1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. **Институт Кибербезопасности и Защиты Информации**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

«**Разработка сервиса для ОС Android**»

1. по дисциплине «Введение в мобильные технологии и средства связи»
2. Выполнил
3. студент гр. 4851003/80802 Сошнев М.Д.

<*подпись*>

1. Преподаватель Мясников А.В.

<*подпись*>

1. Санкт-Петербург
2. 2021

# Цель работы

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

# Задача

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:

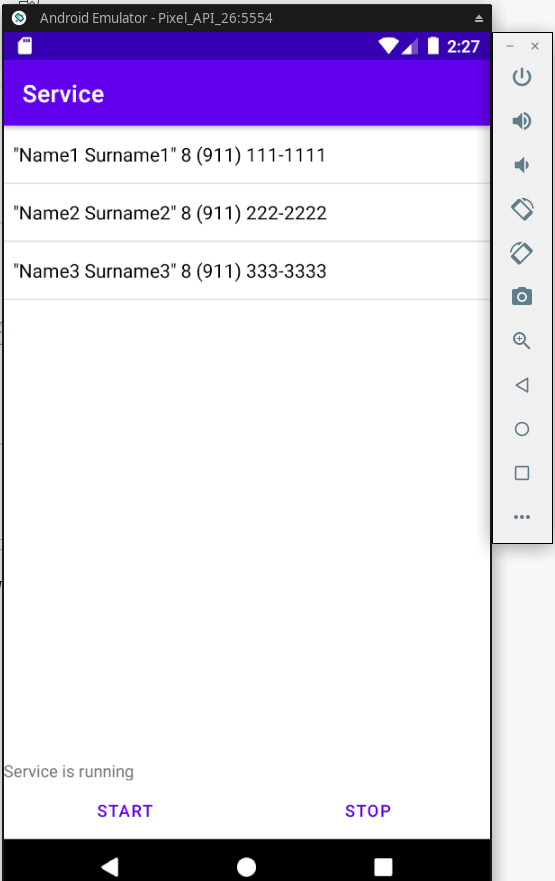
* Выгрузка информации должна осуществляться при наличие любого подключения к сети Интернет и при любом состоянии телефона (включен/выключен дисплей);
* Фоновая работа должна происходить в отдельном потоке ( не в UI thread’е).

Вариант функционала: сбор контактов, способ выгрузки: по протоколу HTTP через POST запрос с написанием серверной части.

# ХОД РАБОТЫ

## **Разработка** клиентской части

Клиентской стороной в данной работе является мобильное приложение под Android реализующее сервер. Приложение было написано на языке Java. Графический интерфейс выглядит следующим образом:

Рисунок 1 — графический интерфейс приложения

Кнопка start — запускает сервис (предварительно проверив, не запущен ли он уже), кнопка stop — останавливает сервис (предварительно проверив, работает ли он в данный момент). Информация о том, запущен ли сервис отображается над кнопками в виде TextView. Также, на главном экране имеется ListView, который отображает последнюю информацию о контактах, которая была отправлена на сервис. Для отправки данных на сервер сервис использует HTTP протокол и метод POST, то есть передаваемая информация записывается в тело запроса следующим форматом:

id=[1,2,3]&name=[name1,name2,name3]&phone=[phone1,phone2,phone3]

Сканирование контактов происходит с помощью интерфейса Cursor.

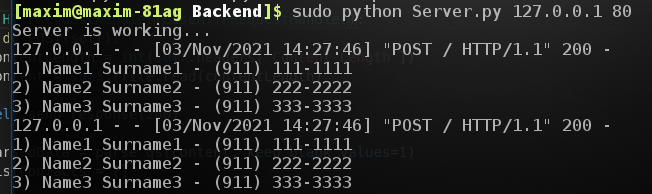
Также в приложении использовались следующие разрешения:

Рисунок 2 — использованные разрешения

Разрешение INTERNET необходимо для отправки POST запросов, разрешение READ\_CONTACTS необходимо для чтения контактов телефонной книги

## **Разработка** серверной части

Сервер разрабатывался на языке Python и представляет из себя небольшой скрипт. Из библиотеки http.server был унаследован класс BaseHTTPRequestHandler и в нем переопределен метод do\_POST. Данный метод срабатывает, когда сервер будет ловить POST-запросы. В данном методе мы десериализуем полученные параметры и выводим в консоль. Далее отвечаем клиенту кодом 200 (код успешной отработки). При запуске сервера надо параметрами указывать ip-адресс и порт, который будем прослушивать (для http это 80). Также для прослушивания портов необходимы root-права (сервер разрабатывался на ОС Linux). Выглядит это следующим образом:

