**Ministerul Educatiei Republicii Moldova**

**Universitatea Tehnica a Moldovei**

**Departamentul ISA**

**Raport**

La lucrearea de laborator № 2

Disciplina “Programarea aplicatiilor mobile”

**Tema:** Organiser Mobile Application (DAILY PLANNER)

A efectuat: st. gr. TI-153 Suruceanu V.

A controlat: lect. univ. C. Rusu

Chisinau 2017

**Scopul Lucrarii:** De realizat aplicație tip Organiser pe platforma aleasă.

La formarea și proiectarea aplicațiilor sunt introduse puncte stricte, ce sunt obligatorii de respectat. La discreția personală rămâne designul aplicației și API-ul/Framework-ul în care va fi dezvoltată lucrarea de laborator #2. Componentele și structura fiecărei Activități va fi descrisă mai jos.

**Sarcina:**

UI Components

Aplicația va conține minim 3 Activități de baza care vor fi numerotate în lucrare sub forma:

* MainActivity (structura/componente)

Text View (default)

Buttons (Add/Remove/Update)

Căutare (caută conform cuvintelor cheie)

* AddActivity

Data/Time controller

Info TextBox

Buttons și altele (la discreție conform specificului aplicației)

* UpdateActivity - practic e una și aceeași activitate de la Add, doar ca completata deja.

Datele operaționale din interiorul aplicației vor fi stocate în fișier/e XML forma cărora rămâne la discreția personală. (cuvinte cheie, XML Serialization).

**Logical/Operational Component**

Toate evenimentele și acțiunile de notificare/semnalizare (sonore/vizuale) intreprinse în Organiser

vor fi tratate într-un serviciu aparte, care logic funcțional va extrage datele din acel **fișier XML**.

La discreția fiecărui rămâne tema/funcționalul aplicației. Exemplu cum se va modifica tema(specificată in raport) -

**Organizer pentru viata de zi cu zi**

**Mersul lucrarii:**

1. Am creat 4 activitati (Main, Add, Update, ShowCalendar)
2. Am creat o baza de date in care va fi salvate datele
3. Scris modelul pe care voi folosi (Reminder)
4. Cu ajutorul PendingIntent si AlarmManager trimit pornesc serviciul de background
5. Citirea datelor din baza de date si afisarea in aplicatie(MainActivity)
6. Serviciul porneste prin intermediul unui receiver (Masuri de securitate)
7. Datele sunt plasate intro stiva si serviciu opreste in momentul in care stiva este goala

