Voorblad

# INHOUDSOPGAVE

[INHOUDSOPGAVE 2](#_Toc536372240)

[1. Inleiding project NabAlexa 3](#_Toc536372241)

[2. Eindproduct 4](#_Toc536372242)

[3. Persoonlijke resultaten 5](#_Toc536372243)

[3.1 David Dijkhoff 5](#_Toc536372244)

[3.2 Jordy Beerse 5](#_Toc536372245)

[3.3 Wijbe Zwager 5](#_Toc536372246)

[3.4 Jesse Piebenga 5](#_Toc536372247)

[3.5 Jeffrey Rozendaal 5](#_Toc536372248)

[3.6 Tsjerk Hiemstra 5](#_Toc536372249)

[4. Conclusie 6](#_Toc536372250)

[5. Reflectie 7](#_Toc536372251)

[5.1 David Dijkhoff 7](#_Toc536372252)

[5.2 Jordy Beerse 7](#_Toc536372253)

[5.3 Wijbe Zwager 7](#_Toc536372254)

[5.4 Jesse Piebenga 7](#_Toc536372255)

[5.5 Jeffrey Rozendaal 7](#_Toc536372256)

[5.6 Tsjerk Hiemstra 7](#_Toc536372257)

# Inleiding project NabAlexa

# Eindproduct

# Persoonlijke resultaten

## David Dijkhoff

Tijdens de stage die ik het derde studiejaar liep heb ik veelvuldig voice applications moeten ontwikkelen voor Amazon Alexa en Google Assistant. Ik had hierdoor een duidelijke voorsprong op de rest van de groep als het aankomt op voice application development, waardoor besloten is dat ik mij volledig zou gaan focussen op het maken van een voice skill voor Amazon Alexa.

Ik heb gekozen om een voice skill te maken die een toevoeging biedt in het dagelijkse, huishoudelijke leven. Zo kunnen er losse taken uitgevoerd worden, zoals het aan en uitdoen van lampen, kachels of ventilators. Ook kunnen er complexere opdrachten uitgevoerd worden. Hierbij kan gedacht worden aan taken die uitgevoerd worden zodra je thuis komt, op bed gaat of wakker wordt.

De voice skill die ik gemaakt heb maakt onder andere gebruik van session attributes en persistent attributes. De session attributes maken het mogelijk om data op te slaan tijdens een conversatie. Deze functionaliteit wordt onder andere gebruikt om op te slaan wat Alexa als laatste zei (voor het geval de gebruiker iets niet verstond en het opnieuw wil horen).

De persistent attributes worden gebruikt om data op te slaan voor conversaties in de toekomst. Dit kan bijvoorbeeld handig zijn om de naam van de gebruiker op te slaan. Hierdoor kan de spraak-assistent meer als een persoonlijke hulp gezien worden.

Verder maakt de voice skill ook gebruik van twee API’s; namelijk die van de Nederlandse Spoorwegen (NS) en filmtotaal. De informatie die uit deze API’s verzameld wordt, wordt per email naar de gebruiker gestuurd.

Daarbovenop kan de voice skill de smartphone van de gebruiker bellen, in geval deze kwijt is. Om dit mogelijk te maken, is gebruik gemaakt van IFTTT (If This Then That).

Alle functionaliteiten die betrekking hebben op ‘home automation’, maken gebruik van de Simple Queue Service (SQS) die Amazon aanbiedt voor gebruikers van Amazon Web Services. Elke keer als een gebruiker iets in het huis aan of uit wil zetten, wordt er een push-bericht gestuurd naar de SQS-service. Vervolgens leest een python-script (genaamd queue-listener) dezelfde queue uit om hier vervolgens apparaten mee aan of uit te zetten.

Een volledige lijst van alle functionaliteiten die de voice skill bezit is als volgt:

*Losse taken:*

* Lampen aan / uit
* Kachel aan / uit
* Ventilatie aan / uit
* Waterkoker aan / uit
* Huidige temperatuur opvragen
* Trein-informatie van huis naar werk opvragen voor het komende uur
* Films die vandaag op de Nederlandse tv spelen opvragen
* Smartphone van gebruiker bellen
* Gebruikersnaam toevoegen of veranderen

*Home Automation taken:*

* Thuis komen:
  + Lampen aan
  + Afhankelijk van huidige temperatuur ofwel kachel of ventilatie aandoen
  + Films per email toesturen
* Wakker worden:
  + Lampen aan
  + Waterkoker aan
  + Afhankelijk van huidige temperatuur ofwel kachel of ventilatie aandoen
  + Trein-informatie voor volgende uur per email toesturen
* Naar bed gaan:
  + Lampen uit
* Het huis verlaten:
  + Lampen uit
  + Afhankelijk van huidige temperatuur ofwel kachel of ventilatie uitzetten

Voor het maken van deze voice skill voor Amazon Alexa is gebruik gemaakt van Alexa Developer Console en Amazon Web Services (AWS).

De Alexa Developer Console wordt gebruikt voor het hosten van de voice skill zelf. De voice skill moet vanaf daar een link leggen met de code die geschreven is voor iedere functie. Hiervoor legt het een link met een lambda-fucntie in Amazon Web Services. Lambda host de code in de cloud, waardoor het altijd bereikbaar is voor de voice skill.

Het is echter erg onhandig en tijdrovend op de gemaakte code iedere keer op de website van AWS Lambda te uploaden. Daarom heb ik besloten een shell-script te schrijven die de code met behulp van de AWS-SDK automatisch upload naar de Lambda-functie die gelinkt staat aan de voice skill op Alexa Developer Console.

