1. Цель работы

Получить навыки разработки фоновых приложений под ОС Android. Научиться работать со сторонними API для организации сетевого приложения в ОС Android.

2. Формулировка задания

Разработать приложение для ОС Google Android, представляющее собой фоновый сервис без графического интерфейса. Данный сервис должен осуществлять периодический сбор и выгрузку информации в соответствии с выданным вариантом.

Требования:

— Выгрузка информации должна осуществляться при наличие любого подключения к сети Интернет и при любом состоянии телефона (включен/выключен дисплей);

— Фоновая работа должна происходить в отдельном потоке (не в UI thread’e).

3. Ход работы

Вариант функционала – сбор контактов, вариант выгрузки – облачный сервис Dropbox.

В ходе выполнения лабораторной работы было реализовано приложение на языке программирования Java в среде Android Studio. Сервис работает в фоновом режиме и периодически считывает данные о контактах (имя контакта и его номер телефона) и выгружает их через облачный сервис Dropbox.

Для начала необходимо создать аккаунт на Dropbox и в разделе для разработчиков создать в консоли приложений своё приложение.

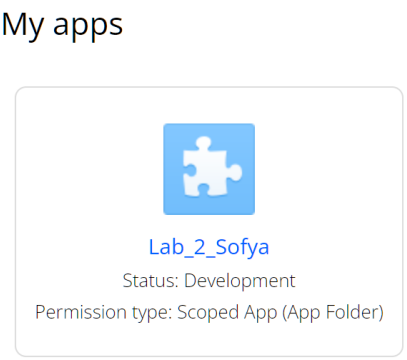

Рис. 1. – Собственное приложение-папка на Dropbox

Также во вкладке Permisions необходимо добавить все разрешения.

Для связи с серверами Dropbox используется API v2. Так как используется интерфейс API v2, то возможно получить доступ не только с помощью двух ключей (OAuth 1), но и вторым способом – OAuth2, используя токен, который можно сгенерировать, нажав на кнопку Generate.

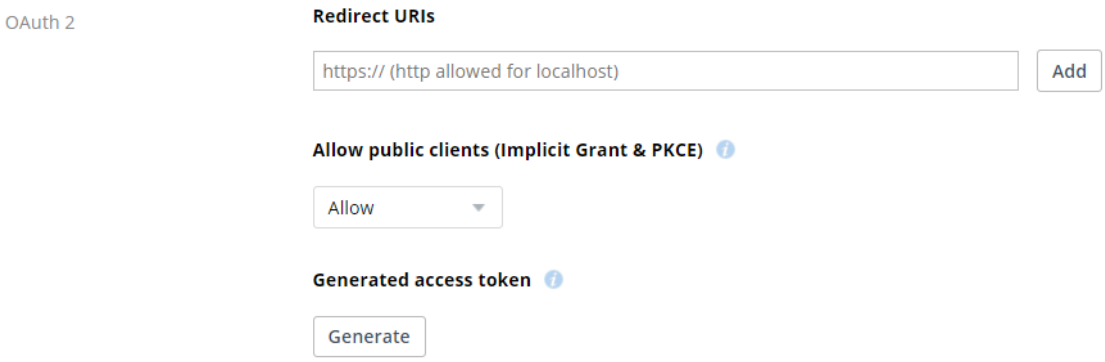

Рис. 2 – Генерация токена для доступа

Затем необходимо было разрешить доступ к приложению, текущий лимит установлен в 500 разработчиков.

Перейдем к разработке своего сервиса. Чтобы иметь возможность использовать в своём приложении API Dropbox, необходимо в gradle:build в dependencies выставить следующую настройку:

```
implementation 'com.dropbox.core:dropbox-core-sdk:3.1.3'
```

Для получения доступа был создан класс DropboxClient, который инициализирует пользователя сгенерированным токеном, а также имеет функцию для получения этого пользователя.

Для получения доступа к контактам, интернету и SD-карте были использованы следующие разрешения:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
```

А также эти разрешения проверяются функциями showContacts, onRequestPermissionsResult, checkPermissionGranted, requestPermission в MainActivity.

Для получения контактов была использована функция getContentResolver. Она возвращает курсор, с помощью которого совершается запрос в базу данных с контактами. Далее идём итератором по колонкам имени и номера. Затем сохраняем их в список, который будет сохранен .txt файлом на SD-карту.

Для того чтобы сделать новый Activity работал как сервис, необходимо выставить в AndroidManifest.xml соответствующую настройку:

```
<service android:name=".mine_service"
    android:enabled="true"
    android:exported="true">
</service>
```

Сервис запускается и останавливается из MainActivity функциями startService и stopService соответственно. Для того чтобы сервис периодически опрашивал контакты и отсылал данные в Dropbox, был создан поток класса Runnable, который перестанет это делать, если будет вызван stopService, который в свою очередь вызовет onDestroy, где поток и сервис будет остановлен.

Рис. 3. – Старт и остановка сервиса

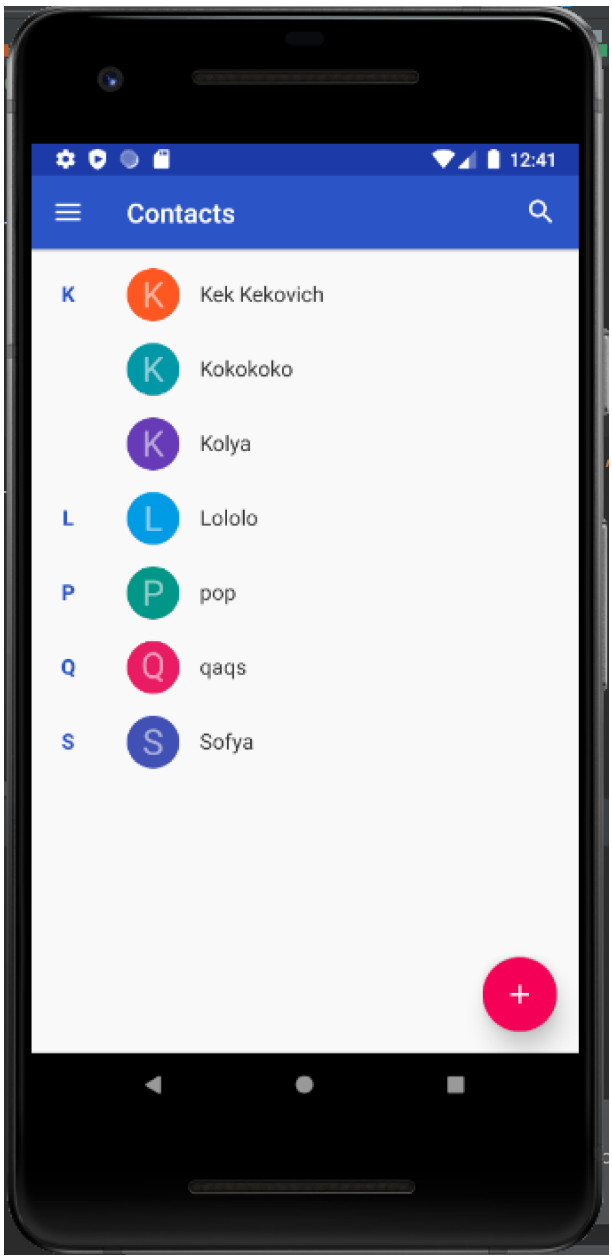

