**Juan Camilo Restrepo Velez 000373886**

Tabla de contenido

[1) API conexión servidor 2](#_Toc38919728)

[1.1) Consumir AUTOMATIC API REST 3](#_Toc38919729)

[1.2) API JSON 3](#_Toc38919730)

[1.3) Servidor AUTOMATIC API REST 4](#_Toc38919731)

[2) Ajuste MariaDB 5](#_Toc38919732)

[3) Conexión FTPS 6](#_Toc38919733)

[3.1) Implementación clase ConexionFTPS 6](#_Toc38919734)

[3.2) Servidor FTPS 8](#_Toc38919735)

[4) Back-End 10](#_Toc38919736)

[4.1) MainActivity 10](#_Toc38919737)

[4.2) ActividadActivity 11](#_Toc38919738)

[4.3) ArchivoActivity 11](#_Toc38919739)

[4.4) SimpleTask 12](#_Toc38919740)

[4.5) NOTA 13](#_Toc38919741)

[4.6) Extra 14](#_Toc38919742)

[Referencias 16](#_Toc38919743)

# 1) API conexión servidor

Para poder consumir el API se ha creado la clase ***ConexionApiRest*** la cual nos facilitará la conexión con la base de datos MariaDB para poder obtener los datos de las tablas que se requieren. Esta clase, es abstracta, ya que cuenta con dos métodos estáticos, uno downloadData() que nos permitirá descargar la información de la base de datos por medio del API LINK que genera AUTOMATIC API REST, la cual queda en formato JSON y getData() que además de descargar los datos (llamando a downloadData()) procesará la información (por medio del **API JSON**) devolviendo una matriz con el siguiente formato:

|  |  |
| --- | --- |
| Columna1 | Value |
| Columna2 | Value |
| Columna3 | Value |
| … | … |

Cabe aclarar que cuando se descarga la información de toda la tabla, esta matriz contendrá al inició la columna ID, mientras que, si se descarga con una sola columna se omiten las demás columnas

**Ejemplo consulta toda la tabla**

**API LINK :** http://dominio/api/get/ftps

**downloadData():** {"data":[{"0":"1","ID":"1","1":"192.168.1.57","IP":"192.168.1.57","2":"admin","User":"admin","3":"1234","Password":"1234"}],"dbInfo":"IIUP"}

**getData() :**

|  |  |
| --- | --- |
| ID | 1 |
| IP | 192.168.1.57 |
| User | Admin |
| Password | 1234 |

**Ejemplo consulta una columna**

**API LINK :** http://192.168.1.53/api/get/actividades/Descripcion/?w=Sesion\_ID:1

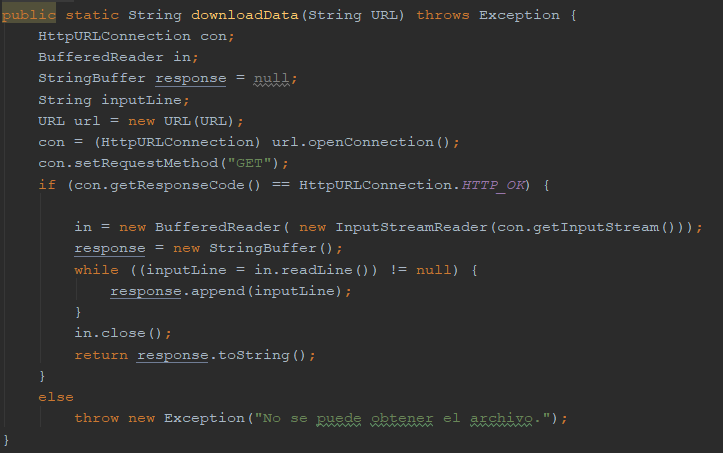
**downloadData():** {"data":[{"0":"Actividad 1 de la sesion de oscar","Descripcion":"Actividad 1 de la sesion de oscar"},{"0":"Actividad 2 de la sesion de oscar","Descripcion":"Actividad 2 de la sesion de oscar"}],"dbInfo":"D"}

**getData() :**

|  |  |
| --- | --- |
| Descripcion | Actividad 1 de la sesion de oscar |
| Descripcion | Actividad 2 de la sesion de oscar |

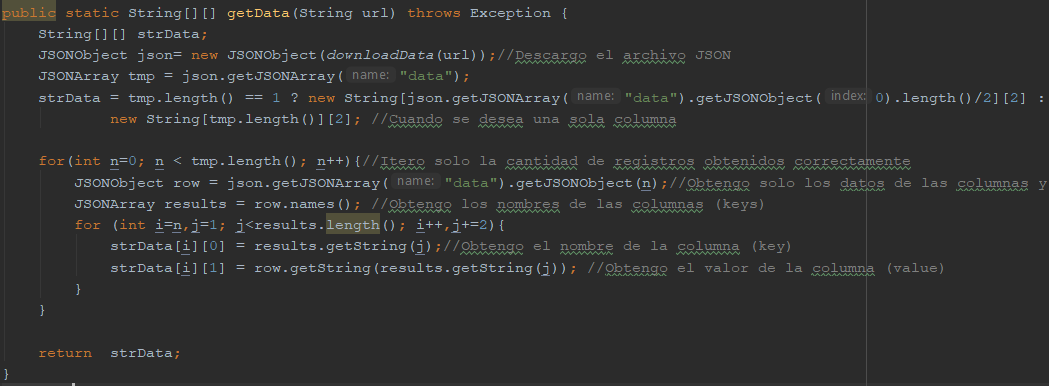
## 1.1) Consumir AUTOMATIC API REST

Ahora bien, el método en cargado de consumir el API REST es el ***downloadData()*** que simplemente se encarga de descargar el JSON por el API LINK que se le pasé y lo retorna en una variable tipo String, queda así:



## 1.2) API JSON

Por otro lado, el método ***getData()*** se encarga de procesar todo el JSON devolviendo la matriz en el formato especificado, que como se dijo, existen dos escenarios, el primero es obtener todos los registros de la tabla y el otro solo una columna, por lo que para obtener la cantidad de registros en la primera se debe dividir entre 2 (se duplican las columnas) el tamaño del JSON después de filtrar por “data”, mientras que, en el segundo es el mismo tamaño del JSON después de aplicar el filtro. En código que:



## 1.3) Servidor AUTOMATIC API REST

Para poder que el API funcione se debe instalar apache y habilitar las conexiones

apt install apache2

Luego, instalar php (minimo la versión 5) para poder que funcione la página

sudo apt-get install apt-transport-https lsb-release ca-certificates

sudo wget -O /etc/apt/trusted.gpg.d/php.gpg <https://packages.sury.org/php/apt.gpg>

sudo sh -c 'echo "deb https://packages.sury.org/php/ $(lsb\_release -sc) main" > /etc/apt/sources.list.d/php.list'

sudo apt update

sudo apt install php7.3

sudo apt-get install php7.3-mbstring php7.3-mysql php7.3-xml php7.3-curl libapache2-mod-php7.3

Una vez con todo el entorno listo, se descarga el archive de API REST [aquí](http://goo.gl/7kPWWP), luego se pasa el .zip al servidor por medio de FTPS y se ubica en la siguiente carpeta

mv Automatic-API-REST-master.zip /var/www/html/

cd /var/www/html/

Si no se cuenta con el paquete unzip se instala con *apt install unzip*

unzip Automatic-API-REST-master.zip

rm Automatic-API-REST-master.zip

mv Automatic-API-REST-master/ ApiRest

nano ApiRest/config.php

define("SERVER","localhost");

define("USER","admin");

define("PASS","1234");

define("DB","EduApps");

define("USERADMIN","admin");

define("PASSADMIN","admin");

mv ApiRest/Server-Configuration/.htaccess .htaccess

nano .htaccess

RewriteEngine on

RewriteRule ^api/ ApiRest/api.php

sudo nano /etc/apache2/apache2.conf

… (Buscar la linea siguiente y dejarla así)

<Directory /var/www/>

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

sudo a2enmod rewrite

Luego se reinicia el servidor apache

service apache2 restart

y se ingresa al navegador ***http://DireccionIP/ApiRest*** se procede hacer el LogIn con los datos definidos (USERADMIN y PASSADMIN), si todo ha salido bien se verán las tablas de la base de datos y al dar click en una se verán las columnas

# 2) Ajuste MariaDB

Se debe adicionar a la base de datos actual la siguiente tabla:

USE EduApps;

CREATE TABLE FTPS

    (ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

     IP VARCHAR(255) NOT NULL,

     User VARCHAR(255) NOT NULL,

     Password VARCHAR(255) NOT NULL);

Además, de realizar las siguientes inserciones:

INSERT INTO FTPS VALUES (NULL,‘192.168.0.101’,‘admin’,‘1234’);

INSERT INTO Recursos VALUES (NULL,1, '/home/admin/www/oscar/1/1/archivo.pdf’),

(NULL,2, '/home/admin/www/oscar/1/2/archivo.apk’), (NULL,2, '/home/admin/www/oscar/1/2/archivo.png’);

# 3) Conexión FTPS

Para realizar la conexión FTPS desde la aplicación de estudiantes al servidor FTPS del nodo Raspberry se hará uso de la misma librería que se utilizó para el prototipo anterior, ***Apache commons* net** (<http://commons.apache.org/proper/commons-net/download_net.cgi>), además se tomará como supuesto que los datos como lo son la IP del servidor, el usuario y contraseña del servidor FTPS ya se han obtenido por medio del API REST, así como la ruta de los paquetes SCORM de las actividades.

## 3.1) Implementación clase ConexionFTPS

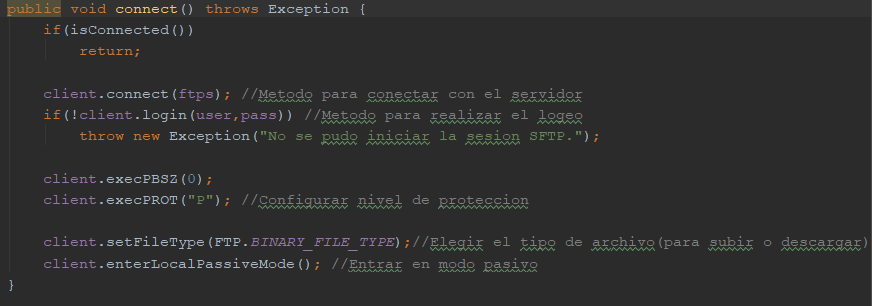
Luego de descargar la librería se debe poner en la carpeta “libs” del proyecto y darle click derecho y “add lib”, de este modo ya puede realizar el import correspondiente asi



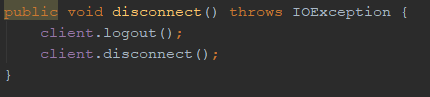
Para tener todo en un menor grado de acoplamiento y mayor legibilidad del proyecto se creará una clase java denominada ConexionFTPS.java que se encargará de todo el manejo de la librería previamente mencionada. Esta clase tendrá los atributos necesarios para realizar la conexión además de los métodos para gestionar la desconexión, la descarga y subida de archivos.

**Constructor:** es el encargado de crear el objeto y recibe como parámetro la dirección Ip del servidor, el usuario y la contraseña.

