Memoria Descriptiva

1. Agente conversacional

La selección de las herramientas informáticas se ha realizado intentando tocar el mayor número de tecnologías existentes.
Se planteado un esquema de agente conversacional del siguiente modo:

1.1 Mobile APP

Hemos implementado la conexión con DialogFlow siguiendo el esquema aportado en el github de Zoraida Callejas:
https://github.com/zoraidacallejas/talk-to-your-assistant

Para disponer de múltiples idiomas en nuestra aplicación, hemos duplicado el archivo "string" dentro de la carpeta "res/values-es", esta vez con los textos en español. Por defecto, la aplicación usará el archivo correspondiente al idioma en el que se encuentre el teléfono.

Del mismo modo, la variable global "languages" obtendrá el valor del sistema "Locale. *getDefault*().toString()", y se usará para indicar al bot en qué idioma debe hablar.

1.2 DialogFlow

Existen varios bloques de *intents*:

- Datos relativos a los dos conjuntos de películas, tanto actores, descripción o año de estreno de forma explícita como por contexto. En el caso de los actores dando la posibilidad de seguir una conversación árbol para centrarse en un actor individual.
- Datos relativos a Jacinto Benavente como su biografía, fecha de nacimiento u obra.
- Datos relativos al contenido de la salas del museo.
- Datos relativos a los cinematógrafos. ¿Qué son y quién los inventó?

Las entities definidas incluyen solo el nombre de los elementos.
Nomenclatura usada: cuando en el nombre de un *intent* se usan paréntesis indica que existen datos proporcionados por el contexto de una pregunta anterior.

1.3 Firebase Functions

Usando la licencia gratuita planteada por Google para el desarrollo en firebase hemos usado el módulo de Firebase Functions. Para realizar el proyecto hacemos uso de Firebase CLI un gestor que permite trabajar de forma local y desplegar las funciones en los servidores de Google.
Referencia de instalación de Firebase CLI:
https://firebase.google.com/docs/cli/?hl=es-4
Referencia al curso realizado para Firebase:
https://www.youtube.com/user/Firebase
Dicho módulo implementa un disparador por medio de una función en javascript nodejs 8 que realiza una conexión con una base de datos NOSQL.

El esquema de fichero *json* que envía DialogFlow por medio de un *fulfillment webhook* a Firebase lo podemos encontrar detallado:
https://dialogflow.com/docs/fulfillment/how-it-works
El fichero *.eslintrc.json* que genera el proyecto reune las caracteristicas que usa nodejs para su propia configuración.
El fichero *package.json* reúne las las dependencias usadas por la función implementada.
El fichero *index.js* contiene el código de la función que ejecuta *webhook.* Dicho fichero contiene la configuración para conectar con F. Cloud, una estructura *map* formada por el par nombre del *intents* y la función que se debe ejecutar en el caso de sea llamado por dicho *intent*. También contiene la implementación de las funciones para cada *intent* que realizan una consulta en la base de dato en forma de promesa javascript.
Desde la plataforma de Google se permite añadir variables de entorno que permiten traspasar información entre ejecuciones. Google no recomienda su uso y es por ello sumado a la memoria limitada la decisión obligada de usar una base de datos.
Nota: el módulo de funciones de Firebase así como las funciones implementada en nodejs 8 se encuentran en versión beta y ocasionan fallos y reinicios inesperados.

1.4 Firebase Cloud

Usa una estructura semejante a la conocida base de datos MongoDB utilizando colecciones y documentos con estructura *json*.
Tenemos 4 colecciones de prueba para películas y actores que solo contienen datos de tipo array y string debido a que se trata de un proyecto didáctico. Siguiendo el curso mencionado anteriormente F. Cloud incluye referencias a documentos que optimizan las consultas como otro tipo de datos que mejoran la compresión.
La estructura de una consulta en F. Cloud es la siguiente:

- Conexión con base de datos haciendo uso de la biblioteca *firebase-admin*.
- Se crea una referencia a un conjunto de documentos especificando la colección y la condición lógica que deben cumplir.
  Ejemplo: db.collection(“peliculas”).where(“year”,”>=”,”1990”)
  Dame las películas que se hayan estrenado posteriormente al año 1990.
- Se ejecuta la consulta mediante el método *get()* sobre la referencia que devuelve una promesa.

Hubiéramos querido probar el módulo de RealDatabase pero la licencia gratuita impone demasiadas limitaciones.
Nota: el módulo de Firebase Cloud se encuentra en versión beta sumado a las limitaciones de la licencia gratuita dispara los tiempos de espera de la consultas.

1.5 Conclusión al agente conversacional

Concluimos que en este ámbito de la inteligencia artificial enfocada a la conversación:

- DialogFlow y SpeechRecognizer conforman la capacidad de captación de información de nuestro agente. A nuestro entender muy dirigida todavía un resultado preestablecido.
- Firebase Functions establece un núcleo de razonamiento del agente. Aunque no hemos explotado la funcionalidad recursiva que nos dan las funciones mientras se

transforma el conocimiento por varios caminos haciendo un símil con los puntos de vista, parece ser firebase un motor con muchas posibilidades.

- Firebase Cloud produce la red de información del agente. Nos ha parecido todo un acierto debido la versatilidad de poder agregar distintos campos independientemente de la colección, pues no existe conjunto de entidades dentro del mismo contexto que el humano procese exactamente de la misma forma debido a la complejidad de sus sentidos.