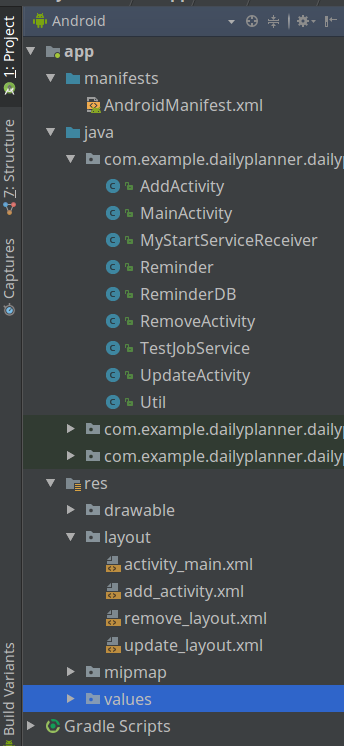


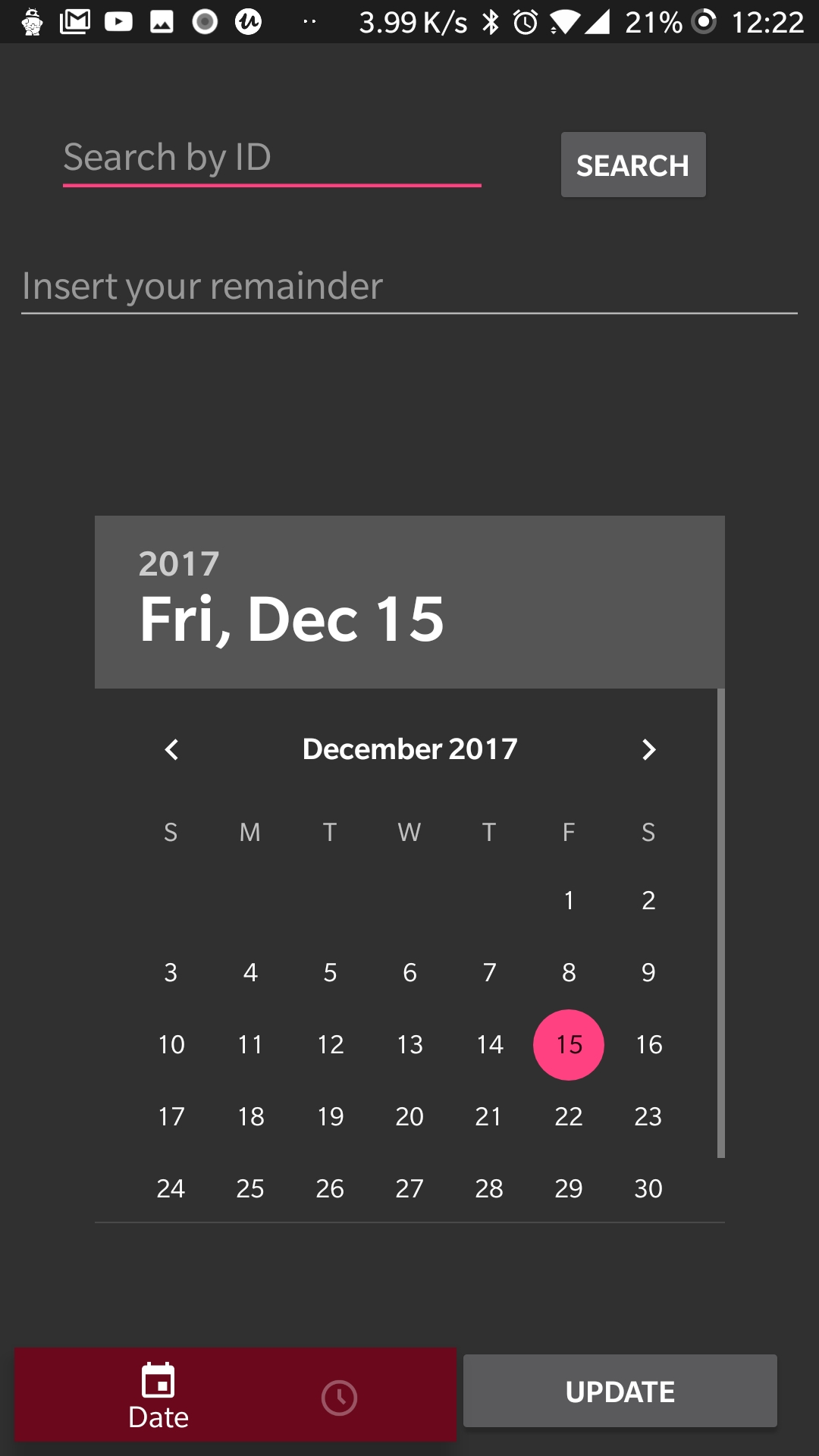
Figura 1 – Structura proiectului

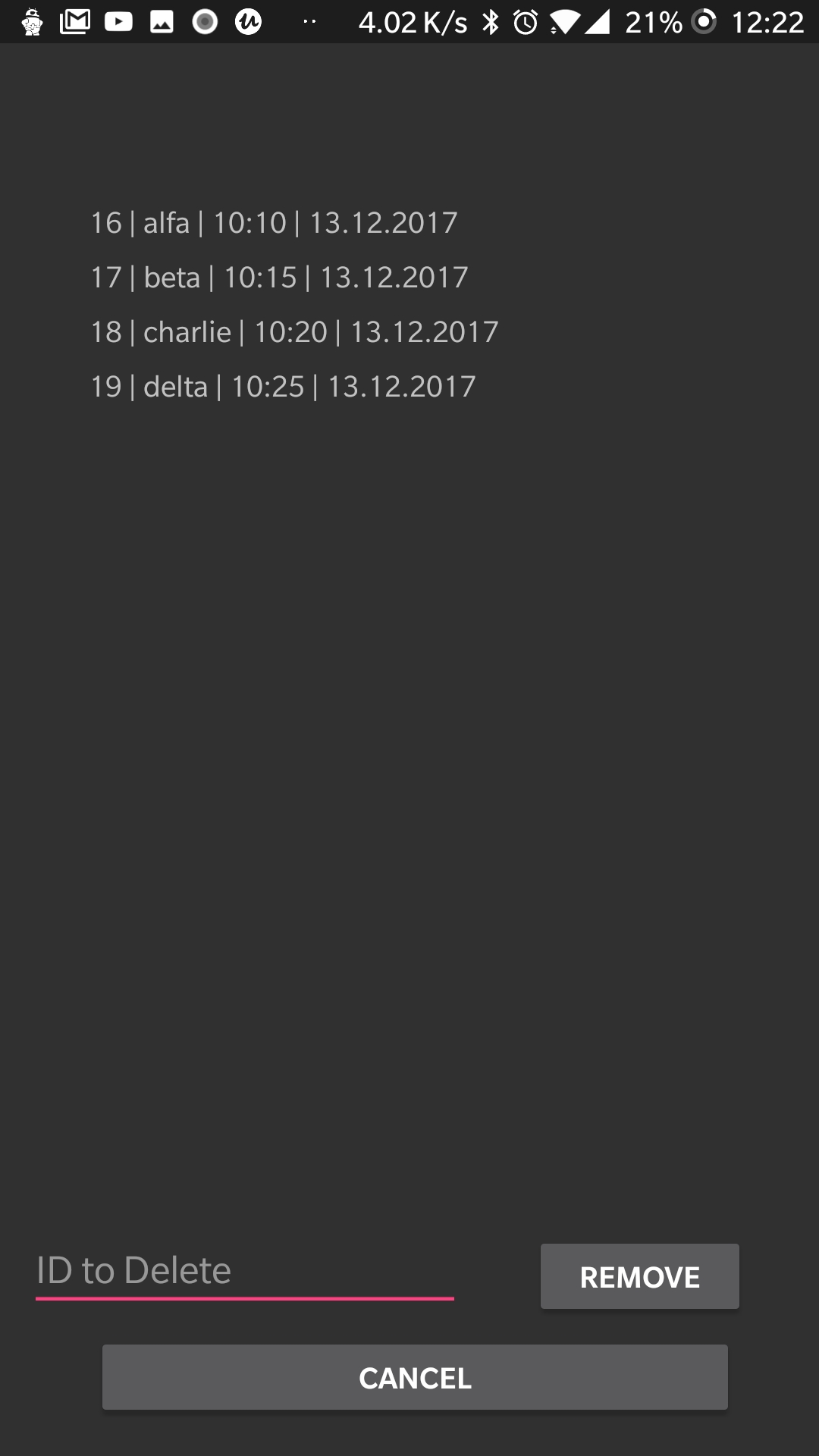


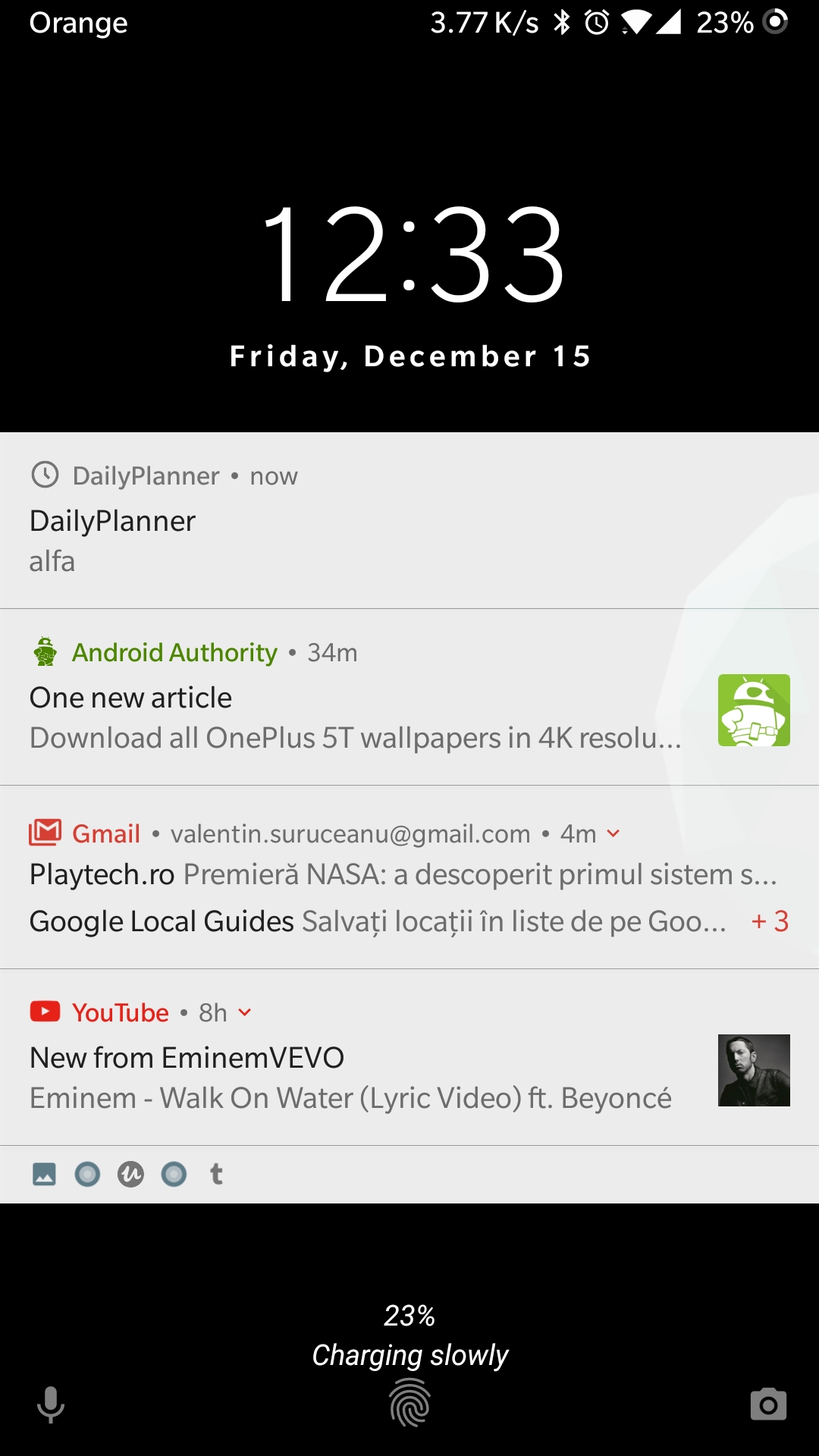
Figura 2 – Activitatea Main



Figura 3 – Activitatea Add

Figura 4 – Actualizarea unui eveniment

Figura 5 – Stergerea unui eveniment

Figura 6 – Notificarea venita din serviciu

**Concluzie**

In urma realizarii laboratorului numarul 2, am aflat metodele de creare a serviciilor pe Android, apelarea serviciilor interne (Alarm, etc..) integrarea valorilor în baze de date, crearea de BroadcastReceiver, crearea de NotificationService

Anexa:

package com.example.dailyplanner.dailyplanner;

/\*\*

\* Created by valentin on 12.12.2017.

\*/

import android.annotation.TargetApi;

import android.app.AlarmManager;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.app.job.JobParameters;

import android.app.job.JobService;

import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Build;

import android.support.annotation.RequiresApi;

import android.support.v4.app.NotificationCompat;

import android.support.v4.app.TaskStackBuilder;

import android.util.Log;

import java.util.List;

import java.util.Stack;

/\*\*

\* JobService to be scheduled by the JobScheduler.

\* start another service

\*/

public class TestJobService extends JobService {

private static final String TAG = "SyncService";

private static final int NOTIFICATION\_ID = 3;

public static Reminder reminder;

public static Stack<Reminder> reminderStack;

public static boolean FIRSTRUN = false;

public static long currentTimerInteval, nextTimerInterval;

String message;

@Override

public boolean onStartJob(JobParameters params) {

message=findMessageByTime(currentTimerInteval);

Log.d("SERVICE", "STARTED");

if (!reminderStack.empty()) {

reminder = reminderStack.pop();

currentTimerInteval = reminder.getDateTime();

