Nuevos Paradigmas de Interacción
grado en ingeniería informática

MEMORIA PRÁCTICA 2

Autores
Alba Casillas Rodríguez
Andrea Navarro Jiménez
Miguel Molino Moreno
Jose Manuel Osuna Luque

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2020-2021

BETWEEN STUDENTS

Between Students es una aplicación enfocada a la Universidad de Granada, con el objetivo de hacer llegar este producto a uno de los grupos más importantes de la ciudad: los estudiantes. La idea principal de esta aplicación nace de la necesidad de facilitar la información sobre cualquier facultad de la UGR de una forma rápida y accesible para todos. Teniendo esta premisa clara, se decidió enfocar la aplicación a información general como dónde se encuentra cada facultad (campus y localización) o el horario de apertura.

Con el fin de aumentar el número de servicios proporcionados, se ha implementado un asistente conversacional en DialogFlow Essential: BEST, de manera que ahora el usuario podrá acceder a información o servicios dentro de una facultad de una forma fluida y atractiva para el usuario.

Los usuarios podrán acceder a BEST de manera cómoda y sencilla realizando un "swipe down" en la pantalla principal de la aplicación y que invita a los usuarios a interactuar con el bot. BEST permite encontrar respuestas sobre localizaciones, horarios y disponibilidades de cualquier facultad perteneciente a la Universidad de Granada, estando integradas también las facultades pertenecientes a Ceuta y Melilla. Toda esta información estará al alcance del usuario con solo indicar la ciudad y facultad donde se encuentra.

Además, se proporciona servicios extra como la oportunidad de interactuar con las citas de secretaria. El usuario podrá realizar las siguientes acciones:

- Pedir una cita: se le proporcionará al usuario una lista de los horarios disponibles y una vez elegida la hora deseada se realizará la reserva proporcionando un correo de contacto.
- Cancelar una cita: el asistente solicitará al usuario el correo electrónico con el cual realizó la reserva y la cancelará.
- Consultar la agenda: el usuario podrá mediante su correo saber si tiene una cita reservada.

Para obtener la mejor experiencia posible para los usuarios, nuestro agente de DialogFlow se apoya en la plataforma adicional Firebase y funciones avanzadas implementadas en código JavaScript que conectan las funcionalidades permitidas con una base de datos en la nube, mediante la que se controlará cualquier información disponible y las modificaciones relacionadas con reservas y cancelaciones de citas.

Con ello se ha construido un sistema con funcionalidades claras y bien definidas, sencillo de usar y que establece una conversación lo más natural posible con el usuario, mediante respuestas personalizadas que tienen en cuenta el contexto de la interacción; siendo así lo suficientemente robusto como para ser capaz de responder de forma predecible a condiciones de error o situaciones inesperadas.

A parte de la plataforma DialogFlow, nuestro agente estará disponible en Kommunicate.io y, como se ha dicho anteriormente, integrado en la propia aplicación de nuestro proyecto: Between Students. De esta manera, se encontrará al alcance de todos los estudiantes y, para aumentar la accesibilidad de los mismos, la comunicación podrá realizarse tanto de manera textual como oral.

Este acceso rápido desde el propio dispositivo del usuario proporcionará comodidad y rapidez en la comunicación con el agente hasta en momentos en los que no se pueda disponer de un total manejo del dispositivo y, frente a la situación actual de COVID-19, evitará el uso de dispositivos comunes en las facultades como por ejemplo la reserva de citas.

IMPLEMENTACIÓN DIALOGFLOW

A continuación se va a comentar los detalles más importantes a la hora de añadir funcionalidades avanzadas mediante el Inline Editor:

Para poder usar los datos almacenados en la base de datos de FireBase, se debe conectar DialogFlow con esta a través del enlace que ofrece la plataforma.

```
// Se inicializa la base de datos. Siempre que se quiera llamar a ella habra que usar la variable admin
admin.initializeApp({
  credential: admin.credential.applicationDefault(),
  databaseURL:'ws://npi-bestbot-rtkv-default-rtdb.firebaseio.com/'
  // In the databaseURL give your own firebase url
});
```

Es posible añadir y resetear contextos mediante código con la siguiente función:

```
// Elimina un contexto
agent.setContext({'name': 'informacion', 'lifespan': '0'});
// Aniade un contexto
agent.setContext({'name': 'informacion', 'lifespan': '5'});
```

Para obtener la funcionalidad de que el agente pueda proporcionar diferentes respuestas cuando el Intent se gestiona desde una función se consigue creando una lista con los diferentes mensajes y seleccionando uno de ellos de manera aleatoria.

```
var posiblesRespuestas = [
    ciudad + ' es muy bonita. Dime en que Facultad te encuentras.',
    'Que suerte vivir en ' + ciudad + '. Dime tu facultad.',
    'Oh. A mi tambien me gustaria vivir en ' + ciudad + ' Dime donde estudias.'
];

var pick = Math.floor( Math.random() * posiblesRespuestas.length );

var respuesta = posiblesRespuestas[pick];
agent.add( respuesta );
```

