|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Materia:** |  | PROGRAMACIÓN MOVIL I | | |
| **Nombre Informe:** |  | AVANCE DE PROYECTO FINAL | | |
| **Alumno(s):** |  | JOSE LUIS GONZALEZ LUIS  LUIS FERNANDO ROJAS MARTINEZ | **Fecha: 05/11/2020** |  |

**Objetivo**

Desarrollar la aplicación móvil solicitada por el cliente, empleando los conocimientos adquiridos a lo largo del semestre para lograr completarla con los requerimientos indicados previamente por el cliente

.

**Temas del plan de estudios**

**Componentes de una aplicación Android**

* Componentes de aplicaciones
* El archivo AndroidManifest
* Actividades
* El ciclo de vida de una actividad.
* Enlazando actividades usando Intents
* Paso y retorno de parámetros entre actividades

**Soporte para diferentes dispositivos**

* Material de diseño de Android
* Recursos de la aplicación.
* Soporte para diferentes lenguajes.
* Soporte para diferentes pantallas
* Soporte para diferentes versiones de plataformas

**Interfaz de usuario dinámica con fragmentos**

* Usando la librería de soporte
* Creación de un fragmento
* Construcción de una interfaz de usuario flexible
* Comunicación entre fragmentos

**Interfaz de usuario**

* Layouts
* Views básicos
* Eventos de controles
* Interfaz de usuario en tiempo de ejecución
* Menús
* Cuadro de dialogo y notificaciones
* Controles para datos
* Adaptadores
* ArrayAdapter
* SimpleCursorAdapter
* BaseAdapter
* ListView, Sppiner, GridView.
* Búsquedas en la aplicación
* Componentes para la navegación en la aplicación.
* Actualización por arrastre (swipe-to-refresh)

**Datos**

* Almacenando datos de tipo llave-valor (Preferencias)
* Almacenamiento en la memoria interna.
* Almacenamiento en la memoria externa.
* Bases de datos (SQLite).

**Multimedia**

* Multimedia en Android
* Reproducir audio y video
* Grabar audio y video
* Aplicaciones del sistema

**Dispositivos Wereables**

* Tipos de dispositivos Wereables
* Android Wear
* Construcción de aplicaciones para Android Wear

**Material**

* Computadoras
* Internet
* Software Pencil
* Practica guiada
* Documento necesidades del cliente.
* Plataforma GitHub
* Android Studio
* Plataforma Google Meet

**Marco Teórico**

Se requiere una aplicación para su smartphone (Android) que le permita registrar notas y tareas multimedia y a su vez estas tengan la capacidad de programarles la cantidad de recordatorios que se requieran.

Se comenta que a las notas puede registrársele la siguiente información: título, una descripción, y podrá adjuntar “n” cantidad de archivos multimedia. Las fuentes para agregar estos archivos pueden ser de videos y fotos capturados en ese mismo momento o seleccionar un archivo del sistema. Por cada archivo que se adjunte podrá agregársele una descripción.

La nota podrá clasificarse como nota o tarea. Para el caso de que sea clasificada como una tarea se podrá asignar fecha y hora en que se debe cumplir la tarea, además de agregar recordatorios. Se espera que cuando se cumple cada fecha de recordatorio la aplicación envié notificación en la barra de tareas, así mismo, también cuando la tarea tenga que ser cumplido.

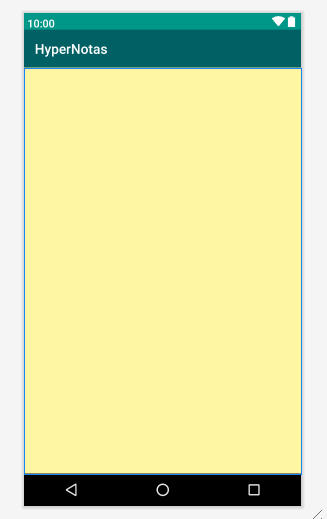
Otra de las características que debe tener una nota clasificada como una tarea es que cuando la tarea se cumpla esta pueda ser marcada como cumplida o que pueda posponerse por más tiempo.

También se quiere que la aplicación sea soportada en smarthphones y tablet’s, que pueda ser instalada en Jelly Bean y superiores, además de soporte al idioma inglés y español.

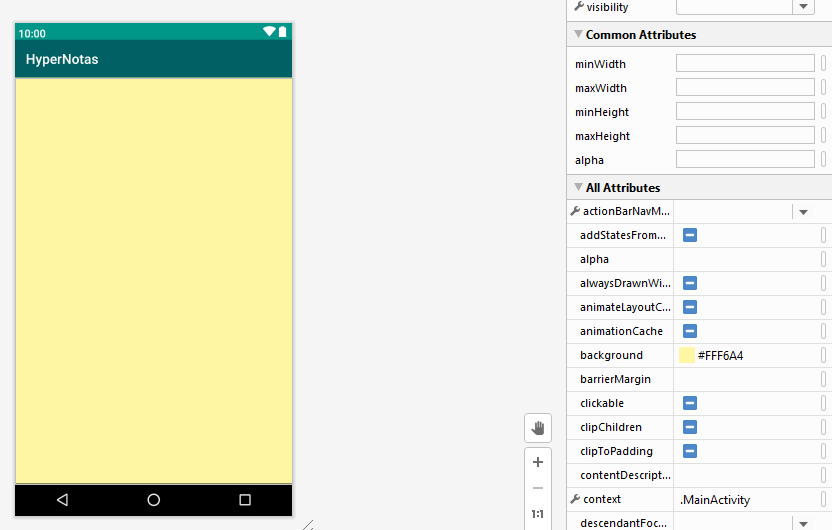
Además, se requiere que sea una aplicación muy fluida y funcional, de tal manera que para hacer un registro o visualizar información tenga que hacer el mínimo de movimientos (Para lograr este requerimiento se debe utilizar controles como listas, cuadros de dialogo, barra de acciones, notificaciones, cejillas o pestañas, grids, menús opcionales y contextuales, controles de fecha, estilos y controles básicos.

* Las notas y tareas registradas pueden ser editadas y eliminadas.
* Se quiere realizar "n" grabaciones de audio y agregarlas a la nota o tarea.
* Se necesita que todos los archivos adjuntados se puedan diferencias fácilmente mediante una miniatura que lo represente.
* También requiere poder realizar búsqueda de notas y tareas de tal manera que cuando se tengan muchos registros puedan buscarse de manera más fácil. Y que la palabra o patrón a buscar debe realizare sobre los campos descripción y nombre de la nota.
* Otra cosa más, las notas deben estar ordenadas por fecha de registro y las tareas por fecha de en qué debe realizarse la tarea.

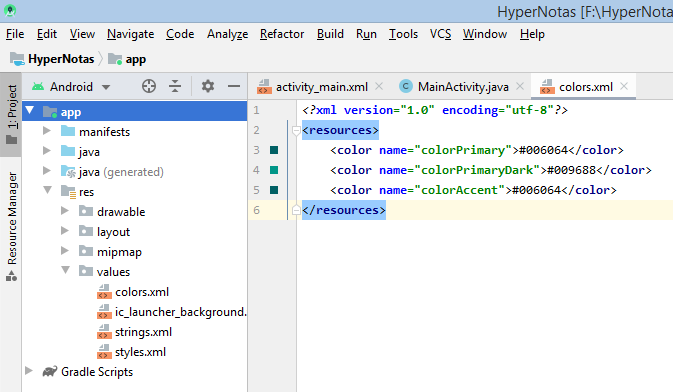
**Desarrollo**

**Diseño general de la Aplicación (Icono, color del ActionBar, Background de las actividades)**

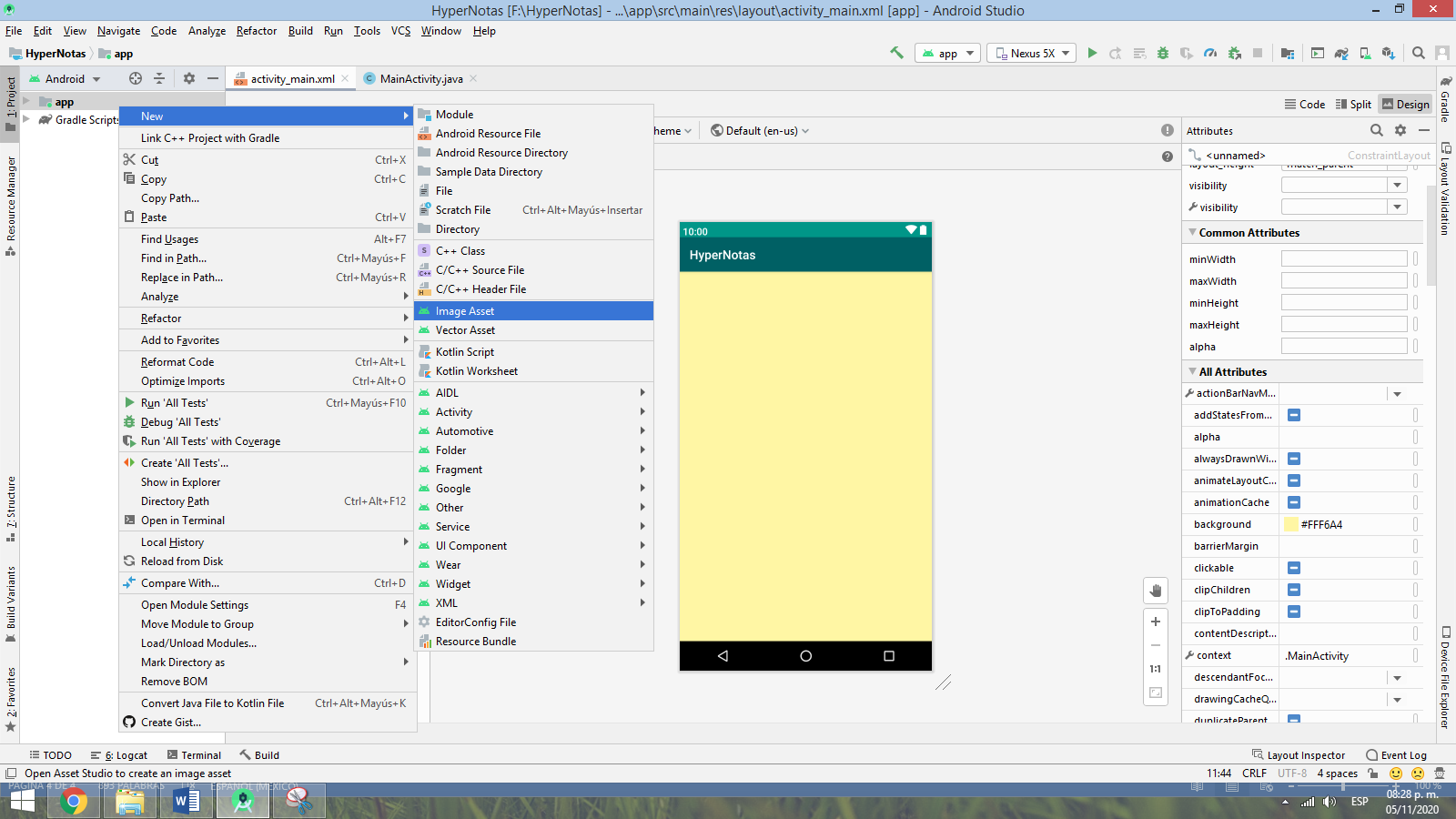
Nosotros decidimos empezar nuestro proyecto con el diseño que va a tener toda la aplicación en general. Esto incluye el icono que se mostrará y el color que tendrán todas las actividades tanto de fondo, como en la ActionBar. El nombre de nuestra aplicación es “HyperNotas”.

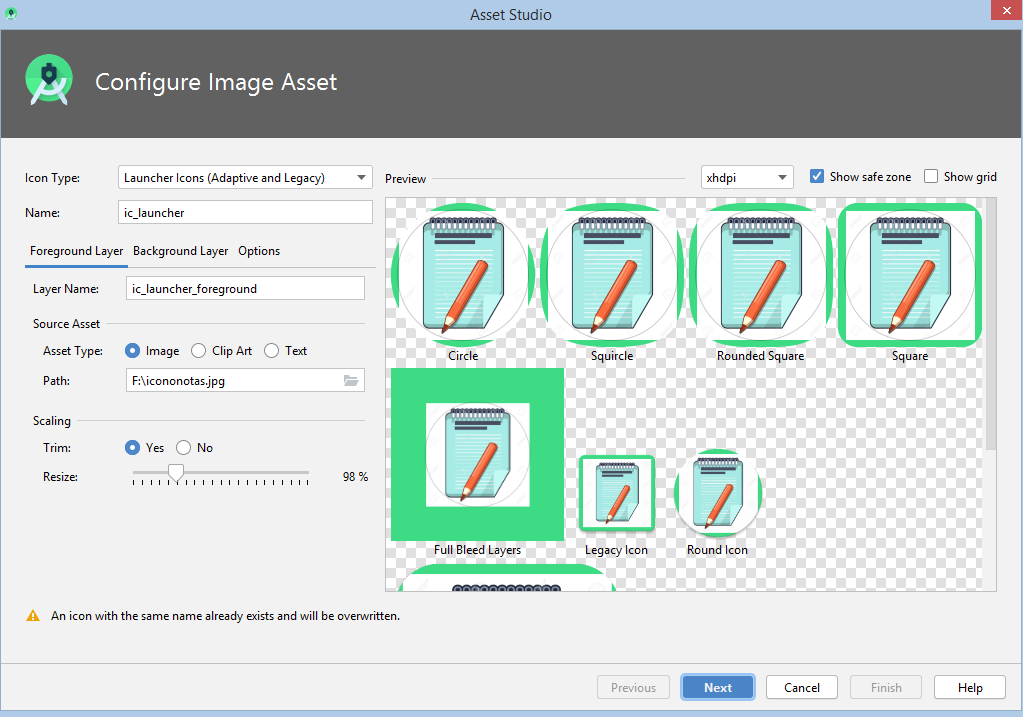


Para colocar el color de fondo en las actividades, se debe ir a la propiedad background e ingresar el código hexadecimal del color que se desea asignar.



Para el color de la ActionBar debemos ir al archivo colors.xml ubicado en la carpeta values. En esté nosotros debemos ingresar los colores en formato hexadecimal que queremos mostrar como diseño en nuestra app.

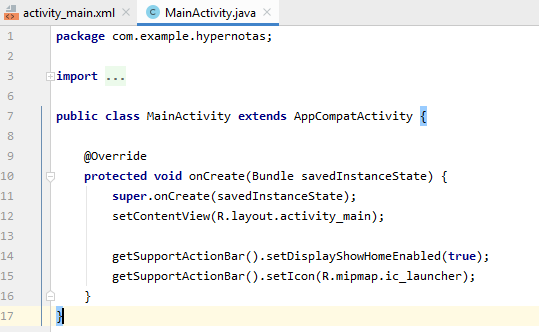


Para ingresar el icono de nuestra aplicación, debemos hacer clic derecho en la carpeta “app”, después seleccionamos “New” y por ultimo damos clic en “Image Asset”.

Aparecerá esta ventana en la que lo único que se recomienda cambiar es: “Path:” para seleccionar la ruta en la que se encuentra el icono, y “Trim” cambiarlo a yes, para que la imagen se ajuste. Luego damos Next.

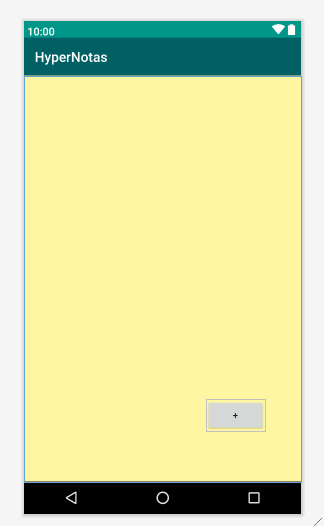


Aparecerá esta ventana la cual marca todas esas palabras en rojo porque al haber dejado el mismo nombre de icono, se sustituirá el que viene por defecto por el nuevo icono. Ahora solo damos en Finish.