Рисунок 3 — работа сервера, вывод контактов

# Выводы

В ходе работы был получен опыт в написании сервисов под ОС Android и был изучен протокол прикладного уровня HTTP на примере POST-запросов. Было написано мобильное приложение под ОС Android, осуществляющее сбор контактов и выгрузку их на сервер в background-thread’e через POST запросы. Также был написан сервер, принимающий эти запросы и выводящий список контактов на экран.

# Приложение А

**Клиентская часть**

package com.example.service;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;

import android.Manifest;

import android.annotation.SuppressLint;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.os.Bundle;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import android.widget.TextView;

import java.util.ArrayList;

import java.util.List;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

private TextView textLabel;

private ArrayAdapter<String> adapter;

public static volatile List<String> lastContacts;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

lastContacts = new ArrayList<>(10);

adapter = new ArrayAdapter<>(this, R.layout.support\_simple\_spinner\_dropdown\_item,

lastContacts);

((ListView)findViewById(R.id.list\_view)).setAdapter(adapter);

// Check permission

if (ContextCompat.checkSelfPermission(this,

Manifest.permission.READ\_CONTACTS) !=

PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this,

new String[]{Manifest.permission.READ\_CONTACTS}, 10);

}

findViewById(R.id.button\_start\_service).setOnClickListener(this);

findViewById(R.id.button\_stop\_service).setOnClickListener(this);

textLabel = (TextView)findViewById(R.id.text\_label);

}

@SuppressLint({"NonConstantResourceId", "SetTextI18n"})

@Override

public void onClick(View v) {

Intent intent = new Intent(this, BackupService.class);

switch (v.getId()) {

case R.id.button\_start\_service:

if (!BackupService.isRunning()) {

startService(intent);

textLabel.setText("Service is running");

}

else {

textLabel.setText("Service is already running");

}

break;

case R.id.button\_stop\_service:

if (BackupService.isRunning()) {

stopService(intent);

textLabel.setText("Service is not running");

}

else {

textLabel.setText("Service is not running yet");

}

break;

}

adapter.notifyDataSetChanged();

}

}

package com.example.service;

import android.annotation.SuppressLint;

import android.app.IntentService;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.provider.ContactsContract;

import android.util.Log;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.lang.reflect.Field;

import java.net.ConnectException;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

abstract class ServiceCheckRunning extends IntentService {

public ServiceCheckRunning() {

super("Service");

}

protected static volatile boolean isServiceRunning = false;

public static boolean isRunning() {

return isServiceRunning;

}

@Override

protected void onHandleIntent(Intent intent) {

isServiceRunning = true;

}

@Override

public void onDestroy() {

isServiceRunning = false;

super.onDestroy();

}

}

class ContactReader {

static class Contact {

public String id;

public String name = "";

public String phone = "";

}

@SuppressLint("Range")

private String getParamFromCursor(Cursor cursor, String param) {

return cursor.getString(cursor.getColumnIndex(param));

}

public ArrayList<Contact> readContacts(Context context) {

ArrayList<Contact> contactsList = new ArrayList<>(10);

Cursor cursor = context.getContentResolver().query(ContactsContract.Contacts.CONTENT\_URI,

null, null, null, null);

if (cursor.getCount() > 0) {

while(cursor.moveToNext()) {

Contact contact = new Contact();

contact.id = getParamFromCursor(cursor, ContactsContract.Contacts.\_ID);

contact.name = getParamFromCursor(cursor, ContactsContract.Contacts.DISPLAY\_NAME);

int hasPhone = Integer.parseInt(

getParamFromCursor(cursor, ContactsContract.Contacts.HAS\_PHONE\_NUMBER));

if (hasPhone > 0) {

// extract phone number

Cursor cursor1 = context.getContentResolver().query(

ContactsContract.CommonDataKinds.Phone.CONTENT\_URI, null,

ContactsContract.CommonDataKinds.Phone.CONTACT\_ID + " = ?",

new String[]{contact.id},null);

while(cursor1.moveToNext()) {

contact.phone = getParamFromCursor(cursor1,

ContactsContract.CommonDataKinds.Phone.NUMBER);

}

cursor1.close();

