Gebruikersanalyse

Rick van Schijndel  
Open Innovation minor

**Geschiedenis**

|  |  |  |
| --- | --- | --- |
| Datum | Auteur | Wijzigingen |
| 9 oktober 2019 | Rick van Schijndel | Eerste opzet, context |
| 11 oktober 2019 | Rick van Schijndel | Test uitvoering, feedback |
| 25 oktober 2019 | Rick van Schijndel | Resultaat |

Inhoudsopgave

[Context 3](#_Toc23157333)

[DialogFlow prototype 4](#_Toc23157334)

[Doelstelling 4](#_Toc23157335)

[Implementatie 4](#_Toc23157336)

[Testsessie 5](#_Toc23157337)

[Scenario’s 5](#_Toc23157338)

[Scenario 1: Absolute tijd 5](#_Toc23157339)

[Scenario 2: Relatieve tijd 5](#_Toc23157340)

[Scenario 3: Complete interactie 5](#_Toc23157341)

[Testdetails 5](#_Toc23157342)

[Resultaten 6](#_Toc23157343)

[Observaties 8](#_Toc23157344)

[Feedback 8](#_Toc23157345)

[Conclusie 8](#_Toc23157346)

# Context

Voor de Open Innovation minor (semester 7, richting Software Engineering) werken studenten zelfstandig of in een groep aan een project. De op te leveren producten, betrokken tooling/technieken en planning is geheel te bepalen door de studenten – mits er hier een externe product owner of stakeholder bij betrokken wordt met overzicht op het gehele traject.

In dit document beschrijf ik gedetailleerd de aard van mijn project, werkmethodiek, tooling en het op te leveren eindproduct.

De Open Innovation minor loopt van 2 september 2019 tot begin februari (ong. 20 weken).

# DialogFlow prototype

## Doelstelling

In deze gebruikersanalyse wil ik met een simpel chatbot prototype de **functionele eisen** van mijn uiteindelijke applicatie achterhalen. In DialogFlow – een “human-AI interactie tool” waarin makkelijk prototypes kunnen worden gemaakt zonder programmeren – stel ik een simpele “demo” op van mijn AI-systeem met een aantal standaard functionaliteiten zoals het reserveren van een ruimte.

Met dit prototype wil ik zonder begeleiding een aantal testgebruikers de taak geven om een kamer proberen te reserveren. Mijn verwachting is dat dit compleet fout zal gaan; maar dit geeft mij juist een duidelijk inzicht in welke uitspraken/commando’s gebruikers het meest gebruiken en welke functionaliteiten de chatbot dus juist af dient te vangen.

## Implementatie

Op 9 oktober heb ik in DialogFlow het prototype opgesteld met als hoofdfunctionaliteit de mogelijkheid om de “huidige” kamer te reserveren met de volgende parameters:

* Starttijd (verplicht)
* Eindtijd (optioneel) **of**
* Duratie (optioneel)

Verder bevat de bot nog een aantal standaardreacties voor begroetingen, complimenten, beledigingen en verwarring wanneer het een commando niet begrijpt. In het volgende hoofdstuk ga ik een aantal gebruikers vragen om zonder mijn hulp een kamer te reserveren en noteer ik de resultaten.

# Testsessie

## Scenario’s

Om gebruikers een *klein* beetje op weg te helpen (een paar, niet allemaal) schrijf ik drie scenario’s die ze kunnen volgen.

### Scenario 1: Absolute tijd

Gebruiker heeft om 15:00 in de middag een vergadering gepland met zijn product owner. De duratie/eindtijd is onbekend. Er dient nog wel een lege ruimte voor te worden gereserveerd. Gebruiker reserveert, in één zin, een lege ruimte vanaf 15:00.

