UNIVERSIDAD DE GRANADA

NUEVOS PARADIGMAS DE INTERACCIÓN

Biodomo Museum APP - Técnica

Prácticas Grupo 1, Lunes 11:30-13:30

Autores:

- Juan Emilio García Martínez
- Adrián Jesús Peña Martínez
- Julio Fresneda García

Granada, 11 de Noviembre de 2018

1.- Introducción

Explicaremos el proceso, a grandes rasgos, de cómo hemos realizado dicho software y de cómo se usa el software.
Además, explicaremos qué partes hemos realizado explícitamente nosotros y cuáles hemos introducido, aplicando algunas modificaciones, de software ya creado y testado, para una realización más rápida de nuestro software utilizando una de las mejores herramientas de las que disponemos los desarrolladores, la reutilización de código.

2.- Uso del software

Aunque en la memoria descriptiva explicamos las acciones que nuestra APP es capaz de realizar, comentaremos la distribución de nuestra aplicación (pantallas o vistas) y la forma de uso de cada una de ellas:

- Pantalla principal, en la que quedan bien definidos los elementos para cumplir el cometido de nuestra APP.
  Dicha pantalla dispone del título de la APP, dispone de los botones con las secciones más importantes de nuestra aplicación, es decir, Explora, lector QR, lector NFC, mapa del Biodomo y precios del Biodomo, en orden de importancia.
  Dispone de los botones de contacto como redes sociales y página web.
  Dispone del botón de la acción principal de nuestra APP, es decir, la interacción vía voz con el asistente.
- Explora: pantalla en la que queda bien definido su cometido, el de explorar via interacción gráfica y tacto, las distintas zonas y elementos de cada una de ellas (animales y vegetales).
  Dicha pantalla dispone a su vez de tres botones, los cuales se refieren a cada una de las tres zonas del Biodomo y por cada una de estas, otra pantalla en la cual quedan definidos tanto el título, imagen representativa y descripción de cada uno de los elementos de cada zona, con botones para un fácil desplazamiento entre cada uno de ellos.
  Se ha de comentar, que en cada una de las zonas, también se dispone de un botón accionador del lector QR, por si en algún momento de la visita, si se está en esta pantalla, se quiere pasar el lector QR sobre alguno de los distintos códigos QR repartidos.
- Lector QR: automáticamente al pulsar el botón “Lector QR” nos aparecerá el lector (cámara) esperando algún código QR. Según el QR leído, nuestra aplicación será capaz de auto redirigirnos hacia el elemento referenciado por el código o en su caso, hacia alguna página web con más información, son los dos posibles usos controlados por nuestra APP.
- Lector NFC: al pulsar dicho botón, se nos abrirá la pantalla correspondiente, la cual, tras comprobar que tenemos el NFC activo (en caso de incluirlo nuestro dispositivo), se queda esperando a que pasemos el móvil por algún tag NFC distribuido por el museo. Al leerlo, nos mostrará la información correspondiente. Aunque la funcionalidad ahora mismo no nos sea de gran utilidad, el principal motivo de su

incorporación fué la de tener varias alternativas para poder obtener información durante el propio recorrido sobre el museo, es decir, según nuestro dispositivo, puede que queramos a veces usar el QR u otras el NFC, pudiendo resolver problemas del tipo “mi móvil no tiene NFC” o “a mi móvil no le funciona la cámara correctamente”, etc..

Esta parte es una de las partes que más desarrollo ha requerido junto con el micrófono.

- Mapa: en dicha sección, se nos aparecerá el mapa del biodomo, el cual dispone de la funcionalidad multitouch “trap-to-...” la cual nos permite navegar a través de él (funcionalidad a incorporar en siguientes versiones: mediante sensor GPS, indicarnos dónde estamos en tiempo real).
- Precios: primera pantalla realizada durante el desarrollo del software, la cual es meramente informativa sobre las tarifas y precios del museo.
- Otros botones: para mayor facilidad a la hora de conocer el museo Biodomo, nuestra APP dispone de botones representativos referentes a las redes sociales y página web del Biodomo, totalmente funcionales.
- **Botón del micrófono:** como hemos comentado antes, es la parte de nuestra APP que más desarrollo ha requerido. Dicho botón contiene la funcionalidad principal de pulsarlo para que el asistente se ponga en funcionamiento y comience a interactuar con nosotros vía voz. El botón estará de color verde, cuando el asistente esté en modo “espera” a que queramos interactuar con el.

  Cuando el asistente comienza a interactuar y estar esperando a que le digamos algo, es decir, está en modo “escucha”, el botón se pone de color rojo. Si pulsamos de nuevo el botón, el asistente parará de escuchar.