Los parámetros que recibe del usuario se pueden recoger del agente definido en la variable 'agent'.

```
const ciudad = agent.parameters.Ciudades;
```

Esto sería un ejemplo de una función avanzada en la que se conecta con la base de datos y hace una búsqueda de las horas disponibles que tiene el usuario para reservar una cita en secretaría. De haberlas, añade un contexto nuevo para que el usuario pueda avanzar a dar una hora y así registrar la cita.

```
return admin.database().ref(ciudad).child(facultad).child(elemento).
    orderByValue().equalTo("LIBRE").once("value").then((hijos) => {
        var numHijos = hijos.numChildren();
        if(numHijos === 0) {
          agent.add("No hay citas disponibles");
        }else{
          respuesta = "HORAS DISPONIBLES";
          hijos.forEach(hora => {
            respuesta = respuesta + "  \n" + hora.key;

          });
          agent.setContext({'name': 'citas', 'lifespan': '5'});
          agent.add(respuesta);
        }
      });
```

Por último, es obligatorio vincular la función implementada al nombre del Intent correspondiente.

```
let intentMap = new Map();
intentMap.set('PedirCiudad', pedirCiudad);
intentMap.set('PedirFacultad', pedirFacultad);
```

IMPLEMENTACIÓN ANDROID

De la misma manera, se explicará los métodos más importantes para implementar el chatbot en Android.

Para que la aplicación Android se pueda conectar al agente conversacional es necesario que el usuario conceda el permiso de conectarse a Internt. Si además quisiera interactuar de manera oral, deberá conceder permisos de grabación de audio.

```
<!-- Para poder usar un navegador interno, es necesario que existan permisos de internet -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- Para poder usar el reconocimiento de voz -->
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

Este método comprueba que los permisos hayan sido concedidos y, de no ser así, se los pide al usuario.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if(requestCode == PERMISSION_REQUEST_AUDIO && grantResults.length > 0){
        if(grantResults[0] == PackageManager.PERMISSION_GRANTED){
            Toast.makeText(this, getString(R.string.audio_permission_granted), Toast.LENGTH_SHORT).show();
            onRestart();
        }
    }
}
```

En este fragmento de código, se instancia la conexión con DialogFlow mediante una credencial obtenida de la consola de Google guardada en un archivo JSON. Si la conexión es correcta, devolverá toda la información necesaria para realizar la conexión y además dejará abierta una sesión para que cuando el usuario realice una consulta, esta sea enviada a DialogFlow y a su vez este devuelva la respuesta correspondiente al agente.

```
private void initV2Chatbot() {
    try {
        InputStream stream = getResources().openRawResource(R.raw.
    npi_bestbot_rtkv_709d8a193b0e);
        GoogleCredentials credentials = GoogleCredentials.fromStream(
    stream);
        String projectId = ((ServiceAccountCredentials)credentials).
    getProjectId();

        SessionsSettings.Builder settingsBuilder = SessionsSettings.
    newBuilder();
        SessionsSettings sessionsSettings = settingsBuilder.
    setCredentialsProvider(FixedCredentialsProvider.create(credentials
    )).build();
        sessionsClient = SessionsClient.create(sessionsSettings);
        session = SessionName.of(projectId, uuid);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Este método mantiene abierta una conexión en segundo plano y hace de intermediario entre el agente y el usuario.

```
@Override
protected DetectIntentResponse doInBackground(Void... voids) {
    try{
        DetectIntentRequest detectIntentRequest =
                DetectIntentRequest.newBuilder()
                        .setSession(session.toString())
                        .setQueryInput(queryInput)
                        .build();
        return sessionsClient.detectIntent(detectIntentRequest);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

Para la implementación oral se crean los objetos speechRecognizer y speechRecognizerIntent y se controla mediante el método de setOnTouchListener asociados al botón de audio haciendo que al pulsarlo se active la escucha, y al soltarlo se recoja lo que el usuario ha dicho.

```
speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
final Intent speechRecognizerIntent = new Intent(RecognizerIntent.
    ACTION_RECOGNIZE_SPEECH);
```

```
fabListening.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_UP){
        speechRecognizer.stopListening();
    }

    if (event.getAction() == MotionEvent.ACTION_DOWN){

        speechRecognizer.startListening(speechRecognizerIntent);
    }

    return false;
        }
    })
```

Referencias

[1] Cloud Console. *Google Cloud Console*
`https://cloud.google.com/dialogflow/docs`

[2] FireBase. *DataBase in FireBase*
`https://firebase.google.com/docs/database/web/lists-of-data`

[3] Emojis. *Emoji's Library*
`https://emojipedia.org`

[4] Doubts. *Page for the doubts during the development of the project*
`https://stackoverflow.com`

[5] ChatBotV2. *Chatbot integration on Android with Version 2*
`https://github.com/abhi007tyagi/DialogflowChat`