### Scenario 2: Relatieve tijd

Gebruiker heeft over 1 uur een stand-up met zijn projectgroep die 15 minuten zal duren. Gebruiker zegt daarom eerst tegen de chatbot dat hij een kamer nodig heeft voor 15 minuten. Wanneer de chatbot pas vraagt om een specifieke starttijd, geeft gebruiker aan **over 1 uur (“in 1 hour”)**, in plaats van een daadwerkelijke tijd.

### Scenario 3: Complete interactie

Gebruiker bevat een goed humeur en begroet de chatbot op vriendelijke wijze (bijvoorbeeld “Good day!”). De chatbot groet terug.  
Dan vraagt gebruiker heel erg netjes om deze kamer te reserveren **tot** 16:00. Wanneer de chatbot vraagt om een starttijd geeft gebruiker 12:00 aan. De chatbot bevestigt de start- en eindtijd. Dan dankt gebruiker de chatbot voor de hulp.

## Testdetails

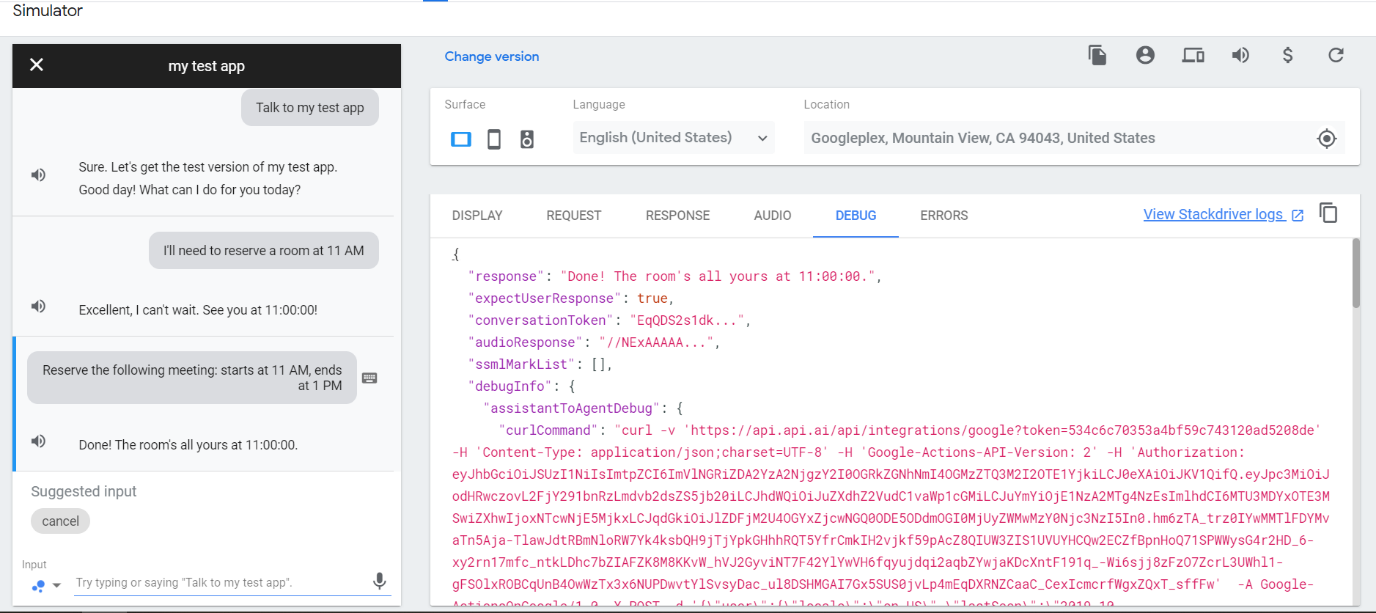
Op donderdag 10 oktober heb ik in totaal 7 verschillende mensen de chatbot laten testen. DialogFlow biedt de mogelijkheid om deze tests op desktop/laptop uit te voeren of op mobiele telefoon; ik koos voor mobiele telefoon omdat mensen misschien gewend zijn om dezelfde soort gesprekken te houden met Google Assistant.

