Default Fallback Intent : En cas d’incompréhension de la requête utilisateur, le message retourné par votre appareil google home sera mentionné dans cette catégorie d’Intent. A nous d’indiquer le message d’erreur retourné;

Default Welcome Intent : Comme son nom l’indique, le message d’accueil par défaut à l’entrée de votre application;



Entities

- - - - x

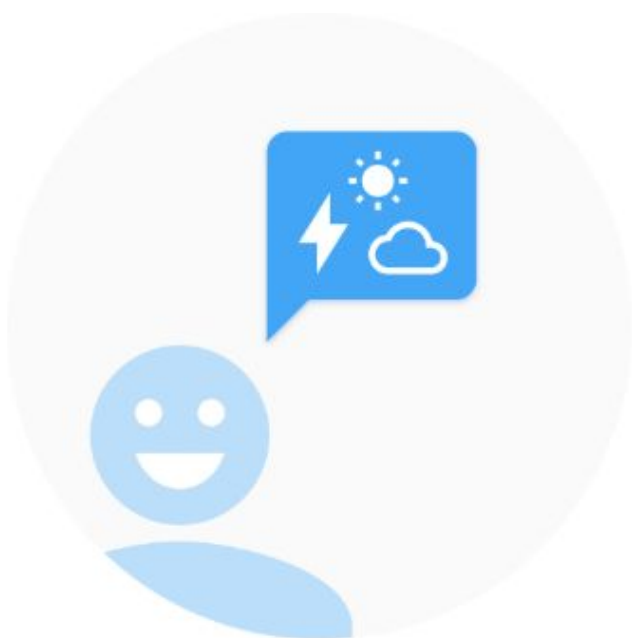
Dialogflow a besoin de savoir **quelles informations seront utiles** pour répondre à la requête de l'utilisateur. Ces données sont appelées des "Entities".

Une entité est une sorte de **mot-clé** que nous souhaitons identifier et récupérer dans la commande vocale de

l'utilisateur.

Particulièrement utile pour les recherches, des **entités réutilisables sont mises à disposition** afin de repérer efficacement tout type d'instructions : musiques, date, météo, etc...

Il est possible de définir des synonymes pour nos entités afin d'élargir les possibilités et de toujours être le plus précis possible dans la réponse attendue.



Context

- - - - x

Afin de **relier plusieurs requêtes**, ainsi que plusieurs réponses, nous avons besoin d'indiquer le **contexte actuel** de la requête utilisateur. C'est aussi utile pour différencier des phrases qui pourraient être vagues ou avoir plusieurs sens.

Par exemple, si jamais un de vos utilisateurs se trouve à Strasbourg, et demande ce qu'il se passe comme événements sur Nancy demain, vous lui fournirez une réponse. S'il demande par la suite : "Dites moi en plus", et bien vous aurez **besoin du contexte de la requête précédente**, c'est à dire la recherche d'événements sur Nancy, donc passer en paramètre via le contexte **"Nancy"** pour votre prochaine requête par exemple.



Fulfillment et Webhook

- - - - x

Quand les réponses que peuvent fournir Dialogflow ne vous correspondent plus, que vous avez besoin de quelque chose de plus poussé, de plus personnalisé, vous activez le “**fulfillment**” afin que dialogflow envoie les informations de l'utilisateur directement à votre **webhook**.

Ce qui **traitera les informations nécessaires** (par le biais de votre code en **node.js**) et répondra en conséquence, comme vous l'aurez indiqué !

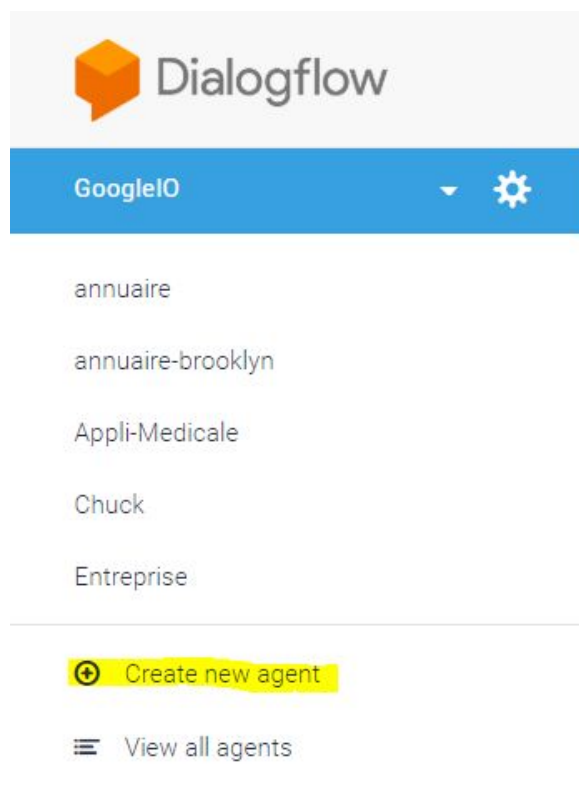
Créons notre application !

