Héctor Álvarez Ibáñez 53541631T

**Entregable: curso de Android**

**Avanzado**

**Índice**

# Práctica 2

## Explicación de la implementación

## Dificultades encontradas y horas trabajadas

# Práctica 3

## Explicación de la implementación

## Dificultades encontradas y horas trabajadas

# Práctica 4

## Explicación de la implementación

## Dificultades encontradas y horas trabajadas

# Práctica 6

## Explicación de la implementación

## Dificultades encontradas y horas trabajadas

# Práctica 7

## Explicación de la implementación

## Dificultades encontradas y horas trabajadas

# Práctica 8

## Explicación de la implementación

## Dificultades encontradas y horas trabajadas

# Conclusión del trabajo

**1.Práctica 2**

En esta práctica se pedía implementar usando fragments la aplicación realizada en la práctica 1. A continuación describiré el proceso seguido para esta práctica.

* 1. **Explicación de la implementación:**

Como la lógica de la práctica 1 estaba ya hecha, solo tuve que hacer el diseño de la interfaz. Para ello declaré un xml por cada fragment que se iba a implementar. En concreto, se implementaron los fragments Main, Contacta, Empresa, Noticias , Catálogo y Detalles.

En el fragment Main, que es el que creó la aplicación por defecto, tras diseñar el xml con la interfaz y de relacionarlo con el fragment por medio del inflater para obtener el View, identifiqué los botones correspondientes mediante la orden findViewById y les puse un onClickListener para que, al pulsar sobre ellos, abrieran el fragment correspondiente en vez de lanzar la actividad, como ocurría en la práctica 1. En las transacciones a los fragments uso el método addToBackStack para que, al pulsar el botón atrás del dispositivo, se vuelva al fragmento que se le indique, en este caso el fragment\_main.

En el fragment empresa, que es el más simple de todos, solo diseñé la interfaz mediante el editor gráfico del xml y lo relacioné con la clase EmpresaFragment (de ahora en adelante cuando se relacione se supondrá que se relaciona con clases ContactaFragment, etc…)

En el fragment de Contacta implementé con el editor gráfico los elementos correspondientes, relacioné el xml con la clase correspondiente y , en la clase, identifiqué con findViewById los elementos e introduje ciertos atributos de forma programática, como el tamaño del texto, el ámbito del spinner y el método que envía los datos del botón envía.

En el fragment de Noticias implementé un scrollView con un layout y luego, en la clase correspondiente, relacioné el xml, identifiqué el layout y, usando el método de la práctica 1, introduje el ejemplo de varias noticias para ver el aspecto final.

En el fragment de Catálogo introduje un scrollView e hice básicamente lo mismo que en Noticias: usé el método implementado en la práctica 1 para cargar los layouts con la información de los pisos. Se sobreentiende también que incluí la base de datos y la clase Piso de la práctica 1. Aparte, en los botones de los pisos que deben abrir los detalles, les añadí un setOnClickListener para que, al ser pulsados, reemplazaran el fragment actual por el fragment de Detalles. Añadí ,al igual que en el main, el método addToBackStab para que vuelva, en este caso, a fragment\_caalogo al pulsar la tecla de atrás del dispositivo.

El fragment de Detalles al igual que en el resto ,diseñé la interfaz con el editor gráfico del xml y lo relacioné con su clase correspondiente. En este caso, El ViewPager puesto sin más me daba problemas, así que puse un LinearLayout vertical en su lugar para luego añadir el viewPager en función de si había imágenes o no. Usé el método usado en la práctica 1 para cargar los campos con los valores correspondientes. He de remarcar aquí que, por motivos estéticos, tuve que definir al ViewPager que su altura fuera la altura de la imagen, pues si no ocupaba el resto de la pantalla apartando los demás campos de texto.

* 1. **Dificultades encontradas y horas trabajadas:**

En cuanto a las dificultades me costó encontrar la manera de volver atrás hasta que descubrí el addToBackStab. La clase Contacta, por motivos que desconozco, no logra mostrar el mensaje al pulsar el botón enviar. Tras debuggear, encontré que algo falla en HttpResponse, pero no logré averiguar qué.

El tiempo total empleado en esta parte ha sido de 2 horas y media.

**2.Práctica 3**

En esta práctica se pide modificar la anterior (práctica 2) para incluir un mensaje de notificación al entrar en el catálogo con dos botones que te lleven a los detalles del piso con id = 1 y al piso con id = 2 respectivamente junto con un sistema de aviso aparte, ya sea vibración o leds.

**2.1 Explicación de la implementación:**

Tras ver los vídeos decidí usar mejor una notificación pequeña, pues el móvil que poseo no admite notificaciones grandes. Dicho esto, para implementar el lanzamiento de la notificación, implementé en la clase CatalogoFragment el método lanzarNotificación a la cual le paso como parámetros la List de pisos sacada de la base de datos. Hice esto porque después lo necesitaría para poder abrir los detalles del piso. Configuré la notificación y creé el pendingIntent en dos botones. Usé el btn\_mas y el btn\_menos de R.android.drawable . Luego, en la clase MainActivity modifiqué el onCreate para que identificara actions del intent y, en función de eso, abriera los detalles del piso correspondiente o no. En los intents de la notificación que pasé mediante los PendingIntents introduje en ellos con el putExtra el piso correspondiente al id el cual debía abrir. Luego, en el MainActivity detecto la action que lleva la intent para diferenciar si es el inicio del programa o es el intent de la notificación y, en función del action, abre DetallesFragment con el xml fragment\_detalles con el piso correspondiente, el cual se lo paso mediante un Bundle al fragment.