  Cuando le requerimos al asistente alguna información, es decir, está en modo “habla”, podemos usar la funcionalidad “tag-and…” implementada por nosotros, para que el asistente deje de hablarnos, por si en ese momento alguien nos habla o nos ocurre cualquier imprevisto y el asistente debe de parar de hablar.

  Cuando decimos que es la parte que más tiempo de desarrollo nos ha ocupado, es porque también incluye la funcionalidad de “hablarnos de cualquier cosa” utilizando la API de la Wikipedia, la cual espera una petición POST y esta devuelve una respuesta con la introducción de la información buscada en ella, sintetizándola por voz.

3.- Realización del software

Una vez explicada la distribución y de lo que es capaz cada una de las partes de nuestra aplicación, explicaremos la parte del “cómo” se ha realizado e implementado cada una de dichas partes.

Solo comentaremos las partes más relevantes, las partes en las que hemos usado recursos externos o las partes en las que hemos usado técnicas no comunes. Nos referimos a técnicas comunes a, por ejemplo, especificar un escuchador de eventos sobre un botón, las cuales las comentaremos de pasada.

3.1. Permisos, sdk y dependencias.

Nuestra APP requiere de los permisos de internet, grabador de audio, permitir comprobar el estado de la red del dispositivo, cámara y NFC.

Nuestra aplicación está desarrollando herramientas y módulos que requieren, como mínimo, del sdk 25, aunque como sdk deseado, el sdk 28, ya que es el sdk con el que se ha desarrollado, testado y lanzado.

Como dependencias necesarias para la compilación de nuestro proyecto, a parte de las básicas de android para el diseño usado, hemos utilizado las siguientes:

- zxing-android-embedded, versión 3.4.0.
- guava versión 26.0
- zoomage, versión 1.2.0.

3.2. Activities

A continuación vamos a proceder a explicar detalladamente cada una de las actividades de nuestra APP interactiva. Detallaremos la clase java y el layout de cada actividad.

3.2.1 MainActivity

En cuanto a cómo silenciamos el asistente mediante el gesto de flip down (poner el dispositivo boca abajo), lo hacemos mediante el uso de el sensor de acelerómetro primero tuvimos que importar la librería de android.hardware.SensorManager, una vez hicimos esto lo siguiente fue crear un objeto SensorManager al que llamamos sensorManager y un booleano para saber en qué momento debemos de escuchar el movimiento del dispositivo y que momentos no, solo queremos que escuche cuando el asistente está dando una respuesta.
Básicamente lo que hacemos para saber si el dispositivo es ver el valor del eje Z de el acelerómetro, entonces de este modo y mediante la clase que calcula la orientación del dispositivo getAccelerometer sabemos la orientación y por tanto silenciamos el asistente. Solo se hará uso de estas funciones cuando el asistente este en medio de una respuesta de este modo evitamos un desgaste de recursos innecesario y por supuesto una vez silenciamos el asistente cambiamos el valor del booleano para así evitar que mientras el dispositivo este cara abajo se llame múltiples veces al método stop().
Además de estas funciones también tenemos las funciones necesarias para poder hacer uso del acelerómetro y por supuesto la clase MainActivity implementa un SensorEventListener.

Botón QR

Para leer códigos QR, hemos implementado un escáner el cual se activa presionando el botón de QR. Para implementar este escáner, hemos usado la biblioteca “zxing”.

Para implementar este botón se han usado dos métodos: *setQrButton()* y *onActivityResult()*.

El primer método implementa el listener del botón, y lanza el escáner cuando lo pulsamos.
El segundo método sirve para obtener el resultado del qr leído. Si el resultado no es nulo (no hemos cancelado el escaneo), el QR nos podrá llevar tanto a una página web como a una especie.

Los QR los hemos generado en qr-code-generator.com. Para las páginas web, los códigos qr simplemente tienen la web. Para las especies, los códigos qr tienen un texto con la siguiente forma: “BiodomoInteractivo:<zona>_<a|v>_<numero_de_animal>”.

Por ejemplo, para el vegetal número 8 del amazonas, el código será “BiodomoInteractivo:ama_v_8”.

Para descifrar este mensaje, entra en juego el segundo método, el cual a partir del texto obtiene el nombre de la especie (nombre, imagen y descripción).

Una vez leído el código, abre un nuevo ExploraViewActivity para mostrar la especie.

3.2.2 ExploraActivity y ExploraViewActivity

3.2.2.1 ExploraActivity

El ExploraActivity.java es una activity a la cual se accede desde MainActivity, y cuya función es mostrar en pantalla tres botones, uno por cada zona del biodomo. Los tres botones llevan a la misma actividad, ExploraViewActivity, y en los tres, al pulsar, se guarda un String con la zona que queremos explorar para que ExploraViewActivity sepa qué zona mostrar. La única diferencia entre los tres botones es que para cada botón se guarda un valor distinto en el String antes mencionado.

