Nuevos Paradigmas de Interacción

UNIVERSIDAD DE GRANADA

Memoria Interacción Oral

Sergio Jesús Jerez Vázquez
Carlos Lao Vizcaíno
Eduardo Muñoz del Pino
Pablo Rodríguez Fernández

Índice

1.- ETSIIT Utilities

ETSIIT Utilities es una aplicación diseñada para facilitar información que puede ser muy útil a los estudiantes y personal de la ETSIIT, pero que en este momento no es posible obtener o, de serlo, está escondida tras innumerables menús y pantallas en un sitio web.

Las herramientas que proporciona la aplicación en su primera versión son:

- **Mapa**: guía al usuario hasta la ubicación elegida dentro de la escuela (aula, despacho,etc).

- **Horario:** Permite saber qué asignatura se está impartiendo en un aula concreta, junto con información del profesor, grupo de prácticas,etc.

- **Parking:** Permite saber, en tiempo real, las plazas de aparcamiento libres en los diferentes aparcamientos con los que cuenta la escuela (coches, motos, bicicletas y patinetes).

- **Comedor**: Permite consultar, de una manera sencilla, el menú semanal.

En esta nueva versión, se busca aumentar las funcionalidades de la herramienta, con el fin de que sea más útil y por tanto más aceptada por la comunidad universitaria. Además, se añaden pequeñas mejoras para hacerla más accesible para personas con algún tipo de discapacidad física, a fin de que pueda ser usada por la mayor cantidad de personas de una manera cómoda.

De cara a mejorar la experiencia de usuario, se ha implementado un asistente conversacional creado en DialogFlow que permite, por un lado, obtener información de una manera más natural (manteniendo una conversación). Además, añade diversas funcionalidades hasta ahora no presentes en la aplicación, como es la **reserva de plazas de parking** y la **consulta de horarios** de diferentes servicios de la escuela, como la **secretaría** o el **comedor**.

Además, buscando hacer la aplicación más accesible, se añadió la posibilidad de navegar entre las diferentes pantallas utilizando la voz. Por ejemplo, decir “quiero ir al aula 3.3” y que se muestre el itinerario asociado, en lugar de tener que seleccionarlo en el menú.

Finalmente, con el mismo fin, se incluyó la lectura en voz alta (síntesis de voz) de las informaciones asociadas a las rutas del mapa (“Sigue recto”, “gira a la derecha”) y las plazas de aparcamiento libres. Esto permite, por ejemplo, seguir la ruta hacia

un aula sin tener que estar mirando la pantalla del dispositivo continuamente, o permitir que personas ciegas puedan utilizar también dicha funcionalidad.

2.- Navegación por voz

A continuación se van a explicar los fragmentos de código más relevantes relacionados con la implementación de la navegación por voz.

Toda la funcionalidad se implementa en la actividad principal de la app, MainActivity.java, ya que es desde la pantalla principal desde donde se puede acceder a dicha funcionalidad. Para ello, se debe pulsar el botón con un micrófono:

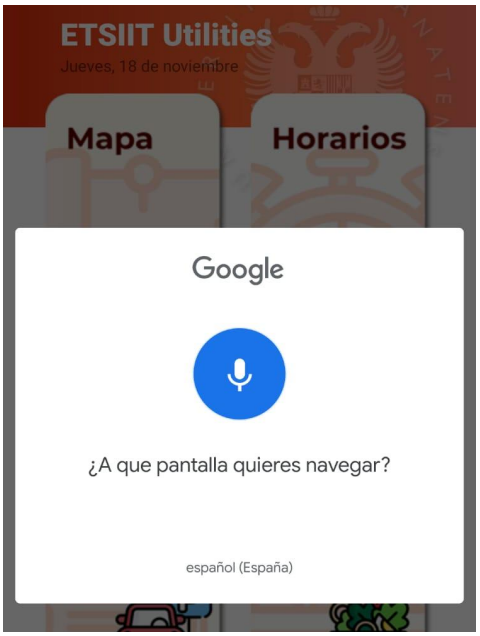
Para poder realizar lo anterior, en primer lugar se debe asociar el evento que detecta la voz con el botón:

```
// Elementos de speech-to-text
-----------------------------------------------------------------
speakButton = (ImageButton) findViewById(R.id.speakButton);
// Activamos el servicio de reconocimiento en la pulsación del botón.
speakButton.setOnClickListener(new OnClickListener() {
   @Override
   public void onClick(View v) {
       startVoiceRecognitionActivity();
   }
});
```

Cuando el usuario hace click, se llama al siguiente método, que es el encargado de realizar el reconocimiento de voz:

```
private void startVoiceRecognitionActivity() {
   Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
   intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
           RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
   intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
           "¿A qué pantalla quieres navegar?");
   startActivityForResult(intent, REQUEST_CODE);
}
```