- - - - x

Etape 1 : Se connecter avec votre compte Google



Etape 2 : Créer votre "agent"



Remarques

- - - - x


Lorsque vous créez un nouvel agent, Dialogflow **crée automatiquement un projet Firebase**, qui est relié au Cloud de Google. C'est pour cela qu'un nouveau Google Project est lié, aussi pour le **partage** du projet si vous avez besoin !

Etape 3 : Humaniser votre Agent : Small Talk

☒ Enable

Small Talk Customization Progress

 100%

 About agent

 100%

 Courtesy

 100%

 Emotions

 100%

 Hello/Goodbye

 100%

 About user

 100%

 Confirmation

 100%

 Other questions and phrases

 100%

Afin de rendre votre agent plus apte à répondre aux petites phrases du quotidien, si tel est votre désir bien sûr, Dialogflow, dans sa version 2 permet aussi, via “Small Talk” de remplir très rapidement des réponses à des petites phrases du genre “Je suis fatigué”, “J’ai faim”, “Comment tu vas?”, “Il fait chaud” et ce, sans avoir à créer d’intents pour chaque requête utilisateur.

Etape 4 : Créez un "intent simple"

Utilité

SAVE

Contexts ?

▼

Events ?

▼

Training phrases ?

Search training phrases 🔍

^

” Add user expression

” Tu sers à quoi?

” Que fais tu ?

” Tu fais quoi?

Responses ?

^

DEFAULT

GOOGLE ASSISTANT

+

Text response

?

🗑

1

Je sers d'exemple pour la conférence !

2

Enter a text response variant

⌵

ADD RESPONSES

☐

Set this intent as end of conversation ?

Si vous souhaitez fermer l'application avec votre intent, vous pouvez cocher "Set this intent as end of conversation" :)

Etape 5 : Créez plusieurs intents qui se suivent avec des entités en paramètres

☐ Default Welcome Intent ^

Add follow-up intent



• ↳ Pseudo

Utiliser “follow-up intent” permet, comme son nom l’indique, d’ajouter un intent qui suivra directement l’autre. Plusieurs intents sont possibles, si vous souhaitez partir de zéro, prenez “custom” !

pseudo

SAVE

☒ Define synonyms ? ☐ Allow automated expansion


Vorthil	Vorthil, Steven
Despendo	Despendo, Allan
Click here to edit entry	


+ Add a row

Créer une entité permettant de stocker des pseudos par exemple. De ce fait, l’assistant google vous laissera entrer seulement si vous êtes un de ces pseudos là ! Sinon, elle ne vous comprendra pas :)

Etape 6 : Tester votre application ! (Bah oui quand même)


Via l'onglet "Integrations", vous avez la possibilité de tester votre application sur différentes plateformes. Choisissons Google Assistant, pour les besoins de notre Workshop.

 Google Assistant



To publish your Google Assistant App, specify invocation intents. Test with Google Assistant simulator, and then proceed to the Actions on Google console to submit the app.

LEARN MORE



Discovery

Explicit invocation *

Sign in required ?

Default Welcome Intent

Specify the intent that is triggered when users request the app by name (for example "Ok Google, talk to Personal Chef."). [Learn more.](#)


Implicit invocation

Sign in required ?

Utilité

Add intent

Specify intents that trigger "deep-link" actions in your app, allowing users to invoke specific functionality, such as "OK Google, ask Personal Chef for a hot soup recipe". Providing good action phrases. [Learn more.](#)




