DOI: https://doi.org/10.30837/csitic52021231818

Model of Integration of Voice Assistants in the Construction of Web Applications

Mykola Slisarenko[1]
Oleksii Liashenko[2]

[1]Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine, mykola.slisarenko@nure.ua
[2]Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv UA-61166, Ukraine, oleksii.liashenko@nure.ua

Abstract. *Today, voice assistants can radically change the interaction of computer users. For many users, the ability to read and print is impossible to access information. Voice assistants can fill the information gap for these users. Embedding these capabilities in today's consumer technology would be much more cost-effective than specially designed devices and many users would find it convenient to control these devices.*

Keywords: *voice assistant, OAuth 2.0, NLU, Google actions, intents, conversation scenes, web-hooks, REST API, JSON, JWT.*

I. Introduction and Problem Statement

The complexity and accuracy of voice recognition technology by voice assistants has increased significantly in recent years. Currently available voice assistant products from Apple, Amazon, and Google allow users to ask questions and translate the answers of assistants into natural language. There are many possible applications of this technology in the field of web applications.

Voice Assistants are best suited for easy and clear use, allowing users to quickly access the information they need. The tasks performed by the voice assistant include:

- queries that can be easily answered. Actions that can be performed using data that does not require additional search, such as time or date. For example, creating a reminder;
- quick but useful actions. They are usually informative and provide benefits in a very short amount of time, such as finding out when their order will be delivered;
- actions that do not require additional tools and are controlled only by voice. For example, listening to reminders for the whole day.

However, integrating voice assistants with a web application is not an easy task. There are many cases. First, most programs have user data to identify them. With this approach, the assistant may require access to the user's profile during the conversation, including the username, email address, and profile image. The user can perform the entire stream through the voice, which provides the ability to log in without additional difficulty [1-7].

Secondly, it is necessary to recognize the user not only after confirming the use of personal data, but also in the subsequent use of the assistant. To achieve this goal, a Bearer token is used, which provides user identification when accessing a web application using a voice assistant. Google Sign-In, Google Secure Authentication, and OAuth2.0 - a standard authentication protocol, are used to link accounts. Third, if you need to access user services (mail, calendar) you need to get additional access from the user. This process is different from the classic ones, because there is no interface through which we can redirect the user to a page where he can confirm his choice. However, the server can provide a response that will involve the mobile device in the access procedure and only then will the server be able to access the user's services [2-3].

Thus, we can identify the problem of implementing an application for a voice assistant and implementing a server part to interact with the application using technologies that will identify users and obtain their data to provide the best user experience.

II. Problem Solution and Results

When building a voice assistant application, it is important to understand the model in which the application interacts with the voice assistant and the assistant with the web server. Actions in Google allow create conversational actions using the Action SDK, Action Designer, or both. This feature allows choosing the best workflow for any application.

The Action SDK provides a standardized file-based schema for building your actions, a library for interacting with the Assistant, and a CLI for deploying and managing your project. Action Builder is built on the same technology as the Action SDK, and allows you to create with an easy-to-use and powerful IDE.

Conversation actions allow extending Google Assistant with its own conversational interfaces that give users access to web application. Actions apply a powerful natural language understanding (NLU) mechanism to the assistant to process and understand natural language input and implement work based on that input. A conversational action is a simple object that defines an entry point in conversations. During a call or conversation, the action may trigger a web hook that notifies the execution service to perform certain tasks [4].

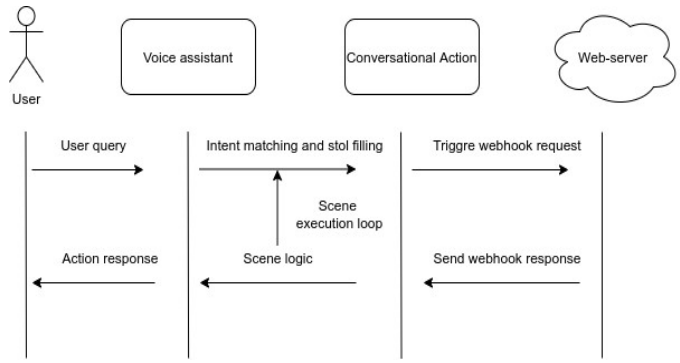

Figure 1. Conversation model using web hook

Figure 1 describes the general way web hooks work for voice assistant application:

- at specific points of action's execution, it can trigger a web-hook that sends a request to a registered web-hook handler (web-server) with a JSON payload.