}

contactsList.add(contact);

}

}

cursor.close();

return contactsList;

}

}

public class BackupService extends ServiceCheckRunning {

private final String DOMAIN = "http://10.0.2.2";

private final ContactReader contactReader = new ContactReader();

@Override

protected void onHandleIntent(Intent intent) {

super.onHandleIntent(intent);

Log.d("onHandleIntent", "isServiceRunning=" + isServiceRunning);

// Payload

System.out.println("Read contacts...");

List<ContactReader.Contact> contactsList = contactReader.readContacts(this);

for (ContactReader.Contact contact : contactsList) {

System.out.println("Contact: " + contact.id + " " +

contact.name + " " + contact.phone);

}

System.out.println("Send to server");

try {

sendToServer(contactsList);

} catch (ConnectException ex) {

System.out.println("Connection exception");

return;

} catch (IOException ex) {

ex.printStackTrace();

}

List<String> contactsListString = MainActivity.lastContacts;

contactsListString.clear();

for (ContactReader.Contact contact : contactsList) {

contactsListString.add("\"" + contact.name + "\" 8 " + contact.phone);

}

}

private String makePostData(final List<ContactReader.Contact> contacts) {

Map<String, List<String>> map = new LinkedHashMap<>();

try {

for (ContactReader.Contact contact : contacts) {

Field[] fields = contact.getClass().getDeclaredFields();

for (Field field : fields) {

String key = field.getName();

String value = (String) field.get(contact);

List<String> list = map.get(key);

if (list == null) {

list = new ArrayList<>();

list.add(value);

map.put(key, list);

} else {

list.add(value);

}

}

}

} catch (IllegalAccessException e) {

e.printStackTrace();

}

StringBuilder postData = new StringBuilder();

for (Map.Entry<String, List<String>> param : map.entrySet()) {

postData.append(param.getKey());

postData.append("=");

postData.append(param.getValue());

postData.append("&");

}

if (postData.charAt(postData.length() - 1) == '&') {

postData.deleteCharAt(postData.length() - 1);

}

return postData.toString();

}

private void sendToServer(final List<ContactReader.Contact> contacts) throws IOException {

String postData = makePostData(contacts);

// Make connection

URL url = new URL(DOMAIN);

HttpURLConnection connection = (HttpURLConnection)url.openConnection();

connection.setDoOutput(true);

connection.setDoInput(true);

connection.setRequestMethod("POST");

connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded; " +

"charset=UTF-8");

connection.setRequestProperty("Content-Length", String.valueOf(postData.length()));

connection.connect();

//Send request

DataOutputStream wr = new DataOutputStream(connection.getOutputStream());

wr.writeBytes(postData);

wr.flush();

wr.close ();

System.out.println("send request");

BufferedReader rd = new BufferedReader(new InputStreamReader(connection.getInputStream()));

String line;

while ((line = rd.readLine()) != null) {

System.out.println("Line: " + line);

}

wr.close();

rd.close();

}

}

# Приложение B

**Серверная часть**

from http.server import HTTPServer, BaseHTTPRequestHandler

from urllib.parse import parse\_qs

from sys import argv

import numpy as np

class HTTPRequestHandler(BaseHTTPRequestHandler):

def do\_POST(self):

contentLength = int(self.headers['Content-Length'])

content = self.rfile.read(contentLength)

self.send\_response(200)

parsedData = parse\_qs(content, keep\_blank\_values=1)

listContacts = []

for key, listValues in parsedData.items():

if key == 'size':

continue

listValues = str(listValues[0])

if listValues[:3] == 'b\'[':

listValues = listValues[3:]

if listValues[-2:] == ']\'':

listValues = listValues[:-2]

listValues = listValues.split(', ')

for i, v in enumerate(listValues):

if len(listContacts) > i:

listContacts[i].append(v)

else:

listContacts.append([v])

for contact in listContacts:

print('{}) {} - {}'.format(contact[0], contact[1], contact[2]))

#send header first

self.send\_header('Content-type','text-html')

self.end\_headers()

if \_\_name\_\_ == '\_\_main\_\_':

if len(argv) < 3:

print('Usage:\n\tpython {} <address> <port>'.format(argv[0]))

exit(0)

address = (argv[1], int(argv[2]))

server = HTTPServer(address, HTTPRequestHandler)

print('Server is working...')

try:

server.serve\_forever()

except KeyboardInterrupt:

exit(0)

exit(0)