Рис. 4. – Список контактов на телефоне

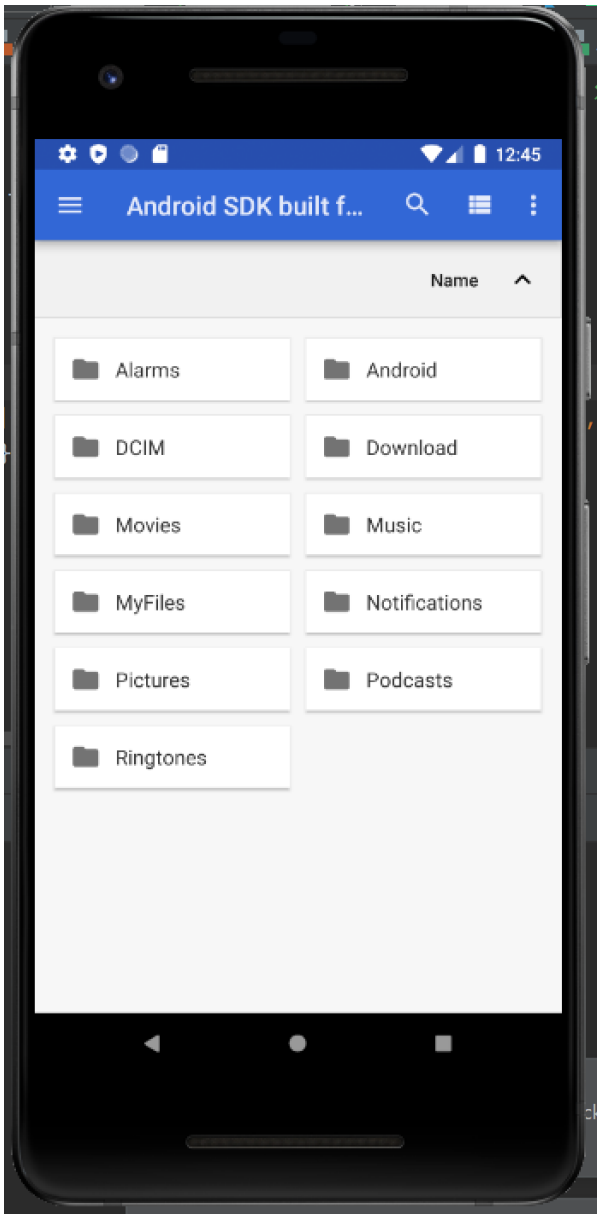

Рис. 5. – Место хранения файла с контактами

Рис. 6. – Файл Contacts.txt

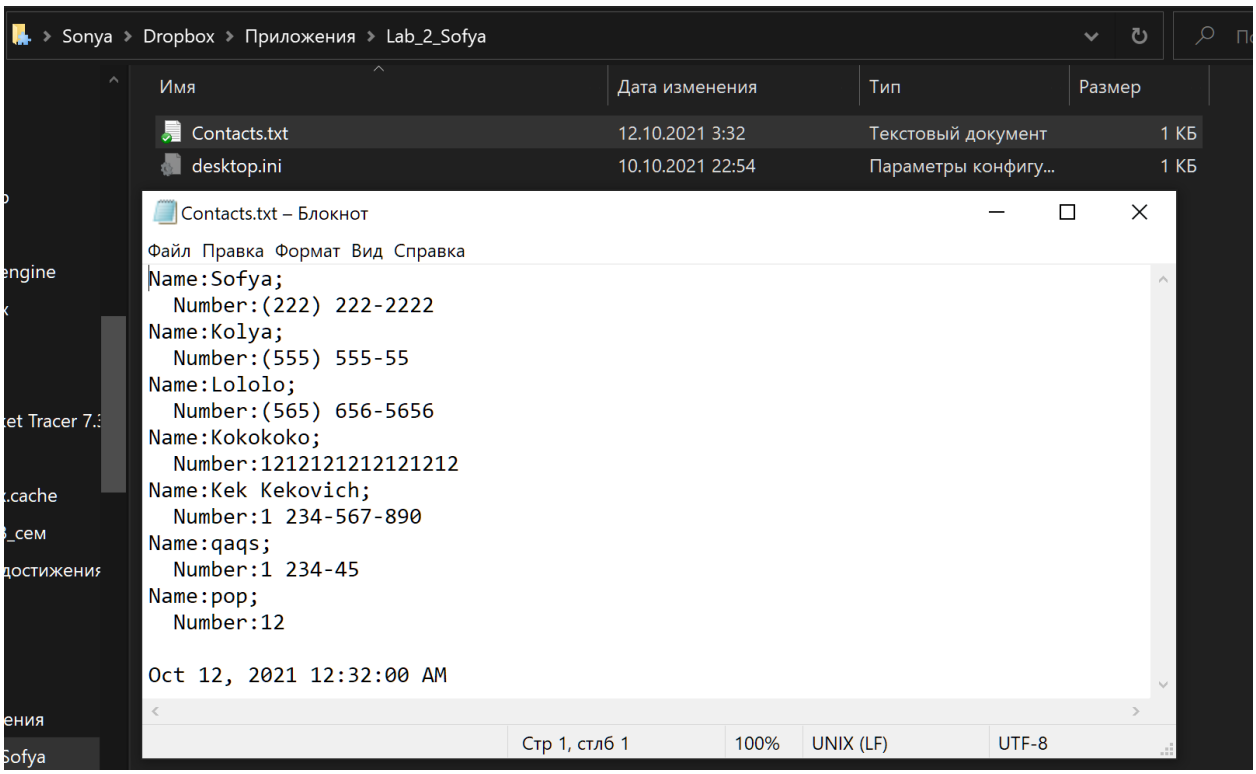

Рис. 7. – Файл, полученный в папке проекта в Dropbox

4. Вывод

В данной лабораторной работе был создан сервис, который работает в фоновом режиме и отсылает в Dropbox список контактов. Для этого был освоен интерфейс для работы с API Dropbox, создание собственного приложения на AppConsole. А также принцип работы сервисов на Android.

Приложение

DropboxClient.java

