МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ПЕРСПЕКТИВНОЙ ИНЖЕНЕРИИ

**Кафедра прикладной информатики**

Лабораторная работа №10

ПО ДИСЦИПЛИНЕ «Программирование мобильных устройств»

НА ТЕМУ:

«Разрешения»

**Выполнил:**

студент группы ПИН-Б-З-22-1

Гадиян Сергей Гариевич

Ставрополь, 2025 г.

Цель работы: Изучить методы использования аппаратных возможностей устройства.

Формируемые компетенции: ПК-7, ПК-8

Ход работы.

Приложение, которое будет выключать/включать wi-fi по расписанию.

package com.example.lab10;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.NotificationCompat;  
  
import android.annotation.SuppressLint;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.TimePickerDialog;  
import android.content.Context;  
import android.net.wifi.WifiManager;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.VibrationEffect;  
import android.os.Vibrator;  
import android.text.format.DateUtils;  
import android.util.Log;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.EditText;  
import android.widget.ListView;  
import android.widget.TimePicker;  
import android.widget.Toast;  
  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.Iterator;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class MainActivity extends AppCompatActivity {  
 Calendar dateAndTime = Calendar.*getInstance*();  
 Timer timer;  
 int TaskId = 1;  
 EditText startTime;  
 NotificationChannel channel;  
 ArrayList<ToDoTask> tasks = new ArrayList<>();  
 ArrayAdapter<String> taskList;  
 WifiManager wifiManager;  
  
   
 Calendar wifiOnCalendar = Calendar.*getInstance*();  
 Calendar wifiOffCalendar = Calendar.*getInstance*();  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
   
 startTime = findViewById(R.id.*toDoTime*);  
 wifiManager = (WifiManager) getApplicationContext().getSystemService(Context.*WIFI\_SERVICE*);  
  
   
 taskList = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, new ArrayList<String>());  
 ListView tasksListView = findViewById(R.id.*tasksList*);  
 tasksListView.setAdapter(taskList);  
  
   
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = "MainChannel";  
 String description = "Channel for ToDo Tasks";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 channel = new NotificationChannel("ch-1", name, importance);  
 channel.setDescription(description);  
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
  
   
 startTime.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 TimePickerDialog.OnTimeSetListener t = new TimePickerDialog.OnTimeSetListener() {  
 public void onTimeSet(TimePicker view, int hourOfDay, int minute) {  
 dateAndTime.set(Calendar.*HOUR\_OF\_DAY*, hourOfDay);  
 dateAndTime.set(Calendar.*MINUTE*, minute);  
 changeStartTime();  
 }  
 };  
 TimePickerDialog dlg = new TimePickerDialog(MainActivity.this, t,  
 dateAndTime.get(Calendar.*HOUR\_OF\_DAY*),  
 dateAndTime.get(Calendar.*MINUTE*), true);  
 dlg.show();  
 }  
 });  
  
   
 EditText wifiOnTime = findViewById(R.id.*wifiOnTime*);  
 wifiOnTime.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Calendar current = Calendar.*getInstance*();  
 TimePickerDialog timePicker = new TimePickerDialog(MainActivity.this, new TimePickerDialog.OnTimeSetListener() {  
 @Override  
 public void onTimeSet(TimePicker timePicker, int hourOfDay, int minute) {  
 wifiOnCalendar.set(Calendar.*HOUR\_OF\_DAY*, hourOfDay);  
 wifiOnCalendar.set(Calendar.*MINUTE*, minute);  
 wifiOnTime.setText(String.*format*("%02d:%02d", hourOfDay, minute));  
 Log.*d*("WiFi On Time", "Setter: " + hourOfDay + ":" + minute);  
 }  
 }, current.get(Calendar.*HOUR\_OF\_DAY*), current.get(Calendar.*MINUTE*), true);  
 timePicker.show();  
 }  
 });  
  
   
 EditText wifiOffTime = findViewById(R.id.*wifiOffTime*);  
 wifiOffTime.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Calendar current = Calendar.*getInstance*();  
 TimePickerDialog timePicker = new TimePickerDialog(MainActivity.this, new TimePickerDialog.OnTimeSetListener() {  
 @Override  
 public void onTimeSet(TimePicker timePicker, int hourOfDay, int minute) {  
 wifiOffCalendar.set(Calendar.*HOUR\_OF\_DAY*, hourOfDay);  
 wifiOffCalendar.set(Calendar.*MINUTE*, minute);  
 wifiOffTime.setText(String.*format*("%02d:%02d", hourOfDay, minute));  
 Log.*d*("WiFi Off Time", "Setter: " + hourOfDay + ":" + minute);  
 }  
 }, current.get(Calendar.*HOUR\_OF\_DAY*), current.get(Calendar.*MINUTE*), true);  
 timePicker.show();  
 }  
 });  
  
   
 timer = new Timer();  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 try {  
 ArrayList<ToDoTask> tasksToRemove = new ArrayList<>();  
 Date currentTime = Calendar.*getInstance*().getTime();  
 for (ToDoTask t : tasks) {  
 if (currentTime.after(t.ToDoTime)) {  
 t.getNotification();  
 tasksToRemove.add(t);  
 }  
 }  
   