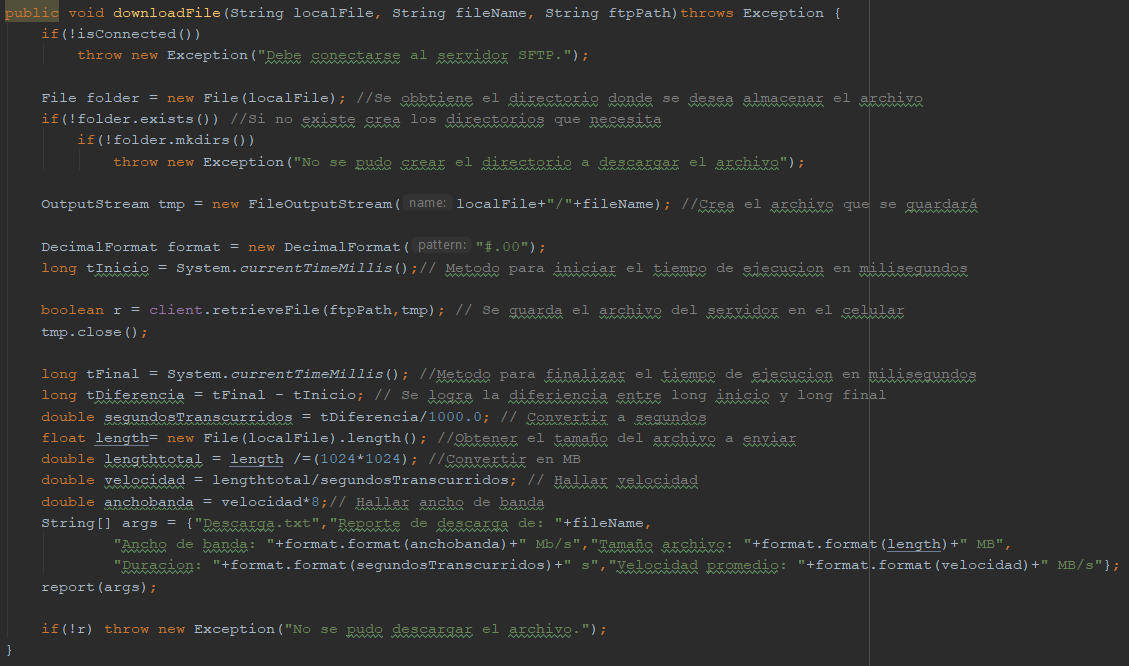
**Conectar:** Se encarga de la conexión para lo cual se debe llamar al método .connect(sftp) ; del objeto FTPSClient de la librería, enviando como parámetros la IP del servidor. Posteriormente se realiza el LogIn al servidor (.login(user,pass) ;), se establecen los parámetros de seguridad y de transferencia de archivos. El método completo es el siguiente:



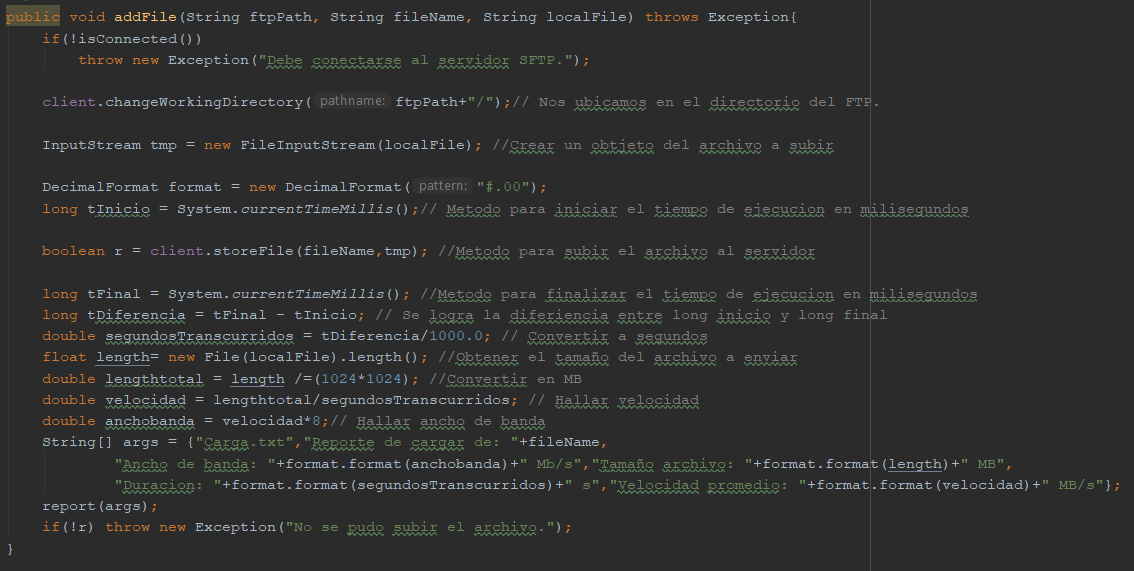
**Desconectar:** Realiza la desconexión llamando al método .logout(); y .disconnect(); del objeto FTPSClient . El método queda asi:



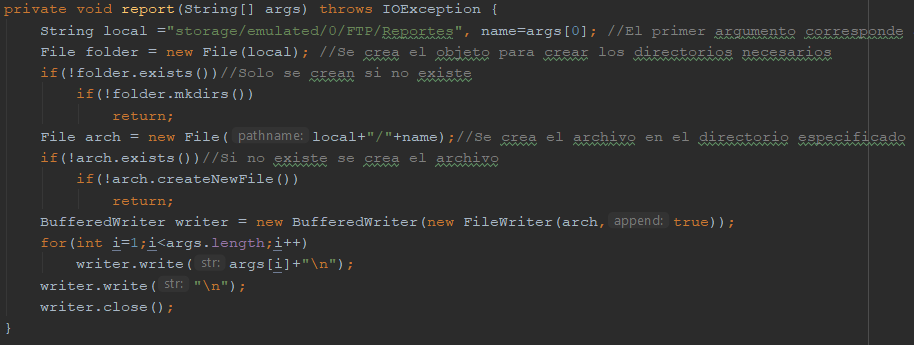
**Descarga:** Para esta función se debe se recibe por parámetro tres variables tipo String, una para la ruta del archivo a descargar que está alojado en el servidor (ftpPath, incluye el nombre del archivo), otra con la ruta donde se va a guardar localmente en el dispositivo móvil (localFile) y el nombre con el cual se guardará local (fileName), para ello se utilizará el método .retrieveFile(ftpPath,tmp); del objeto FTPSClient donde la variable tmp corresponde a un objeto de la clase FileOutputStream(localFile+”/”+fileName); que en el constructor recibe la ruta y el nombre del archivo a guardar. Además, en este método se tomarán los tiempos de descarga y demás parámetros para llamar a una función de reporte para generar estos. El método queda:



**Carga:** Brinda la posibilidad de subir archivo para lo cual se debe especificar, por medio del objeto FTPSClient . Este método recibe tres parámetros, uno correspondiente a la ruta del archivo, en disco, que se desea subir (localFile, incluye el nombre del archivo), otro que indica el directorio al cual se desea subir el archivo (ftpPath) y el nombre con el cual se quedará guardado en el servidor (fileName). Para ello, nos ubicamos en el directorio en el servidor por medio del método .changeWorkingDirectory (ftpPath);, luego se debe crear un objeto FileInputStream (tmp) cuyo constructor recibe la ruta del archivo del servidor concatenado con el nombre del archivo (ftpPath y fileName), por último se llama el método .storeFile(fileName,tmp); del objeto FTPSClient que recibe el nombre del archivo y el objeto previamente creado. Este método también cuenta con los cálculos para generar los informes. El método completo es:



**Reporte:** Para la función de Descargar y Carga se debe generar un reporte de estas dos operaciones, desde las cuales se llama al método private void report(String[] args) que recibe el un vector que contiene el nombre del archivo a escribir (args[0]) y el cuerpo del archivo (las demás posiciones), donde cada posición del vector es una línea (si el archivo existe, escribe desde la última línea, es decir que no sobre escribe). Los parámetros son calculados en los métodos Carga y Descarga, el método queda:



## 3.2) Servidor FTPS

Una vez se tenga el servidor FTP funcionando, para hacer que soporte autentificación por TLS/SLL para cifrar la conexiones, primero se debe tener instala OpenSSL (***apt-get install openssl***) para generar un certificado

Se crea el certificado

sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 –keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem

Preguntará por algunos datos para incluir en el certificado (se ponen cualquiera) y luego se procede a editar el archivo vsftpd.conf descomentariando o agregando las siguientes lineas

nano /etc/vsftpd.conf

ssl\_enable=YES

rsa\_cert\_file=/etc/ssl/private/vsftpd.pem

rsa\_private\_key\_file=/etc/ssl/private/vsftpd.pem

allow\_anon\_ssl=NO

force\_local\_data\_ssl=YES

force\_local\_logins\_ssl=YES

require\_ssl\_reuse=NO

Guardamos los cambios y reiniciamos el servidor. Cuando se vaya a establecer una conexión por medio de un cliente FileZille nos mostrará una ventada que nos dice que el servidor utiliza un certificado autofirmado.

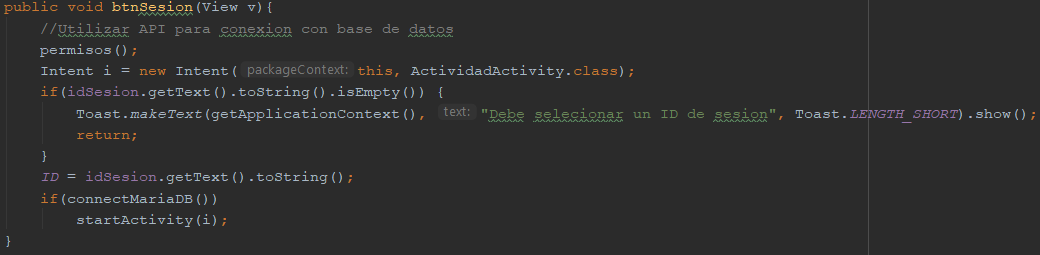
Más información en: <https://wakervall.wordpress.com/2017/12/17/instalacion-de-un-servidor-ftp-en-linux-ubuntu-16-04-server/>

# 4) Back-End

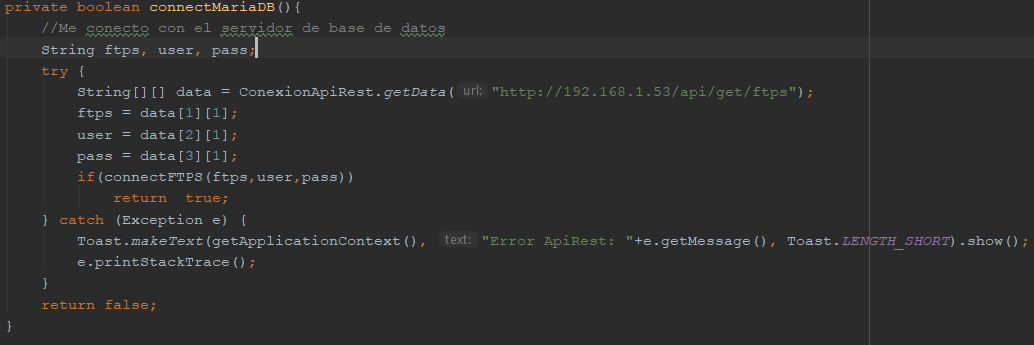
Luego de tener todas las clases de conexión listas se procede a realizar los llamados respectivos desde la aplicación, ya que, todos los datos, como lo son los del servidor FTPS (Ip, Usuario y Contraseña), se obtendrán a partir del API REST que se comunica con el servidor MariaDB para no quemar los datos y tener una aplicación más segura.