Después en el método onCreate de la clase MainActivity debemos ingresar las 2 líneas de código que se muestran en la imagen. La primera indica que el icono se muestre siempre, y la segunda le dice que imagen mostrar a través de la ruta indicada.

**Clasificar el tipo de nota**

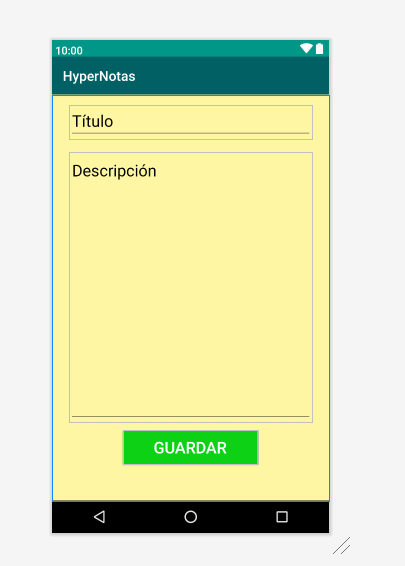


Para llevar el control de si el archivo es una nota simple o tarea multimedia, nosotros añadimos un botón (botón flotante más adelante) en la parte inferior derecha de la interfaz principal. El objetivo de este botón es que al presionarlo abra un AlertDialog que nos indique las opciones de “Nota”, “Tarea Multimedia” y “Cancelar”. Cada opción debe mandar al usuario a su activity correspondiente.

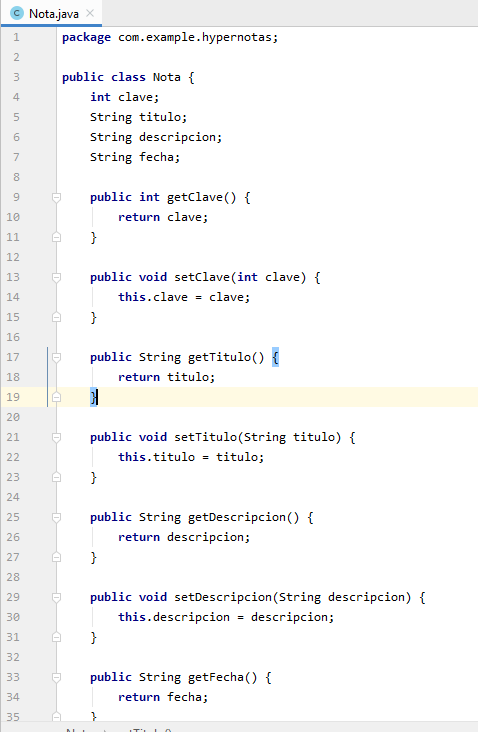


Este es el método que permite crear y mostrar el cuadro de dialogo. Se tiene el título y después las opciones a escoger, si se elige la opción Nota nos enviara a la actividad CrearNota en donde se lleva a cabo el registro de las notas de texto, en el caso de elegir la opción Tarea Multimedia se envía a su actividad correspondiente también. Se agregó otro elemento para que nos dé la posibilidad de cancelar nuestra elección.

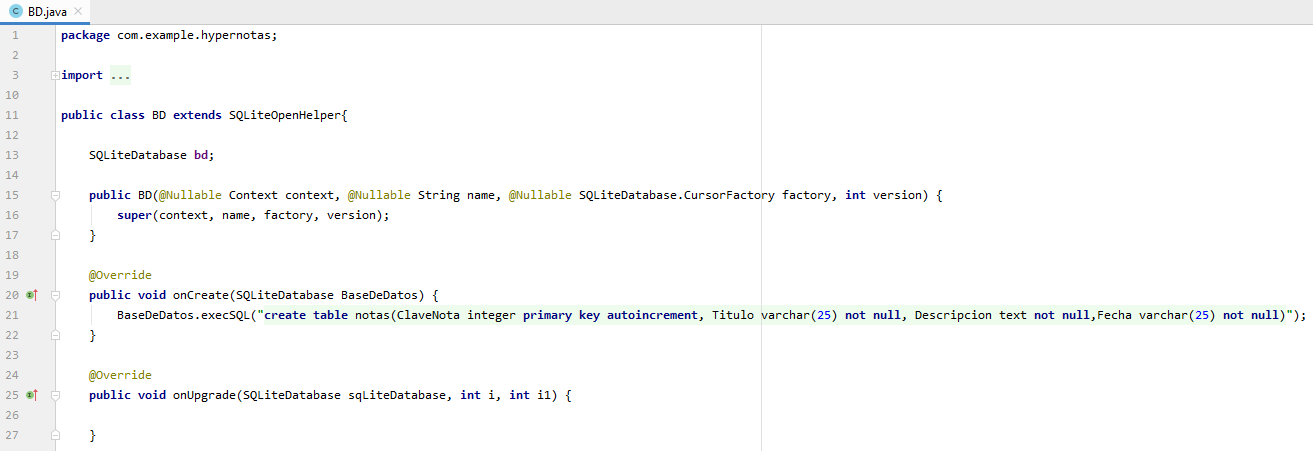
**Crear nota**



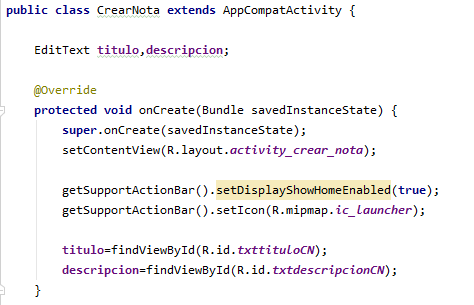
La interfaz para crear nota contiene 3 componentes: un EditText para el título, un MultiText para escribir el contenido de la nota, y un botón para guardarla. Además se crea la fecha actual al presionar el botón guardar para almacenarse en la base de datos, y el titulo debe tener como máximo 25 caracteres.



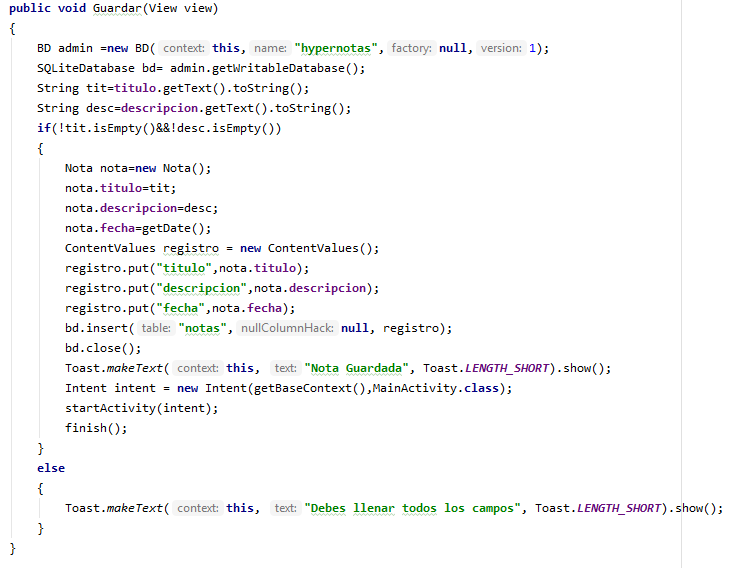
Se creó la clase Nota para que sean los objetos de tipo Nota de texto, estos contienen 4 atributos: la clave, el título, su descripción y la fecha en la que se crea la nota. Además se agregaron los getters y setters. No se añadió constructor porque estos objetos se utilizarán como elementos de una lista, y también ocuparemos llamar está clase sin el constructor para varias acciones más adelante.



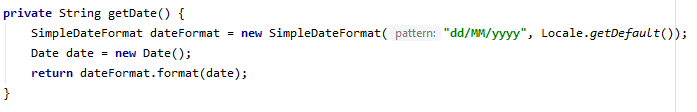
Se creó la clase BD donde se empieza a desarrollar la base de datos, por el momento solo está creada la tabla notas, esta consta de 4 campos: una clave para la nota, su título, descripción y fecha de creación. La clave de la nota es auto incrementable.



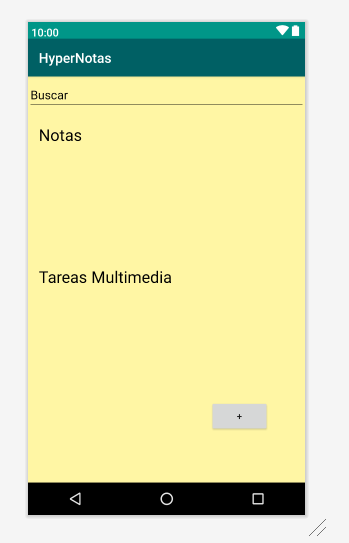
En la clase CrearNota, dentro del método onCreate, primero se vuelve a indicar que nos muestre el icono de la app en la barra de acciones, después se asigna la relación para que puedan interactuar las variables de tipo EditText y los componentes que se tienen en el layout activity\_crear\_nota.



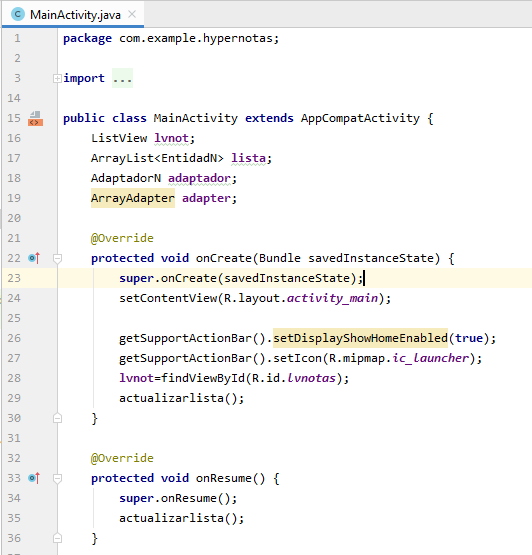
El método Guardar permite que se inserte un nuevo registro en la base de datos hypernotas. Se verifica que los campos no estén vacíos, si lo están manda un mensaje indicando que debe llenarlos. Cuando se valida correctamente comienza el proceso para insertar los datos en la tabla notas, añadiendo primero cada variable a un nuevo objeto de tipo Nota en sus atributos correspondientes, después se crea un ContetValues para enviar todos los datos individuales como un solo registro, al terminar se cierra la BD, manda un mensaje notificando que se guardó la nota. Por último se finaliza la actividad y nos regresa a la interfaz principal.

También en esta clase está el método getDate, el cual obtiene la fecha actual del registro, en el formato indicado, para posteriormente asignárselo al atributo fecha del objeto tipo Nota.

**Lista de notas guardadas**

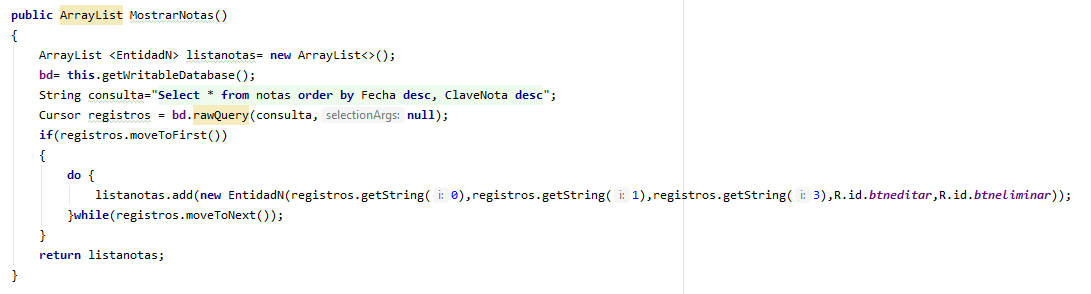


Para mostrar las notas que están en la base de datos, se actualizo la interfaz principal. Se incluyó un EditText que será el que permita buscar las notas y tareas multimedia, se agregaron 2 TextViews uno para el título de la sección de Notas y el otro para Tareas Multimedias, debajo de estos hay 1 ListView por sección, ya que ahí es donde se mostraran las notas y tareas guardadas.

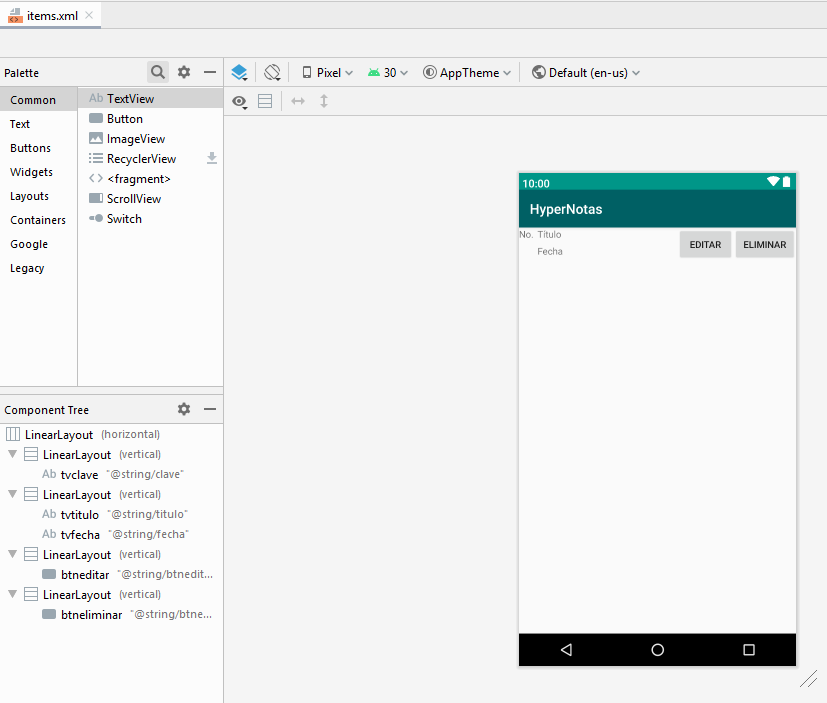


En la clase MainActivity se agregaron variables globales. En el método onCreate se necesita mandar llamar el método actualizarlista. Porque al iniciar la app se deben mostrar las notas registradas. Tambien este método se debe llamar en el onResume, para que de igual manera se muestren siempre los cambios realizados en la app.

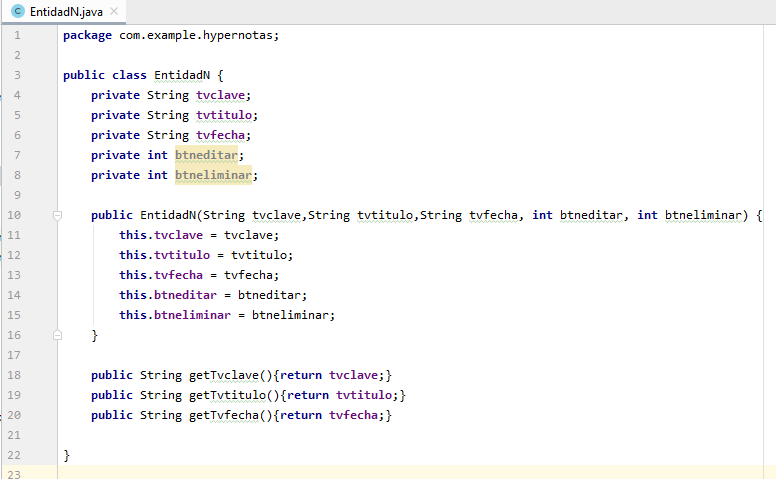


En el método actualizarlista se codificaron las acciones necesarias para que al crearse la actividad se nos muestre el ListView lleno de las notas que se encuentran en la base de datos. También se validó que si no hay registros solamente muestre un mensaje en el ListView indicando que no hay notas.