}

while (currentTimerInteval < System.currentTimeMillis()) {

if(currentTimerInteval > System.currentTimeMillis()) break;

if (!reminderStack.empty()) {

Log.i("currentTime",Long.toString(currentTimerInteval));

nextTimerInterval = reminderStack.peek().getDateTime();

currentTimerInteval = nextTimerInterval;

//Log.i("systemTime",Long.toString(System.currentTimeMillis()));

} else {

break;

}

reminder = reminderStack.pop();

}

Log.d("REMINDER", findMessageByTime(currentTimerInteval));

if(FIRSTRUN) {

processStartNotification(message, getResources().getString(R.string.app\_name), R.drawable.ic\_time);

}

else {

FIRSTRUN =true;

}

if (!reminderStack.empty()) {

if (reminderStack.peek().getDateTime() > System.currentTimeMillis()) {

Intent notifyIntent = new Intent(this, MyStartServiceReceiver.class);

PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, notifyIntent, PendingIntent.FLAG\_UPDATE\_CURRENT);

AlarmManager alarmManager = (AlarmManager) this.getSystemService(Context.ALARM\_SERVICE);

alarmManager.set(AlarmManager.RTC\_WAKEUP, currentTimerInteval, pendingIntent);

Log.i("nextTime", Long.toString(currentTimerInteval));

}

}

return true;

}

@Override

public boolean onStopJob(JobParameters params) {

return true;

}

@TargetApi(Build.VERSION\_CODES.O)

@RequiresApi(api = Build.VERSION\_CODES.O)

private void processStartNotification(String message, String title, int iconID) {

// Do something. For example, fetch fresh data from backend to create a rich notification?

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

intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);

PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /\* Request code \*/, intent,

PendingIntent.FLAG\_ONE\_SHOT);

NotificationManager mNotificationManager =

(NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);

// The id of the channel.

String id = "my\_channel\_01";

// The user-visible name of the channel.

CharSequence name = getString(R.string.channel\_name);

// The user-visible description of the channel.

String description = getString(R.string.channel\_description);

int importance = NotificationManager.IMPORTANCE\_HIGH;

NotificationChannel mChannel = new NotificationChannel(id, name, importance);

// Configure the notification channel.

mChannel.setDescription(description);

mChannel.enableLights(true);

// Sets the notification light color for notifications posted to this

// channel, if the device supports this feature.

mChannel.setLightColor(Color.RED);

mChannel.enableVibration(true);

mChannel.setVibrationPattern(new long[]{100, 200, 300, 400, 500, 400, 300, 200, 400});

mNotificationManager.createNotificationChannel(mChannel);

// The id of the channel.

String CHANNEL\_ID = "my\_channel\_01";

NotificationCompat.Builder mBuilder =

new NotificationCompat.Builder(this, CHANNEL\_ID)

.setSmallIcon(iconID)

.setContentTitle(title)

.setContentText(message);

// Creates an explicit intent for an Activity in your app

Intent resultIntent = new Intent(this, MainActivity.class);

// The stack builder object will contain an artificial back stack for the

// started Activity.

// This ensures that navigating backward from the Activity leads out of

// your app to the Home screen.

TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);

// Adds the back stack for the Intent (but not the Intent itself)

stackBuilder.addParentStack(MainActivity.class);

// Adds the Intent that starts the Activity to the top of the stack

stackBuilder.addNextIntent(resultIntent);

PendingIntent resultPendingIntent =

stackBuilder.getPendingIntent(

0,

PendingIntent.FLAG\_UPDATE\_CURRENT

);

mBuilder.setContentIntent(resultPendingIntent);

// mNotificationId is a unique integer your app uses to identify the

// notification. For example, to cancel the notification, you can pass its ID

// number to NotificationManager.cancel().

mNotificationManager.notify(NOTIFICATION\_ID, mBuilder.build());

mBuilder.setContentIntent(pendingIntent);

//mBuilder.setDeleteIntent(MyReceiver.getDeleteIntent(this));

final NotificationManager manager = (NotificationManager) this.getSystemService(Context.NOTIFICATION\_SERVICE);

manager.notify(NOTIFICATION\_ID, mBuilder.build());

}

public String findMessageByTime(long time) {

List<Reminder> reminders = new ReminderDB(this).read();

for (Reminder obj : reminders) {

if (obj.getDateTime() == currentTimerInteval) {

return obj.reminder;

}

}

return "";

}

}