AMAZON ALEXA WEATHER SKILL IMPLEMENTATION

Andrej Dobrik
Bachelor Degree Programme (3), FEEC BUT
E-mail: xdobri01@stud.feec.vutbr.cz

Supervised by: Jiri Pokorny
E-mail: jiri.pokorny@vutbr.cz

Abstract: In the past few years there has been a quite significant shift in how customers interact with their smart devices. Strong emphasis is laid on the usage convenience of the device. For people, the most convenient way to communicate is through voice interaction. Many companies have realized this change and started to develop software and hardware to support it. The main focus of this work is the development of a new unique Amazon Alexa skill that recommends appropriate windsurfing equipment based on weather conditions.

Keywords: AWS, Amazon Echo, custom skill, resource linking, Lambda function, smart home

1 INTRODUCTION

The interaction shift challenged the idea of simplified household interaction with news, media or smart home appliances. To accompany this interaction there are several voice assistants on the market. Most popular among the users are considered Amazon Alexa, Google Home and Apple Siri.

This work is devoted to the development of a new Amazon Alexa skill for the Amazon Voice Service (AVS). The AVS technology was first introduced alongside the Amazon Echo in the November of 2014 [1]. It introduced new voice–forward experience and enabled developers to build countless new skills that were able to connect to other services and products. Through the AVS, Amazon has simplified the creation of conversational interfaces for device makers [2]. This allows developers to add Alexa and intelligent voice control to new products, bringing convenient voice experience to customers [3].

2 DEVELOPMENT OF THE SKILL

This skill is aimed at windsurfers, in particular windsurfers that are in the initial process of learning the sport and could potentially have trouble with determining the right equipment for the wind conditions. This skill will require a multilayer interaction model, because the interaction will consist of multiple communication sequences. For instance, the invocation phrase will include the respective location followed by the response of the weather forecast details with an option whether the user wishes to receive the suggestions for the suitable equipment.

The weather forecast data requires extraction from a weather forecast source. The decision was to opt for an API source website OpenWeatherMap.org. This API service offers 5 subscription plans: free, startup, developer, professional and enterprise. The distinguishing element between these subscription plans is the API requests per minute, ranging from 60 calls per minute to 200k calls per minute. For the purpose of this Alexa skill, 60 calls per minute will suffice. A call can be specified as an API request issued to the provider.

Furthermore, this API provides an access to current weather forecast for any location, including over 200k cities, with frequent updates based on global models and the data from more than 40k weather

stations. The API data is available in JSON, XML or HTML format. For the API to be accessible however, an account needs to be registered to acquire the API key. The http request includes the city ID element and the API key element [1].

The API responds with JSON data structure that includes all necessary weather forecast data. This chunk of JSON data was rather illegible, because the API response had no proper text formatting to clearly read out the data. An extension for the Chrome browser was downloaded for the purpose of easier navigation through the data. This extension formatted the chunk of data into the root elements of the API response that include the individual data provided by the API. After reviewing the documentation and the API request output, an additional adjustment needed to be done to the URL. By adding `&units=metric` to the URL the unit system that the data is represented in was changed to metric system.

2.1 Interaction Model of the Skill

The user invokes the skill with a predefined invocation phrase and Alexa triggers the Launch Request with the Lambda response to the user, asking the user to specify the particular place for which the weather forecast should be looked up. User then responds with an utterance specifying the windsurfing spot location.

Based on the utterance, an Intent Request is triggered and an API request is issued, this process is illustrated in Figure 2. After parsing the data pulled from the API request a response is structured, including the forecast data, followed by a question whether the user wishes to receive a recommendation for an appropriate windsurfing equipment. Based on the response, the session either ends or the skill continues with the session. This process is illustrated in Figure 1.

