|  |  |  |  |
| --- | --- | --- | --- |
| **Alumno(s):** | Ricardo Fabián Juárez Santoyo  Juan Carlos Hurtado Morales  José Guadalupe Rivera Alcántar | **Fecha:** | 29/11/2018 |

|  |
| --- |
| OBJETIVO |
| El objetivo de este trabajo es crear una aplicación que le permita al usuario crear notas y recordatorios, las notas y los recordatorios pueden llevar archivos multimedia además de contener una breve descripción. Las notas pueden ser editados y/o eliminadas. |

|  |
| --- |
| MARCO TEÓRICO |
| Soporte de Idiomas  Lo más común para crear los soportes de idioma es con las herramientas dadas que crean un directorio res/ en el nivel superior del proyecto. Dentro de este directorio res/ se encuentran diferentes tipos de recursos en varios subdirectorios. El que nos interesa para este tipo de soporte es el res/values/strings.xml, el cual contendrá los valores Stings de la app (las cadenas de texto).  Lo más común es crear archivos string.xml lo cual con el string nos referimos a las etiquetas o textos que son visibles para el usuario.  Ilustración 1. Archivos xml para idioma.  Un ejemplo de esto entre el inglés y el español sería:   * Inglés:   <?xml version="1.0" encoding="utf-8"?>  <resources>      <string name="title">My Application</string>      <string name="hello\_world">Hello World!</string>  </resources>   * Español:   <?xml version="1.0" encoding="utf-8"?>  <resources>      <string name="title">Mi Aplicación</string>      <string name="hello\_world">Hola Mundo!</string>  </resources>  Lo que está en el calor name será la referencia que estará en la app que especifica que ese campo tendrá ese texto en el idioma seleccionado  Permisos Externos  Comprobar si existen permisos.  int permissionCheck = ContextCompat.*checkSelfPermission*(this,Manifest.permission.*CAMERA*  Si la app tiene el permiso, el método se compara con  **[PackageManager.PERMISSION\_GRANTED](https://developer.android.com/reference/android/content/pm/PackageManager.html?hl=es-419" \l "PERMISSION_GRANTED)** y esta puede continuar con la operación dentro del if. Si la app no tiene el permiso, el método se puede comparar con **PackageManager.** [**PERMISSION\_DENIED**](https://developer.android.com/reference/android/content/pm/PackageManager.html?hl=es-419#PERMISSION_DENIED)o poner en el else si se compara con permission\_granted.  if(permissionCheck == PackageManager.*PERMISSION\_GRANTED*){  }else{  }  Solicitar Permisos  public void Permiso(int i){  int permissionCheck = ContextCompat.*checkSelfPermission*(this,Manifest.permission. *CAMERA*);  if(permissionCheck == PackageManager.*PERMISSION\_GRANTED*){  Toast.*makeText*(getApplicationContext(),"Hay permiso",Toast.*LENGTH\_SHORT*).show();  saveExternal();  if (i==1){  saveExternal();  }else{  openExternal();  }  }else{  Toast.*makeText*(getApplicationContext(),"No hay permiso",Toast.*LENGTH\_SHORT*).show();  if (ContextCompat.*checkSelfPermission*(this,Manifest.permission. *CAMERA*)!= PackageManager.*PERMISSION\_GRANTED*) {  if ( ActivityCompat. *shouldShowRequestPermissionRationale* ( this , Manifest . permission . *CAMERA*)) {  } else {  ActivityCompat . *requestPermissions* ( this , new String [] { Manifest . permission . *CAMERA* },  MY\_PERMISSIONS\_REQUEST\_READ\_ESTORAGE );  }    }  }  }  Para solicitar permisos dentro de la aplicación se utiliza este método el cual está dentro del botón agregar foto que está en la sección multimedia el cual pide permiso para acceder a la cámara y así poder tomar fotos relacionados con la nota o tarea.  Notificaciones  Añadir la biblioteca de notificaciones  dependencies {      implementation "com.android.support:support-compat:28.0.0"  }  Creación de la notificación  Una notificación en su forma más básica y compacta (también conocida como forma contraída) muestra un icono, un título y una pequeña cantidad de texto de contenido.  Para comenzar, debe configurar el contenido y el canal de la notificación utilizando un[NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html)objeto. El siguiente ejemplo muestra cómo crear una notificación con lo siguiente:   * Un pequeño icono, establecido por [setSmallIcon()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setSmallIcon(int)). Este es el único contenido visible por el usuario que se requiere. * Un título, fijado por [setContentTitle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setContentTitle(java.lang.CharSequence)). * El texto del cuerpo, fijado por [setContentText()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setContentText(java.lang.CharSequence)). * La prioridad de notificación, establecida por [setPriority()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setPriority(int)). La prioridad determina qué tan intrusiva debe ser la notificación en Android 7.1 y más bajo. (Para Android 8.0 y superior, debe establecer la importancia del canal, que se muestra en la siguiente sección).   