PRACTICA 2

Jonathan García Ruiz 48732127-A

Contenido

[ENTORNO 2](#_Toc94275845)

[ACTIVITIES 3](#_Toc94275846)

[TICKETS\_MAINACTIVITY 3](#_Toc94275847)

[CATEGORIA\_ACTIVITY 3](#_Toc94275848)

[ADDTICKET\_ACTIVITY 4](#_Toc94275849)

[ADDCATEGORIA\_ACTIVITY 4](#_Toc94275850)

[DTO 5](#_Toc94275851)

[DTOTICKET 5](#_Toc94275852)

[DTOCATEGORIA 5](#_Toc94275853)

[HELPERS 5](#_Toc94275854)

[CategoriaHelper 5](#_Toc94275855)

[ImageHelper 5](#_Toc94275856)

[OCRHelper 5](#_Toc94275857)

[TicketsHelper 6](#_Toc94275858)

[CONSTRUCTOR(Context,CursorFactory,version) 6](#_Toc94275859)

[onCreate(SQLiteDatabase) 6](#_Toc94275860)

[onUpgrade(SQLiteDatabase, versionAnterior,versionNueva) 6](#_Toc94275861)

[obtenerTickets() 6](#_Toc94275862)

[obtenerValoresTicket() 6](#_Toc94275863)

[asignarValoresTicket 6](#_Toc94275864)

[agregarTicket(…) 6](#_Toc94275865)

[borrarTicket (…) 6](#_Toc94275866)

[ADAPTERS 6](#_Toc94275867)

[INTERFACES 6](#_Toc94275868)

# ENTORNO

Para poder acceder a la cámara y almacenar los archivos en memoria externa he necesitado modificar el Android Manifest y el build.gradle

Para los permisos de lectura y escritura, de cámara y de actualización de la galería:

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.CAMERA" />  
<protected-broadcast android:name="android.intent.action.MEDIA\_MOUNTED"/>  
<uses-feature android:name="android.hardware.camera" android:required="true" />

Por otra parte, para las versiones sdk 24 en adelante, las rutas con file:// ya no funcionaban, esto lo hizo Android para que el fichero lo gestione la clase que haga el intent y no la propia cámara.

En conclusión, para resolver este problema tuve que implementar la clase FileProvider, añadirla en el AndroidManifest, y crear un xml con el path.

<provider  
 android:name="android.support.v4.content.FileProvider"  
 android:authorities="${applicationId}.provider"  
 android:exported="false"  
 android:grantUriPermissions="true">  
 <meta-data  
 android:name="android.support.FILE\_PROVIDER\_PATHS"  
 android:resource="@xml/file\_paths"/>  
</provider>

Una vez hecho esto, las rutas serán content:// y no file://

Por otra parte, primero hice la aplicación desde una blank activity y cuando la tuve casi acabada creé un proyecto Navigation Drawer e importé todo el proyecto. Puntos destacables, como no tengo fragments tuve que crear más layouts y definir en el AndroidManifest que no tenían Action Bar (ya que usa toolbar):

<activity  
 android:name=".Activities.Categoria\_Activity"  
 android:theme="@style/AppTheme.NoActionBar"/>

En el gradle, en pos de usar el gson y el ocr, implementé estas 2 APIs

implementation 'com.google.code.gson:gson:2.8.5'  
implementation 'com.google.android.gms:play-services-vision:17.0.2'

# ACTIVITIES

## TICKETS\_MAINACTIVITY

Actividad principal, se encarga de la interfaz de los tickets.  
Instancia las vistas.  
Pone en marcha el toolbar y el drawer.  
Abre una conexión con la BD y carga los tickets en un arraylist.  
Carga los tickets del arraylist a una vista ListView.  
Para mostrar los tickets usa un adaptador que se nutre de una plantilla(tipo list por defecto).  
Añade un menú contextual al ListView.  
Da funcionalidad al fab que contiene un intent para crear/modificar un ticket.

Mediante el Toolbar podemos elegir que adaptador mostrará nuestro ListView.  
Mediante el Drawer incorporamos un menú lateral para navegar por la apk.  
Podemos mantenernos en tickets, ir a categorías, abrir Google y mostrar la ficha del profesor, abrir los contactos, un query a Google, etc.

Para el menú contextual tenemos 2 comportamientos, o ver ticket (desde aquí podemos editarlo) o borrar un ticket.