Y en la clase BD se codifico el método MostrarNotas el cual devuelve una lista, que por medio de una consulta a la base de datos se llena de objetos de tipo EntidadN, este tipo de objeto es necesario para darle un aspecto visual al ListView con el cual se muestra la clave, título y fecha de la nota, pero además los botones editar y eliminar por ítem de la lista. La consulta sql devuelve todas las notas registradas de manera descendente por su fecha y después por su clave, de esta manera en el ListView siempre se mostrará la nota más reciente primero.



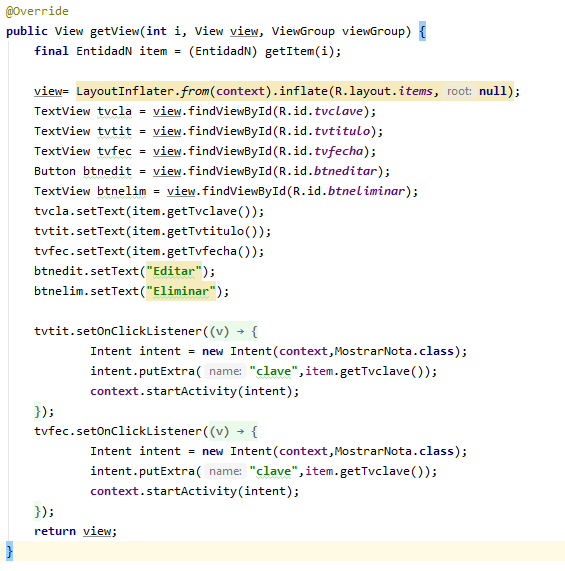
El archivo ítems.xml es el que ayuda a darle una mejor apariencia a la manera en la que se muestran los elementos de nuestro ListView. Está compuesto de varios LinearLayout para separar los componentes.



La clase EntidadN es la que ayuda a agregar la información a los elementos de nuestro ListView, son 5 atributos, los mismos que tiene el layout ítems.xml.

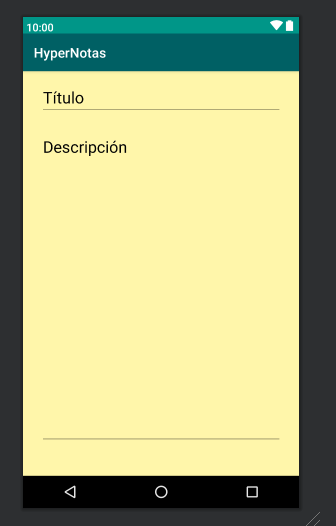


Se necesita la clase personalizada AdaptadorN, para establecerle esté adaptador a nuestro ListView y sea posible lograr el apartado visual planeado.

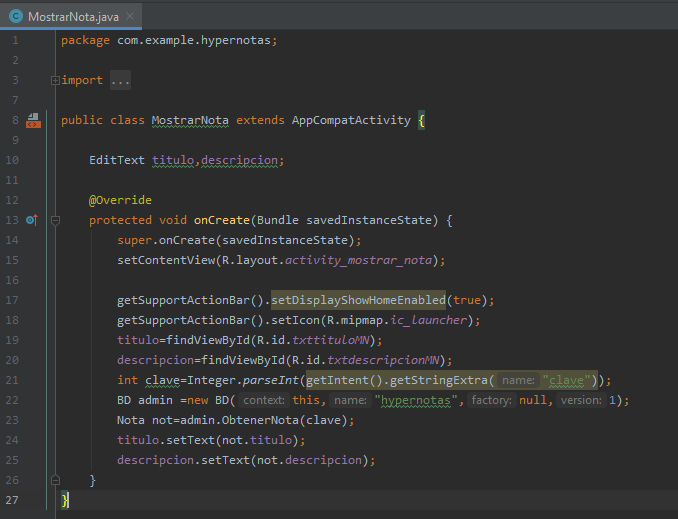


El método getView de esta clase es el que nos permite llenar de información cada uno de los elementos EntidadN del ListView. Se tiene que decir que se va a inflar la vista con el archivo ítems.xml, después se debe hacer la relación de variables con los componentes del archivo ítems.xml y por último se agrega la información y acciones que se quieran establecer en los componentes mostrados en la lista cada vez que se cree uno nuevo.

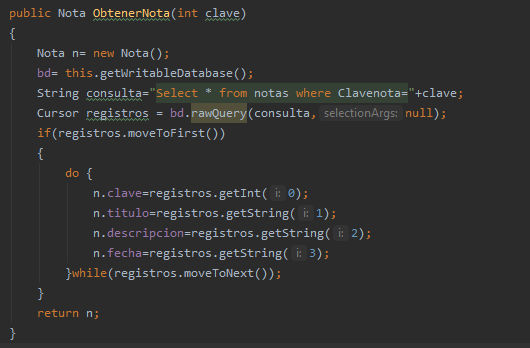
**Mostrar nota**



La interfaz para mostrar el contenido de una nota consta de 2 EditText que solo permiten visualizar los datos, no se puede ingresar nada. Además no hay botón de guardado ni otra cosa, porqué esta interfaz solo es para visualizar el contenido de la nota seleccionada sin modificar nada por error.

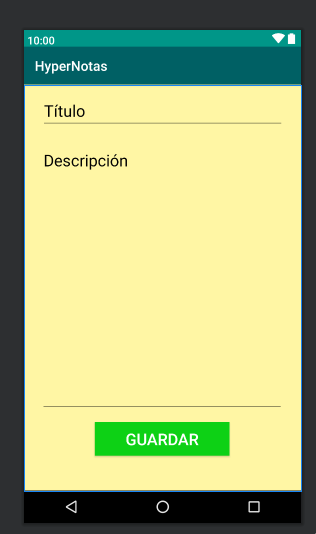


La clase MostrarNota permite obtener la clave de la nota seleccionada previamente al hacer clic sobre el título o fecha del ListView de notas, en la actividad principal. Como la clave se envió desde otra actividad en esta clase la obtenemos y ejecutamos el método ObtenerNota, pasando la clave como parámetro, para traer los datos de esta nota desde la BD, después se colocan los valores título y descripción en sus correspondientes componentes. Todo esto se hace en el método onCreate.

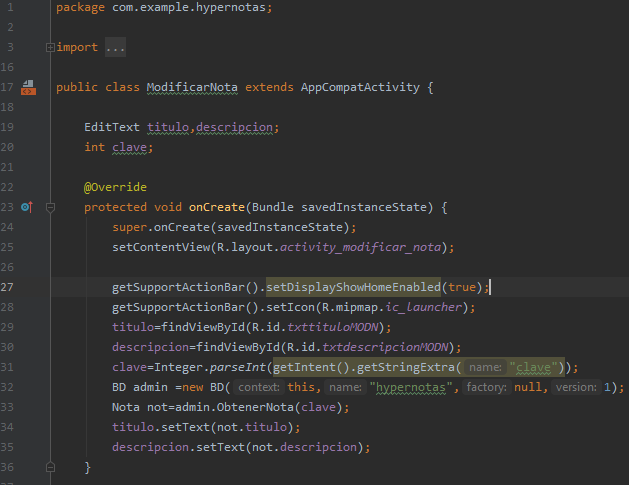


El método ObtenerNota ayuda a obtener los valores de una nota en específico desde la base de datos, por medio de una clave.

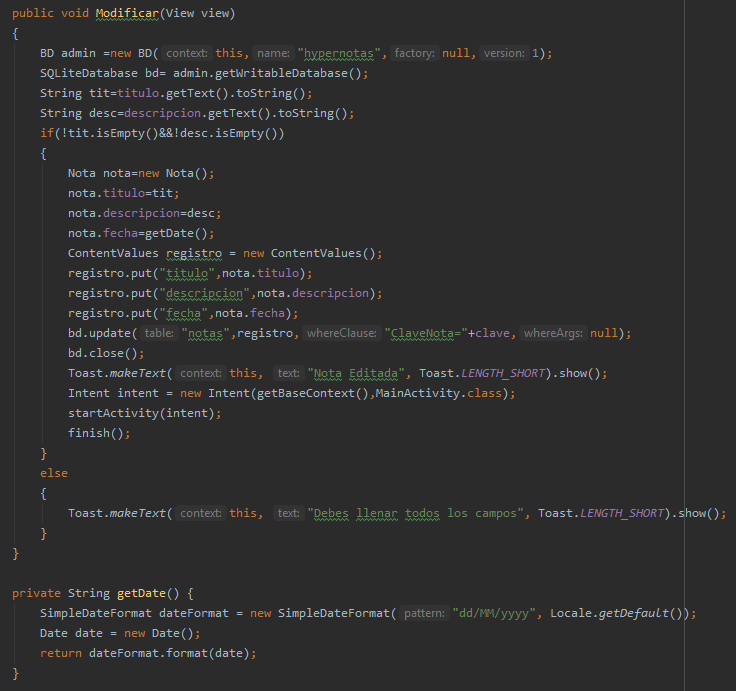
**Modificar nota**



La interfaz para modificar nota es el mismo diseño utilizado para la creación de una nota. En este si se permite editar el contenido que se mostrara al cargarse la actividad.

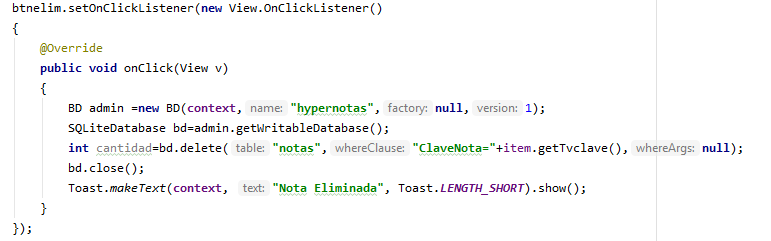


Del mismo modo se utilizó código idéntico para traer los datos de la nota que se selecciona en la actividad principal para modificarla. En la clase ModificarNota se obtiene la clave, se manda llamar el método ObtenerNota y después se pasan los valores a los componentes, luego el usuario modifica lo que guste.



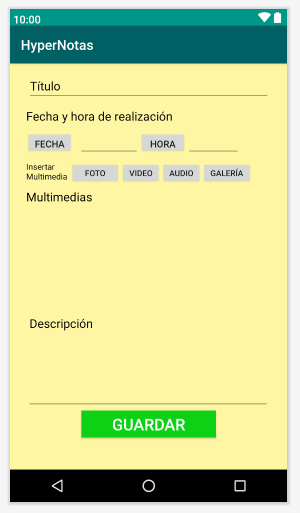
También el proceso para modificar una nota es casi idéntico al proceso de creación, en lugar de insert, usamos update para pasar los nuevos valores a modificar. Teniendo en cuenta que la clave es lo que nos permite que solo la nota seleccionada se vea afectada. Otra cosa es que modificamos la fecha con la actual en la que se realizan modificaciones, para que así esta nota sea de las principales en visualizarse en la interfaz principal.

**Eliminar nota**



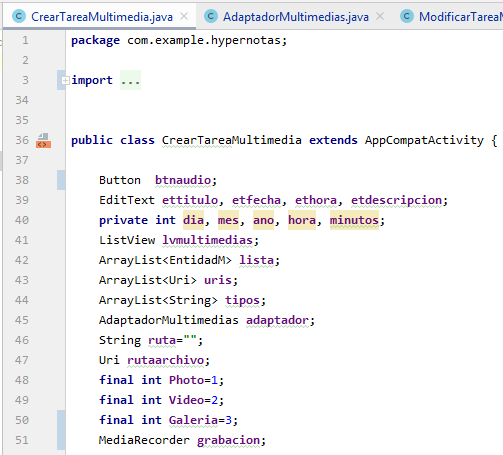
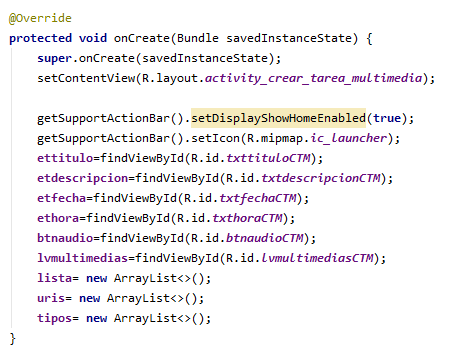
Para eliminar las notas se añadió comportamiento a los botones dinámicos en los que se elimina el dato de la base de datos. Por el momento aún no se actualiza la lista en tiempo de ejecución, se debe volver a cargar la actividad principal.

**Crear tarea multimedia**

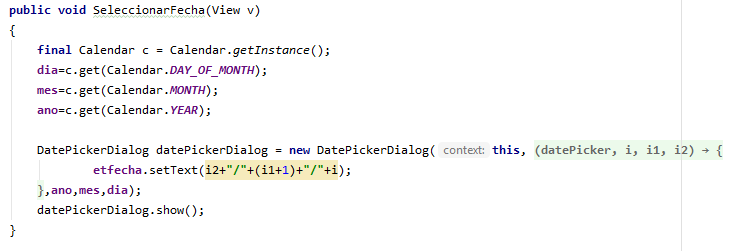


La interfaz de crear una tarea multimedia permite ingresar el título, una fecha y hora de realización, diferentes tipos de multimedia y una descripción.

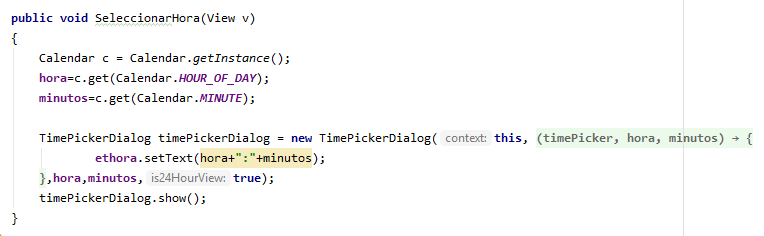
La fecha y hora se escogen con ayuda de su botón correspondiente. Para las multimedia al presionar el botón que se desee se realizara una acción diferente, para foto y video nos manda a otra aplicación que puede realizar dicha acción de capturar foto o video. Para el audio se comienza la grabación al presionar el botón, para finalizar la grabación se debe volver a pulsar el botón. Por último el botón de galería nos envía a la app en donde se pueden seleccionar las fotos que están almacenadas en el dispositivo.



En la clase CrearTareaMultimedia lo primero que se tiene es declarar las variables globales, después en el método onCreate se hace la relación entre componentes y variables, se asigna el icono a la actividad y se inicializan arreglos que se ocuparan posteriormente.



El método SeleccionarFecha es el que permite visualizar y seleccionar de un DatePickerDialog la fecha en la que se debe realizar a más tardar la tarea multimedia.

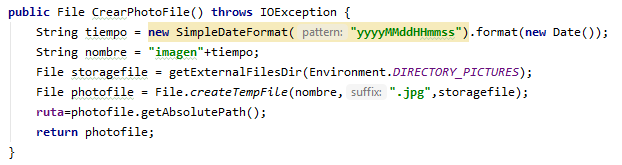


Mientras que el método SeleccionarHora nos permite realizar las mismas acciones pero en un TimePickerDialog.

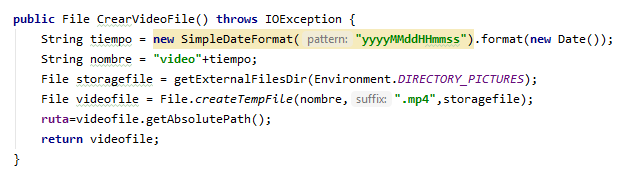
Para ambos métodos se les dio formato de cómo se mostraran los datos obtenidos.