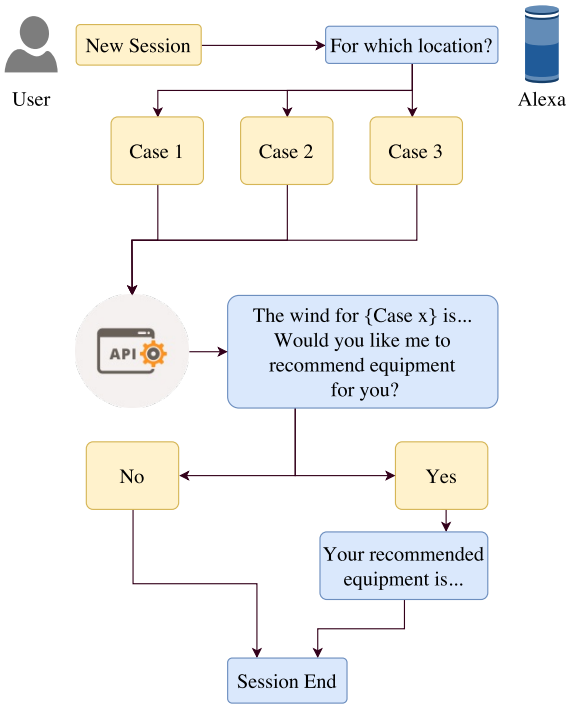

Figure 1: Control Flow Diagram

```
case "IntentRequest":
    console.log(`INTENT REQUEST`)

    switch(event.request.intent.name) {
     case "GetCaseOneData":
       var endpoint = "http://api.openweathermap.org/data/2.5/
           weather?id=3061186&APPID=21ae391f98005fcae6
           5c368747276905&units=metric"

       var body = ""
       https.get(endpoint, (response) => {
        response.on('data', (chunk) => { body += chunk })
        response.on('end', () => {
          var data = JSON.parse(body)
          var windSpeed = data.wind.speed
          var windDeg = data.wind.deg
          context.succeed(
            generateResponse(
              buildSpeechletResponse(`The weather forecast for
              {city} is ${windSpeed} meters per second with
              ${windDeg}  degrees  direction.`, true),
              {}
            )
          )
        })
       })
       break;
```

Figure 2: Data pull code logic

[1] `http://api.openweathermap.org/data/2.5/weather?id=CITYID&APPID=APIKEY&units=metric`

2.2 SKILL EXECUTION

The logic behind the skill itself is executed from a linked Lambda function. This source linking allows for a real time testing of the skill, with the help of the Amazon Resource Number that is generated with the function. The function requires a trigger for it to be executed, therefore an Alexa Skills Kit trigger was selected. The Amazon CloudWatch log and the inline text editor where a Node.js 6.10 runtime was selected, reside within the Lambda console. With this configuration, the JavaScript code was written and executed. Whenever any function is written on Lambda, it needs to be wrapped inside a function handler with a signature `exports.handler = (event, context) {}`, which is a default signature for the Lambda function. The custom skill itself, or rather any skill in particular, consists of four separate events in the whole life cycle of the skil, which are: New Session, Launch Request, Intent Request, Session End Request. To handle the functionalities of these requests, helpers are used to generate the actual data structure returned from the Lambda function onto the Amazon Alexa Service to be then passed to the Echo device. The `generateResponse` helper will serve this purpose.

The `speechletResponse` is a data structure that specifies the output text for the device to present to the user, as well as whether it should end the session or leave the session open for more requests. Additional functionalities can be implemented, such as the reprompt text which is what the Amazon Echo says when it does not receive an input for a period of time. For a value to be returned from each request, a call out to `context.succeed` and `generateResponse` passed to it from the helper is used. A simple way to demonstrate this is in the launch Request, where after receiving the `context.succeed` a response is generated and the session is consequently eneded. This code can be tested directly in the Lambda console or through the ARN in the Amazon Developer portal. For the intent request, a `switch` statement with `(event.request.intent.name)` parameters is used to execute the domain logic for each intent within. The endpoint to which the function will call out is in a URL form of Openweathermap data API. After the initial HTTP request, a response is generated. The response obtains the data from the weather API. The resulting data are parsed from the response followed by the `content.succeed` call out.

3 CONCLUSION

This work described the development and functionality of a custom Alexa skill. APIs of the weather skill were successfully requested and their data parsed. Based on their values a response was structured. This skill was subjected to different testing scenarios to verify its functionality and possible errors, enclosing the development process in Amazon developer portal and source linking of the Lambda function. These tests were conducted with the help of the Amazon developer console, Lambda test events and the Echosim.io Alexa emulator. The Alexa emulator replaced the need of physical Echo device to be present. The skill behaved according to the programming with a few speech recognition failures.

REFERENCES

[1] L. Nicholls, Y. Strengers, and S. Tirado, "Smart home control: Exploring the potential for off-the-shelf enabling technologies in energy vulnerable and other households," 2017.

[2] N. Chandak and A. Joshi, "An intelligent remote controlled system for smart home automation," *International Research Journal of Engineering and Technology (IRJET)*, 2018.

[3] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.