## Jordy Beerse

## Wijbe Zwager

## Jesse Piebenga

## Jeffrey Rozendaal

## Tsjerk Hiemstra

## *Alexa skills*

Een van de eerste dingen waar ik mee aan de slag ging was het maken van een simpele Hello World skill voor Alexa. Aanleiding hiervoor was dat ik nog nooit met iets als Alexa had gewerkt, en wou leren hoe dit werkt, zodat ik later in het project zou kunnen helpen met het maken van skills die iets toevoegen aan de functionaliteiten van het konijn.

Echter, ondanks meerdere pogingen met gebruik van meerdere tutorials, is hier geen resultaat uit gekomen. Op de online console van Alexa kwam een leeg antwoord, waardoor het ook niet duidelijk is waardoor dit komt. Zelfs als alles hetzelfde was als bij David, met uitzondering van het account dat wordt gebruikt, komt er een leeg antwoord.

## *RFID Plugin*

Vervolgens ben ik aan de slag gegaan met het konijn: voornamelijk het gebruik van de RFID tags. In het begin waren de waardes van de tags nog onbekend, tot deze werden uitgelezen met gebruik van de SystemOutLogger Plugin. Deze werden vervolgens in een switchcase gezet, zodat er onderscheid kan worden gemaakt tussen de RFID tags vanuit de code.

Daarna werd deze plugin gebruikt om het konijn zelf dingen te laten doen, zoals het draaien van de oren en het veranderen van de kleur van de leds. Al snel was ontdekt hoe je een choreography aanmaakt, een bestand dat zegt wat de oren en leds moeten doen. Dit wordt toegevoegd aan een messageblock, dat vervolgens wordt toegevoegd aan een packet dat vanaf de server wordt gestuurd naar het konijn.

Nadat ontdekt werd hoe je met de server de oren naar een absolute positie (van 0 tot 18) kunt sturen, werd ook al snel ontdekt hoe je ze naar een relatieve positie kunt sturen. Beiden kunnen voorwaarts en achterwaarts draaien. Ook het veranderen van de kleur van alle vijf led lampjes los van elkaar volgde snel. Deze leds kunnen elke kleur krijgen die je wilt, door de rgb waarden meegeven.

Het afspelen van muziekbestanden volgde hierna. De manier om dit de doen werd redelijk snel ontdekt, maar de Nabaztag kan maximaal 50 seconden aan audio afspelen.

Vervolgens werd er gekeken naar het inladen van andere plugins vanuit de code. Ook dit werkte redelijk snel. Als meerdere plugins gebruik maken van hetzelfde event, dan wordt ook de code van beide plugins aangeroepen.

Ten slotte is er ook nog een choreography gemaakt die de leds en oren beweegt in een loop, terwijl er muziek wordt gespeeld.

## *API’s*

Alexa en Google Assistant kunnen al veel dingen, maar nog niet alles. Gelukkig bestaan er API’s waar je informatie uit kunt halen, en die informatie kun je doorgeven naar Alexa en Google Assistant. Daarom heb ik gekeken naar interessante API’s die wat kunnen toevoegen aan de functionaliteit van de Nabaztag.

Al snel werd duidelijk dat er veel is dat of niet interessant genoeg is, of dat deze services al kunnen doen wat de API’s kunnen doen.

De API’s waar ik mee heb gewerkt zijn van NS. Vanwege het eerder beschreven probleem met Alexa, heb ik deze niet daarvoor kunnen schrijven. In plaats daarvan heb ik de API’s geschreven voor een simpele webpagina in Javascript, zodat David het makkelijker kan schrijven voor Alexa. De API die uiteindelijk is toegepast op Alexa is de Journeyplanner. Je geeft op van welk station je wilt vertrekken, waar je wilt aankomen, of je de vertrektijd of aankomsttijd meegeeft en hoe laat je vertrekken of aankomen wil. Vervolgens filtert de code de treinen die al vertrokken zijn, sinds de API deze treinen wel meegeeft, en geeft de eerste twee treinen terug.

Verbinding maken met de API bleek lastiger dan gedacht. Alle pogingen om een API call met Authorization te doen gaven een Cross-Origin Resource Sharing error. Om dit te omzeilen wordt er niet direct naar de API een call gedaan, maar via CORS-Anywhere.

## *Radio Plugin*

Het konijn kon ooit een radio stream afspelen. Dit moest weer terug komen. Al snel werd ontdekt hoe dit moet. Je voegt aan een messageblock een addPlayStreamCommand toe, waar je een url aan meegeeft. Echter, het was niet zo simpel als het klinkt. Vele url’s zijn geprobeerd. Deze url’s komen onder andere uit .pls en .m3u bestanden die gedownload zijn van Shoutcast.com, maar ook url’s van andere sites.

Uiteindelijk werd ontdekt dat in de broncode van de server van de makers van Nabaztag zelf deze stream niet was geimplementeerd. Uiteindelijk is het niet gelukt om een radio plugin aan de praat te krijgen.

# Conclusie

# Reflectie

## David Dijkhoff

## Jordy Beerse

## Wijbe Zwager

## Jesse Piebenga

## Jeffrey Rozendaal

## Tsjerk Hiemstra

Tijdens dit project heb ik op verscheidende onderdelen gewerkt. Omdat de radio plugin niet gelukt is en Alexa op mijn account niet wou werken, heb ik niet genoeg werkende eindproducten als ik zou willen hebben. Ondanks dat vind ik dat ik goed gewerkt heb aan het product, en mijn steentje bijgedragen heb.