La estructura del layout consiste en un LinearLayout con tres ConstraintLayout dentro, uno por cada botón.

Se ha usado un LinearLayout con orientación vertical, para que los tres botones se re-escalen con igual tamaño, dando igual en qué resolución estemos. El problema de los LinearLayout es que si queremos superponer objetos (uno sobre otro) a priori no podemos. Por eso, dentro del LinearLayout he introducido tres ConstraintLayout, uno por cada botón. Dentro de cada uno de esos ConstraintLayout tenemos el propio botón, una ImageView con una imagen decorativa (vemos que en la esquina superior izquierda del botón hay un animal, es esto), y dos textos: Uno para el nombre de la zona y otro para una pequeña descripción.

3.2.2.2 ExploraViewActivity

El ExploraViewActivity es la activity donde se mostrarán los animales de la zona que hemos elegido en ExploraActivity.

Su estructura consiste en un título arriba, una imagen del animal en el centro, una descripción del animal abajo y un botón flotante de QR. Si esta actividad se ha cargado desde ExploraActivity, además tendremos un botón a la derecha y otro a la izquierda de la imagen: Estos botones sirven para pasar a la especie siguiente o a la anterior.

Al ExploraViewActivity se puede acceder a través del ExploraActivity, a través del botón de QR o a través del NFC. Por eso, debemos diferenciar para qué caso lanzamos el ExploraViewActivity. Esto lo hacemos con las siguientes comprobaciones:

A través de ExploraActivity

Si el String de zona que se guarda en ExploraActivity no es nulo, quiere decir que el ExploraActivity ha guardado algún valor, por lo que sabemos que venimos de ahí. Por lo tanto, se procede a cargar las especies de la zona elegida.

Para cargar las especies vamos a necesitar tres arrays: Una para los nombres, otro para las imágenes y otro para los textos. Según el valor de nuestro String de zona recuperado, a estos arrays les asignaremos unos valores u otros. Esto se hace en el método *insertSpecies(String zone).*

Una vez cargadas las especies, se modifican los colores de la interfaz, según la zona donde estemos.

Para controlar la posición en la que estamos, vamos a usar un valor de posición, que empieza en 0. Si pulsamos el botón de la derecha (siguiente especie), el valor de la posición aumenta en 1, y si pulsamos el botón de la izquierda (especie anterior), el valor de la posición disminuye en 1. De esta forma podemos recorrer los arrays de nombre, imagen y descripción de forma coherente.

Para que nuestras especies estén distribuidas de forma aleatoria, se usa un array auxiliar de enteros con los índices, llamado shuffleIndices. Este array contiene todos los números desde el 0 hasta el número de especies menos 1. Por ejemplo, si tenemos 20 especies, nuestro array de índices sería {0,1,2,3,...,19}.

Una vez hecho esto, le hacemos shuffle al array de índices.

Cada vez que queramos acceder a una posición, obtendremos la posición a través de este array. Por ejemplo, para acceder a imagen de la tercera especie, usamos *imagenes.get(shuffleIndices.get(3))*.

De esta forma cada vez que entremos a explorar veremos una distribución distinta de las especies.

A través del lector QR

Si el String de zona que usamos antes es nulo, pero recuperamos un String “id” y no es nulo, hemos accedido al ExploraViewActivity a través del lector QR.
Los QR contienen un animal en concreto, por lo que no nos harán falta los botones de derecha e izquierda, ya que sólo mostraremos ese animal.
Para hacer esto, obtenemos a partir de la información del QR el nombre del título, imagen y descripción y los mostramos en pantalla.

A través del lector NFC

El funcionamiento es exactamente igual que el del QR, solo que en este caso obtenemos el nombre de la zona, y mostramos una imagen decorativa para una zona en concreto, y una descripción de la zona abajo.

Botón flotante de QR

ExploraViewActivity contiene un botón flotante que lanza el escáner de QR, para no tener que volver al menú si queremos leer algún animal. Los métodos usados son exactamente iguales a los explicados en el MainActivity, con la diferencia de que aquí, si leemos más de un animal seguidos con QR, finalizamos la actividad anterior. Esto es para que, por ejemplo, podamos leer desde el ExploraView el QR de diez animales seguidos, y al volver hacia atrás volvamos al ExploraView con todos los animales de la zona, y no al animal leído antes por QR.

En cuanto al Layout, está activity usa ConstraintLayout. Tiene una ImageView con un texto arriba de la pantalla, donde se muestra el nombre de la especie. En el centro de la pantalla, hay una ImageView que contiene la imagen de la especie, y un botón a cada lado para pasar a la especie siguiente o anterior. Abajo hay un ImageView con una imagen de un marco, y dentro un texto con la descripción de la especie. De fondo tenemos el mismo background que en el resto de actividades.