Cabe decir que la aplicación proporciona feedback al usuario para que sea que se está realizando el reconocimiento, y qué se está entendiendo exactamente de sus palabras:



Cuando se termina de hablar, se invoca inmediatamente el método siguiente, el cual es el encargado de, en función de la respuesta del usuario (comprobando la existencia de ciertas palabras “clave”), moverse a la pantalla apropiada. Debido a la extensión del método, se muestra un pequeño fragmento de este, permitiendo ver la lógica básica tras su funcionamiento:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   if (requestCode == REQUEST_CODE && resultCode == RESULT_OK) {

       ArrayList<String> matches =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);


       if ( matches.get(0).contains("mapa") ) {
           startActivity(new Intent(this, ActivitySelectRoute.class));
       } else if ( matches.get(0).contains("aula") ){
           Intent intent = new Intent(this, ActivityMap.class);
           intent.putExtra(RUTA, "Banho_Clase");
           startActivity(intent);
       }
```

3.- Síntesis de voz

Como se comentó anteriormente, para mejorar la accesibilidad se incluye la lectura de las instrucciones del mapa y de las plazas de aparcamiento disponibles.

Si bien en ambas funcionalidades se implementan de manera similar, existe una gran diferencia entre ellas: en el parking solamente se lee una frase, mientras que el mapa debe leer frases continuamente. Además, como el mapa muestra indicaciones cada cierto número de pasos, puede darse el caso de que, si un usuario se mueve rápido, se intente leer una indicación cuando la anterior aún se esté leyendo. Por esta razón, se comentará más en detalle la parte asociada al mapa, siendo la del parking similar (pero más simple).

Si bien la funcionalidad se implementa en la clase “ActivityMap.java”, que es la asociada al guiado a través de un ruta por la escuela, la creación del objeto TextToSpeech se realiza en la MainActivity. De esta manera, el objeto puede ser utilizado en cualquier parte de la app, y en particular, utilizar el mismo para el mapa y para el parking.

```
//Inicializacion del textSpeech
textToSpeechEngine = new TextToSpeech(this, new TextToSpeech.OnInitListener() {
   @Override
   public void onInit(int status) {
       if (status != TextToSpeech.SUCCESS) {
           Log.e("TTS", "Inicio de la síntesis fallido");
       }
   }
});
```

Cuando se realiza un cambio en una indicación se modifica la imagen que se muestra por pantalla, el texto asociado y ahora, además, se lee en voz alta dicho texto. El código asociado a esta lectura es el siguiente:

```
//Leemos la indicación si el texto es diferente o si no se está leyendo nada
if(!textoAnterior.equals(texto) || textToSpeechEngine.isSpeaking()==false) {
   textToSpeechEngine.speak(texto, TextToSpeech.QUEUE_FLUSH, null, "tts1");
}
//Almacenamos el texto que se acaba de leer para poder compararlo en la próxima
indicación
this.textoAnterior = texto;
```

Como se comentó, puede darse el caso de que se intente leer la nueva indicación mientras aún se lee la indicación. En dicho caso, por defecto, TextToSpeech comienza a leer el nuevo texto, terminando bruscamente el anterior.

Tras diversas pruebas, se decidió actuar de la siguiente manera:

- Si no se está leyendo la indicación anterior, se lee la indicación nueva.
- Si se está leyendo una indicación y la nueva tiene el mismo texto (situación muy habitual debido a la longitud de los pasillos de la escuela. Las rutas suelen tener muchas indicaciones cuyo texto es varias veces seguidas “sigue recto), se ignora y se deja que termine de leerse el texto actual.
- Si se está leyendo una indicación y la nueva tiene distinto texto, se interrumpe la lectura y se lee la nueva. La razón de esto es similar a la decisión anterior. Como muchas indicaciones se repiten varias veces seguidas, aunque se interrumpa, el usuario ya sabrá cual era, por eso se le da prioridad a la nueva.

4.- Asistente

La última funcionalidad añadida en esta versión es la más novedosa: un asistente tipo chatbot que permite realizar consultas sobre diversos servicios (horarios de la secretaría, consultar el menú del miércoles en el comedor) y reservar plazas de aparcamiento en el parking de la escuela (diferenciando el tipo de vehículo: coche, moto, patinete o bicicleta).

El asistente está basado en DialogFlow, herramienta de Google que permite la creación de bots conversacionales. Además, se utiliza como herramienta de apoyo PubNub, una API que permite ejecutar funciones complejas para mejorar la experiencia con el asistente.

En primer lugar se presentará cómo se integra el asistente dentro de la aplicación, para pasar a comentar algunos detalles de la implementación del asistente.

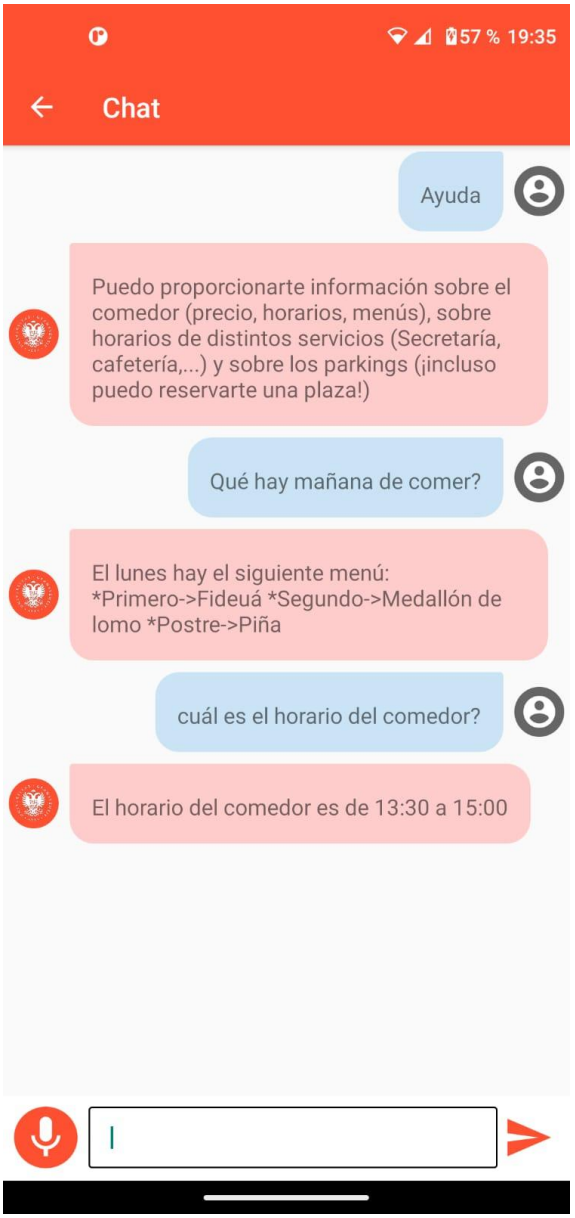
4.1 - Integración en la aplicación

Para iniciar el asistente se debe pulsar en la pantalla principal el botón que dice “Chatbot” (se puede ver en la página 3 de este documento), ejecutándose el siguiente código:

```
public void openChatbot(View view) {
   // provide your Dialogflow's Google Credential JSON saved under RAW folder in
resources

DialogflowCredentials.getInstance().setInputStream(getResources().openRawResourc
e(R.raw.etsiit_utilities_chatbot));

   ChatbotSettings.getInstance().setChatbot( new Chatbot.ChatbotBuilder()
           .setDoAutoWelcome(false)
           .setChatBotAvatar(getDrawable(R.drawable.pajaro))
           .setShowMic(true).build());
   Intent intent = new Intent(this, ChatbotActivity.class);
   Bundle bundle = new Bundle();