- Web-server processes the request, such as calling a REST API to do some data lookup or validating some data from the JSON payload. A very common way to use fulfillment is to generate a dynamic prompt at runtime so conversations are more tailored to the current user;
- Web-server returns a response back to action containing a JSON payload. It can use the data from the payload to continue its execution and respond back to the user.

To handle any action intents has to be created. They represent for assistant application the scenario needs exact action to be carried out. Such as some user input that needs processing or a system event that you need to handle. Intents needs to be used to build invocation and conversation models.

When creating user intentions, you must specify the following items:

- global indication of intent, which determines whether the execution time of the assistant can correspond to the specified user's intention during the call, as well as during the conversation. By default, the assistant can only meet the user's intentions during a call. Only intentions that are marked as public are eligible for a call from the beginning of the conversation;
- learning phrases are examples of what a user can say to match intent. The Assistant NLU system naturally extends these learning phrases to include other, similar phrases. Providing a large set of high-quality examples increases the quality of intent and accuracy of compliance;
- parameters - is the data collected that must be obtained from user input. To create a parameter, you need to annotate the learning phrases in types to inform the NLU mechanism that parts of the custom input have been extracted. You can use system types or create your own custom types for parameters.

When the NLU mechanism detects a match in a user's input, it retrieves the value as a typed parameter, so you can make logic with it in the scene. If the intent parameter has the same name as the scene slot, the Run Time automatically fills the scene slot with the value from the intent parameter [4-5].

When a user's response doesn't match one of intents, voice assistant attempts to handle the input and asks user to repeat his input. This behavior facilitates users changing actions in the middle of a conversation. In combination with intents, scenes are the other major building block of conversation model. Scenes represent individual states of conversation and their main purpose is to organize conversation into logical chunks, execute tasks, and return prompts to users.

After building the voice assistant application there is important part to understand how to build web-application that can process JSON request and interact with it. After the user authorizes action to access their Google profile, web-application will receive a Google ID token that contains the user's Google profile information in every subsequent request to any action. To access the user's profile information, the next steps should be followed:

- decoding and verification the token using JWT-decoding library and Google's public keys;
- verification that the token's issuer (is field in the decoded token) is https://accounts.google.com and that the audience (aud field in the decoded token) is the value of Client ID issued by Google to voice assistant application;

The following is an example of a decoded token.

```
{
 "sub": 1234567890, // The unique ID of the user's Google Account
 "iss": "https://accounts.google.com,
 "name": "Mykola Slisarenko",
 "given_name": "Mykola",
 "family_name": "Slisarenko",
 "email": "mykola.slisarenko@nure.ua",
 "locale": "en_US"
}
```

After validation the JSON it's possible to get user's information and store it. When the creation is completed, there should be exact response to make voice assistant application send access token in each request to let web application identify exact user.

```
{
 "token_type": "Bearer",
 "access_token": "ACCESS_TOKEN",
 "expires_in": SECONDS_TO_EXPIRATION
}
```

In this response "access_token" is the key which will be sent in the following requests and by which user can be identified [6].

III. CONCLUSIONS

During the work, the areas of voice assistants and tasks that are best suited for their use were analyzed. The model of work of the voice assistant and agglomeration of construction of the application for it was described. Conversational intentions and conversational scenes, key elements for integration, were analyzed.

Also, tools have been proposed to achieve the goal of identifying a user who uses a web application using a voice assistant. JSON user data models and ways to decode this information have been described.

As a result of the analysis, a model of voice assistant integration when working with web applications was proposed.

REFERENCES

[1] G. Blodick, “Google Assistant: A complete guide to self-assessment”, 2018, 138 p.

[2] C. Adams, “A powerful assistant called Google: Learn to master Google and make full use of it for your business”, 2019, 79 p.

[3] M. Alexander, “Amazon Alexa: 55 benefits of your Amazon Alexa. Amazon Alexa vs. Google Assistant”, 2019, 48 p.

[4] M. Adams, “Google Home: Google Home Guide and Google Home Guide with settings, features”, 2017, 117 p.

[5] T. Schillerhof, “Google Home: Guides, Settings, Features”, 2016, 56 p.

[6] H. Lee, “Voice UI Projects: Create voice-enabled applications using Dialogflow for Google Home and Alexa Skills Kit for Amazon Echo”, 2018 , 404 p.

[7] Anbazhagan K., “IOT project using Google Assistant and other web technologies: the best approach using web technologies with a simple explanation”, 2019, 326 p.