**Juan Camilo Restrepo Velez 000373886**

Tabla de contenido

[1) EU Estudiante 2](#_Toc40629566)

[1.1) MainActivity 2](#_Toc40629567)

[1.1.1) Iniciar el estado de la actividad 4](#_Toc40629568)

[1.2) ActividadActivity 6](#_Toc40629569)

[1.2.1) Sincronizar SQLite local con nodo MariaDB 9](#_Toc40629570)

[1.2.2) Actualizar el estado de la actividad 10](#_Toc40629571)

[1.3) Modelo de sincronización 16](#_Toc40629572)

[Referencias 18](#_Toc40629573)

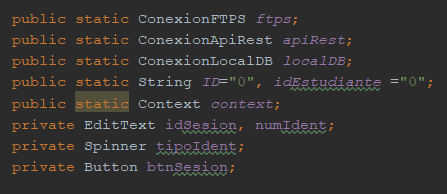
# 1) EU Estudiante

Se hará una breve recopilación de los llamados a las clases auxiliares para manejar la conexión con el servidor MariaDB y con el servidor FTP, así como las clases necesarias para el manejo del front y las nuevas implementaciones que abarca el manejo de la base de datos SQLite interna y todo el código para iniciar, actualizar y sincronizar los estados de las actividades.

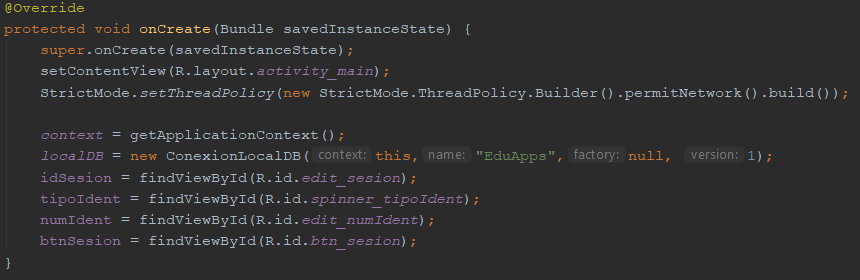
## 1.1) MainActivity

Esta actividad se encarga de iniciar el estado de la actividad, si es primera vez que el estudiante inicia sesión para dichas actividades, insertando los datos del nodo a la base de datos local o actualizando las tablas locales con la base de datos que está en el nodo (sin modificar la tabla que tiene el estado de las actividades), esta última opción si ya se ha iniciado sesión en el dispositivo con anterioridad. Además, si el dispositivo ya tiene información de la sesión que se desea iniciar y no hay conexión con el nodo, se permite entrar en estado *offline* para que el estudiante pueda visualizar las actividades y poder cambiar el estado de cada una de ser necesario.

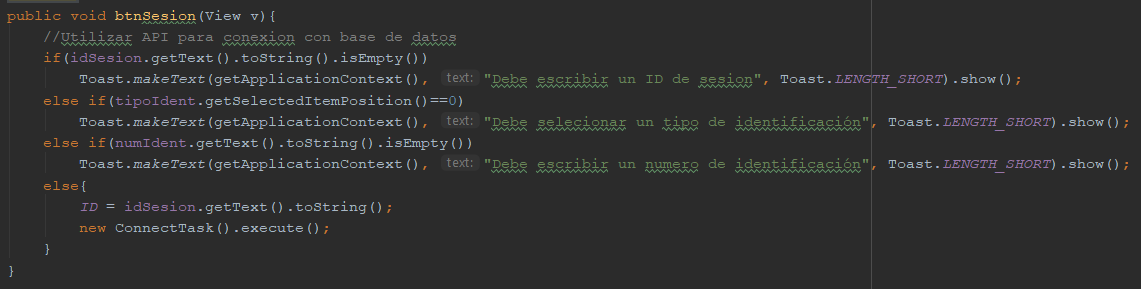
Primero, se establecen las variables para los objetos de las vistas y las instancias de conexión global como estáticas para no estar creando múltiples objetos con el mismo fin:



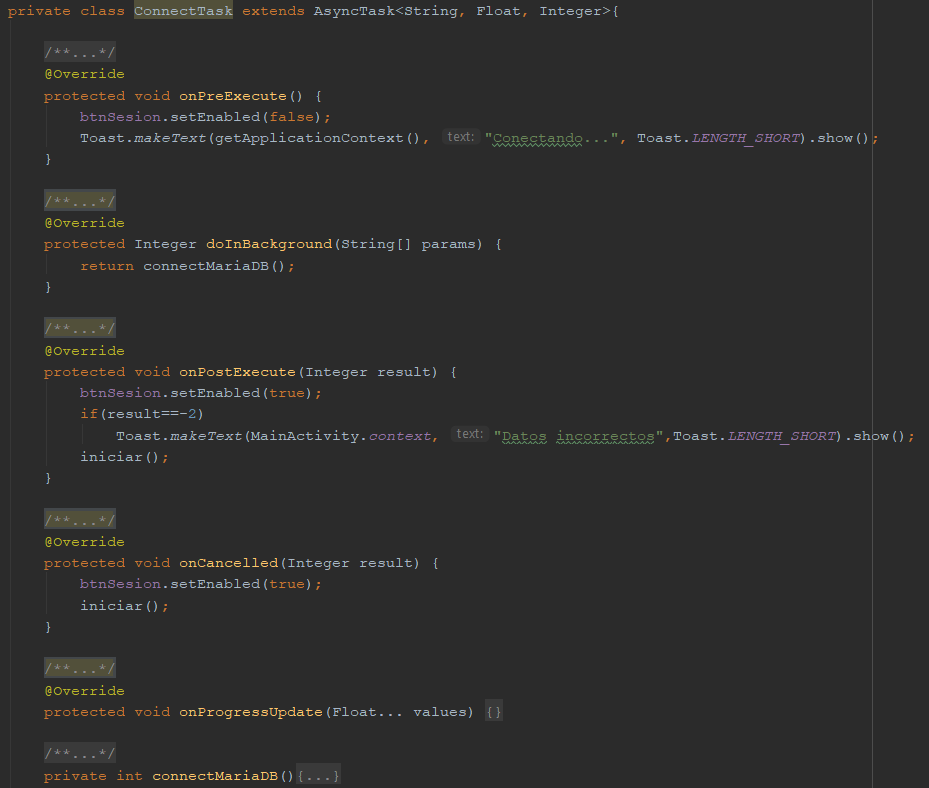
Luego, en el onCreate se hacen las asignaciones necesarias y se habilita la regla de descargar la información desde el hilo principal con la línea de *StrictMode*:



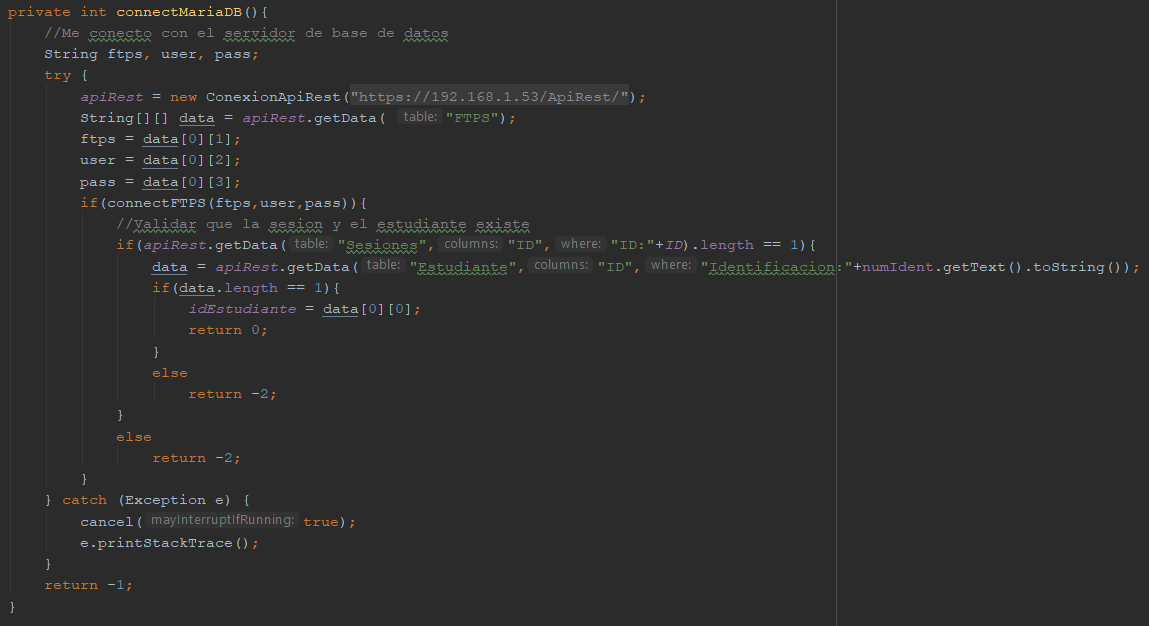
Cuando se dé clic en el botón *Iniciar sesión* se obtiene la información de los objetos haciendo una primera validación de que estos no estén nulos:



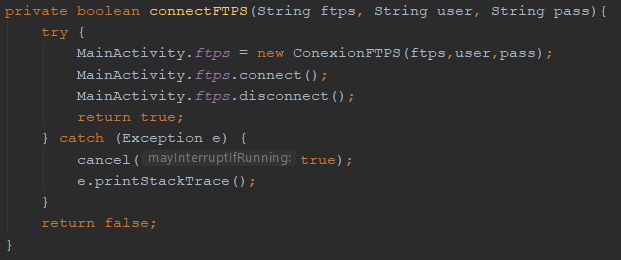
Luego se ejecuta un hilo para probar la conexión con el API REST, que puede retornar 3 valores; 0 (conexión exitosa), -1 (conexión fallida), -2 (Datos erróneos). Además, antes de que se ejecute el hilo se deshabilita el botón de iniciar para evitar que el usuario haga múltiples peticiones:



En el método *connectMariaDB* se realiza la prueba con el API REST, que si es exitosa se establece la variable *online* en *true* y se obtienen los datos del servidor FTP, la sesión y el estudiante para validar que los datos ingresados sean correctos:

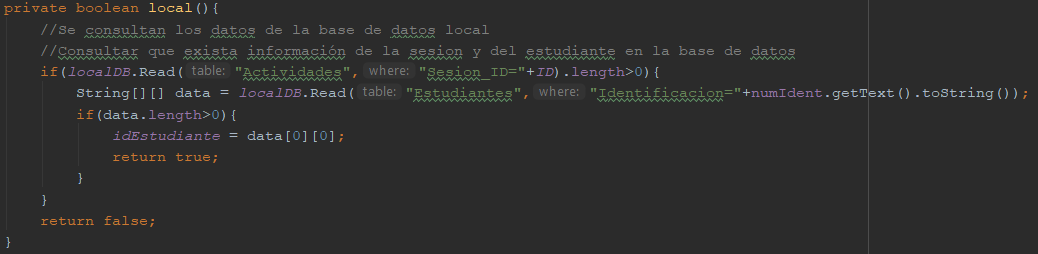


Luego, en el método *connectFTPS*, se valida la conexión con el servidor FTPS según los datos del nodo:



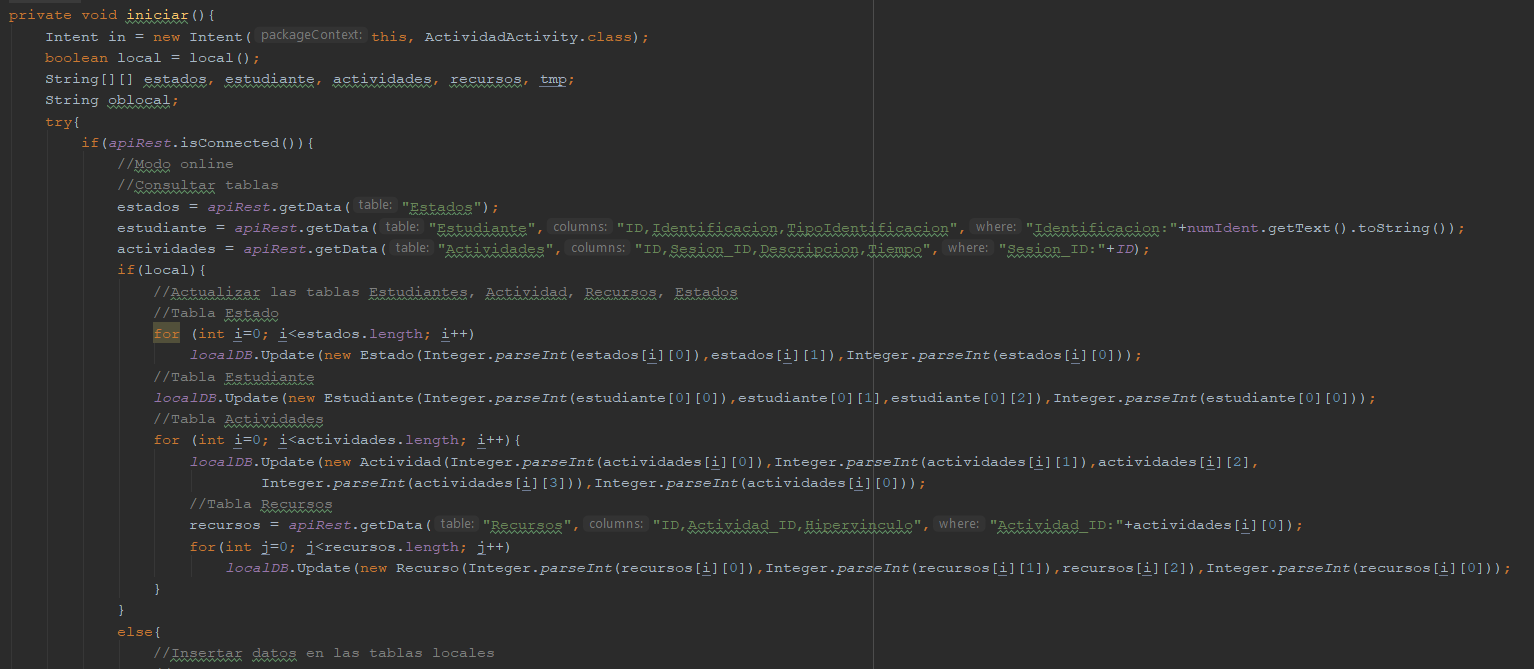
### 1.1.1) Iniciar el estado de la actividad

Posterior a la ejecución del hilo, se ejecuta el método *iniciar* que llama al método *local* para saber que hay información de la sesión en la base de datos local:

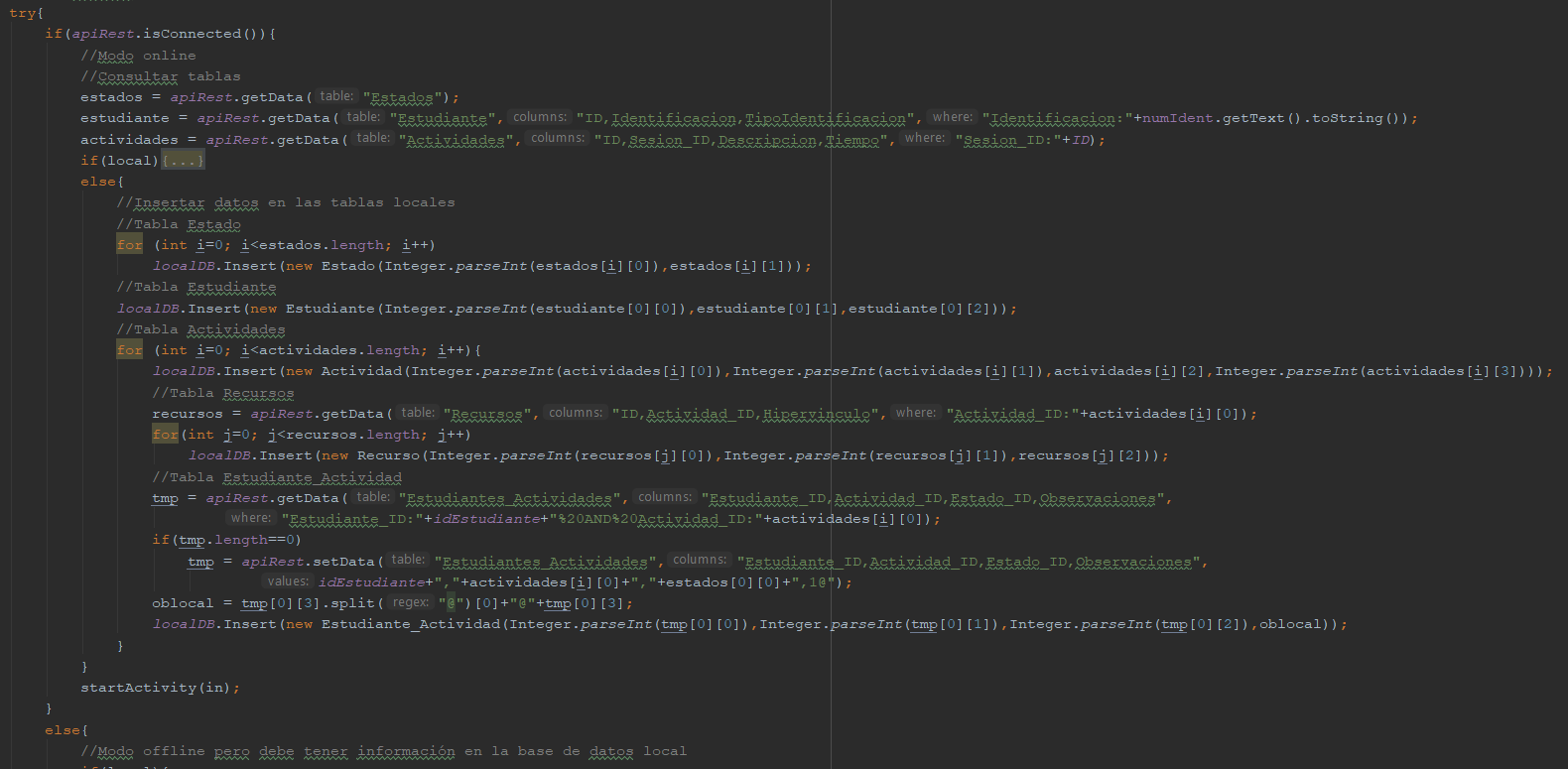


Luego de esto puede pasar lo siguiente:

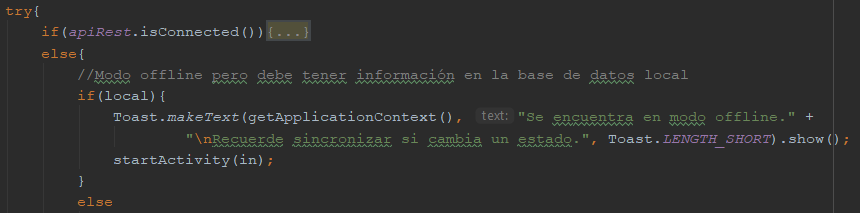
* Si se está en modo *online* y hay información en la base de datos local se actualizan las tablas locales excepto la tabla que contiene los estados de las actividades ya que posteriormente (en la *ActividadActivity* se deben sincronizar), y luego se inicia la actividad siguiente:



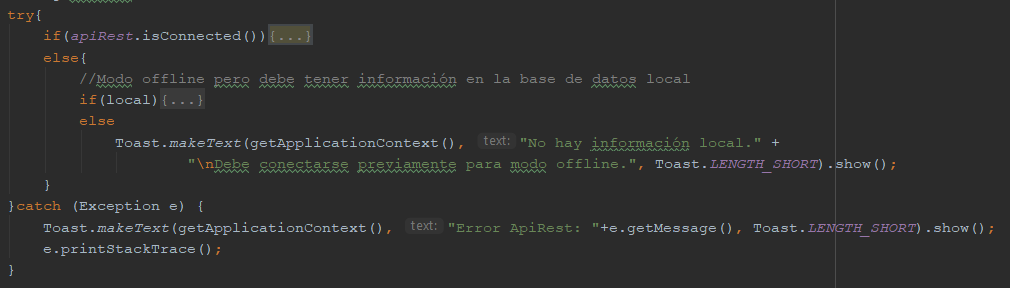
* Si se está en modo *online* pero sin datos en la base de datos local, se insertan en todas las tablas de la base de datos SQLite local y se consulta si es la primera vez que se va a ingresar a la sesión consultando los datos de los estados, **si no existe información se insertan los datos y se ponen las actividades en estado *Iniciada***, y luego se inicia la *ActividadActivity*:



* Si se está en modo *offline* y con datos locales se inicia la *ActividadActivity* que se encarga de cargar la información para visualizar las actividades:

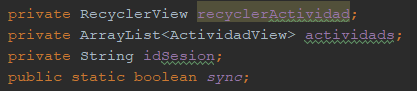


* Si se está en modo *offline* pero sin datos en la base de datos local, no se puede iniciar debido a que debe haber un ingreso previó *online* para poder tener información en la base de datos local:

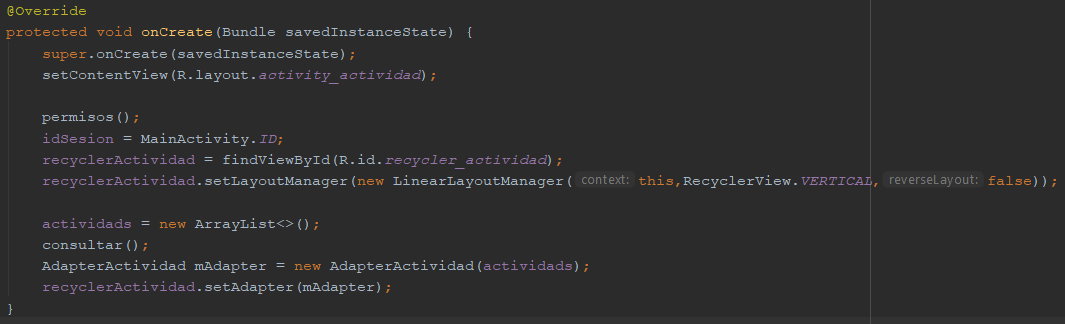


## 1.2) ActividadActivity

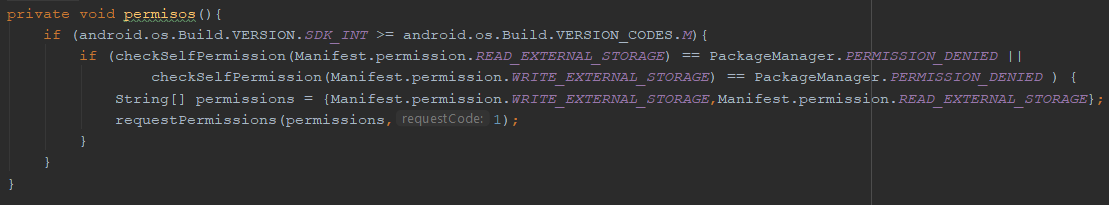
Se establecen las variables que se utilizaran, así como una lista de *ActividadView* que es una clase que nos ayuda al manejo del Front junto con la clase *AdapterActividad*, para establecer todas las actividades de una sesión, además de una variable boolean para saber si aún no se ha sincronizado el dispositivo con el nodo:



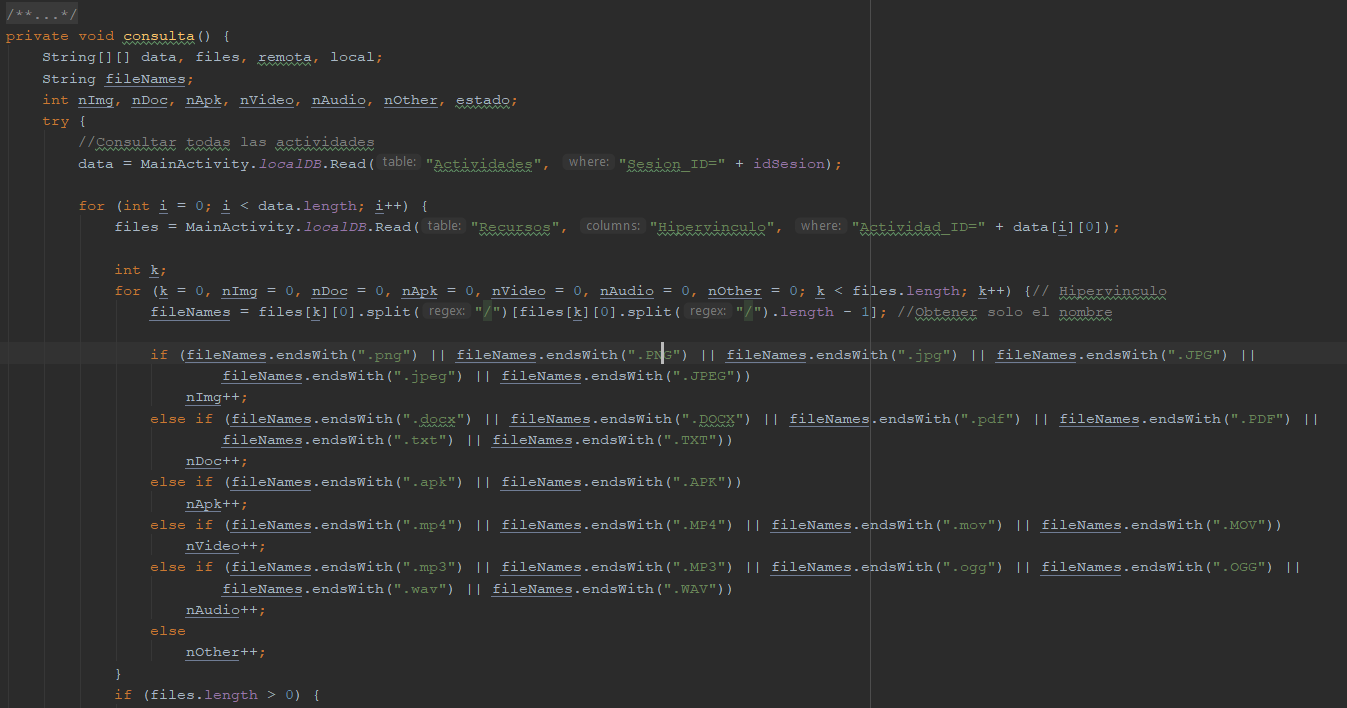
Luego en el onCreate, se piden los permisos de almacenamiento, así como obtener y establecer el objeto de la vista (RecyclerView) y luego consultar las actividades de la sesión para ponerlas en la lista:



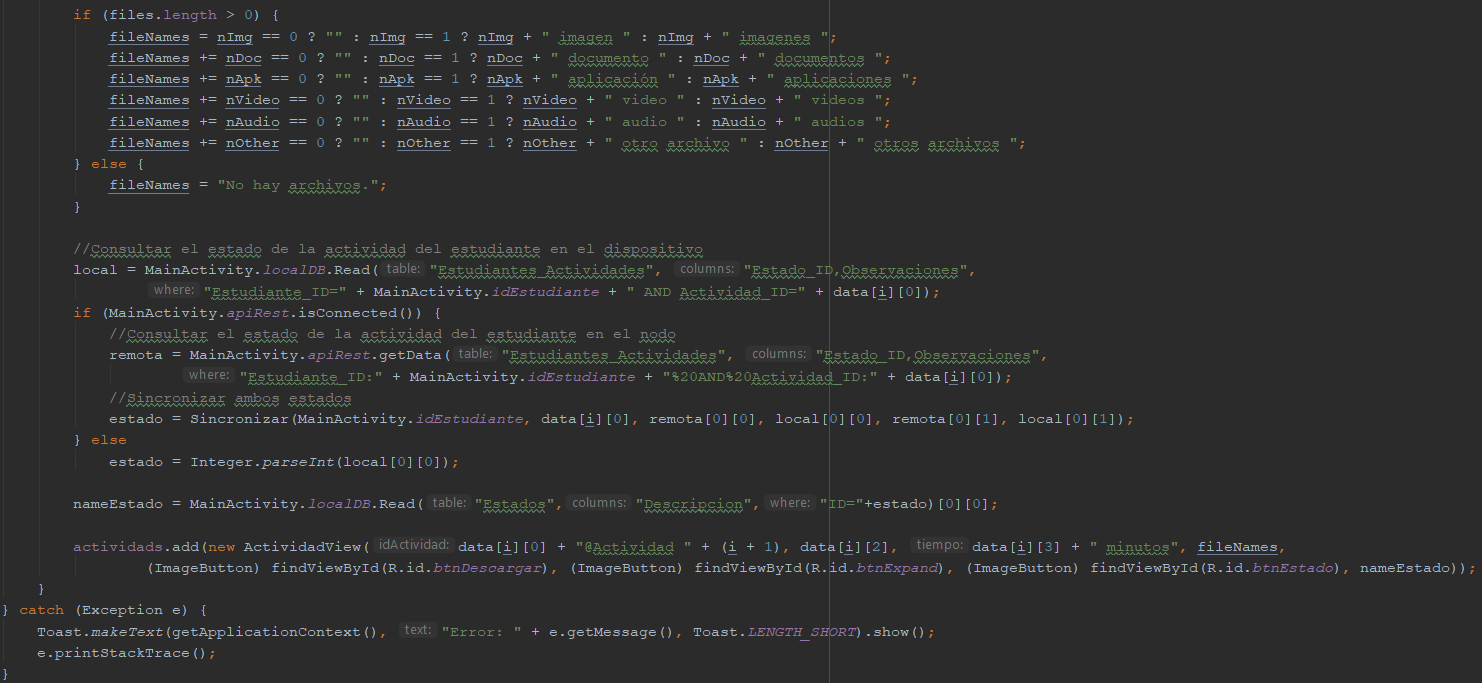
El método de los permisos es el mismo que se ha utilizado en los sprint pasados:



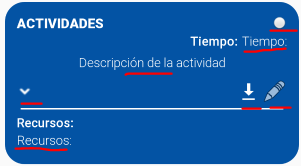
Y el de consulta cambia realizando todas las consultas sobre la base de datos local, ya que al iniciar sesión se valida que se cuente con ella para poder estar en modo *offline*, además con esto ahorramos recursos y procesamiento al no tener que descargar la información en una página porque ya se tiene localmente lo cual permite un mejor rendimiento de la app:



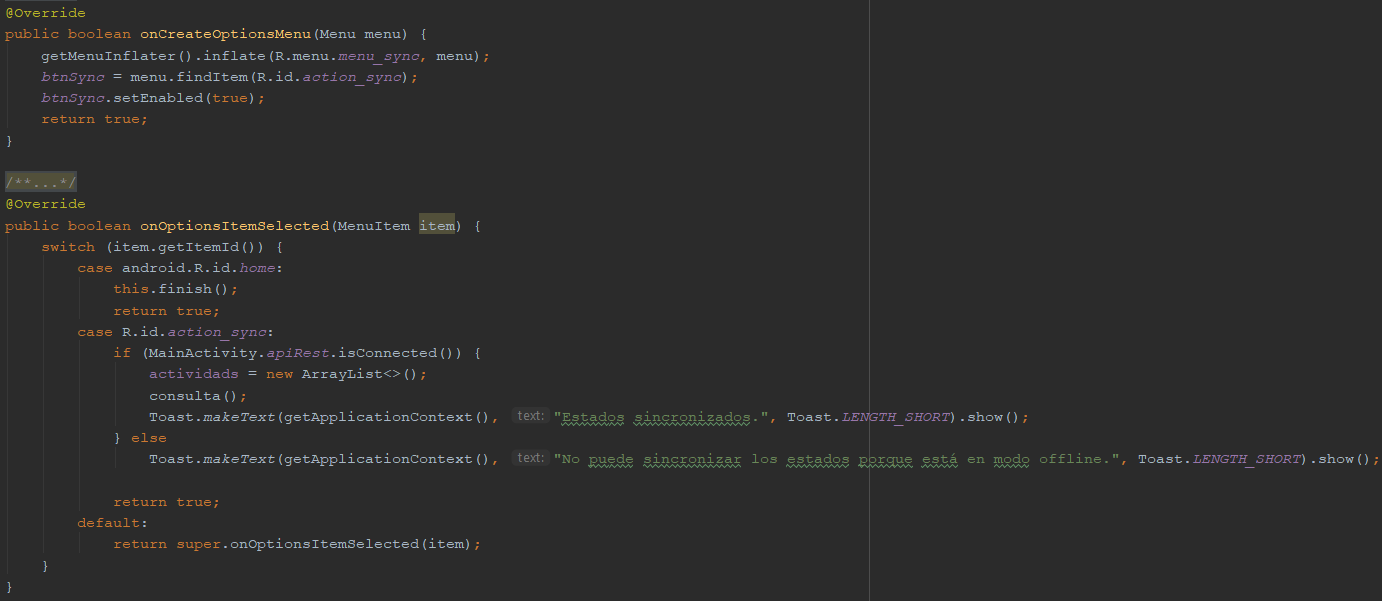
Se analizan los archivos para mostrar de una forma más agradable los recursos de las actividades y luego si se está *online* se realiza la sincronización de forma automática, para esto se consulta el estado local y remoto de las actividades, así como las observaciones que contienen los metadatos que permitirán aplicar el modelo de sincronización (***ver 4.3 Modelo de sincronización***) que se realiza en la función *sincronizar*:



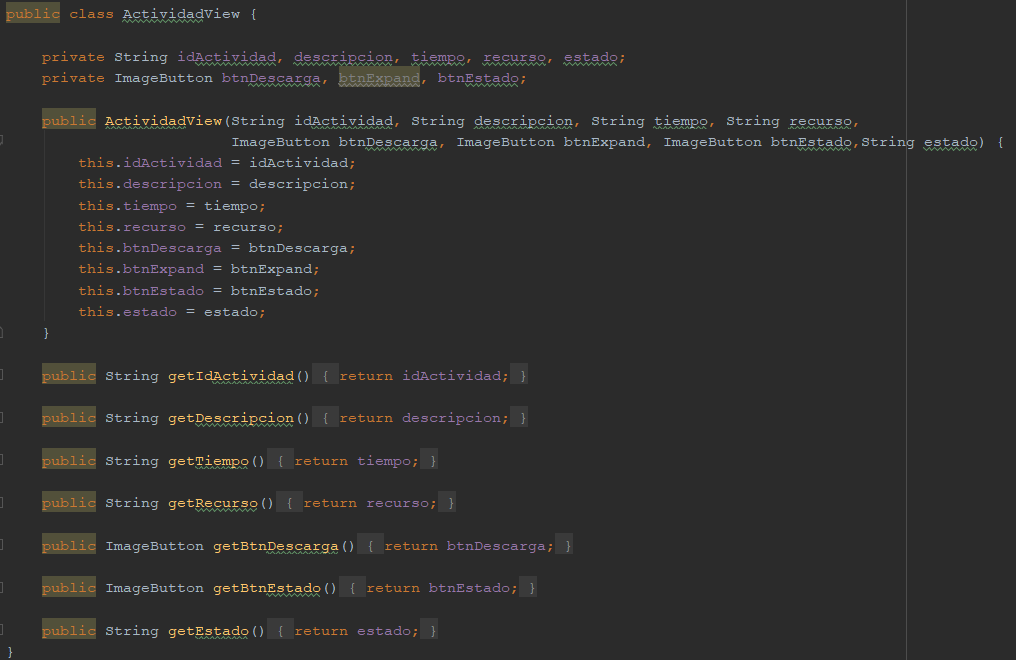
Cabe aclarar, que aumentan los parámetros del constructor de *ActividadView* porque deben estar todos los elementos (líneas rojas) que se cambian o interactúan del *CardView*:



También en la *ActividadActivity* se asigna un menú para poder sincronizar el estado local de las actividades con el estado en el nodo por si se estuvo trabajo offline no perder el progreso del estudiante, al presionar el botón vuelve a consultar la información que a su vez llama al método sincronizar:



La clase *ActividadView* contiene los atributos para cada uno de esos, el *estado* es el que me permite identificar el icono que se debe mostrar para la actividad:

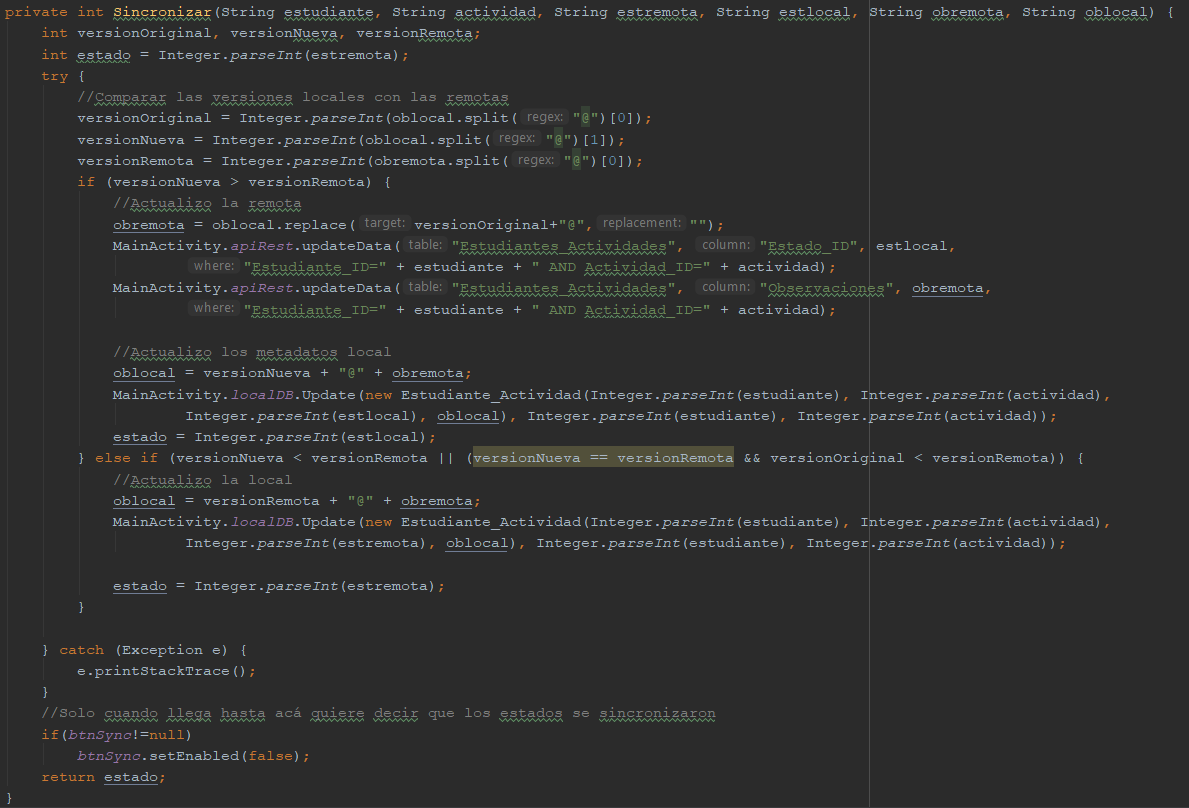


### 1.2.1) Sincronizar SQLite local con nodo MariaDB

Gracias al método *isConnected* la aplicación siempre que tenga haya una conexión los estados se sincronizará automáticamente, de la siguiente forma:

* Si se inició en modo *online* pero se perdió la conexión, al cambiar el estado aparecerá un mensaje diciendo que está en modo offline, y si vuelve la conexión la cambiar un estado, este automáticamente se sincronizarán los estados.
* Si se inició en modo *offline* y se hicieron cambios aparece un mensaje recordando que luego se debe sincronizar, sin embargo, si vuelve la conexión al cambiar un estado se sincronizará este automáticamente.

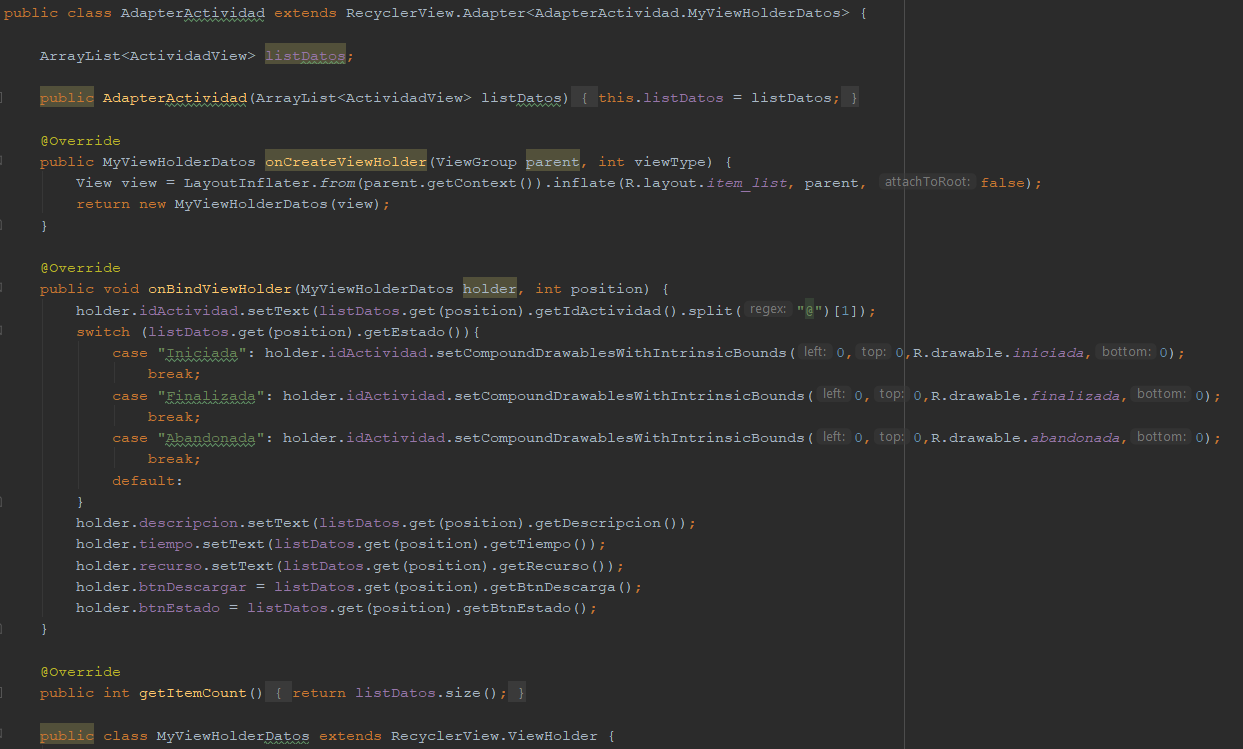
No obstante, estas opciones sincronizarían solo el estado que se acaba de cambiar en ese instante por ello, si se hicieron cambios en modo *offline* y se vuelve a iniciar sesión con conexión se sincronizarán todos los estados automáticamente o si estando en modo *offline* vuelve la conexión se puede dar clic en el botón *sincronizar* (ubicado en la esquina superior derecha) para sincronizar todos los estados que vuelve a realizar la consulta que llama al método sincronizar que me devolverá el estado sincronizado para mostrar en la lista:



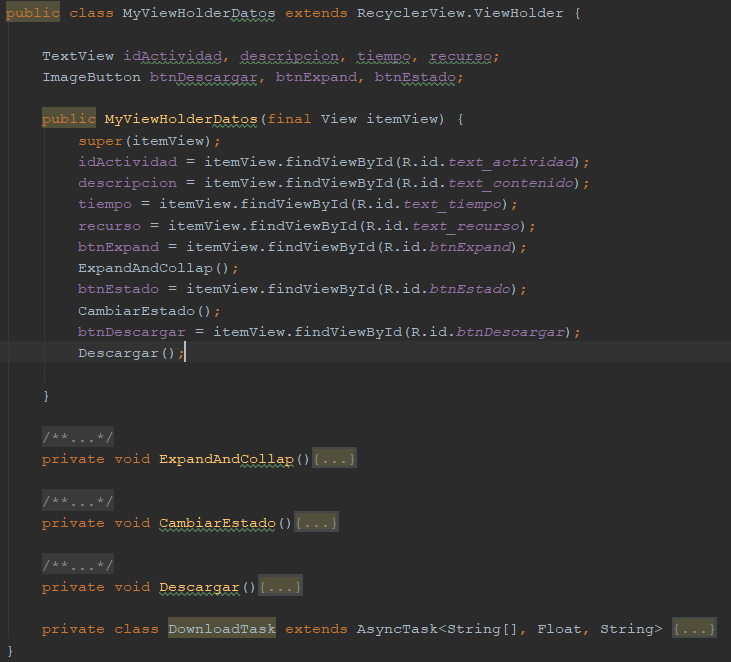
### 1.2.2) Actualizar el estado de la actividad

Ahora bien, la clase *AdapterActivity* es la que se encarga de varias cosa:

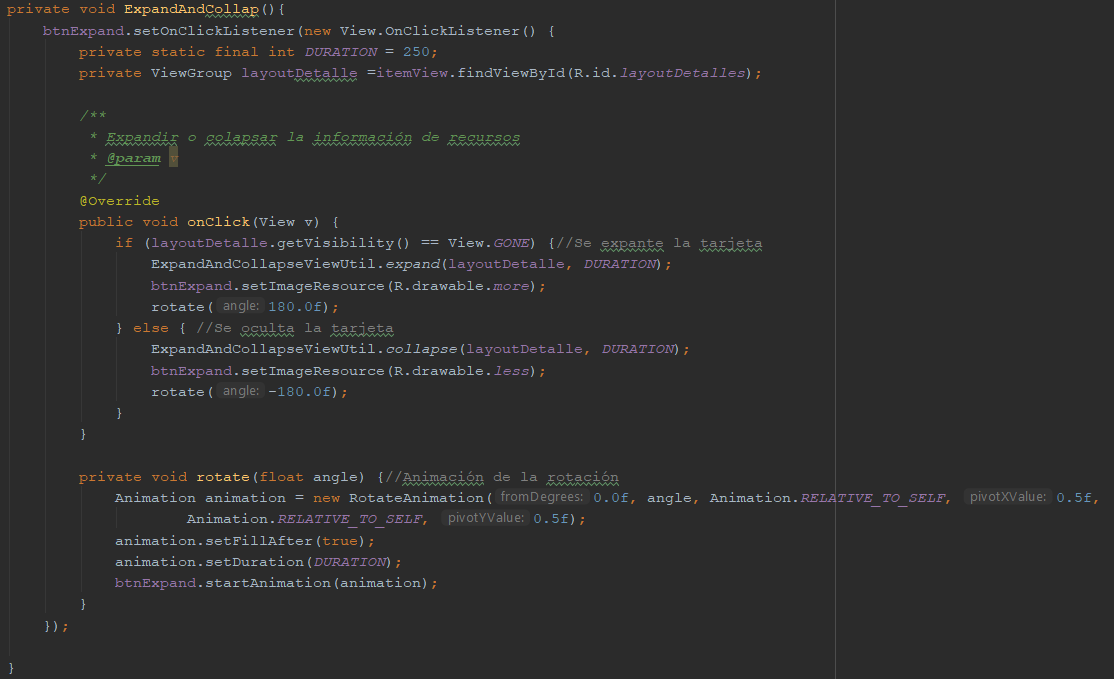
* Replicar la tarjeta tantas veces como actividades hayan, ademas es allí donde se codifica toda la logica cuando se hace clic en un botón de la *CardView*:



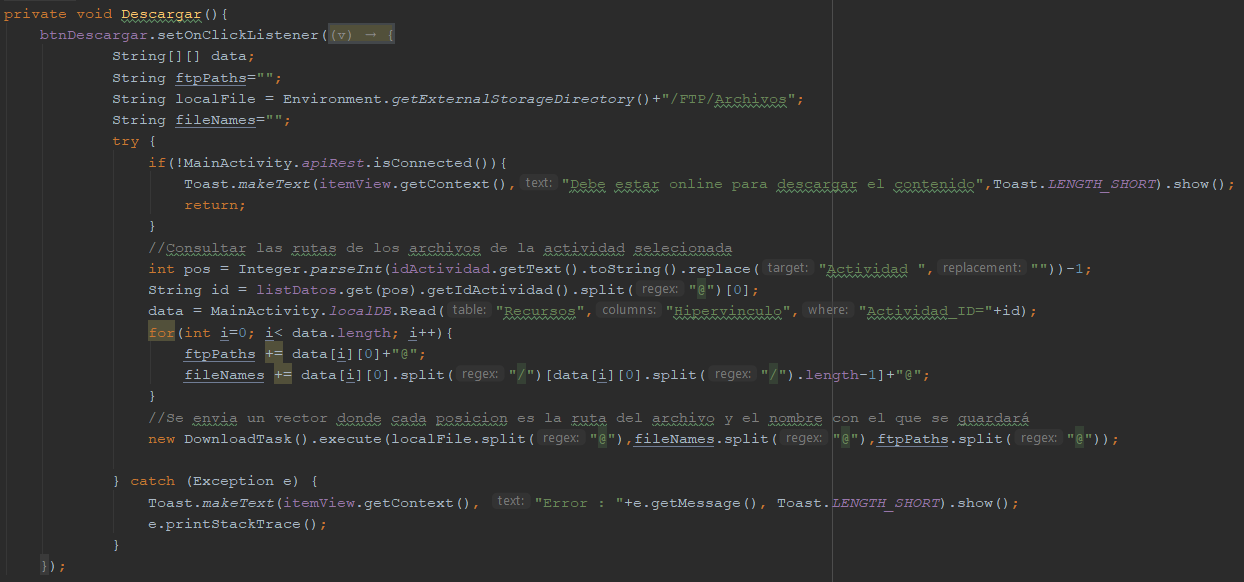
* Para los eventos cuando se hace clic se debe asignar el método *setOnClickListener* en la parte de la clase *MyViewHolderDatos*, primero se establecen las variables y para que el código sea más legible cada evento para los clics (expandir, cambiar estado y descargar):



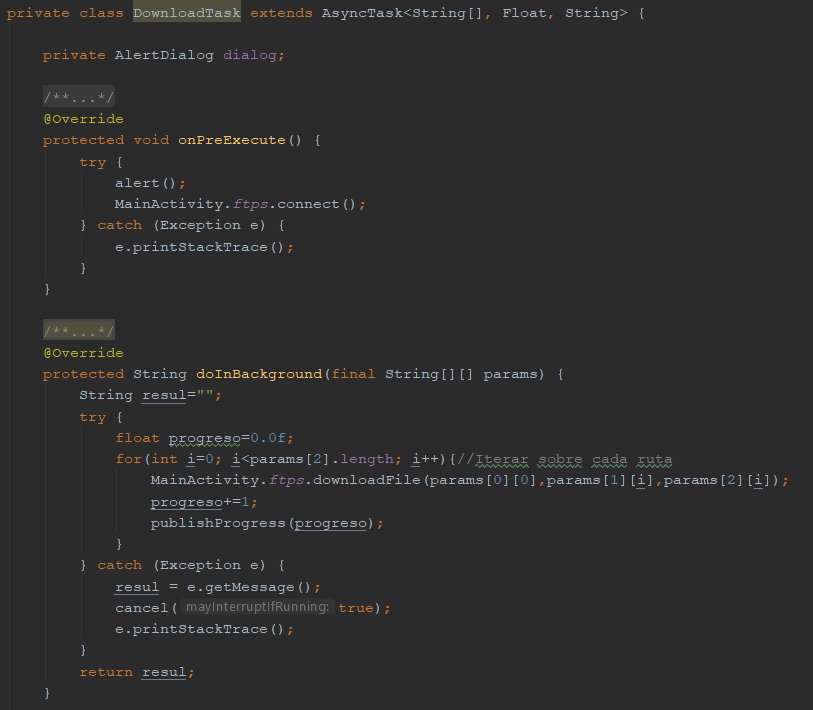
* El método para asignar el evento cuando se desea expandir o colapsar la información de la actividad:



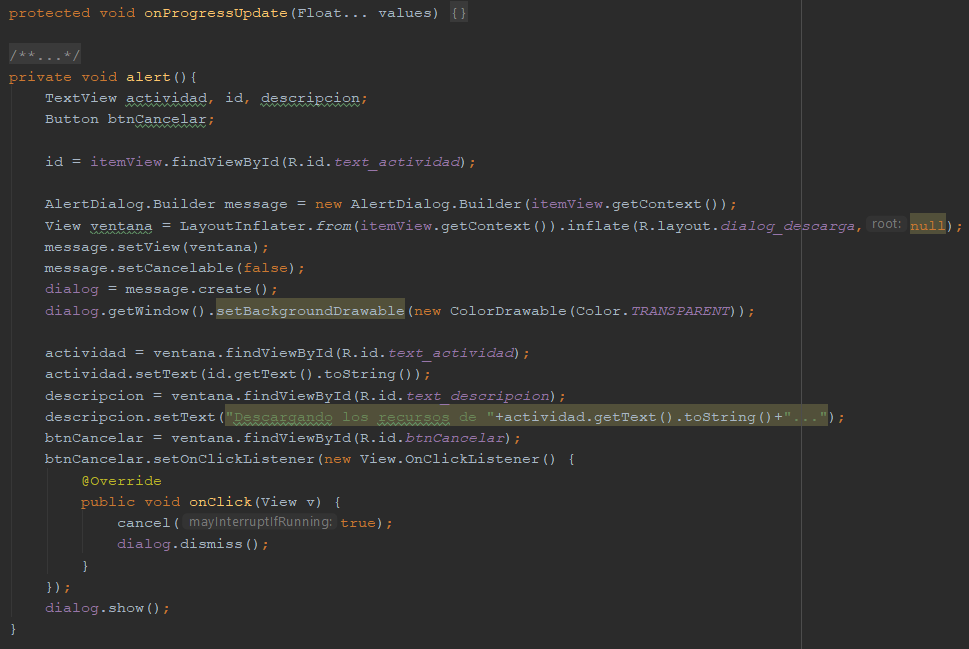
* El método para descarga los recursos de una actividad, que hace uso de un hilo para no bloquear el funcionamiento del hilo principal y tener una mejor experiencia de usuario:



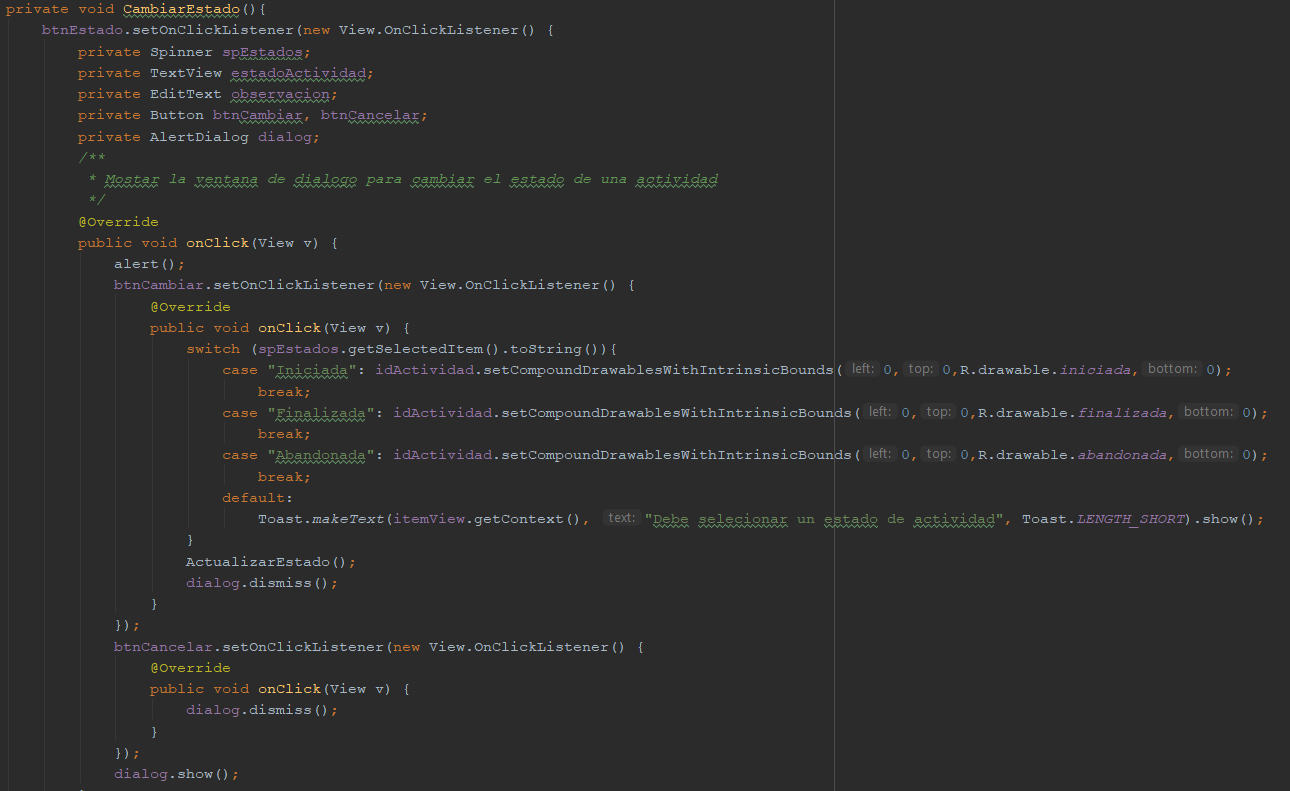
El código de la clase *DownloadTask*, que es una clase para realizar tareas asincrónicas en segundo plano, que establece la ventana de dialogo por el método *alert* y en el método *doInBackground* descarga los recursos:

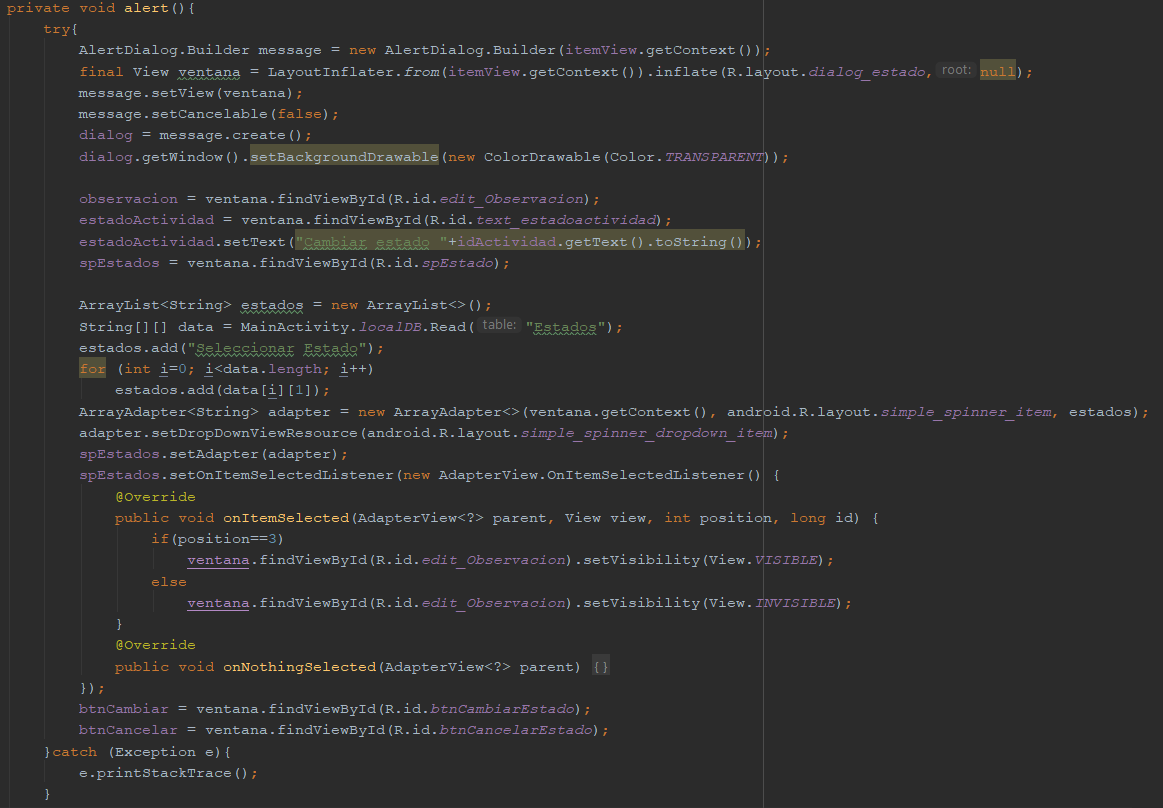


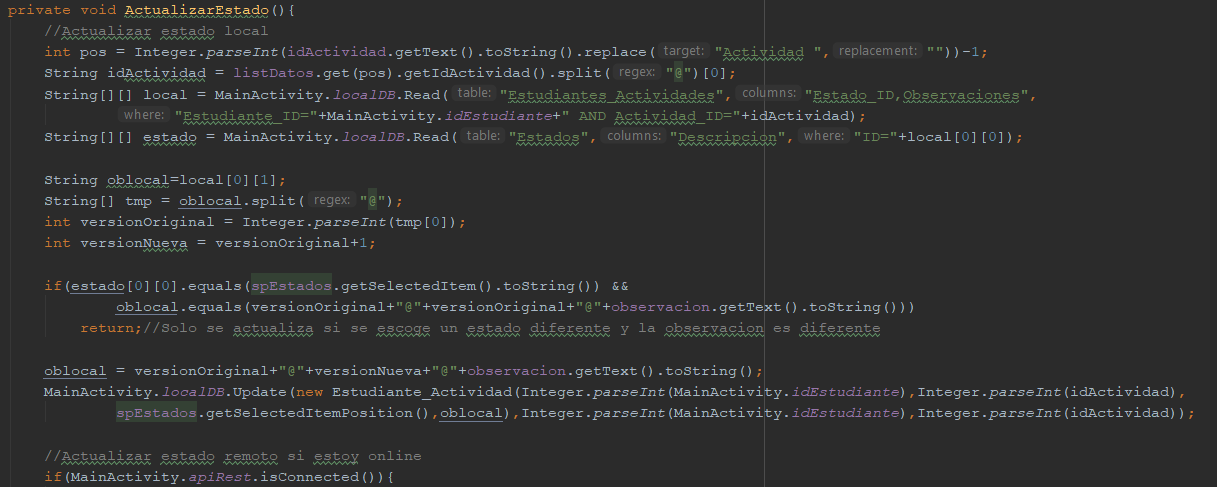


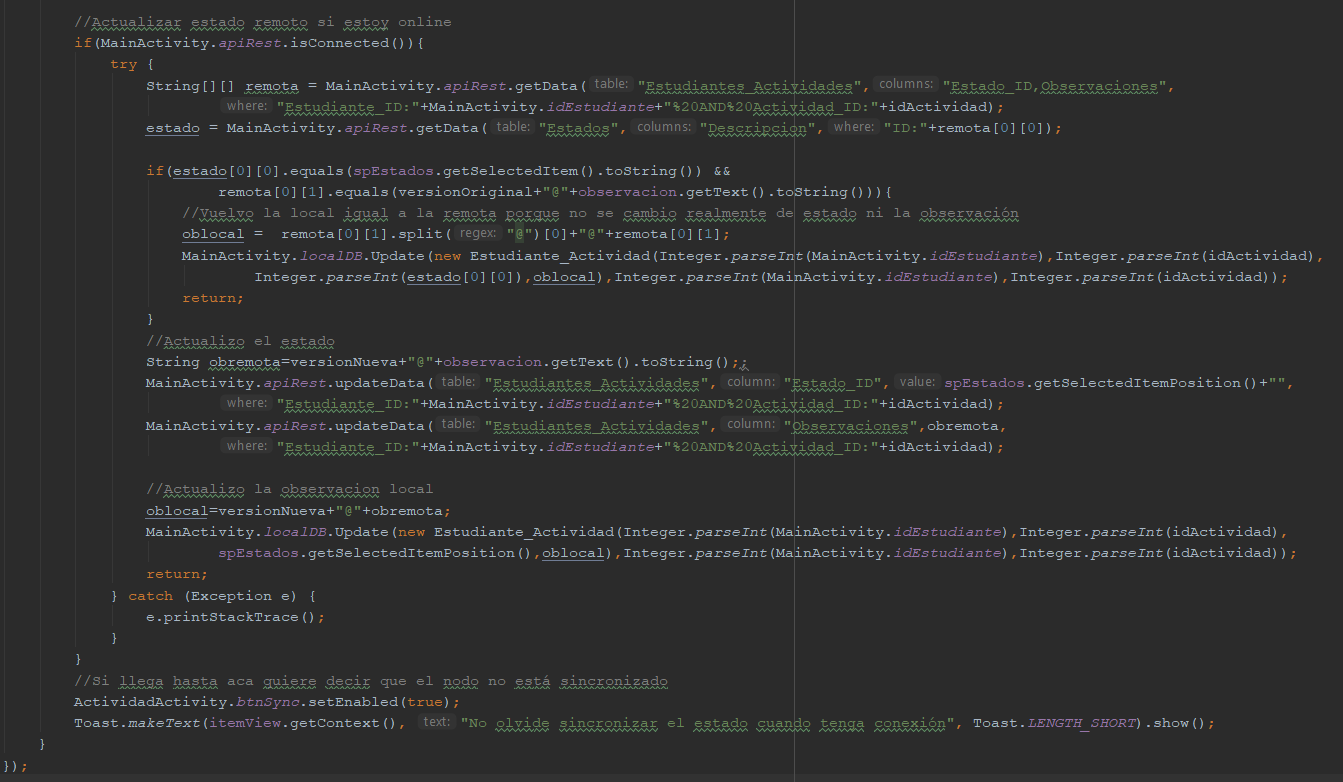


* El método de cambiar el estado se encarga de establecer la ventana de dialogo por medio de la función *alert* (que solo cuando se elige el estado *Abandonada* se muestra el cuadro de texto para ingresar la observación), también actualiza la imagen que se muestra y, lo más importante, de **actualizar el estado en la base de datos local y si esta online, realiza la actualización en el nodo**, siempre y cuando se cumpla: el nuevo estado es diferente al anterior o la observación es diferente, es decir, se actualizará si en verdad se realizó algún nuevo cambio por medio del método *ActualizarEstado*:









## 1.3) Modelo de sincronización

Al iniciar sesión se comparan las versiones locales con la versión remota, y se sincronizará de la siguiente forma:

* Si la versión nueva local es mayor que la versión remota se actualiza la remota con la local y la versión original local pasa a ser la nueva versión remota
* Si la versión nueva local es menor que la versión remota se actualiza la local con la remota, poniendo la versión nueva y original versión local iguales a la remota.
* Si la versión nueva local es igual que la versión remota pasan dos cosas:
  + Si la versión original local es menor que la versión remota se actualiza la local con la remota, debido a que en el nodo está la información real.
  + Si la versión original local es igual que la versión remota no se actualiza

**Nota:** La versión original local nunca es mayor que la versión remota, por lo que así se cubren todos los escenarios posibles.

Además, esto mismo sucede cuando se da clic en el botón Sincronizar, que solo estará activo cuando la nueva versión y la versión original son diferentes que sucede cuando se actualiza un estado y no se logra actualizar en el nodo (modo offline).

* Formato del campo Observaciones en el nodo

|  |  |
| --- | --- |
| <Versión> | <Información> |

* Formato del campo Observaciones en el dispositivo

|  |  |  |
| --- | --- | --- |
| <Versión original> | <Versión nueva> | <Información> |

El flujo es el siguiente:

1. Al iniciar sesión se comparan las versiones para sincronizar o para habilitar el botón de sincronizar (offline)
2. Al actualizar el estado de una actividad la *versión nueva* se vuelve la *versión original* + 1, se intenta la sincronización o si no da se habilita el botón de sincronizar (offline)
3. Al dar en el botón sincronizar, se aplica el modelo de sincronización.

# Referencias

Diseño Android: Tarjetas con CardView. danielme.com. (2020). Retrieved 5 May 2020, from <https://danielme.com/2015/08/12/diseno-android-tarjetas-con-cardview/>

Optimizando la interfaz Android - Compound Drawables. Optimizando la interfaz Android - Compound Drawables. (2020). Retrieved 5 May 2020, from <https://elbauldelprogramador.com/optimizando-la-interfaz-android-compound-drawables/>

Diseño Android: Spinner. danielme.com. (2020). Retrieved 5 May 2020, from <https://danielme.com/2013/04/25/diseno-android-spinner/>

Cómo interpretar el ciclo de vida de una actividad. developer. (2020). Retrieved 6 May 2020, from <https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es>

¿Cuál es la mejor forma de iterar un Cursor de Android?. Stackoverrun.com. (2020). Retrieved 14 May 2020, from <https://stackoverrun.com/es/q/2841081>

como habilito e inhabilito las opciones de un menú. Stack Overflow en español. (2020). Retrieved 14 May 2020, from <https://es.stackoverflow.com/questions/44790/como-habilito-e-inhabilito-las-opciones-de-un-men%C3%BA>