2. Botones para salas

Tres botones con su método OnTouch definidos, que se encargarán de modificar el estado del botón pulsado a “seleccionado” y poner los otros dos al estado “default”. Además enviarán un mensaje a DialogueFlow indicando en qué sala se encuentran para que éste informe sobre qué temas preguntar.

3. Multitouch

Para gestionar la interacción multitouch, la función “SetOnTouchListener” define un OnTouchListener con un método onTouch que, en función de dedos en pantalla, controlará diferentes gestos (función handleTouch).

3.1. Deslizar 3 dedos para moverse a la habitación contigua

Si se detectan 3 dedos en pantalla, guardará la posición en la que se colocó el primer dedo inicialmente. Después, conformes deslices los dedos, irá calculando la diferencia entre ambas posiciones. Si esta diferencia es mayor o menor que el valor de la variable MIN_DESPLACEMENT se registrará el cambio de habitación.

Una vez calculada la habitación resultante en función de cuál estuviera seleccionada, se llamará al botón de dicha función.

3.2. Doble tap con 2 dedos para iniciar conversación

Si por lo contrario, el número de dedos detectado es igual a 2 pasará a detectar si ocurre un doble tap. Para llevar a cabo este proceso, se lanzará una nueva hebra después de el primer tap que durante 300ms estará a la escucha de un posible segundo tap. Si esto ocurre, se lanzará la acción del botón “speech_btn”.

https://www.youtube.com/watch?v=PEpQT4x-kKc

3.3. Slide con 2 dedos para iniciar la actividad principal

Si detectamos un deslizamiento en la pantalla realizado con 2 dedos, se llamará a la función que lanzará el intent de la actividad principal de nuestra aplicación, tras realizar una animación en la actividad inicial de modo que se simula el efecto de cerrar una claqueta cuando se comienza a rodar una escena de película.

3.3.1. Animación de claqueta

Una vez es detectado el slide con 2 dedos, iniciamos la animación definida en la carpeta anim, que aplicaremos a la parte superior de la plaqueta, y tras ésta, se ejecutará en una hebra aparte el lanzamiento de la actividad principal.
La animación consistirá en una rotación de modo que simularemos como si la claqueta se estuviera cerrando.

http://www.androidcurso.com/index.php/recursos/35-unidad-4-graficos-en-android/269-animaciones-tween

4. Acelerómetro para sacudir teléfono

Creamos los objetos de la clase SensorManager y de la clase Sensor, definidos como Sensor. TYPE_LINEAR_ACCELERATION si está disponible en el dispositivo. Si no está disponible, se intentará con TYPE_ACCELEROMETER.

El funcionamiento en ambos casos será el mismo (función onSensorChanged adaptada del la adquirida en el enlace https://stackoverflow.com/questions/5271448/how-to-detect-shake-event-with-android).
Cuando la velocidad a la que se agite el móvil sea mayor que SHAKE_THRESHOLD se llamarán a los métodos stop y stopListening del ChatBot.

El Listener para el SensorManager se inicializará en el onResume y se pausará en el onPause, de este modo la actividad no quedará en segundo plano esperando a detectar un cambio.

5. Lectura de luminosidad para activar cámara

Si detectamos que baja la luminosidad de la sala B cuando indicamos que nos encontramos en ésta, siendo el valor leído igual a 0, se lanzará la actividad que nos mostrará una proyección en el dispositivo.

https://developer.android.com/guide/topics/sensors/sensors_environment

6. Activación de cámara para lectura de códigos QR

Si accionamos el botón asociado a QR, se lanzará la llamada a una nueva actividad que será la encargada de gestionar la apertura de la cámara, así como de comprobar los permisos disponibles para ello y solicitarlos en caso de ser necesario, y tras ello, gestionar la captura del código QR y lanzar una nueva actividad para reproducir un vídeo relacionado con el museo, como podría ser un trailer de alguna película, o un vídeo informativo sobre algún tema del cine.

6.1. QR Activity

Como hemos mencionado, esta es la actividad donde gestionaremos la lectura del código QR ayudándonos de la librería `ZXingScannerView`, con la que crearemos un manejador, lanzaremos la cámara, tras comprobar o solicitar los permisos oportunos, y gestionaremos la respuesta del código QR lanzando la actividad creada para reproducir los id de los vídeos obtenidos de la lectura del cÓdigo QR.

https://www.numetriclabz.com/android-qr-code-scanner-using-zxingscanner-library-tutorial/

6.2 Player Activity

Este activity es el encargado de reproducir los vídeos asociados a los id obtenidos en las urls de los códigos QR leídos.
Para ello parseamos la dirección y obtenemos el id para poner la dirección en el formato que nos interesa.
Una vez disponemos de la dirección, instanciamos un webView y lanzamos una búsqueda de la url del vídeo obtenida.

En este activity hemos incluido además, en la función de retroceso, una variable que mandamos a la actividad principal para evitar que se muestre de nuevo la ayuda, y otra para indicar que se pueda cargar la animación de nuevo en caso de volver a la actividad inicial de la aplicación.

https://developer.android.com/guide/webapps/webview

7. Error Enum

Hemos definido un enumerado para discriminar los diversos errores del cliente que podamos obtener por el manejo de las distintas funcionalidades implementadas para nuestra aplicación.