var mBuilder = NotificationCompat.Builder(this, CHANNEL\_ID)          .setSmallIcon(R.drawable.notification\_icon)          .setContentTitle(textTitle)          .setContentText(textContent)          .setPriority(NotificationCompat.PRIORITY\_DEFAULT)  De forma predeterminada, el contenido del texto de la notificación se trunca para ajustarse a una línea. Si desea que su notificación sea más larga, puede habilitar una notificación expandible agregando una plantilla de estilo con [setStyle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setStyle(android.support.v4.app.NotificationCompat.Style)). Por ejemplo, el siguiente código crea un área de texto más grande:  var mBuilder = NotificationCompat.Builder(this, CHANNEL\_ID)          .setSmallIcon(R.drawable.notification\_icon)          .setContentTitle("My notification")          .setContentText("Much longer text that cannot fit one line...")  **.setStyle(NotificationCompat.BigTextStyle()**  **.bigText("Much longer text that cannot fit one line..."))**          .setPriority(NotificationCompat.PRIORITY\_DEFAULT)  Soporte para Tabletas  Para agregar soporte a tabletas es necesario especificar un nuevo layout que es el que contendrá el soporte.  Para ello en la carpeta res/layout seleccionamos click derecho, new, Layout Resource File.  Ilustración 2. Paso 1 Layout.  Seleccionamos la opción Screen Width y presionamos el botón que contiene >> para agregarlo al layout.  Ilustración 3. Paso 2 Layout.  Una vez agregado nos pedirá que indiquemos el tamaño de la pantalla a lo ancho, en mi caso usamos 400, y una vez agregado el tamaño lo único que queda por hacer es ponerle el nombre a nuestro layout, pero no cualquier nombre, tiene que ser específicamente el mismo nombre del XML del diseño al que le queramos agregar este soporte.  Ilustración 4. Paso 3 Layout.  Una vez ingresado todos los datos, damos en OK y se nos generará una carpeta con los 2 modelos de XML para nuestro diseño de los layout.  Ilustración 5. Vista final del Layout.  Lo único que nos queda por hacer es comenzar a darle diseño a nuestra interfaz y listo, ahora cada vez que giremos el teléfono o se detecte una tableta, el diseño de la interfaz parecerá al de la imagen anterior.  SQLite  Uno de los principios principales de las bases de datos SQL es el esquema: una declaración formal de cómo está organizada la base de datos. El esquema se refleja en las sentencias de SQL que utiliza para crear su base de datos. Puede resultarle útil crear una clase complementaria, conocida como clase de contrato, que especifique explícitamente el diseño de su esquema de manera sistemática y autodocumentada.  Una clase de contrato es un contenedor para constantes que definen nombres para URI, tablas y columnas. La clase de contrato le permite usar las mismas constantes en todas las otras clases en el mismo paquete. Esto le permite cambiar el nombre de una columna en un lugar y hacer que se propague a lo largo de su código.  Una buena manera de organizar una clase de contrato es colocar definiciones que sean globales para toda su base de datos en el nivel raíz de la clase. Luego crea una clase interna para cada tabla. Cada clase interna enumera las columnas de la tabla correspondiente.  Por ejemplo, el siguiente contrato define el nombre de la tabla y los nombres de las columnas para una sola tabla que representa una fuente RSS:  Una vez que haya definido el aspecto de su base de datos, debe implementar métodos que creen y mantengan la base de datos y las tablas. Aquí hay algunas declaraciones típicas que crean y eliminan una tabla:  Al igual que los archivos que guarda en el [almacenamiento interno](https://developer.android.com/guide/topics/data/data-storage.html#filesInternal) del dispositivo, Android almacena su base de datos en la carpeta privada de su aplicación. Sus datos están seguros, porque de forma predeterminada, esta área no es accesible a otras aplicaciones o al usuario.  Insertar  Inserte datos en la base de datos pasando un [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html) objeto al [insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String,%20java.lang.String,%20android.content.ContentValues)) método:  Mostrar  Para leer desde una base de datos, use el [query()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "query(java.lang.String,%20java.lang.String[],%20java.lang.String,%20java.lang.String[],%20java.lang.String,%20java.lang.String,%20java.lang.String)) método, pasándole los criterios de selección y las columnas deseadas. El método combina elementos de [insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String,%20java.lang.String,%20android.content.ContentValues)) y [update()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "update(java.lang.String,%20android.content.ContentValues,%20java.lang.String,%20java.lang.String[])), excepto que la lista de columnas define los datos que desea obtener (la "proyección"), en lugar de los datos que se deben insertar. Los resultados de la consulta se le devuelven en un [Cursor](https://developer.android.com/reference/android/database/Cursor.html)objeto.  