   // provide a UUID for your session with the Dialogflow agent
   bundle.putString(ChatbotActivity.SESSION_ID, UUID.randomUUID().toString());
   intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_NO_HISTORY);
   intent.putExtras(bundle);
   startActivity(intent);
}
```

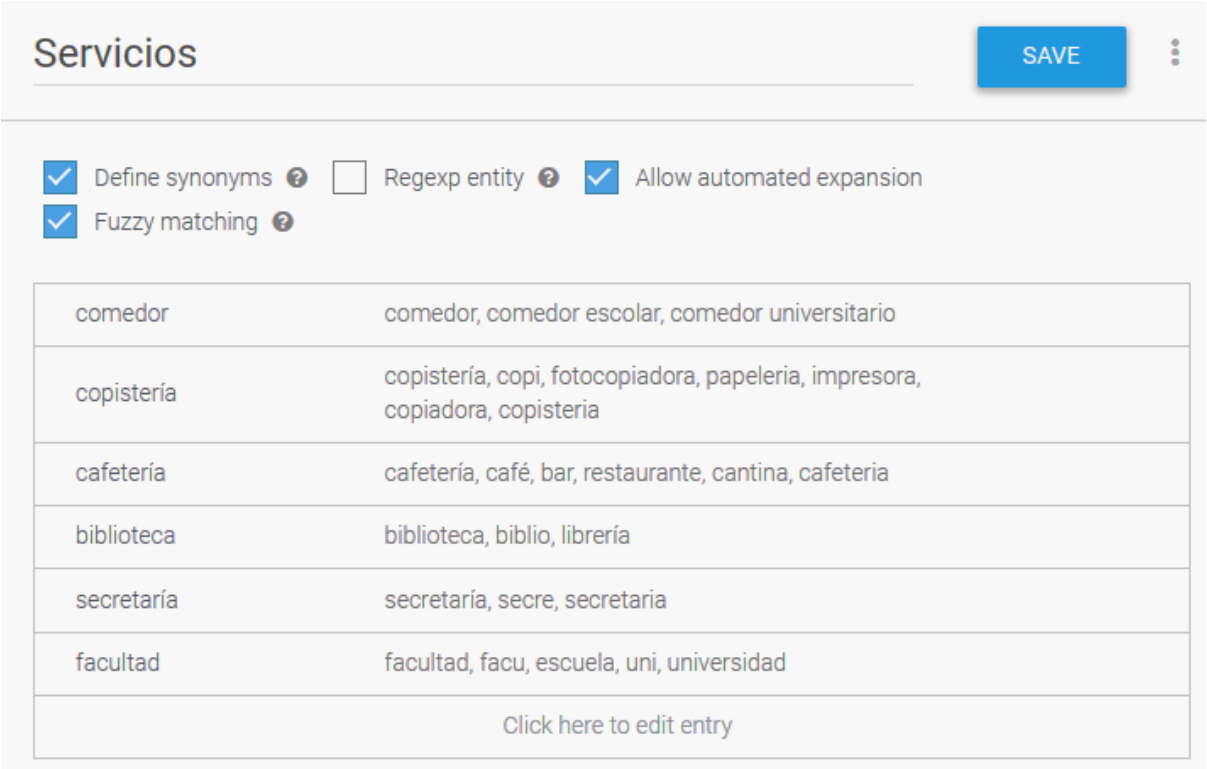
En el código anterior se le indica el fichero donde se encuentran los datos para poder acceder al chatbot (etsiit_utilities_chatbot.json), así como se fijan parámetros del asistente, como la imagen que aparecerá como icono, la posibilidad de dictar el texto por voz en lugar de escribirlo,etc…

Una vez se ejecuta la función anterior, aparece una nueva pantalla con el asistente, la cual es generada automáticamente por la herramienta. A continuación se muestra un ejemplo de uso real:



4.2 - Detalles de implementación

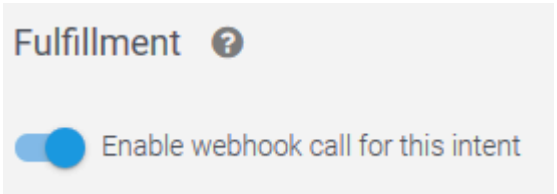
Como ya se comentó anteriormente, el asistente está realizado empleando la herramienta de Google DialogFlow. Ésta está basada en el empleo de “Intents”, que serían las acciones concretas que se pueden realizar con el asistente (comprobar un horario) , “Entities”, que son los elementos necesarios para completar las acciones (“de qué” se comprueba un horario) y “Fullfilments”, que es la integración con otros servicios. En este caso, ese servicio sería PubNub, donde se ejecutan funciones complejas (obtención del horario para el servicio elegido).



Ejemplo de una entity del asistente

Para la ejecución de funcionalidades complejas, en lugar de escribir todas las posibles respuestas en el propio Intent, se puede vincular una función de tal manera que, cuando el usuario pregunte la información, se llame a esa función y utilizando los datos obtenidos en la conversación, se decida qué responder.

Para ello, se debe indicar que se utilice una “webhook call” en el Intent:



A continuación se muestra un fragmento de la función (escrita en javascript) alojada en PubNub que se ejecuta cuando un usuario pregunta por el horario de un determinado servicio de la escuela:

```
export default (request, response) => {
    let bodyString = request.body;
    var entradaDialogFlow = JSON.parse(bodyString);
    var servicio=entradaDialogFlow['queryResult']['parameters'].servicio;

    // Horarios de los servicios
    var horario="No se conoce el servicio";

    if (servicio ==="comedor"){
        horario="de 13:30 a 15:00";
    }else if (servicio ==="copistería"){
        horario="de 08:30 a 19:30";
    }else if (servicio ==="cafetería"){
        horario="de 08:30 a 17:00";
    }else if (servicio ==="biblioteca"){
        horario="de 08:30 a 20:30";
    }else if (servicio ==="secretaría"){
        horario="de 09:00 a 13:00";
    }else if (servicio ==="facultad"){
        horario="de 07:30 a 22:30";
    }else{
        servicio = null;
    }

    let respuestaHorario ="El horario del "+servicio+" es "+horario;
```