La parte del borrado y editado ha sido lo más complicado de esta actividad, en parte por cómo plantear cómo editarlo (plantear el intent, qué parámetros pasarle y cómo gestionarlos) y que lo editara de verdad y por otro lado eliminarlo. El problema era que como lo tenía planteado el AUTOINCREMENT del id en la BD sobraba, ya que provocaba que los ids no casaran.

Por último, implementa 2 interfaces, una para que funcione el Drawer y otra con las constantes que se pasan por los intents.

## CATEGORIA\_ACTIVITY

Actividad, se encarga de la interfaz de las categorías.  
Instancia las vistas.  
Pone en marcha el toolbar y el drawer.  
Carga las categorías en un arraylist.  
Carga las categorías del arraylist a una vista ListView.  
Para mostrar las categorías usa un adaptador quesenutre de una plantilla(tipo tile por defecto)  
Añade un menú contextual al ListView.  
Da funcionalidad al fab que contiene un intent para crear/modificar una categoría.

Mediante el Drawer incorporamos un menú lateral para navegar por la apk.  
Podemos mantenernos en tickets, ir a categorías, abrir Google y mostrar la ficha del profesor, abrir los contactos, un query a Google, etc.

Para el menú contextual tenemos 2 comportamientos, o ver categoria (desde aquí podemos editarlo) o borrar una categoría.

También ha sido la parte más complicada, el de editar no porque hice primero el del ticket y ya sabía cómo hacerlo. Para el borrado, al principio pensaba que borraba bien, luego no, luego si, en resumen, no conseguía gestionar adecuadamente qué categoría borrar porque en vez de pasarle el índice con info.position usaba arrayCategorias.get(info.position).getId() y otras varias combinaciones más que tampoco funcionarion, pero la actual sí funciona.

## ADDTICKET\_ACTIVITY

Actividad, se encarga del formulario para crear o modificar un ticket.  
Instancia las vistas.  
Instanciamos un ImageHelper para ayudarnos con el almacenamiento de la foto.  
Instanciamos el OCRHelper para ayudarnos con el OCR.  
Abre una conexión con la BD y carga los tickets en un arraylist.  
Reserva memoria para un objeto de tipo Ticket.  
Comprueba si le han llegado parámetros de Tickets\_Mainactivity (para saber si es un ticket a editar o a crear). Si es para editar, carga las vistas con el ticket.  
Comprueba si tenemos permisos de escritura y de acceso a la cámara.  
Da funcionalidad al fab que puede o tomar una foto, o cogerla de la galería o poner OCR. Para el OCR, primero debemos haber cargado la foto con tomar una foto o desde la galería.

Da funcionalidad al botón de aceptar, aquí comprobamos si es un ticket a editar o a crear y rellena con datos la memoria reservada para el objeto de tipo Ticket.

Aquí tuve un problema, y es que cuando era un ticket a editar no se le asignaba la foto (ya que lo gestionaba desde el callback pero no tomé en cuenta que si lo editaba no llamaba al callback, por tanto, intentaba insertar en la BD un campo nulo (la foto) y en la BD era NOT NULL.

Con los métodos obtenerThumbnail y obtenerImagenCompleta accedo a la cámara para tomar una foto, se diferencian en que el primero, al devolver un thumbnail no genera uri y el segundo sí. Igualmente obtenerThumbnail no se usa aunque esté. Con el segundo, tuve el problema ya mencionado del File Provider.

Con el método obtenerGaleria hago un intent a la galería, pudiendo elegir qué galería.

Para guardar el estado del path de la foto cuando vuelvo del intent tengo los métodos onSaveInstanceState y onRestoreInstanceState.

Luego hay métodos que verifican que se tengan los permisos.

Por último tenemos el callback (onActivityResult).  
Si el intent fue de la cámara asignamos la imagen a la vista mediante el path.  
Si el intent fue de la galería asignamos la imagen a la vista mediante el uri.  
En ambos casos rellenamos en la memoria del objeto ticket la ruta de la foto.

Por último, implementa 2 interfaces con constantes para trabajar con los intents y las fotos.

## ADDCATEGORIA\_ACTIVITY

Mismo Código, cambiando tickets por categorías.  
Reservamos memoria para un Arraylist de categorías.  
Cargamos las categorías mediante CategoriaHelper.  
Inicializamos el arraylist con las categorías cargadas.  
Reservamos memoria para un objeto de tipo categoría.