Eliminar  Para eliminar filas de una tabla, debe proporcionar criterios de selección que identifiquen las filas del [delete()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "delete(java.lang.String,%20java.lang.String,%20java.lang.String[]))método. El mecanismo funciona igual que los argumentos de selección para el [query()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "query(boolean,%20java.lang.String,%20java.lang.String[],%20java.lang.String,%20java.lang.String[],%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String))método. Divide la especificación de selección en una cláusula de selección y argumentos de selección. La cláusula define las columnas a observar y también le permite combinar pruebas de columna. Los argumentos son valores con los que se debe probar que están vinculados a la cláusula. Debido a que el resultado no se maneja de la misma manera que una instrucción SQL normal, es inmune a la inyección de SQL.  Multimedia  Poner en Manifest el siguiente código:  <uses-feature  android:name="android.hardware.camera"  android:required="true" />  Esto activa la cámara de modo que no pedirá permisos especiales al usuario.  Código de vista:  <ImageView  android:id="@+id/imgPhotoUser"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:clickable="true"  app:srcCompat="@drawable/ic\_ic\_rg\_add\_photo"  tools:layout\_editor\_absoluteX="307dp"  tools:layout\_editor\_absoluteY="8dp" />  Imagen de una cámara darle la propiedad al imageview, el método de click =  android:clickable=”true”  En la clase java, asociar la imagen y darle el evento onClick.  ImageView addPhoto = (ImageView) findViewById(R.id.imgPhotoUser);  addPhoto.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  //codigo  }  });  Dentro se crea un Intent, la cual llamará a la acción de capturar una imagen y si se va a visualizar en ImageView.  Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  startActivityForResult(takePictureIntent, REQUEST\_IMAGE\_CAPTURE);  }  startActivityForResult es un método de android que es llamado cuando el intent trae algo para mostrar, así que pasamos el intent y request image que es una variable static en 1, esta variable es una respuesta de correcto.  Lo siguiente es pasar la foto que va anexado en un extra de data, data es el intent que enviamos anteriormente, y luego lo compartimos en bitmap para poder mostrarlo en nuestra pantalla.  @Override  protected void onActivityResult(int requestCode, int resultCode, Intent data) {  if (requestCode == REQUEST\_IMAGE\_CAPTURE && resultCode == RESULT\_OK) {  Bundle extras = data.getExtras();  Bitmap imageBitmap = (Bitmap) extras.get("data");  addPhoto.setImageBitmap(imageBitmap);  }  }  Notificaciones  public void showNotification() {    NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);    Podrás mostrar el icono de la notificación, en este caso una alerta.  Notification notification = new Notification(android.R.drawable.stat\_sys\_warning,  "Notificación", System.currentTimeMillis());    CharSequence titulo = "Alerta";    Clase de Notification  Intent notificationIntent = new Intent(this, NotificationActivity.class);  PendingIntent contIntent = PendingIntent.getActivity(this, , notificationIntent, );  notification.setLatestEventInfo(this, "Aviso de notificación", "Esto es un ejemplo de notificación", contIntent);    notification.flags |= Notification.FLAG\_AUTO\_CANCEL;    int not\_id = 1;  notificationManager.notify(not\_id, notification);  Fragments  Un [Fragment](https://developer.android.com/reference/android/app/Fragment.html?hl=es-419) representa un comportamiento o una parte de la interfaz de usuario en una [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=es-419). Puedes combinar múltiples fragmentos en una sola actividad para crear una IU multipanel y volver a usar un fragmento en múltiples actividades. Puedes pensar en un fragmento como una sección modular de una actividad que tiene su ciclo de vida propio, recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando (algo así como una "subactividad" que puedes volver a usar en diferentes actividades).  Un fragmento siempre debe estar integrado a una actividad y el ciclo de vida del fragmento se ve directamente afectado por el ciclo de vida de la actividad anfitriona. Por ejemplo, cuando la actividad está pausada, también lo están todos sus fragmentos, y cuando la actividad se destruye, lo mismo ocurre con todos los fragmentos. Sin embargo, mientras una actividad se está ejecutando (está en el estado del ciclo de vida reanudada), puedes manipular cada fragmento de forma independiente; por ejemplo, para agregarlos o quitarlos. Cuando realizas una transacción de fragmentos como esta, también puedes agregarlos a una pila de actividades administrada por la actividad; cada entrada de la pila de actividades en la actividad es un registro de la transacción de fragmentos realizada. La pila de actividades le permite al usuario invertir una transacción de fragmentos (navegar hacia atrás) al presionar el botón Atrás.  Un objeto POJO es una instancia de una clase que no extiende ni implementa nada en especial. Por ejemplo, un [Servlet](https://es.wikipedia.org/wiki/Servlet" \o "Servlet)tiene que extender de HttpServlet y sobrescribir sus métodos, por lo tanto no es un POJO. En cambio, si se define una clase 'Persona', con sus atributos privados y sus correspondientes getters y setters públicos, una instancia de esta simple clase es un objeto POJO. |