```
package com.example.imageuploaddropbox;

import com.dropbox.core.DbxRequestConfig;
import com.dropbox.core.v2.DbxClientV2;

class DropboxClient {
    private static DbxClientV2 sDbxClient;

    public static void init(String accessToken) {
        if (sDbxClient == null) {
            DbxRequestConfig requestConfig =
DbxRequestConfig.newBuilder("BookShare").build();
            sDbxClient = new DbxClientV2(requestConfig, accessToken);
        }
    }

    public static DbxClientV2 getClient() {
        if (sDbxClient == null) {
            throw new IllegalStateException("Client not initialized.");
        }
        return sDbxClient;
    }
}
```

MainActivity.java

```
package com.example.imageuploaddropbox;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity
{
    private static final int PERMISSIONS_REQUEST_READ_CONTACTS = 100;

    Button Start_service;
    Button Stop_service;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        checkPermissionGranted();
        showContacts();
        Start_service = findViewById(R.id.Start);
        Stop_service = findViewById(R.id.stop);

        Start_service.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                startService(new Intent(MainActivity.this,
mine_service.class));
                finish();
            }
        });

        Stop_service.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                stopService(new Intent(MainActivity.this,
mine_service.class));
                Toast.makeText(getApplicationContext(), "SERVICE Successfully
STOPPED.", Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void showContacts() {
        // Check the SDK version and whether the permission is already
granted or not.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M &&
checkSelfPermission(Manifest.permission.READ_CONTACTS) !=
PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new
String[]{Manifest.permission.READ_CONTACTS},
PERMISSIONS_REQUEST_READ_CONTACTS);
            //After this point you wait for callback in
onRequestPermissionsResult(int, String[], int[]) overriden method
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
        if (requestCode == PERMISSIONS_REQUEST_READ_CONTACTS) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                showContacts(); // Permission is granted
            } else {
                Toast.makeText(this, "Until you grant the permission, we
canot display the names", Toast.LENGTH_SHORT).show();
            }
        }
    }

    private void checkPermissionGranted() {
        if(( ActivityCompat.checkSelfPermission(getApplicationContext() ,
Manifest.permission.READ_EXTERNAL_STORAGE )
                != PackageManager.PERMISSION_GRANTED ) && (
ActivityCompat.checkSelfPermission(getApplicationContext() ,
Manifest.permission.WRITE_EXTERNAL_STORAGE )
                != PackageManager.PERMISSION_GRANTED )) {
            Toast.makeText(this, "Permission Not Granted",
Toast.LENGTH_SHORT).show();
           requestPermission();
        }
        else {
            Toast.makeText(this, "Permission Granted",
Toast.LENGTH_SHORT).show();

        }
    }

    private void requestPermission() {
```

```
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE,
                        Manifest.permission.WRITE_EXTERNAL_STORAGE} ,123);
    }
}
```

mine_service.java

```
package com.example.imageuploaddropbox;

import androidx.annotation.Nullable;

import android.content.ContentResolver;
import android.database.Cursor;
import android.app.Service;
import android.content.Intent;
import android.os.Environment;
import android.os.IBinder;
import android.provider.ContactsContract;
import android.util.Log;
import android.widget.Toast;

import com.dropbox.core.v2.files.FileMetadata;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.util.Date;

public class mine_service extends Service {

    private String ACCESS_TOKEN = "";
    public static String TEXT1 = "";

    final String DIR_SD = "MyFiles";
    final String FILENAME_SD = "Contacts.txt";

    String uri_string;
    String mPath;

    final String LOG_TAG = "myLogs";

    MyRunnable myRunnable = new MyRunnable();

    public class MyRunnable implements Runnable
    {
        private boolean doStop = false;
        public synchronized void doStop() {
            this.doStop = true;
        }
        private synchronized boolean keepRunning() {
            return this.doStop == false;
        }

        @Override
        public void run() {
            while(keepRunning())
            {
                // keep doing what this thread should do.
```

```
                System.out.println("Running");
                getContactList();
                uploadFile(uri_string);
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(LOG_TAG, "onCreate");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        super.onStartCommand(intent, flags, startId);
        Log.d(LOG_TAG, "onStartCommand");
        DropboxClient.init(ACCESS_TOKEN);

        Thread thread = new Thread(myRunnable);

        thread.start();
        return Service.START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        try
        {
            stopSelf();
            myRunnable.doStop();

        } catch (Exception e)
        {
            e.printStackTrace();
            stopSelf();
        }
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        Log.d(LOG_TAG, "onBind");
        return null;
    }

    private void uploadFile(String uri_string) {
        new UploadFileTask(this, DropboxClient.getClient(), new
UploadFileTask.Callback() {
            @Override
            public void onUploadComplete(FileMetadata result) {
                String message = result.getName() + " size " +
result.getSize() + " modified " +