Para las fotos se crearon 2 métodos, el de TomarFoto es el principal ya que este es el que te permite ir hacia una actividad que realice la captura de fotografías y manda a llamar el resto de métodos.



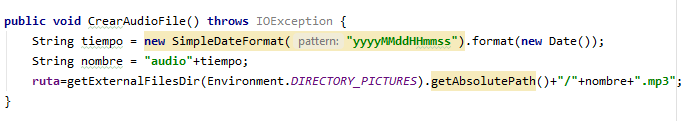
En CrearPhotoFile es donde le establecemos el nombre a nuestras fotos que capturamos, además también el formato en el que se quiere la fotografía. Otra cosa muy importante es que aquí obtenemos la ruta de la foto, la cual posteriormente enviaremos a la BD.



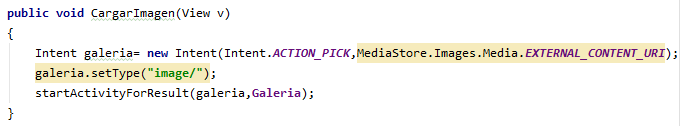
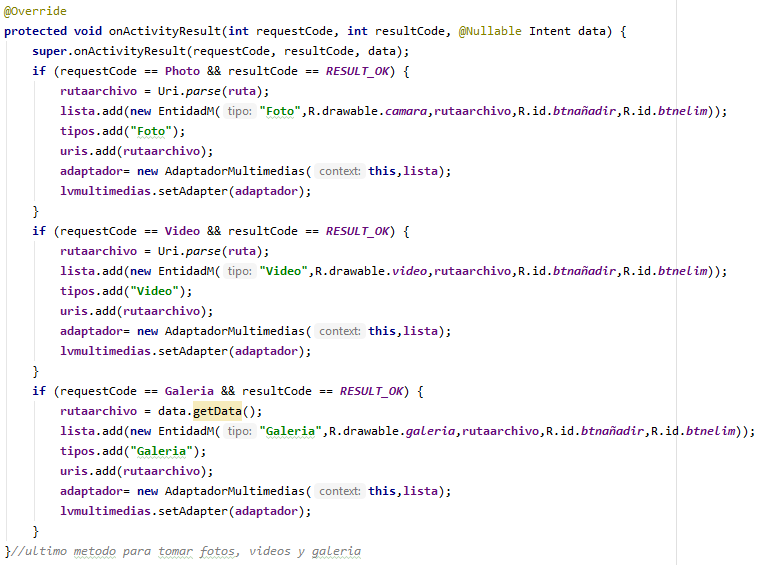
Los métodos para tomar un video tienen las mismas funciones que los métodos para tomar fotos, solo que cambia en algunas variables clases, funciones y métodos que se usan, ya que estos están específicamente empleados para videos.



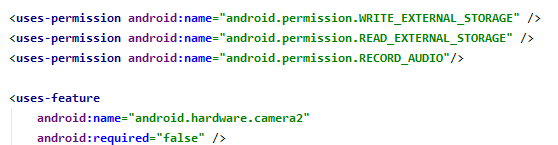




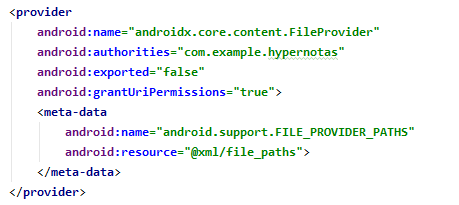
Para grabar los audios se emplea también uso del método CrearAudioFile con el que se crea el archivo mp3 con su nombre. El método GrabarAudio es el que permite utilizar el micrófono del celular para generar la grabación. En este método se asigna los elementos al LstView y se guarda la Uri del archivo. Además se cambia el texto del botón para que se diferencie cuando está grabando de cuando no.

Solo se necesita del método CargarImagen para poder seleccionar alguna foto de la galería de nuestro dispositivo móvil.

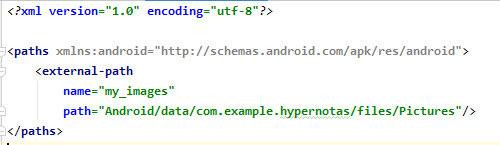
Por ultimo en onActivityResult después de tomar la foto, video o seleccionar la imagen de la galería; se transforma la ruta en Uri, se añade a una lista una nueva EntidadM, además se agrega a otras 2 listas el tipo y la Uri, para hacer acciones posteriores. Ya al final se le establece la lista al adaptador y se muestra en el ListView. Se hicieron validaciones para realizar diferentes funciones correspondientes a si es foto, video o imagen de galería.



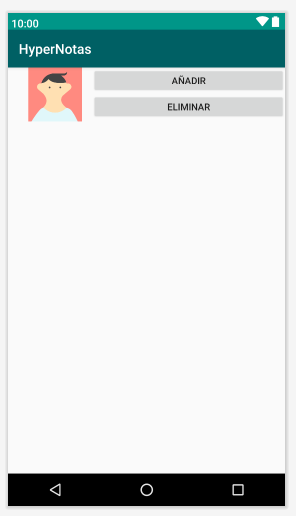
También asignamos los permisos en el Archivo Manifestó para poder escribir o leer en la memoria externa y usar la cámara o microfono.



E igual se utilizó el provider para dar los permisos, y decirle en donde se van a guardar las multimedias.



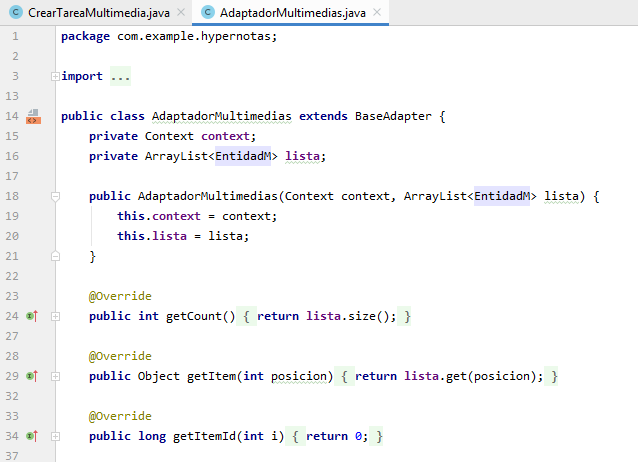
El archivo xml file\_paths es el que da acceso a la carpeta donde se van a guardar los archivos. Se usa esta ruta para guardar todas las multimedias (Fotos, Videos, Audios).



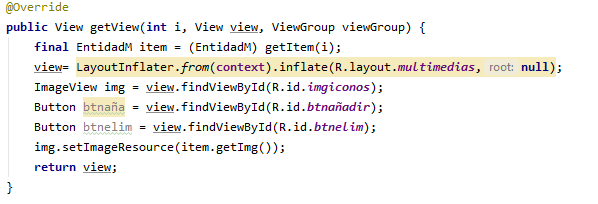
Se creó una vista llamada multimedias.xml compuesta de un ImageView, que es donde se muestra un icono representativo de cada tipo de multimedia (Foto, Video, Audio, Galería) y 2 botones.

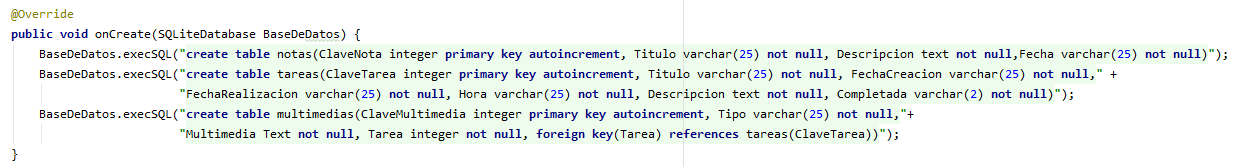


La clase EntidadM fue necesaria para enviar los valores a cada vista generada para el ListView de las multimedia capturadas.



Del mismo modo la clase AdaptadorMultimedias es la que permite asignar un comportamiento a los componentes de nuestra vista y también asignarle este adaptador al ListView.

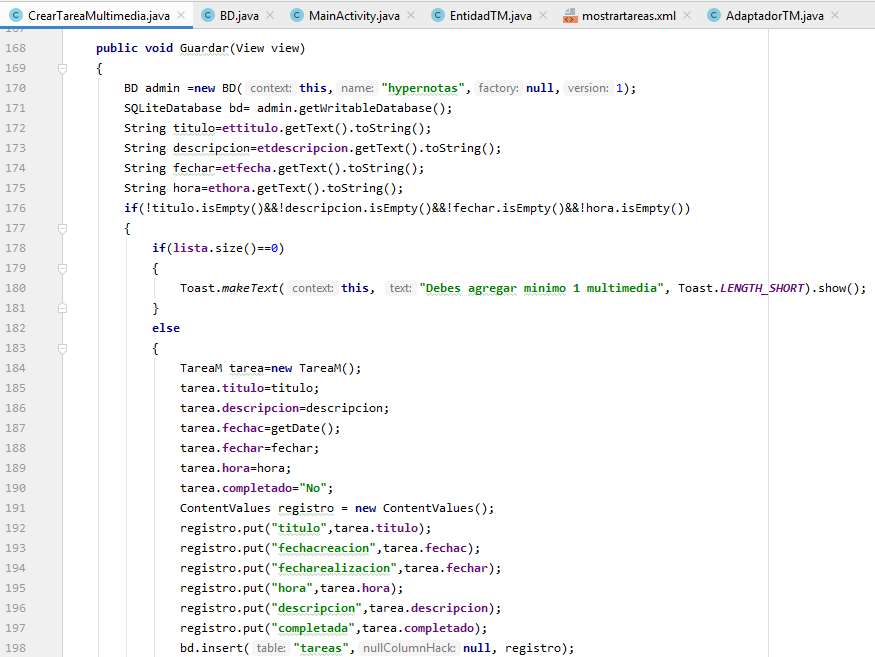


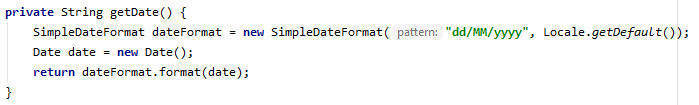
Es en el método getView donde se asigna la relación y el comportamiento a los componentes.

Se actualizo la base de datos con otras 2 tablas, una para las tareas multimedia en general y otra para las multimedia asignadas a su tarea correspondiente.



También se crearon las clases Multimedia y TareaM con sus respectivos atributos, para acceder como objetos de dicho tipo y almacenarlos en la BD.

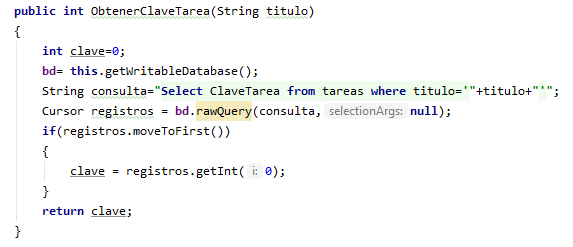


El método Guardar que está dentro de la clase CrearTareaMultimedia es el que permite insertar los datos a la BD, primero valida que todos los datos estén llenos, después valida que la lista de multimedia tenga al menos un elemento. Una vez realizada la validación se prosigue con la asignación de valores al objeto de tipo TareaM, luego se añaden los valores a la tabla tareas.

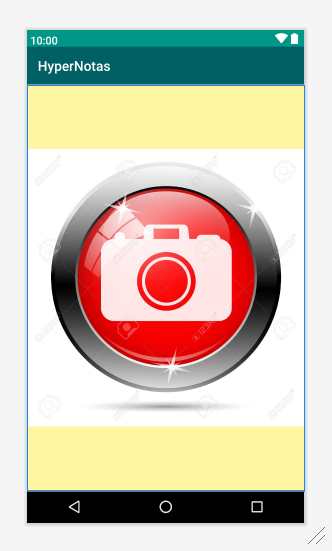
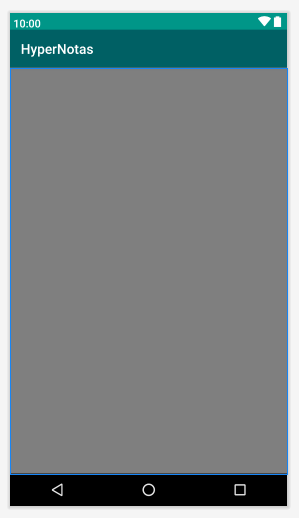
Para la fecha de creación de nuestra tarea se utilizó de nuevo el método getDate, el cual da la fecha actual en la que se realiza el llenado de los datos.



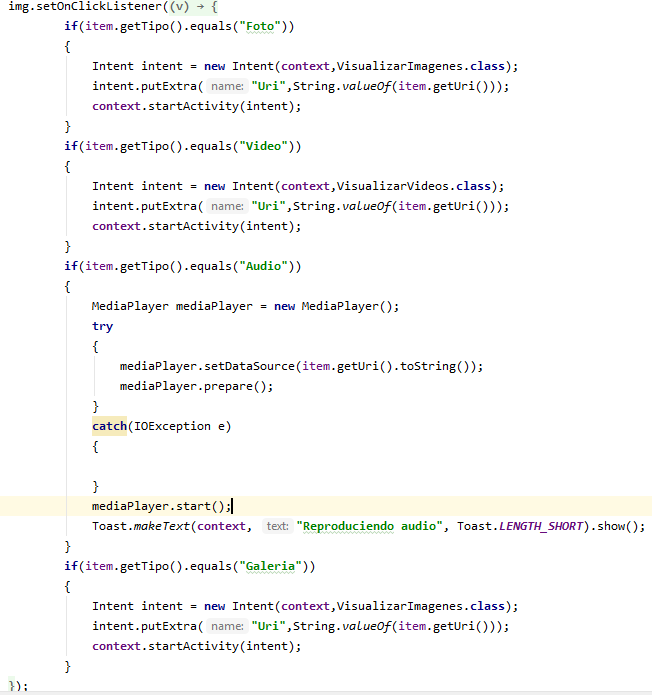
Continuando con el método Guardar, después de insertar la tarea se recorre la lista de uris para añadir una por una las multimedia que el usuario capturo. Luego se insertan los datos en la BD y se termina la actividad. Para la clave de la tarea se realizó una consulta sql.



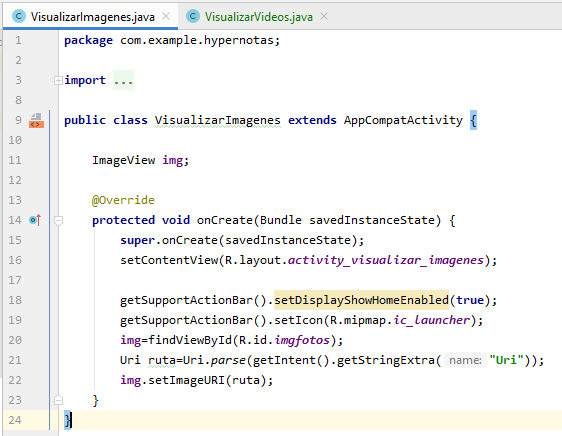
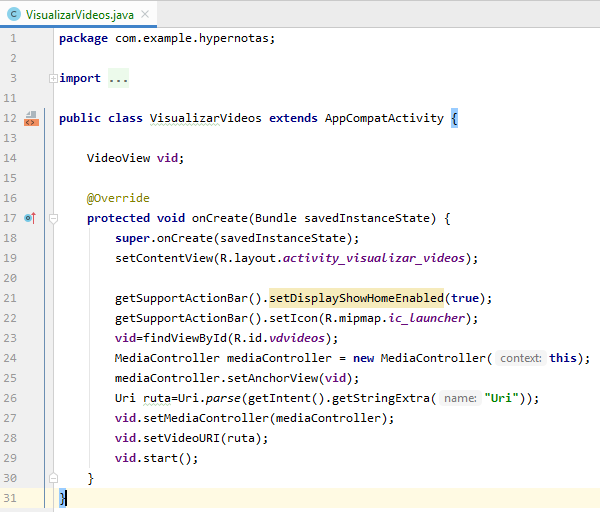
Este es el método que devuelve la clave de la tarea que se acaba de guardar en la BD, se necesita el título de la tarea para encontrarla en la base de datos.