|  |
| --- |
| DESARROLLO |
| * Model * DaoMultimedia: Sirve para la llamada y guardada de los datos de la sección multimedia * DaoNotas: Sirve para la llamada y guardada de los datos de la sección Notas * DaoRecordatorios: Sirve para la llamada y guardada de los datos de la sección multimedia * MiDBRecordatorios: Especifica lo que contiene la base de datos y como esta armada y vinculada * POJO : Esta sección solo contiene los getter y setter para la creación de los obj de cada clase y asi facilitar el uso de los atributos de cada obj en las demás actividades cuando se llaman los datos de la bd. * Multimedia * Notas * Recordatorios * ActivityRecordatorios * GridViewAdapter * ListViewAdapter1 * ListViewAdapter2 * MainActivity: Este archivo se encarga de la inicialización de la app cuando se ejecuta en el Android y muestras la ventanas * SeccionsPageAdapter * Servicio * Tab1Fragment * Tab2Fragment * actividadMultimedia * menuActivity |

|  |
| --- |
| CONCLUSIONES |
| * Se desarrolló una aplicación que cumple con lo requerido. * Se implementaron todos los requerimientos que se necesitan para su correcto funcionamiento. * Se integró la mayor parte de los conocimientos adquiridos durante la materia a través de las practicas que se realizaron durante el semestre. |

|  |
| --- |
| BIBLIOGRAFÍA |
| <https://developer.android.com/guide/components/fragments?hl=es-419>  <https://androidstudiofaqs.com/tutoriales/notificaciones-barra-estado-android-studio>  <https://unprogramador.com/tomar-foto-en-android/>  <https://unprogramador.com/tomar-foto-en-android/> |

Ilustración 6. Estructura de la base de datos.

Ilustración 7. Declaración de la base.

Ilustración 8. Insertar datos en una tabla.

Ilustración 9. Mostrar datos de una tabla.

Ilustración 10. Eliminar datos de la base.

Ilustración 11. Ejemplo de Fragments.