Helaas bevat DialogFlow of Google Assistant op dit moment een bug waarbij de microfoon input automatisch sluit na een paar seconden. Dit betekent dat gebruikers zeer snel moeten praten of – in mijn geval – ik hun uitspraken handmatig overgetyp naar de chatbot.

De resultaten van de testsessie noteer ik hieronder.

## Resultaten

Van de 7 tests die zijn uitgevoerd gingen er maar 2 mis – de chatbot haalde soms parameters door de war of begreep de gebruiker niet goed. Dit is in het algemeen al een stuk positiever dan ik verwacht had, maar dat betekent ook dat ik voornamelijk de positieve interacties mee moet nemen naar het volgende prototype.

  
*Het DialogFlow/Google console dashboard op desktop-weergave.*

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

*Gesprekken met als resultaat een succesvolle reservering.*

|  |  |
| --- | --- |
|  |  |

*Gesprekken die iets “minder goed” zijn verlopen.*

## Observaties

Als positief aspect is op te merken dat (wegens het gebruik van standaard “sys.time” parametertypes in DialogFlow) er veel ondersteuning is voor verschillende manieren om een tijd te bepalen:

* Om 3 uur
* Om 15:00
* Over 10 minuten

Het is voor mij onbekend of een vergelijkbaar systeem al bestaat in C# of Java. Zo niet zal ik dit handmatig moeten programmeren.

Ook was het een aangename verassing dat DialogFlow standaard uit een lijst van opgegeven responses alleen responses kiest die de door jouw opgegeven parameters bevatten.  
*“The room’s all yours for 30 minutes, starting at 15:00:00”* zal bijvoorbeeld nooit getoond worden als de gebruiker niet specifiek aangeeft dat de kamer maar voor 30 minuten nodig is.

Soms lukt het de chatbot niet om parameters meteen in één zin te herkennen. Het kan bijvoorbeeld *“I want a room until 13:00”* verkeerd opvatten als een reservering **vanaf** 13:00. Waarschijnlijk betekent dit dat ik niet genoeg gevarieerde oefeningszinnen opgegeven heb in DialogFlow.

Een ander… niet zo handig detail is dat de chatbot blijkbaar standaard de chat verlaat na een aantal mislukte commando’s. Deze standaardfunctionaliteit kan ik in DialogFlow eenvoudig uitschakelen – en in mijn uiteindelijk hoofdproduct kan ik zelfs de bot zo programmeren dat deze steeds meer geïrriteerd raakt en daardoor andere reacties geeft.

## Feedback

Aangezien de testsessie een heel stuk beter ging dan verwacht is het voor mij makkelijk om de specifieke tekortkomingen aan te geven en te verbeteren in de volgende versie van mijn chatbot;

* Parameters worden soms niet goed in-line herkend.
* Standaardinstellingen zorgde ervoor dat de chatbot een beperkt aantal pogingen doet tot interpretatie.
  + Echter is dit wel een leuk idee dat ik eventueel toe zou kunnen passen in een latere sprint.

Interessant om op te merken is dat ik geen verschil heb gezien tussen de verschillende gebruikersgroepen en relevante interacties. Zowel de studenten als docenten die mijn prototype hebben getest zijn merendeels tot een succesvol resultaat (gereserveerde kamer) gekomen en beide groepen zijn ieder verantwoordelijk voor één “mislukte” reservering.

## Conclusie

Het doel van deze analyse was om te achterhalen welke functionaliteit en spraak gebruikers verwachten van het reserveringsysteem en op welke manier zij deze functionaliteit proberen te gebruiken. Betreft het opvangen van commando’s concludeer ik dat ik zeker goed zit met de testregels en herkenning die het prototype op dit moment bevat. De eerder genoteerde misinterpretatie en ongewenste “time-out” kan ik eenvoudig verbeteren mits DialogFlow als framework nog steeds gebruikt wordt in de volgende chatbot iteratie.