**Ver multimedia creada**

Se crearon 2 actividades para que una muestre la foto o imagen de la galería, mientras que la otra es para ver el video.

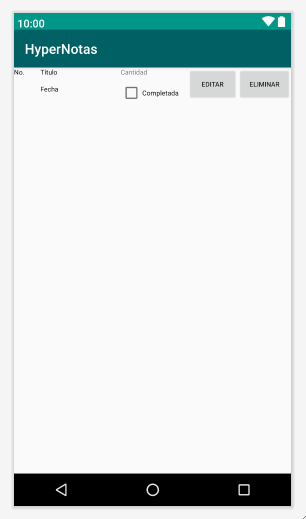


En el AdaptadorMultimedias se agregó comportamiento al componente ImageView para que al momento de que el usuario presione el icono de la multimedia lo envié a la actividad que corresponde con la Uri como parámetro. En caso de si es un audio se reproduce el archivo.

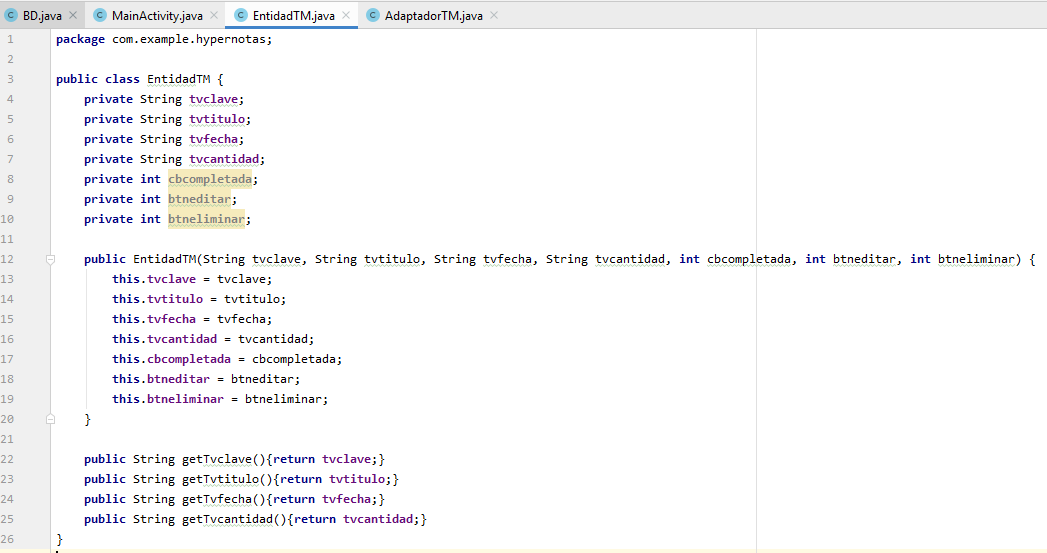


Y en sus clases correspondientes se obtiene la Uri, para después establecérsela a su componente y que se muestre la foto o video.

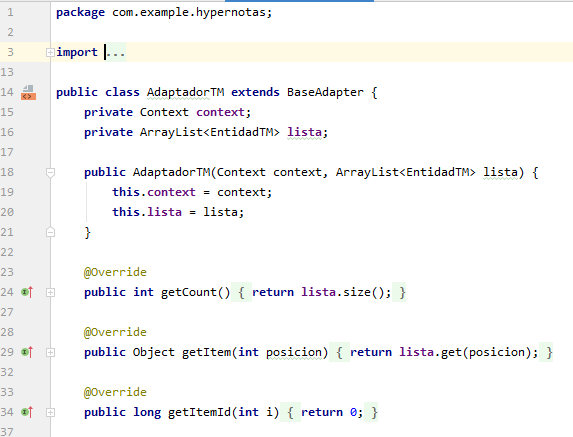
**Lista de tareas multimedia guardadas**

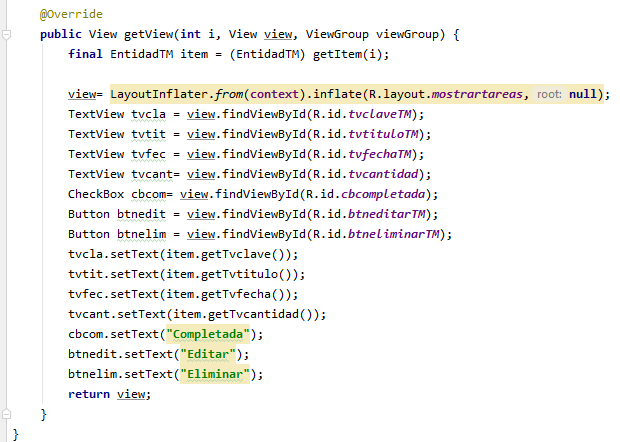


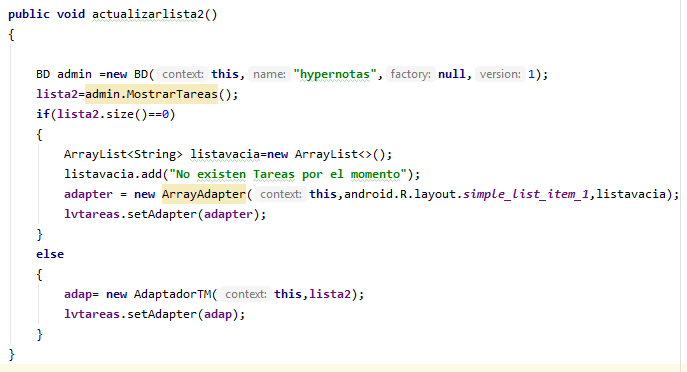
Se creó la vista para la lista de tareas multimedia, se compone de la clave, titulo, fecha de realización con la hora concatenada, cantidad de multimedia de esa tarea, el checkbox para saber si se ha completado y 2 botones.



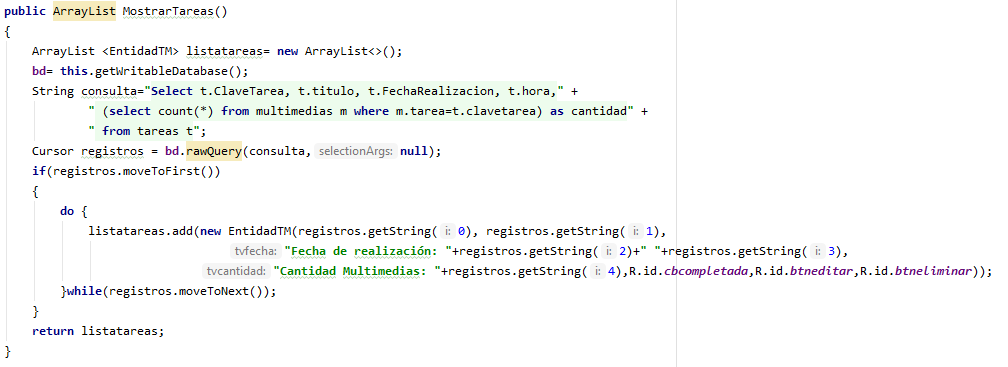
La clase EntidadTM para asignarle los valores a cada componente de la vista.





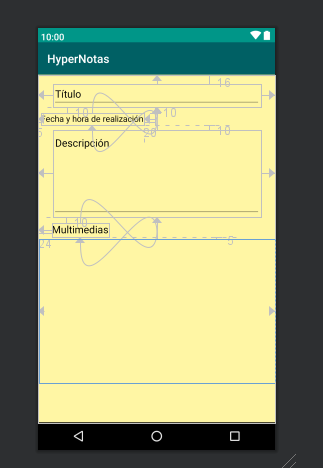
Y de nuevo otro adaptador para la vista de las tareas multimedia.

Ya en el MainActivity se creó el método actualizarlista2 que básicamente tiene la misma funcionalidad que el método actualizarlista, solo que esté llena la lista con tareas multimedia y hace uso de las clases referentes a estas.

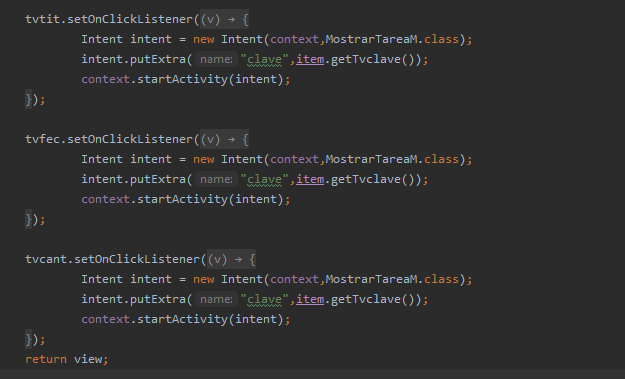


Es el método MostrarTareas el que permite llenar la lista con las tareas multimedia de la BD, se hace uso de una subconsulta para obtener la cantidad de multimedias de esta tarea.

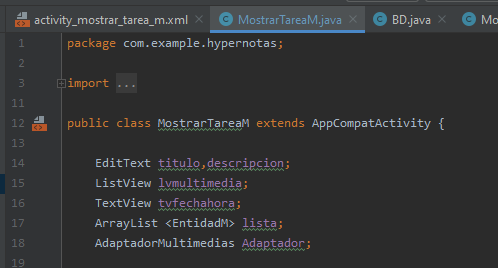
**Mostrar tarea multimedia**

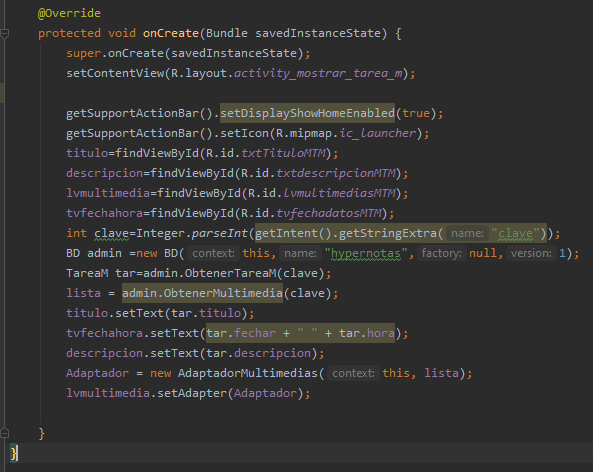


Se creó la actividad para poder visualizar el contenido de la tarea multimedia que se esté seleccionando en la aplicación, los componentes título y descripción están inhabilitados para que el usuario no pueda escribir, en esta actividad solo se puede visualizar.



Se añadió en la clase AdaptadorTM el comportamiento a varios TextView para que envíen la clave de la tarea multimedia seleccionada a otra actividad.



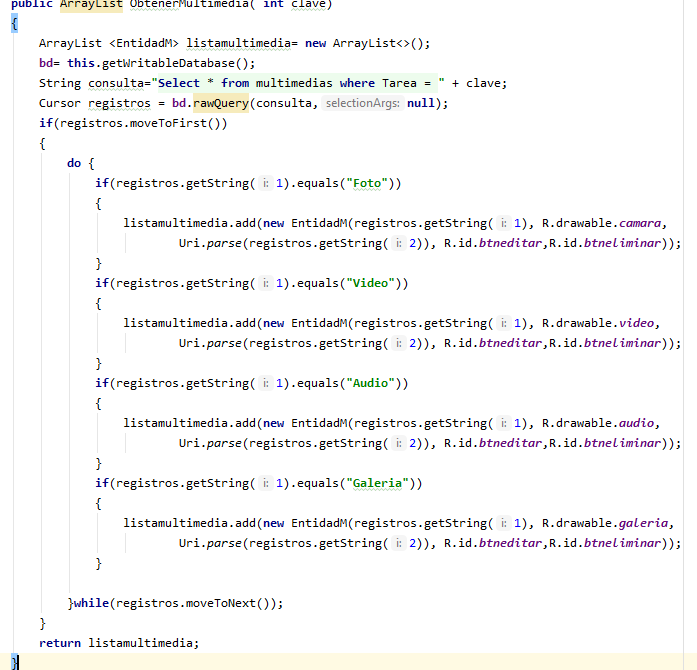


En la clase tarea multimedia se hizo referencia a los componentes de la interfaz gráfica, además

Se guarda la clave recibida desde el adaptador para posteriormente enviarla como parámetro en varios métodos los cuales obtienen todos los datos referentes a la tarea multimedia, por último, se asignan los valores a los componentes para que se muestren al iniciarse la actividad.

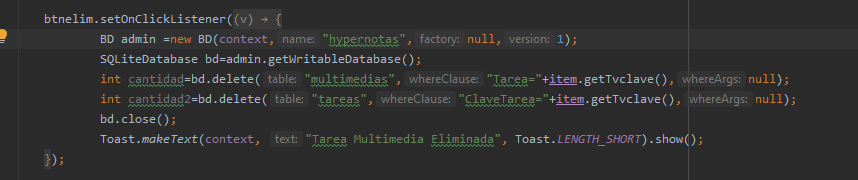


El método ObtenerTareaM recibe la clave de la tarea que se desea obtener todos los datos, estos se guardan en un objeto de tipo TareaM para posteriormente utilizarlo.



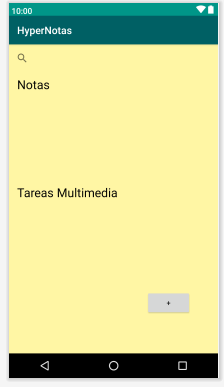
El método obtenerMultimedia también recibe la clave de la tarea de la cual se quiere obtener todas sus multimedia correspondientes. Se hicieron validaciones para que cambien algunos valores si es una foto, video, audio o imagen de la galería.

**Eliminar tarea multimedia**

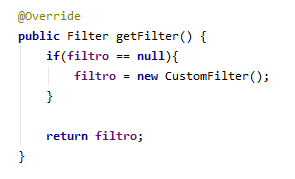


En el adaptadorTM se le agrego el comportamiento al botón de eliminar para que obtenga la clave de la tarea multimedia de la cual va a eliminar de la base de datos. Al fin muestra un mensaje notificando que la eliminación fue exitosa.

**Buscar notas y/o tareas multimedia**

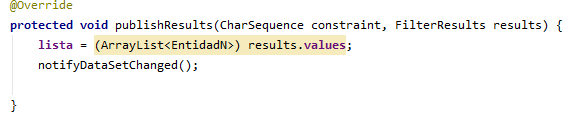


En la actividad principal se utiliza un SearchView para buscar las notas y tareas en base a letras o palabras ingresadas.

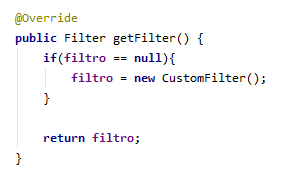
En la clase AdaptadorN se agrego el metodo getFilter, para que se realice la busqueda de notas en el ListView en base a las letras ingresadas en el SerachView.