# DTO

## DTOTICKET

Clase con únicamente las propiedades del objeto, en este caso un ticket, con sus pertinentes getters, setters y constructor para garantizar la encapsulación y modularización.

## DTOCATEGORIA

Clase con únicamente las propiedades del objeto, en este caso una categoría, con sus pertinentes getters, setters y constructor para garantizar la encapsulación y modularización.

# HELPERS

## CategoriaHelper

Se encarga de crear el fichero categorías.json en memoria interna.  
Para añadir, borrar y editar una categoría, las carga en un arraylist y de ahí se encarga de gestionarlo (.add,.remove,.set), después de hacer los cambios guarda los cambios en el fichero categorías.json.

Para obtener las categorías una vez cargadas se usa el metofo getCategorias, que devuelve un arraylist con todas las categorías.

Para guardar las categorías busca si existe el fichero y lo crea si no existe. Si existe escribe sobre el fichero en formato json con los objetos categoría.

Para cargar las categorías busca el fichero en memoria interna, lo recorre y transforma los datos de formato json a formato objeto y los carga en el arraylist de categorías.

## ImageHelper

Se encarga de gestionar la ruta y el almacenamiento en memoria externa de las fotos.  
Cuando se instancia la clase, se crea si no existe, la ruta:   
/storage/emulated/0/Android/data/dadm.practica2/files/Pictures

El método getPhotoPath se encarga de crear un fichero con el nombre en base a la hora en que se cree y devuelve la ruta de la misma.

El método setImageFromPathString se encarga de crear el bitmap a partir del path del fichero, una vez se tiene el bitmap se le inicializa al imageView que se le pase por parámetros.

El método setImageFromPathUri se encarga de crear el bitmpat y asignarlo al imageView que se le pase por parámetros a partir de la uri.

El método saveBitmapOnStorage, solo es llamado cuando se trabaja con la galería, ya que si trabajamos con la cámara, desde el propio intent se nos crea el fichero. Cuando es llamado por la galería tiene 3 funciones, crear el bitmap a partir de una uri, crear el fichero y devolver su path.

## OCRHelper

Se encarga de devolver el texto que encuentre en un bitmap.

## TicketsHelper

### CONSTRUCTOR(Context,CursorFactory,version)

Metodo para instanciar la BD

### onCreate(SQLiteDatabase)

Metodo llamado por Android cuando se instancie la BD

### onUpgrade(SQLiteDatabase, versionAnterior,versionNueva)

Metodo llamado por Android cuando se cambie la version de la BD.  
Hace falta seguir una estrategia para salvar los datos, si deja hacer un ALTER TABLE no hay problema, si no deja hay que salvar los datos y Volver a crear u otra estrategia

### obtenerTickets()

Abrimos la BD en modo escritura con getWritableDataBase, hacemos un query que almacenamos en un cursor, lo recorremos (llamamos al método obtenerValores y añadimos el ticket creado al arraylist de tickets)

### obtenerValoresTicket()

Instancia un objeto y le da valores (obteniendolos desde el cursor). Este método es usado en obtenerTickets()

### asignarValoresTicket

Método que usaremos para establecer el par clave valor de los campos y su valor para la base de datos mediante la clase ContentValues y su método put. Puede ser utilizado tanto en el método agregarTicket como en el método editarTicket.

### agregarTicket(…)

Para este método tenemos 2 alternativas.  
Alternativa 1: Usar el método execSQL con la sentencia cruda “INSERT INTO …”  
Alternativa 2: Usar el método insert habiendo previamente usado correctamente la clase ContentValues para instanciar un objeto par clave valor.   
Ambas alternativas dan como resultado la inserción del registro en la tabla deseada.

### borrarTicket (…)

Abrimos la BD en modo escritura y usamos el método delete para borrar el registro/ticket y el método close para cerrar la conexión con la BD.

# ADAPTERS

Tenemos 3 adapters para los tickets y 1 para las categorías.  
Son todos iguales, lo único en que se diferencian es qué plantillas usan (y qué vistas de estas plantillas trabajan)

Con el constructor cargamos el arraylist con los objetos.  
El método getView se llama para cada ítem de un listview, desde aquí instanciamos las vistas y las rellenamos (el título, la descripción, la foto). Coge del array de datos al adaptador y le aplica una plantilla, aplicandola en el ListView.

# INTERFACES

Guardan constantes para los intents.