DateFormat.getDateTimeInstance().format(result.getClientModified());
                Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT)
                        .show();
```

```
                Toast.makeText(getApplicationContext(), "Successfully
Uploaded.", Toast.LENGTH_SHORT)
                        .show();
            }

            @Override
            public void onError(Exception e) {
                Log.e("ERROR ", "Failed to upload file.", e);
            }
        }).execute(uri_string, mPath);
    }

    public void getContactList() {
        TEXT1 = "";

        ContentResolver cr = getContentResolver();
        Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
                null, null, null, null);

        //Запускаем цикл обработчик для каждого контакта:
        if ((cur != null ? cur.getCount() : 0) > 0) {
            while (cur != null && cur.moveToNext()) {
                String id =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts._ID));
                String name =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));

                if
(cur.getInt(cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER)) >
0) {
                    Cursor pCur = cr.query(

ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                            null,
                            ContactsContract.CommonDataKinds.Phone.CONTACT_ID
+ " = ?",
                            new String[]{id}, null);

                    while (pCur.moveToNext()) {
                        String phoneNo =
pCur.getString(pCur.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUM
BER));
                        Log.i("Srv", "Name: " + name);
                        TEXT1 += "Name:" + name + "; " + "\n";
                        Log.i("Srv", "Phone Number: " + phoneNo);
                        TEXT1 += "  Number:"+ phoneNo + "\n";
                    }
                    pCur.close();
                }
            }
        }
        if(cur!=null){
            cur.close();
        }

        String currentDateTimeString =
DateFormat.getDateTimeInstance().format(new Date());
        TEXT1 += "\n" + currentDateTimeString;
        writeFileSD();
    }

    void writeFileSD() {
        // проверяем доступность SD
```

```
        if
(!Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
            Log.d(LOG_TAG, "SD-карта не доступна: " +
Environment.getExternalStorageState());
            return;
        }

        File sdPath = Environment.getExternalStorageDirectory();  // получаем
путь к SD
        sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);  //
добавляем свой каталог к пути
        sdPath.mkdirs(); // создаем каталог
        File sdFile = new File(sdPath, FILENAME_SD);  // формируем объект
File, который содержит путь к файлу
        try {
            BufferedWriter bw = new BufferedWriter(new FileWriter(sdFile));
// открываем поток для записи
            bw.write(TEXT1);  // пишем данные
            bw.close();  // закрываем поток
            Log.d(LOG_TAG, "Файл записан на SD: " +
sdFile.getAbsolutePath());
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        uri_string=sdPath.toString();
        mPath=sdFile.toString();
    }
}
```

UploadFileTask.java

```
package com.example.imageuploaddropbox;

import android.content.Context;
import android.os.AsyncTask;
import android.util.Log;

import com.dropbox.core.DbxException;
import com.dropbox.core.v2.DbxClientV2;
import com.dropbox.core.v2.files.FileMetadata;
import com.dropbox.core.v2.files.WriteMode;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

public class UploadFileTask extends AsyncTask<String , Void , FileMetadata> {
    private final DbxClientV2 mDbxClient;
    private final Callback mCallback;

    public interface Callback {
        void onUploadComplete(FileMetadata result);
        void onError(Exception e);
    }

    UploadFileTask(Context context, DbxClientV2 dbxClient, Callback callback)
{
        mDbxClient = dbxClient;
        mCallback = callback;
```

```
    }

    @Override
    protected void onPostExecute(FileMetadata result) {
        super.onPostExecute(result);
       if (result == null) {
            mCallback.onError(null);
        } else {
            mCallback.onUploadComplete(result);
        }
    }

    @Override
    protected FileMetadata doInBackground(String... strings) {
        File localFile = new
File("/storage/emulated/0/MyFiles/Contacts.txt");

        if (localFile != null) {
            // Note - this is not ensuring the name is a valid dropbox file
name
            String remoteFileName = localFile.getName();

            try  {
                InputStream inputStream = new FileInputStream(localFile);
                return mDbxClient.files().uploadBuilder(  "/" +
remoteFileName)
                        .withMode(WriteMode.OVERWRITE)
                        .uploadAndFinish(inputStream);
            } catch (DbxException | IOException e) {
                Log.e("Exception" , e.getLocalizedMessage());
            }
        }
        return null;
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.imageuploaddropbox">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".mine_service"
            android:enabled="true"
            android:exported="true">
        </service>

    </application>

</manifest>
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:gravity="center">

    <Button
        android:id="@+id/Start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Service" />

    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop Service" />
</LinearLayout>
```

Activity_mine_service.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".mine_service">

</androidx.constraintlayout.widget.ConstraintLayout>
```