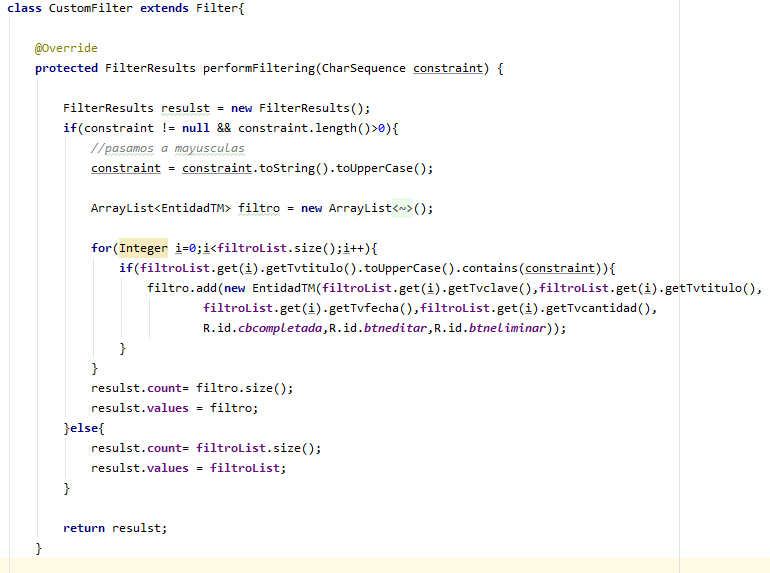


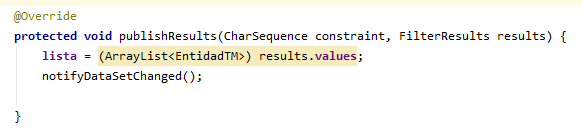
En la misma clase se crea otra llamada CustomFilter la cual se encarga de ir llenando una lista de tipo EntidadN con todas las coincidencias encontradas en el Listview de las notas.



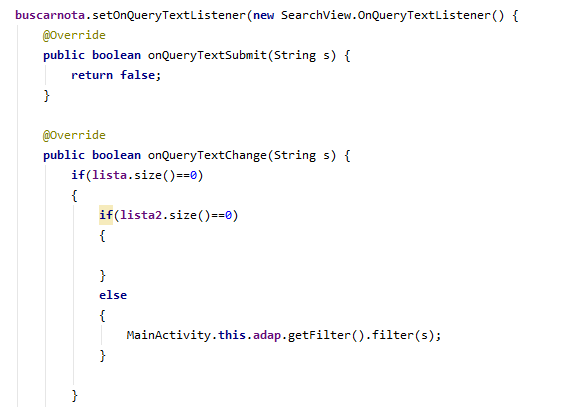
Por ultimo el metodo publishResults asigna los valores a la lista actual para que se muestren solo las notas que coincidan en el titulo con lo ingresado al SerachView.





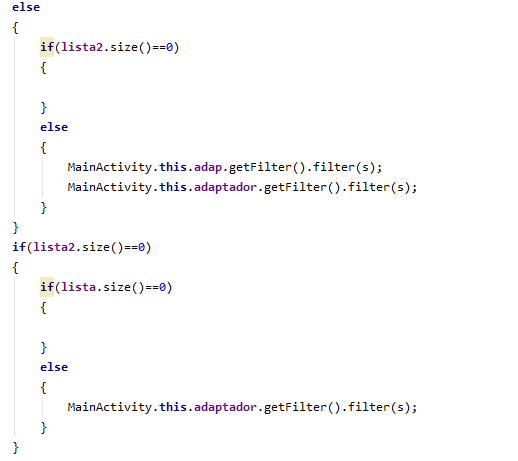


Se utilizaron los mismos metodos y clase en el AdaptadorTM para realizar el mismo porcedimiento de filtrado, pero esta vez va dirijido a las EntidadesTM, mostrando asi una lista de tareas multimedia.



En la clase MainActivity se le agrego el comportamiento al SearchView de modo que al nosotros ingresar caracteres este efectue un filtro de busqueda sobre los ListView de notas y tareas multimedia.

Se realizaron varias validaciones para evitar errores en la aplicación.

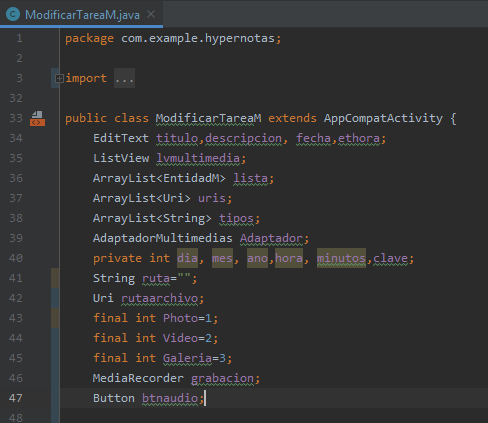


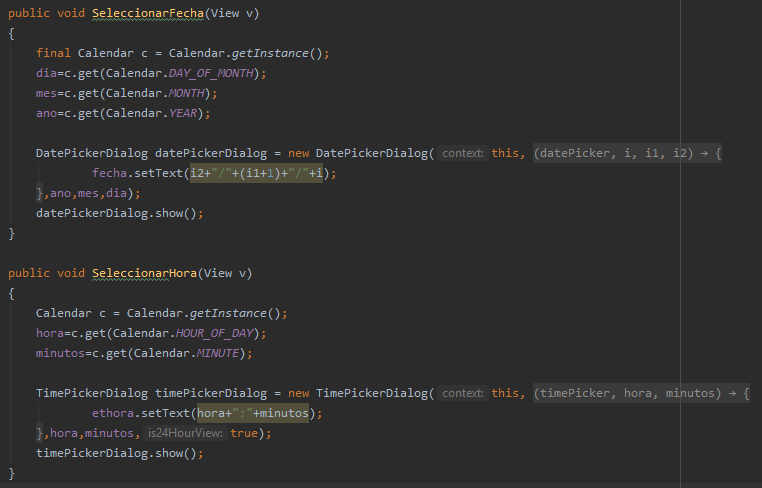
Las validaciones generadas son para evitar fallos en la app ya que si no se cuenta con notas o tareas multimedia en el ListView el filtrado de busqueda produce errores, esto se soluciono indicando que solamente realice la filtracion si hay datos en los 2 ListView o por lo menos en 1 a la vez.

**Modificar tarea multimedia**

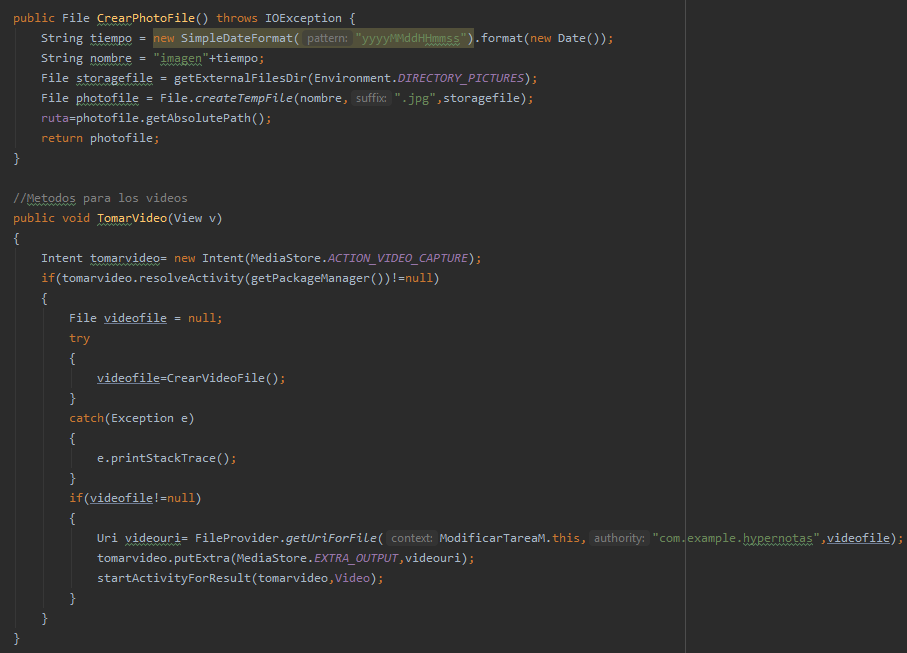


La interfaz para modificar una tarea multimedia es exactamente la misma, por lo que los métodos también son los mismos. Solamente hay cambios en el método para guardar y una que otra variable.

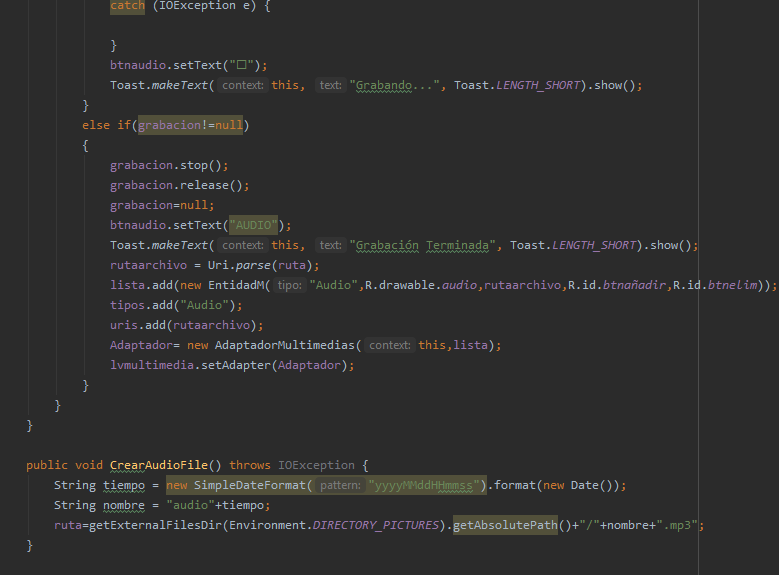




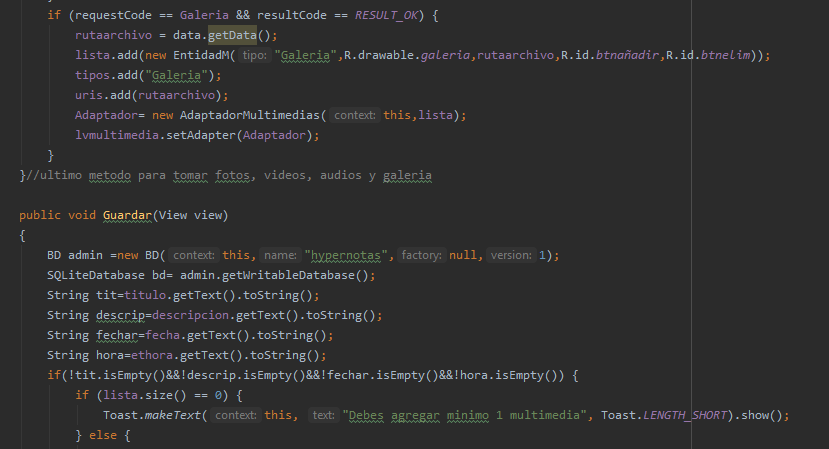




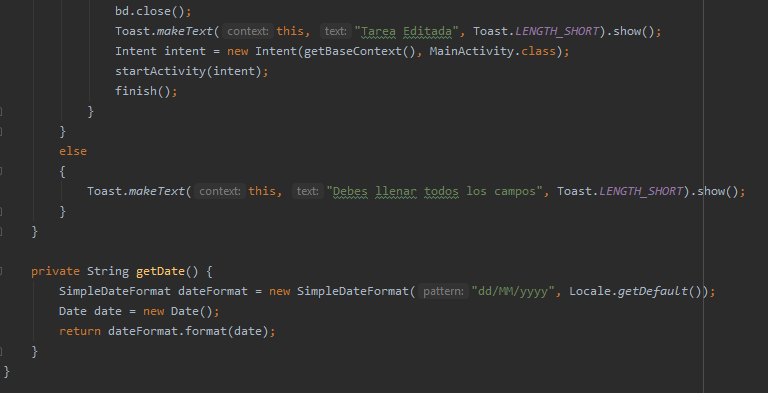








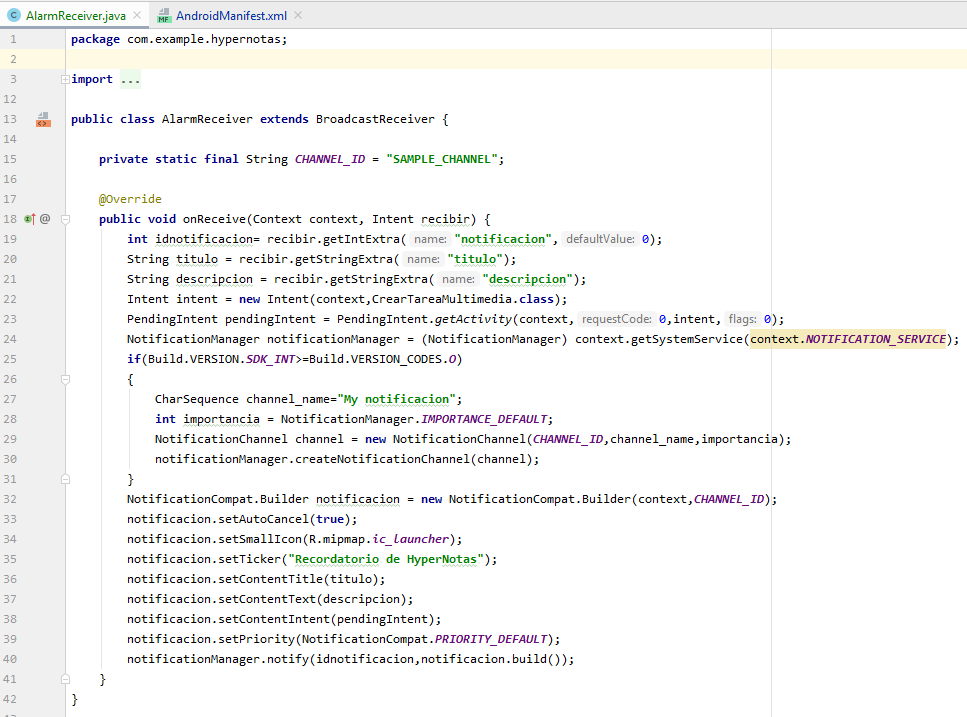




El método Guardar es el único que tiene cambios ya que en lugar de insertar hacemos un update a la tabla tareas, mientras que la acción a la tabla de multimedias sigue siendo la misma debido a que se pueden añadir aún más multimedias si así lo desea el usuario.

**Notificaciones en la app**

El método Recordatorio se ejecuta cada vez que se presiona el botón de Recordatorio, para crear las notificaciones se necesita tener llenos los campos título, descripción, fecha y hora. Estos datos se envían a otra clase llamada AlarmReceiver la cual se encarga de darle formato a la notificación. En este método hacemos referencia y uso de los servicios de alarma del dispositivo móvil, además aquí es donde se pasa el parámetro de la hora a la que se quiere recibir la notificación.

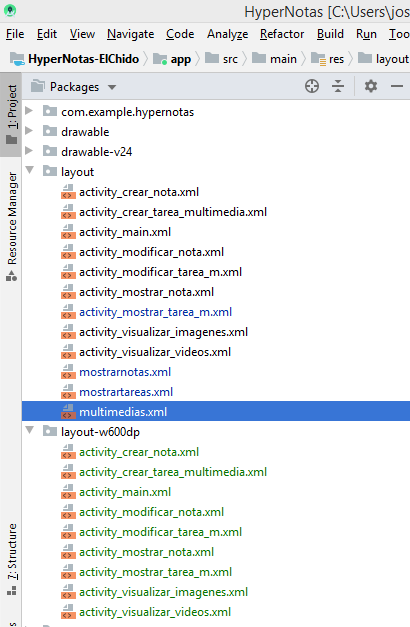


La clase AlarmReceiver extiende de BroadcastReceiver para poder obtener los parámetros enviados desde la otra clase. En esta clase se le da el formato a la notificación como: título, icono, mensaje, etc. Y se construye la notificación.



También en el archivo manifiesto se añadió esta línea de código para poder usar la clase AlarReceiver.