**2.1 Dificultades encontradas y horas trabajadas:**

La mayor dificultad que me supuso fue el hecho de pasar el piso para la intent de la notificación. Intenté hacer que en vez de intents que me abren otra activity intercambiara los fragments, pero no hallé la manera. En total el tiempo que me llevó programar esta práctica sin el tiempo invertido en investigación por mi cuenta es de 1 hora y media aproximadamente. Con investigación serían alrededor de 3 horas y media, porque aparte en cierto punto se me borró el método lanzarNotificación entero y tuve que rehacerlo.

**3. Práctica 4**

En esta práctica se pide modificar la práctica 2 para que el programa, al detectar una conexión de cascos en mi caso, abriera la vista principal, un servicio que incrementara en 1000 el precio de la base de datos al pulsar un botón de la vista principal y, por último, que dicho servicio se lance automáticamente cada 60 segundos.

**3.1 Explicación de la implementación:**

Debido a que el hecho de identificar la conexión de cascos me pareció más simple, empecé por ahí. Implementé en el ActivityMain un BroadcastReceiver que lo que hace es sustituir el container con un nuevo PlaceholderFragment para que, esté donde esté la aplicación, lo sustituya. Luego, en el onCreate puse un registerReceiver con el broadcast para que la aplicación lo implementara, si no como si nada. Le puse el intent correspondiente a la acción de conectar los cascos. Y con esto terminé lo que sería el primer punto.

Para la siguiente fase creé una clase SubirPrecioService y la incluí en el AndroidManifest como servicio. Luego, incluí un Button en la vista principal que, al ser pulsado, inicia la actividad.

Para lograr que la base de datos se actualizara, modifiqué la clase PisosDB y creé un método subirPrecio que, tomando una SQLDatabase, ejecuta un código SQL para que incremente el precio en 1000 cada vez que se llame.

En el servicio, en el onStart hice que se abriera la base de datos y se modificara.

Para que se modifique automáticamente cada 60 segundos, en el servicio implementé en su método onCreate un Timer que, cada 60 segundos, llama al método subirPrecios del servicio, el cual modifica la base de datos, al igual que el botón.

**3.2 Dificultades encontradas y horas trabajadas:**

Los principales problemas que se me presentaron fue el hecho de modificar la base de datos. También el hecho de cómo hacer que cada 60 segundos se ejecutara, problema que solucioné con el Timer. Aparte, cuando conectas los cascos te intercambia el Fragment, pero luego hace cosas raras la aplicación. No supe arreglarlo, así que me temo que no esta 100 por 100 funcional. El tiempo empleado para programar esta práctica fue más o menos 1 hora y media, aparte de media hora de investigación para encontrar el Timer.

**4 Práctica 6**

En esta práctica se pide crear un estilo que modifique, al menos, 4 propiedades distintas de cualquiera de los controles utilizados.

**4.1 Explicación de la implementación:**

Para hacer esta práctica, simplemente seguí los pasos de los vídeos para crear mi estilo llamado MiEstiloPractica6. En él, modifico 4 atributos, gravity, textColor, textSize y textColorLink y aplico el estilo a la aplicación entera usándolo en el AndroidManifest.

**4.2 Dificultades encontradas y horas trabajadas:**

En esta práctica no encontré dificultad alguna y me llevó un total de 30 minutos el realizarla.

**5 Práctica 7**

En esta práctica se nos pide implementar un sistema de reconocimiento de voz y un sistema de dictado de texto para la aplicación hecha en la práctica 2.

**5.1 Explicación de la implementación:**

Primero implementé el sistema de dictado de texto y, para ello, puse un botón en la interfaz de los detalles de los pisos. Luego, creé un objeto TextToSpeech e hice que la clase implementara un OnInitListener. Sobreescribí el método onInit con la configuración pertinente del TextToSpeech y , en el botón, cree un onClickListener para que, al pulsarlo, lea la descripción del piso correspondiente.

Para la segunda parte, en el MainActivity, añadí un botón que lanza un Intent que llama al sistema de reconocimiento del móvil. Sobreescribí el método onActivityResult para que, al detectar el sistema de voz ciertas palabras, cambie el fragmento con el correspondiente a la palabra dicha.

**5.2 Dificultades encontradas y horas trabajadas:**

En cuanto a dificultad la palabra contacto no la detecta como tal, sino como contactar, así que tuve que poner un toast para ver qué entendía el teléfono y así poder crear los casos. Aparte de eso, el resto no llevó mayor problema. El tiempo usado para esta práctica ha sido de un total de 1 hora.

**6 Práctica 8**

En esta práctica se pide crear una aplicación que se comunique con una calculadora web mediante un servicio SOAP.

He de decir que la práctica está casi hecha, solo que me da un error al mandar los parámetros x e y como Floats. Si los mando como Strings me devuelve un String vacío, así que la aplicación esta “casi hecha”, solo falta arreglar ese problema para el cual no he encontrado solución. El tiempo que me llevó hasta rendirme fue de 1 hora.

**7 Conclusión del trabajo**

Ante todo, quiero agradecerle por esta segunda oportunidad. El anterior entregable fue un total desastre y lo cierto es que me supo mal entregarle en su momento aquello. Espero que este le haga ver que no es que no lo entienda o me lo tome a la ligera. Muchas gracias por todo y, ante cualquier tipo de duda, no dude en contactarme. En general, el curso es muy completo y he aprendido mucho. Seguiré entrenando con el material que me ha dado para, en el futuro, poder aprovechar todo el contenido de este curso, no solo la parte referente a los entregables.