## 4.1) MainActivity

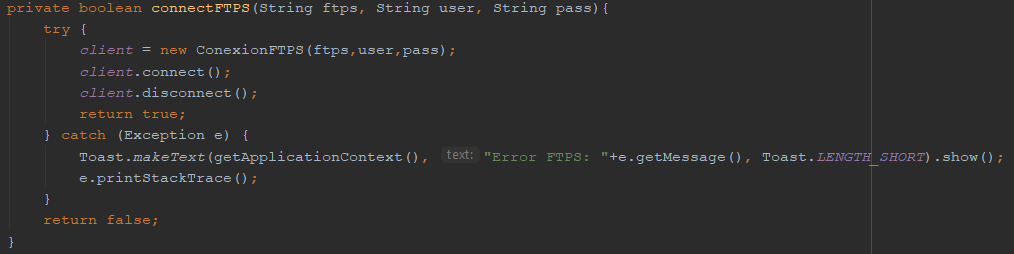
El código para el botón “Iniciar Sesión” valida que haya ingresado los campos y posterior realiza la conexión al API REST por medio de un método:



El método consultará la información del servidor FTPS para poder crear el objeto ConexionFTPS, como se ve en la siguiente imagen:



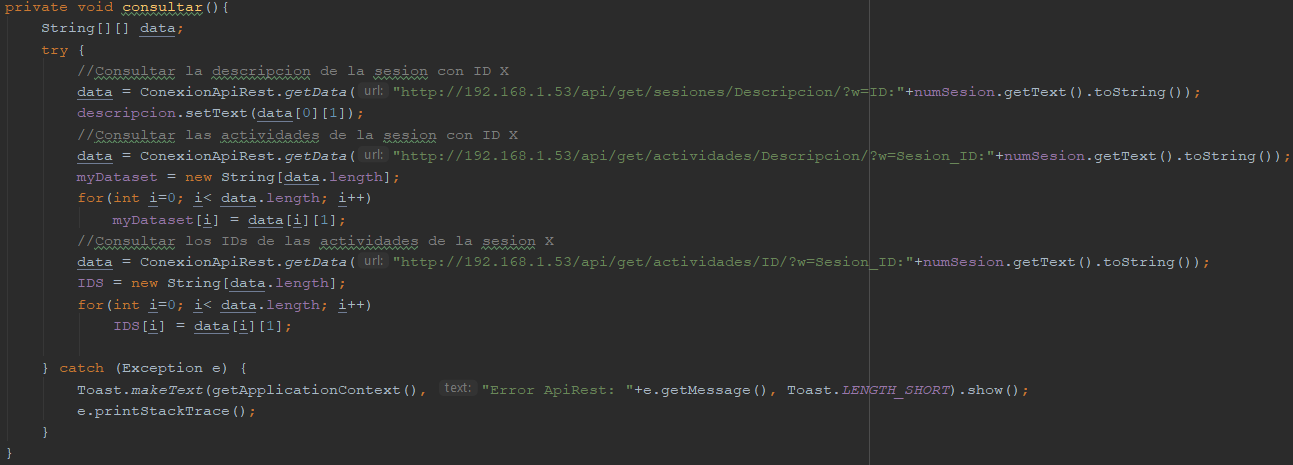
Una vez obtenidos estos parámetros le pasarán al método auxiliar para probar la conexión con el servidor FTPS, creando el objeto correspondiente que llamará al método connect() para validar los datos:



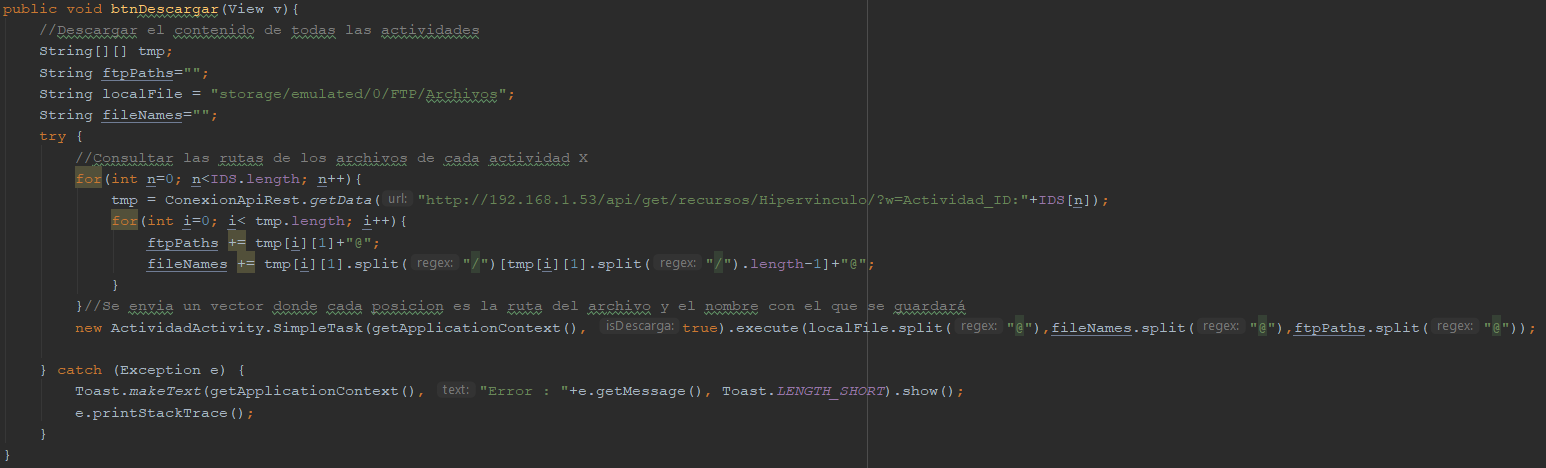
Toda esta secuencia debe ser exitosa al iniciar sesión porque que si falla en la conexión no podrá ingresar.