**Soporte para diferentes pantallas**

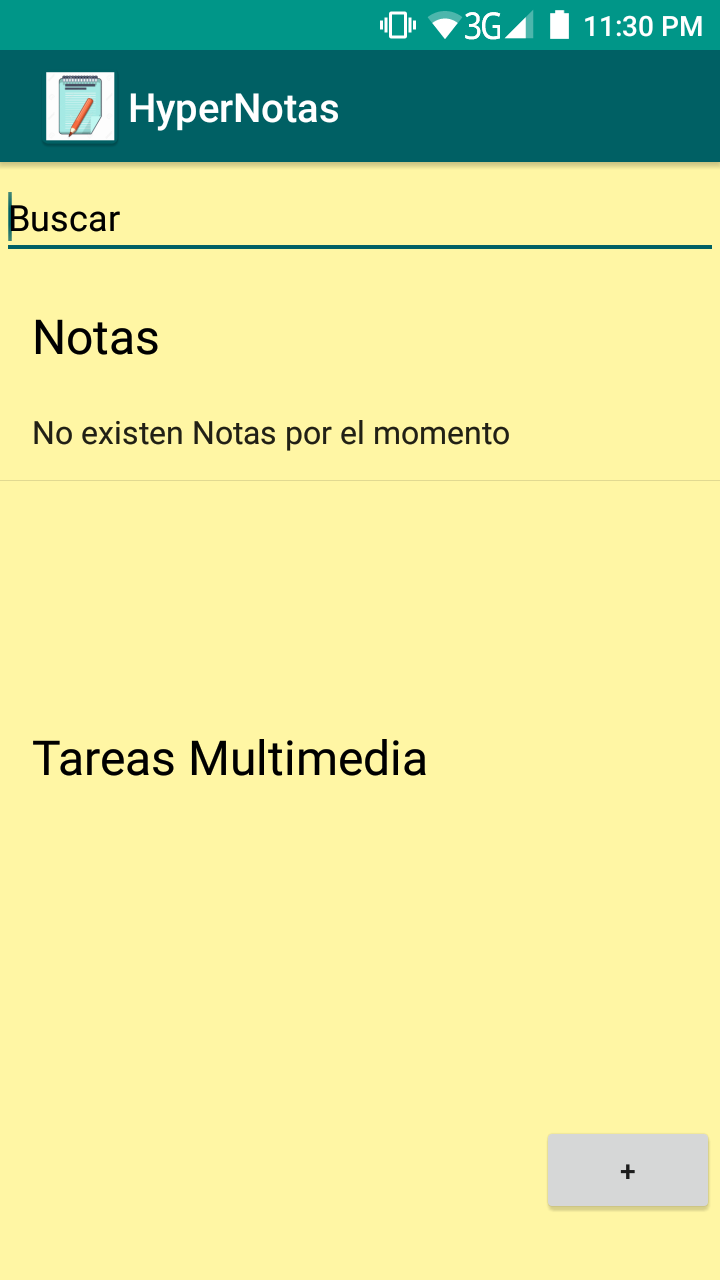


Se realizó el ajuste de los componentes de cada layout para un ancho de pantalla de 600dp. Todas las interfaces fueron diseñadas tanto para la pantalla de un celular como para la de una tableta. Los archivos mostrarnotas, mostrartareas y multimedias.xlm no fueron adaptados debido a que estos archivos no son interfaces como tal, sino que son los adaptadores de los distintos ListViews utilizados. Las interfaces finales se podrán apreciar en la parte de Resultados en este documento.

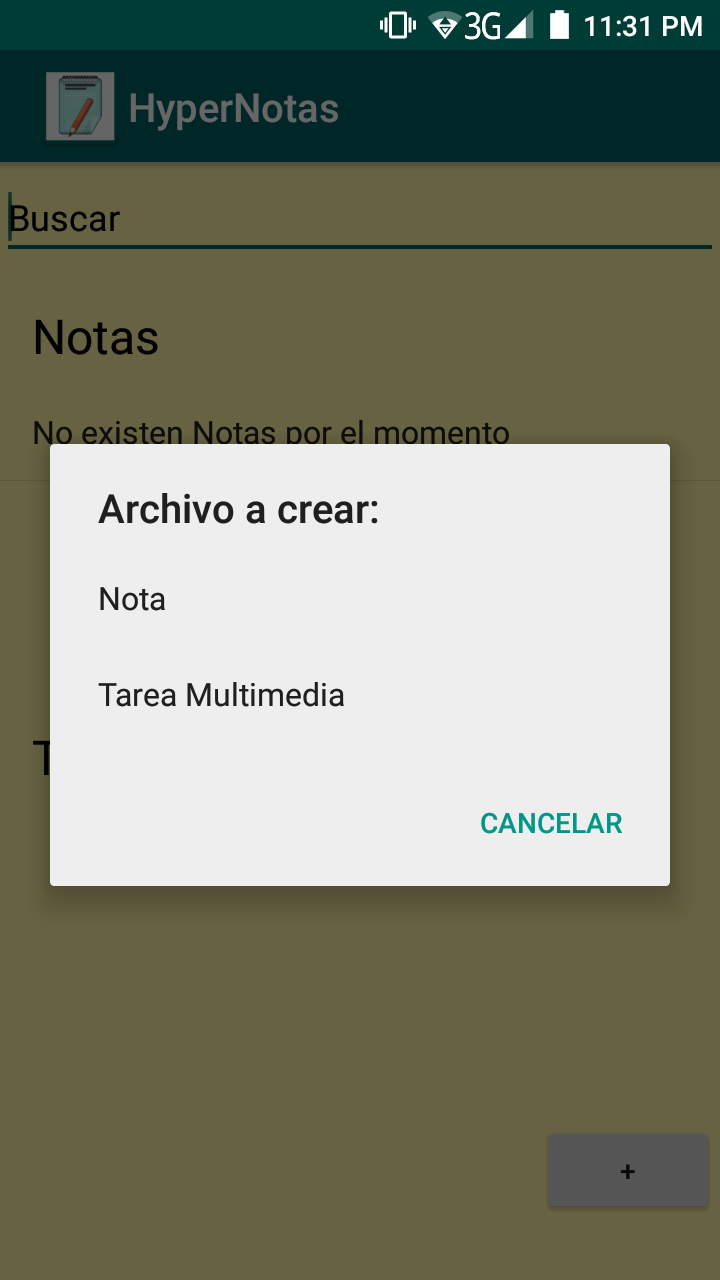
**Resultado**

(Esta parte aún no está actualizada)

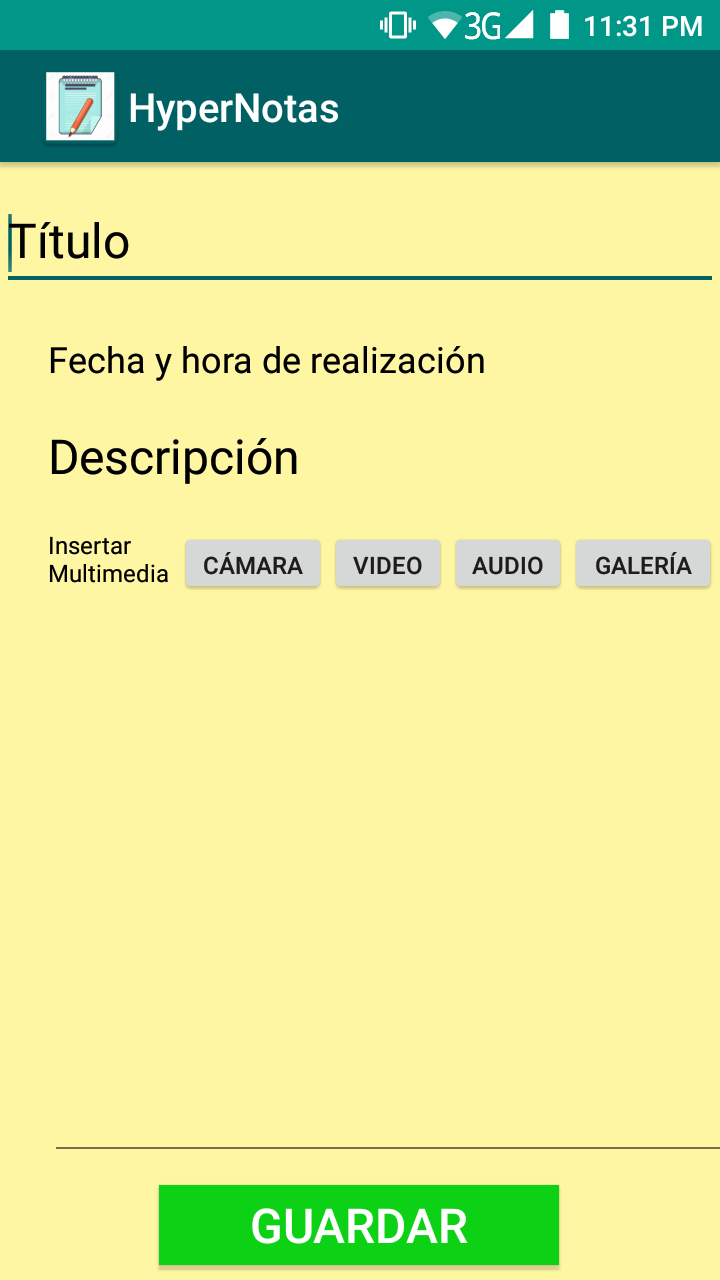
Estos fueron los resultados obtenidos de ejecutar la aplicación en el celular.



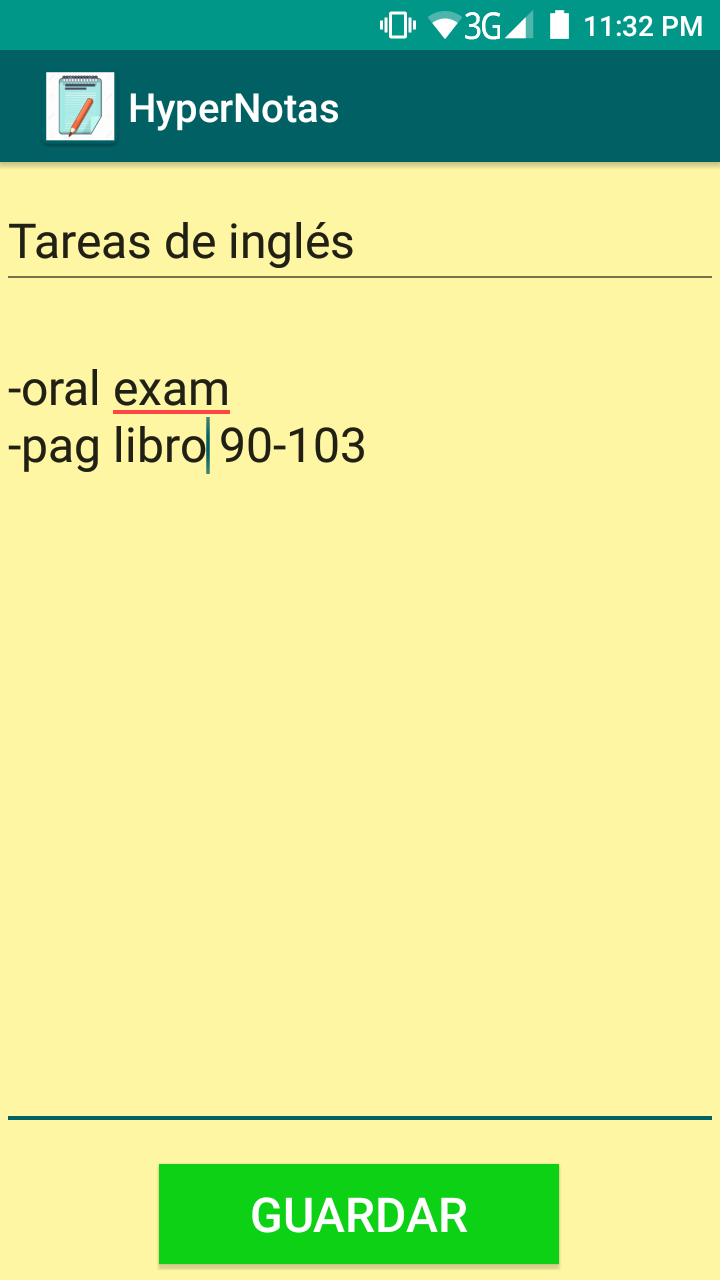
La interfaz principal indicando que no hay notas registradas.



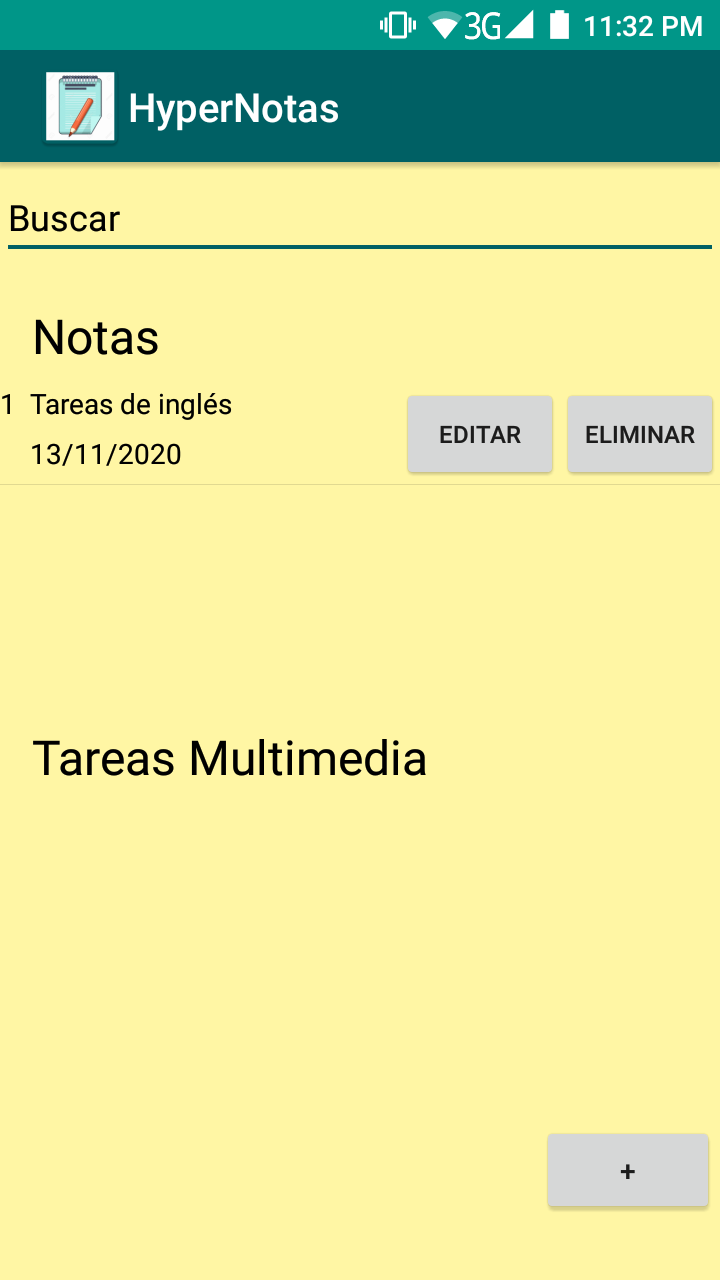
El cuadro de opciones preguntando que archivo se desea crear.