Auto-preview changes

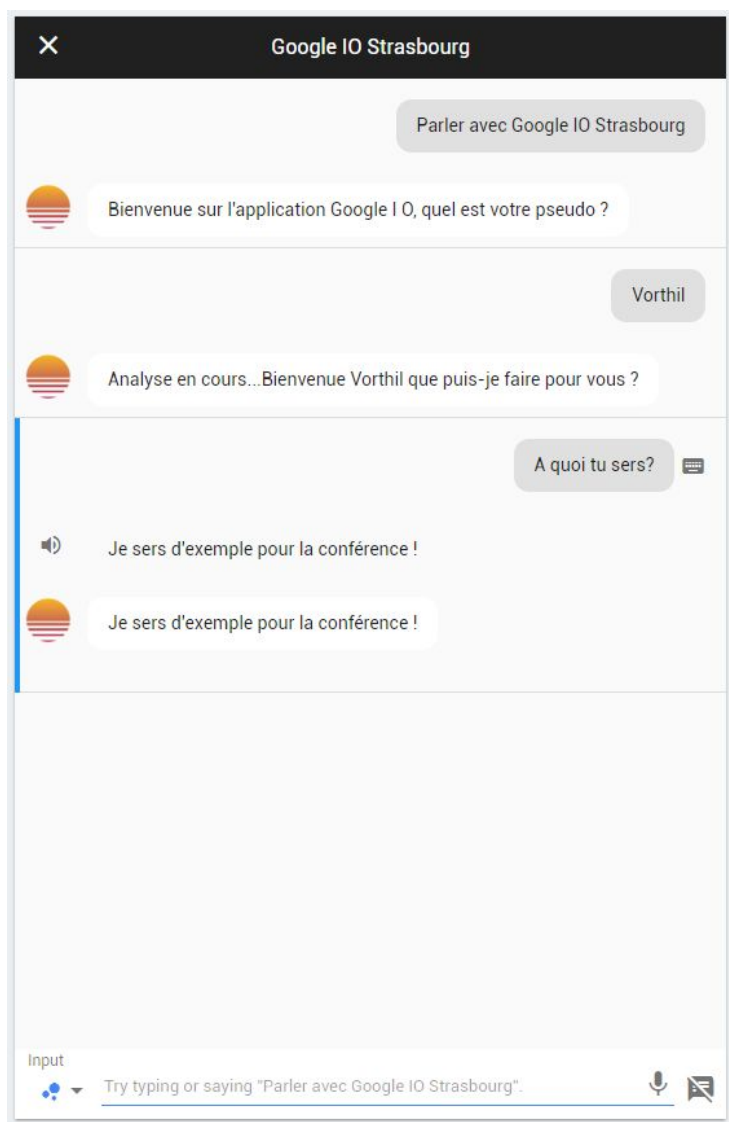
Dialogflow will propagate changes to the Actions Console and Assistant Simulator automatically.

CLOSE

TEST

MANAGE ASSISTANT APP 

Tester votre application sera possible même sans devices à portée de main ! Un environnement de test vous est proposé et vous permet de tester par la voix, ou par le texte votre agent.



Etape 7 : Allons plus loin, programmons un intent nous même

Comme expliqué précédemment, il est possible de créer votre agent entièrement avec `node.js`.

Pour ce faire, nous allons utiliser l'éditeur en ligne fourni par Dialogflow, utilisant les Cloud Functions pour Firebase.

Via un intent créé que nous appellerons sobrement "code", nous allons utiliser l'entity pseudo.

Cet intent nous permettra d'associer un nom à un âge, donc d'utiliser un paramètre d'entrée pour demander l'âge des utilisateurs existants dans notre entité.

Il faut activer le fulfillment dans votre intent afin de laisser la gestion de la réponse au code que nous allons écrire.

Responses ?



DEFAULT GOOGLE ASSISTANT +

Text response



1 Enter a text response



ADD RESPONSES

☐ Set this intent as end of conversation ?

Fulfillment ?



☒ Enable webhook call for this intent

☒ Enable webhook call for slot filling

Lien du code : https://github.com/StevenVorthil/Workshop_GoogleIO2018/

Inline Editor (Powered by Cloud Functions for Firebase)

ENABLED



Build and manage fulfillment directly in Dialogflow via Cloud Functions for Firebase. [Docs](#)

```
index.js  package.json  
```

```
1  'use strict';
2
3  const {dialogflow} = require('actions-on-google');
4  const functions = require('firebase-functions');
5
6  const app = dialogflow({debug: true});
7
8  app.intent('code', (conv, {pseudo}) => {
9    console.log('Pseudo : ' + JSON.stringify(pseudo));
10    let age = '42';
11    let pseudos = pseudo;
12    switch (pseudos) {
13      case `Vorthil`:
14        age = '24';
15        conv.ask(`L'âge de ${pseudos} ` + `est de ` + age + " ans");
16        break;
17      case `Despendo`:
18        age = '26';
19        conv.ask(`L'âge de ${pseudos} ` + `est de ` + age + " ans");
20        break;
21      default:
22        conv.ask(`L'utilisateur demandé n'existe pas dans mon système`);
23    }
24    //conv.close(`You said ${pseudo}`);
25  });
26
27  exports.dialogflowFirebaseFulfillment = functions.https.onRequest(app);
```