## 4.2) ActividadActivity

Al iniciar esta ventana se consultará la descripción de la sesión seleccionada para mostrar en la ventana y consultar las descripciones de las actividades de la sesión y sus IDs, por lo que queda el siguiente método:



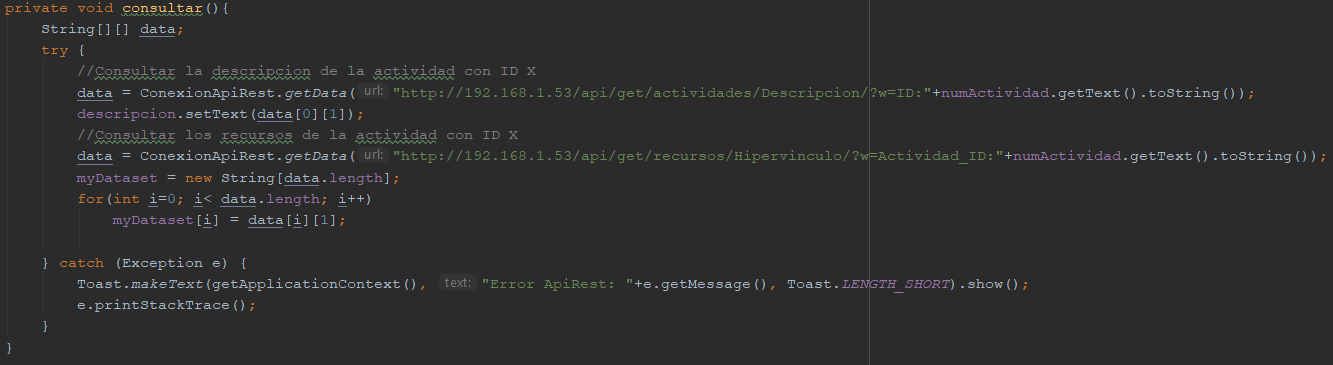
Por otro lado, si se presiona en el botón de “Descargar”, la aplicación descargará todos los recursos de todas las actividades por lo que consultará los recursos de cada actividad, luego, se envían todas las rutas a un hilo (de la clase SimpleTask) que se encargará de la descarga de estos recursos.



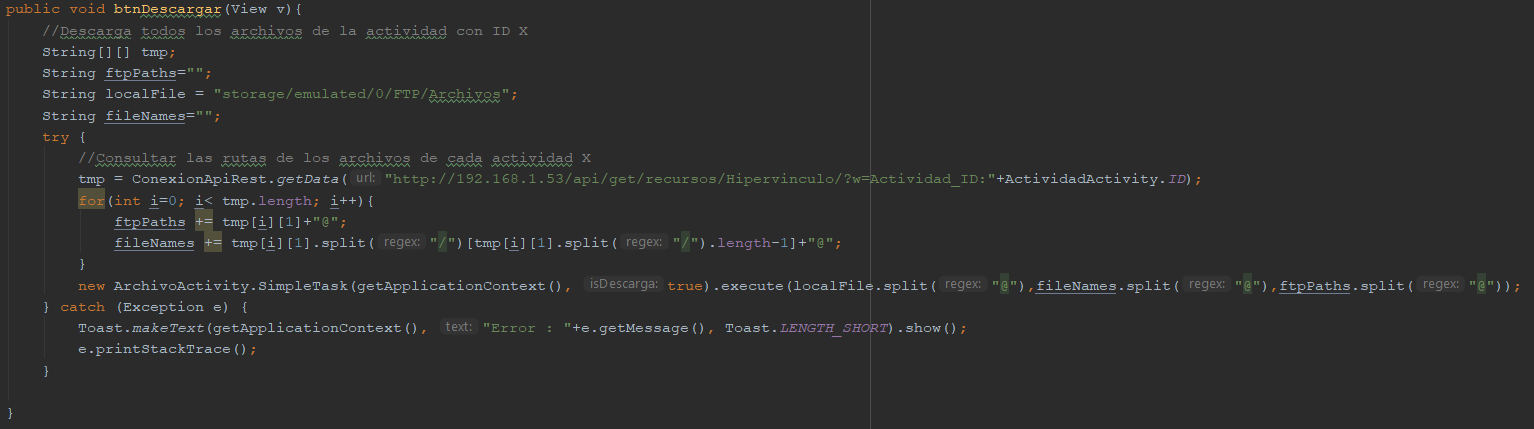
Por último, se tiene la opción de al darle clic a una actividad aparecerá se abre la ventana ***ArchivoActivity*** con la información de esta.

## 4.3) ArchivoActivity

Al igual que la ventana anterior, esta también cuenta con el un método para consultar los valores de descripción de la actividad y los recursos de esta para mostrar:



Además, está la misma opción de “Descargar” pero esta solo se aplica sobre los recursos de la actividad actual, por lo que descargará todos los recursos de la actividad, lo único que cambia es la consulta del API LINK, después, se le asigna a un hilo (de la clase SimpleTask) descargar todos los recursos que se consultaron.