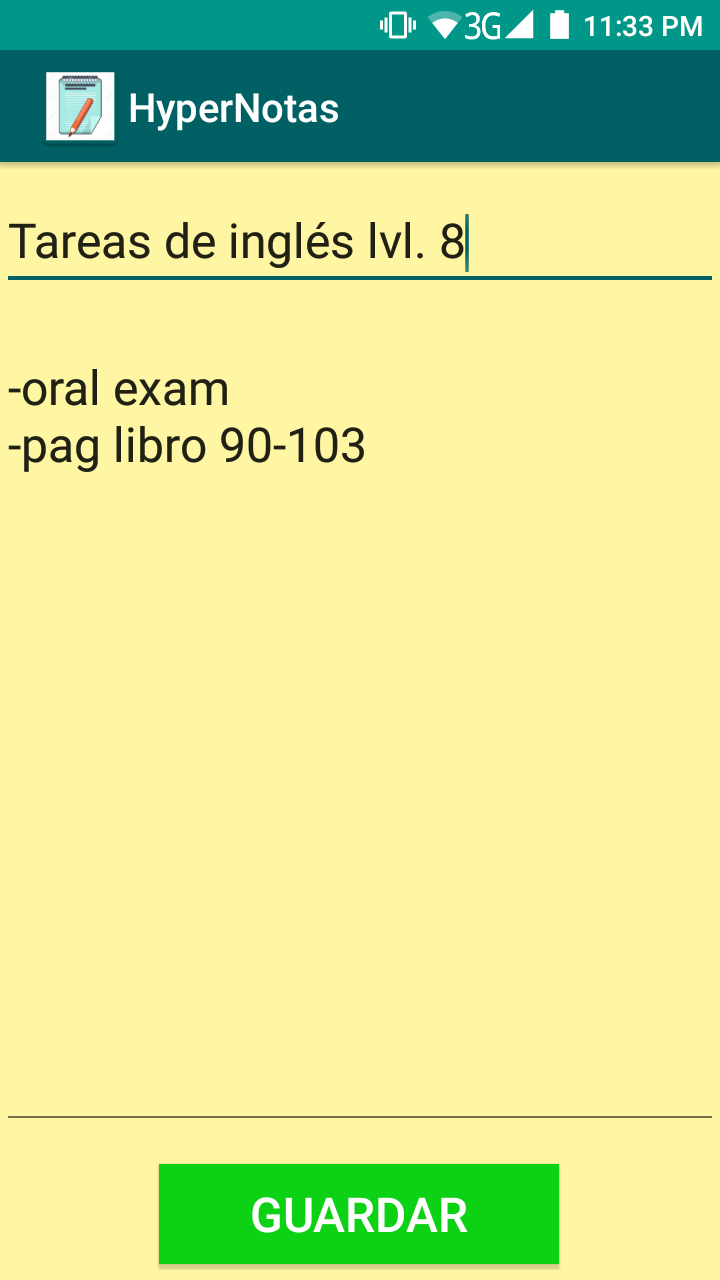
La actividad que nos permite crear tareas multimedia, por el momento solo es visual, no hay nada programado.



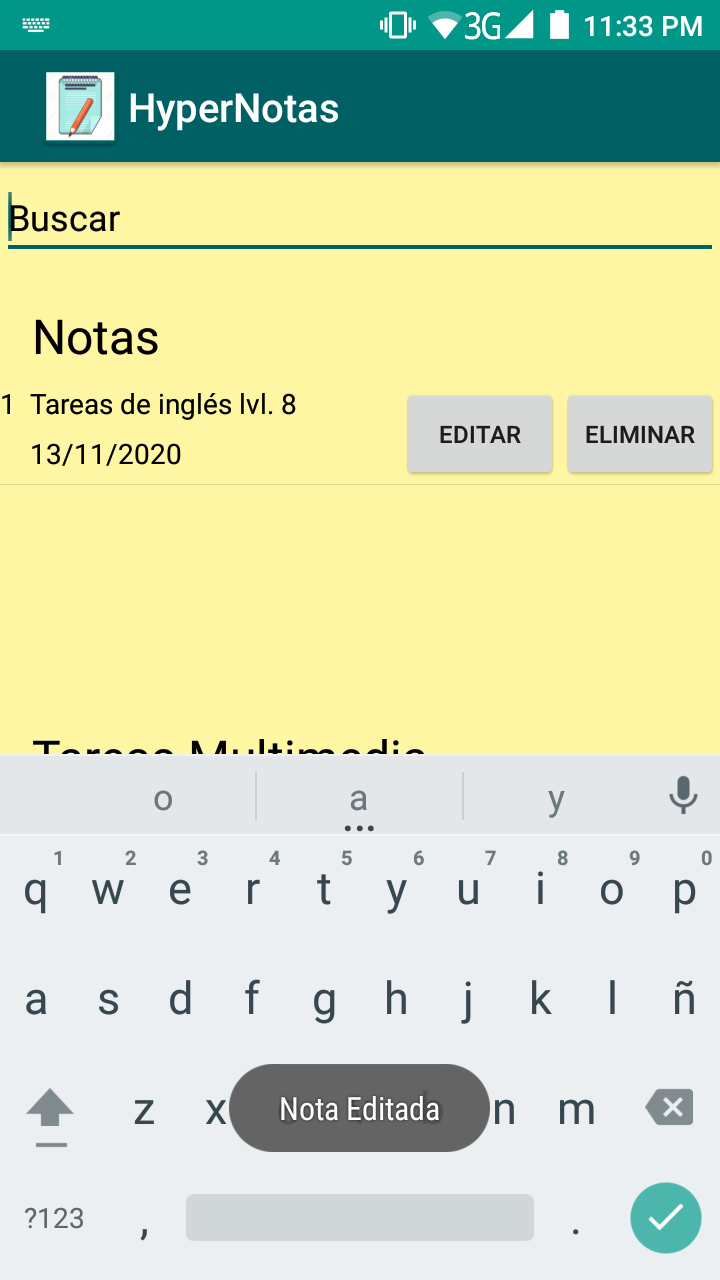
En cambio la actividad para crear notas si está programada y nos permite guardar la nota.



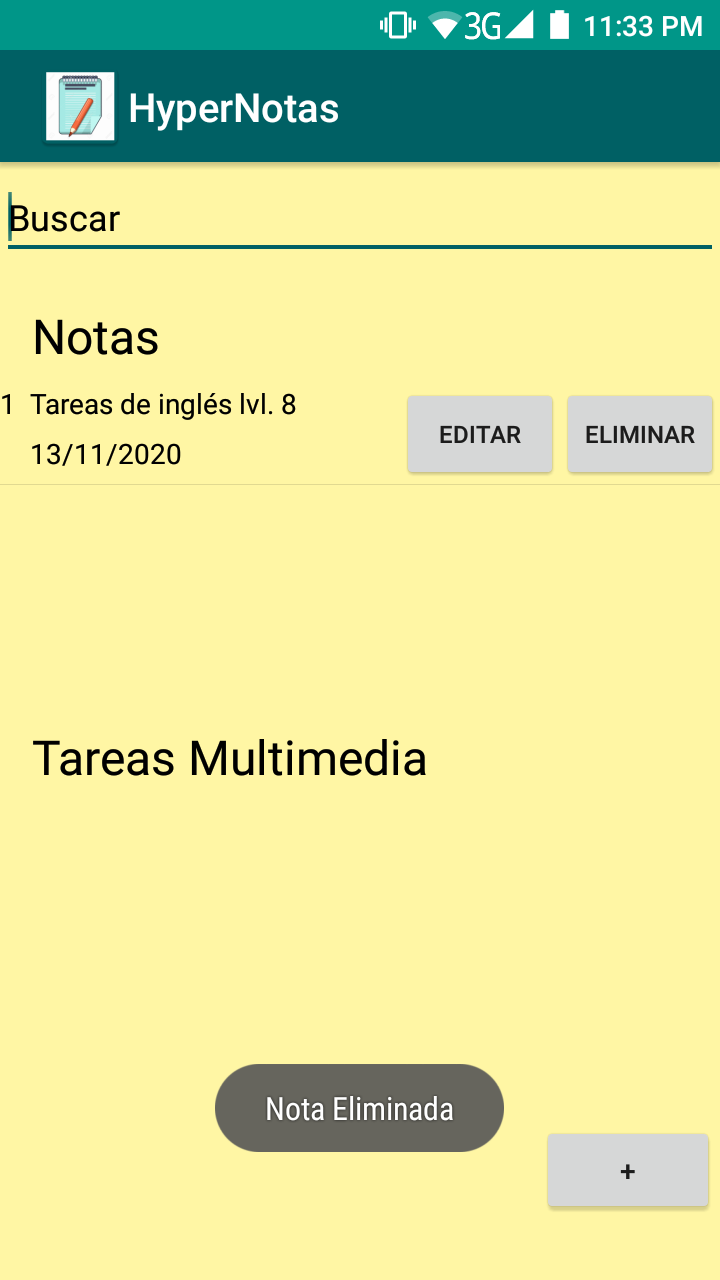
Se muestra la nota guardada en la interfaz principal.



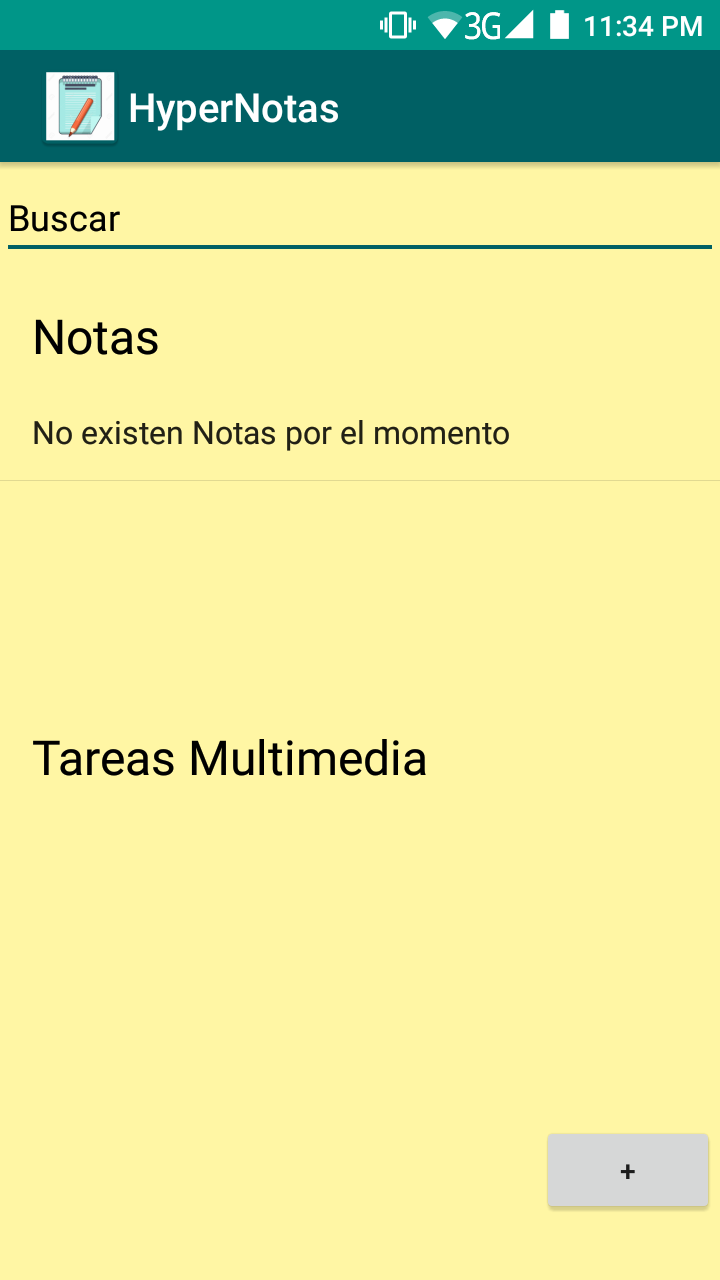
Se modifica el título de la nota y después guarda.



Notifica que se editó de forma correcta y ahora la nota aparece con los cambios efectuados en la actividad principal.



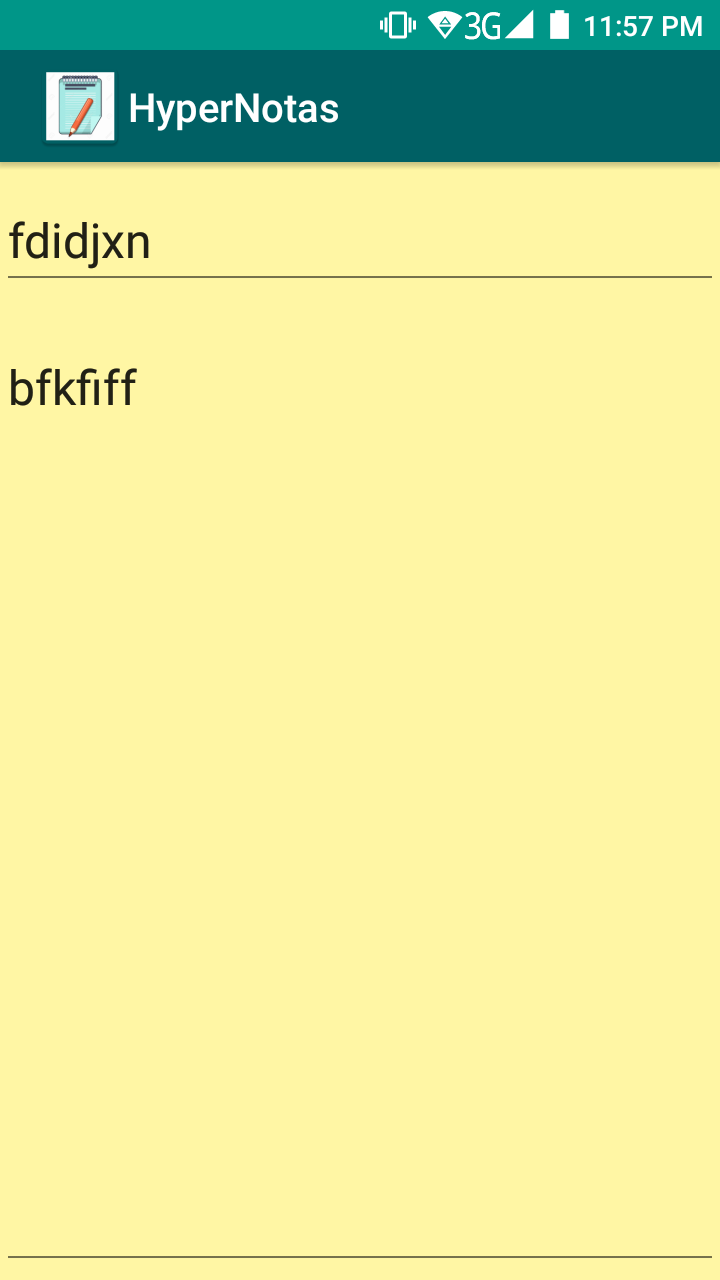
Se elimina la nota de manera correcta, pero para visualizarse los cambios es necesario volver a cargar la actividad. Este es un problema que esperamos poder resolver más adelante.



Como se observa la actividad se volvió a cargar y ya nos indica que no hay notas.



Se agregaron más notas para que se aprecie el funcionamiento del ListView. Se visualizan las notas empezando por las de fecha y clave mas reciente.



Por ultimo si se da clic sobre el título o la fecha nos envía a la actividad que permite ver el contenido de la nota. Solo visualizar, no es posible editar.

**Evaluación**

**Conclusiones**

En lo que llevamos de avance del proyecto final pudimos hacer uso de algunos de los conocimientos comprendidos a lo largo del curso, hay algunas otras cosas en las que tuvimos problemas, pero logramos resolver los conflictos realizando búsquedas en internet. Esperamos que en lo que falta del semestre podamos cumplir con la aplicación de manera correcta y si no se puede por falta de conocimientos o tiempo, entregaremos la aplicación en el mejor estado posible.

**Bibliografía**