3.2.3 NFCActivity

En la ejecución de esta actividad simplemente se mostramos una imagen (imageView) y un texto indicando que pasemos el dispositivo con NFC por un tag NFC (viewText).

Al pasar el dispositivo por un tag NFC, comparamos dicho valor obtenido con algunos de nuestros tags específicos del Biodomo para mostrar la zona correspondiente (en un futuro utilizaremos esta tecnología para más funcionalidades).

El extraer el valor de dicho TAG nfc, lo realizamos modificando el código utilizado en un anterior proyecto realizado por Google y compartido por este en el “Android Open Source Project”: el funcionamiento básico es el siguiente: tras crear las clases correspondientes con la lectura y procesamiento de los tags NFC (y más funcionalidades) en los paquetes “parser” y “record”, los usamos de la siguiente manera: la interfaz ParsedNdefRecord contiene el método str() , el cual nos devuelve al final un String con los datos leídos de la tarjeta NFC. Dicha interfaz la usaremos para la creación de las clases UriRecord, TextRecord y SmartPoster (URI+Text).

Los datos intercambiados usando NFC usan el formato “NDEF”. Para “parsear” dicho formato utilizamos la clase NdefMessageParser, devolviendo el contenido de los mensajes obtenidos a través del NFC en una lista de ParsedNdefRecords.

Una vez tenemos las clases y métodos que implementan la funcionalidad necesaria para poder utilizar el NFC de nuestro dispositivo, lo usamos en nuestra actividad NFC Activity.
En dicha actividad definimos algunas propiedades como NfcAdapter del SDK de Android estándar, un PendinIntent el cual utilizamos para lanzar la actividad cuando un nuevo TAG es escaneado.

Controlamos en todo momento que el NFC esté activo o disponible en nuestro teléfono, parando dicha actividad si esto ocurriera. Si el NFC estuviera disponible, pero no estuviera activo, mandamos al usuario a los ajustes de Wireless para que este lo active.

Lo que controlamos nosotros para conocer lo que nos llega de la lectura, es la acción recibida mediante el Intent creado, la cual puede ser un ACTION_TAG, ACTION_TECH o ACTION_NDEF. Posteriormente, analizamos dicha acción y mandamos ejecutar el Intent correspondiente con las actividades de ExploraView para mostrar la zona correspondiente.

Implementamos otros métodos para, en un futuro, implementar otras funcionalidades utilizando NFC, como pueden ser toDec, toHex, toReverseHex, etc..

3.2.4 MapaActivity

En esta activity simplemente hemos añadido un imageView con el atributo zoomable gracias a la librería zoomage que permite que se le pueda hacer zoom mediante el gesto pinch-to-zoom. El archivo Java solo tiene el onCreate, el Layout sí que usa un objeto propio de la librería, el cual contiene la imagen.

3.2.5 TarifasActivity

Para esta actividad simplemente dejamos la clase como viene por defecto, es decir, solo con el método onCreate pues solo queremos que nos muestre los precios como hemos establecido en el layout.
El layout de esta actividad es simplemente varios textView unidos unos con otros dentro de un constraint layout de tal manera que en todas las pantallas se guarde una relación de aspecto entre ellos.

3.2.6 VoiceActivity

Esta actividad es la que venimos usando desde el principio de la asignatura, una vez vimos la funcionalidad tanto del sintetizador de voz (TextToSpeech) como el reconocedor de voz (Automatic Speech Recognizer).

Básicamente la utilizamos en MainActivity, implementando los métodos abstractos que hay en esta y utilizando sus métodos, correctamente comentados en ella y sin necesidad de explicar aquí toda su funcionalidad.

4.- Bibliografía y referencias

Para la realización del gesto de silenciar al asistente (dispositivo cara abajo) nos hemos basado en el código de estás dos webs pero lo hemos modificado bastante:

- https://stackoverflow.com/questions/17774070/android-detect-when-the-phone-flips-around
- http://www.vogella.com/tutorials/AndroidSensor/article.html

Para la realización del escáner QR, hemos usado la biblioteca “zxing” (“Zebra Crossing”):

- https://opensource.google.com/projects/zxing
- https://github.com/zxing/zxing

Para la realización del zoom en el mapa, hemos usado la librería “zoomage”:

- https://github.com/jsibbold/zoomage

La implementación del Layout con zoomable la hemos hecho a partir del ejemplo de este github.

Para la realización de la interfaz (Imágenes, iconos, etc) se ha usado Photoshop CC, apoyándonos en la librería de iconos de Google:

- design.google
- https://material.io/tools/icons/?style=baseline

Por supuesto, hemos usado la web del biodomo: biodomogranada.com