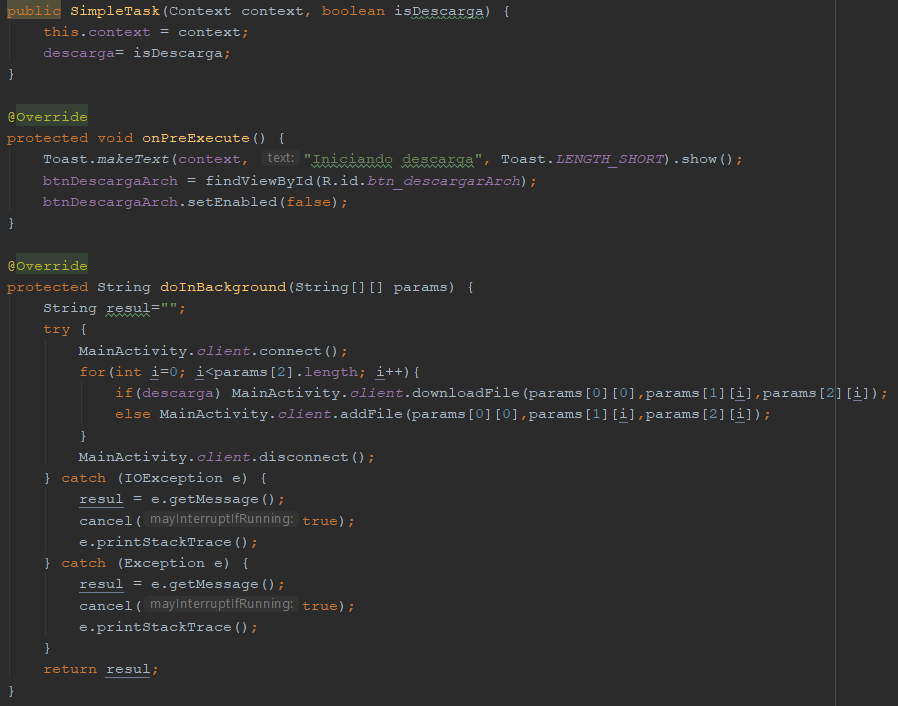


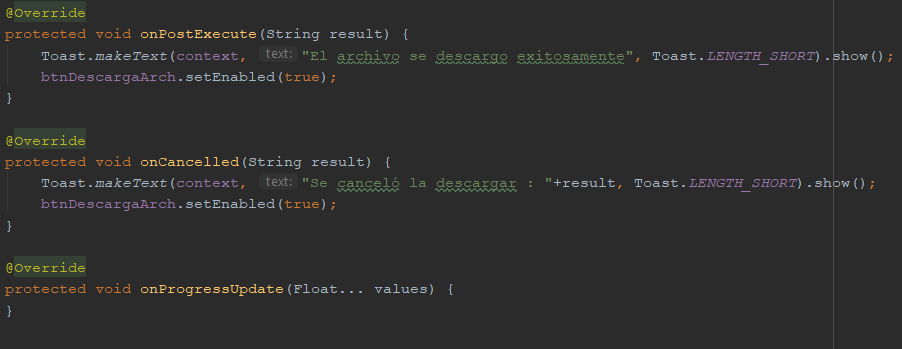
Por último, si se selecciona un solo archivo de la actividad se descargará únicamente este:



## 4.4) SimpleTask

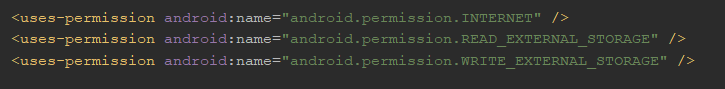
La clase SimpleTask, es una clase interna para realizar tareas asincrónicas, esto es asignar a un hilo el procesamiento de una función que en este caso es la descarga de archivo como se muestra a continuación:



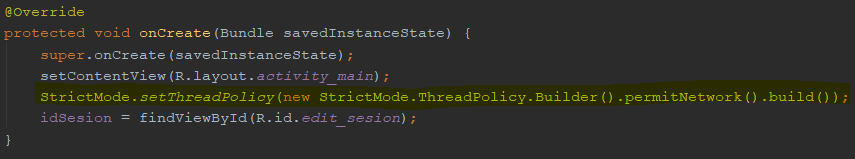


## 4.5) NOTA

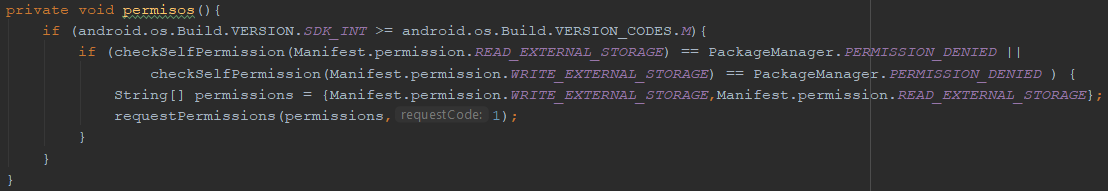
Se debe agregar en el AndroidManifest.xml las siguientes líneas para permitir el acceso a Internet, permitir la lectura y escritura de archivos:



También agregar la siguiente línea en el método onCreate del MainActivity para poder descargar los archivos JSON desde el hilo principal:

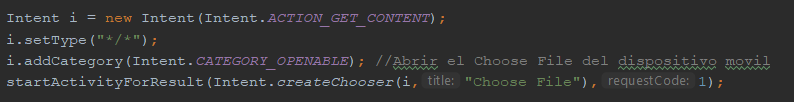


Y crear un método en la clase ActividadActivity denomino permiso() que será llamado en el método OnCreate para pedir los permisos (en tiempo de ejecución) que se establecieron en cuanto a la lectura y escritura de archivos, el cual es requerido para los métodos Cargar() y Descargar(), contiene el siguiente código:

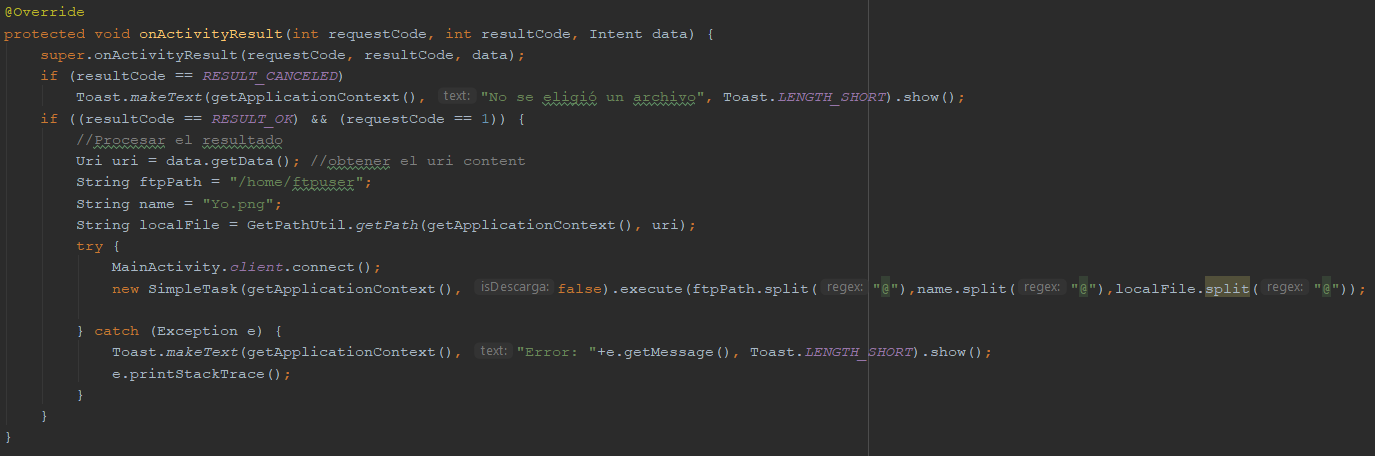


## 4.6) Extra

Aunque se implementó la posibilidad de **subir archivos**, no se ha identificado la acción por la cual el estudiante pueda hacer uso de esta función, sin embargo para hacerlo, el usuario deberá elegir un archivo por medio del "***File Chooser/File picker***" nativo de Android:



Luego se invoca un método estático de una clase de biblioteca (GetPathUtil) que se implementó para encontrar la ruta del archivo elegido, después, se creará un hilo con la misma estructura de los anteriores, solo que en su constructor se pasa un false, para que pueda llamar el método addFile() del objeto ConexionFTPS :



Al ser la misma de la clase SimpleTask, también se le puede asignar al hilo que cargue varios archivos.

# Referencias

Programacion en Castellano, S. (2020). Conectar vía SFTP con Java. Programación en Castellano. Retrieved 21 April 2020, from <https://programacion.net/articulo/conectar_via_sftp_con_java_1147>

Transferring a File Through SFTP in Java | Baeldung. Baeldung. (2020). Retrieved 21 April 2020, from <https://www.baeldung.com/java-file-sftp>

Cómo conectarse a FTP a través del servidor TLS/SSL (FTPS) en Java. Stackoverrun.com. (2020). Retrieved 21 April 2020, from <https://stackoverrun.com/es/q/10008268>

FTPSClient (Apache Commons Net 3.6 API). Commons.apache.org. (2020). Retrieved 21 April 2020, from <http://commons.apache.org/proper/commons-net/javadocs/api-3.6/org/apache/commons/net/ftp/FTPSClient.html>

WakerVall, V. (2020). Instalación de un servidor FTP en Linux Ubuntu 16.04 Server. WakerVall. Retrieved 21 April 2020, from <https://wakervall.wordpress.com/2017/12/17/instalacion-de-un-servidor-ftp-en-linux-ubuntu-16-04-server/>

AsyncTask en Android - Jarroba. Jarroba. (2020). Retrieved 21 April 2020, from <https://jarroba.com/asynctask-en-android/>

AsyncTask en Android . Cursos de Programación de 0 a Experto Garantizados. Cursos de Programación de 0 a Experto Garantizados. (2020). Retrieved 21 April 2020, from <https://unipython.com/asynctask-en-android/>

Automatic API REST Droidcon. Slideshare.net. (2020). Retrieved 26 April 2020, from <https://www.slideshare.net/AlejandroEsquiva/automatic-apirestdroidcon>

Instalar php5.6 en Ubuntu 18.04 – Ekiketa. Ekiketa.es. (2020). Retrieved 26 April 2020, from <https://ekiketa.es/instalar-php5-6-en-ubuntu-18-04/>

Cómo crear un usuario en MySQL/MariaDB y concederle permisos para una base de datos desde la línea de comandos | daniloaz.com. daniloaz.com. (2020). Retrieved 26 April 2020, from <https://www.daniloaz.com/es/como-crear-un-usuario-en-mysql-mariadb-y-concederle-permisos-para-una-base-de-datos-desde-la-linea-de-comandos/>

Nextcloud 14 en la Raspberry PI con PHP 7.2 (3). Servidor web, PHP 7.2 y base de datos. El blog del aburrido. Retrieved 26 April 2020, from <https://polkillas.net/blog/blog/2018/11/02/nextcloud-14-en-la-raspberry-pi-con-php-7-2-3-servidor-web-php-7-2-y-base-de-datos/>

Packages.sury.org. (2020). Retrieved 26 April 2020, from <https://packages.sury.org/php/README.txt>

How to Set AllowOverride all. Stack Overflow. (2020). Retrieved 27 April 2020, from <https://stackoverflow.com/questions/18740419/how-to-set-allowoverride-all>

Class in charge of the connections management with AutomaticAPIREST. Gist. (2020). Retrieved 28 April 2020, from <https://gist.github.com/MiguelCatalan/e36a610f4a3a8d08008a>

Taller Android Party: Automatic API REST + Notificaciones PUSH. Slideshare.net. (2020). Retrieved 28 April 2020, from <https://www.slideshare.net/AlejandroEsquiva/taller-androidparty>