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 tasks.removeAll(tasksToRemove);  
 updateTaskList();  
 }  
 });  
 } catch (Exception e) {  
 Log.*e*("Timer Error", e.getMessage());  
 }  
 }  
 }, 15000, 15000);  
 }  
  
 public void updateTaskClick(View v) {  
 updateTaskList();  
 }  
  
 protected void updateTaskList() {  
 try {  
 ArrayList<String> tasksTitles = new ArrayList<>();  
 for (ToDoTask t : tasks) {  
 tasksTitles.add(t.Title + " - " + t.ToDoTime.toString());  
 }  
 taskList.clear();  
 taskList.addAll(tasksTitles);  
 taskList.notifyDataSetChanged();  
 } catch (Exception e) {  
 Log.*e*("ListView Error", e.getMessage());  
 }  
 }  
  
 protected void changeStartTime() {  
 startTime.setText(DateUtils.*formatDateTime*(this,  
 dateAndTime.getTimeInMillis(),  
 DateUtils.*FORMAT\_ABBREV\_TIME* | DateUtils.*FORMAT\_SHOW\_TIME*));  
 }  
  
 public void addTaskClick(View v) {  
 try {  
 String title = ((EditText) findViewById(R.id.*toDoTitle*)).getText().toString();  
 String description = ((EditText) findViewById(R.id.*toDoDescription*)).getText().toString();  
 String toDoTimeOnString = ((EditText) findViewById(R.id.*toDoTime*)).getText().toString();  
  
 if (title.isEmpty() || description.isEmpty() || toDoTimeOnString.isEmpty()) {  
 Toast.*makeText*(MainActivity.this, "Пожалуйста, заполните все поля!", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");  
 Date toDoTime = formatter.parse(toDoTimeOnString);  
  
   
 Calendar taskCalendar = Calendar.*getInstance*();  
 taskCalendar.set(Calendar.*HOUR\_OF\_DAY*, dateAndTime.get(Calendar.*HOUR\_OF\_DAY*));  
 taskCalendar.set(Calendar.*MINUTE*, dateAndTime.get(Calendar.*MINUTE*));  
 taskCalendar.set(Calendar.*SECOND*, 0);  
 taskCalendar.set(Calendar.*MILLISECOND*, 0);  
 Date currentTime = new Date();  
 if (taskCalendar.getTime().before(currentTime)) {  
 taskCalendar.add(Calendar.*DATE*, 1);   
 }  
  
 ToDoTask newTask = new ToDoTask(TaskId++);  
 newTask.Title = title;  
 newTask.Description = description;  
 newTask.ToDoTime = taskCalendar.getTime();  
 tasks.add(newTask);  
  
   
 ((EditText) findViewById(R.id.*toDoTime*)).setText("");  
 ((EditText) findViewById(R.id.*toDoTitle*)).setText("");  
 ((EditText) findViewById(R.id.*toDoDescription*)).setText("");  
  
 Toast.*makeText*(MainActivity.this, "Задача добавлена!", Toast.*LENGTH\_SHORT*).show();  
 updateTaskList();  
 } catch (Exception e) {  
 Log.*e*("Add Task Error", e.getMessage());  
 Toast.*makeText*(MainActivity.this, "Ошибка при добавлении задачи!", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 public void setWifiSchedule(View v) {  
 try {  
 String wifiOnTimeString = ((EditText) findViewById(R.id.*wifiOnTime*)).getText().toString();  
 String wifiOffTimeString = ((EditText) findViewById(R.id.*wifiOffTime*)).getText().toString();  
  
 if (wifiOnTimeString.isEmpty() || wifiOffTimeString.isEmpty()) {  
 Toast.*makeText*(MainActivity.this, "Пожалуйста, установите оба времени Wi-Fi!", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");  
 Date wifiOnTimeDate = formatter.parse(wifiOnTimeString);  
 Date wifiOffTimeDate = formatter.parse(wifiOffTimeString);  
  
 Calendar current = Calendar.*getInstance*();  
  
   
 Calendar wifiOnCalendar = Calendar.*getInstance*();  
 wifiOnCalendar.set(Calendar.*HOUR\_OF\_DAY*, wifiOnTimeDate.getHours());  
 wifiOnCalendar.set(Calendar.*MINUTE*, wifiOnTimeDate.getMinutes());  
 wifiOnCalendar.set(Calendar.*SECOND*, 0);  
 wifiOnCalendar.set(Calendar.*MILLISECOND*, 0);  
 if (wifiOnCalendar.getTime().before(current.getTime())) {  
 wifiOnCalendar.add(Calendar.*DATE*, 1);  
 }  
  
   
 Calendar wifiOffCalendar = Calendar.*getInstance*();  
 wifiOffCalendar.set(Calendar.*HOUR\_OF\_DAY*, wifiOffTimeDate.getHours());  
 wifiOffCalendar.set(Calendar.*MINUTE*, wifiOffTimeDate.getMinutes());  
 wifiOffCalendar.set(Calendar.*SECOND*, 0);  
 wifiOffCalendar.set(Calendar.*MILLISECOND*, 0);  
 if (wifiOffCalendar.getTime().before(current.getTime())) {  
 wifiOffCalendar.add(Calendar.*DATE*, 1);  
 }  
  
   
 scheduleWifiTask(wifiOnCalendar.getTime(), true);  
  
   
 scheduleWifiTask(wifiOffCalendar.getTime(), false);  
  
 Toast.*makeText*(MainActivity.this, "Расписание Wi-Fi установлено!", Toast.*LENGTH\_SHORT*).show();  
 } catch (Exception e) {  
 Log.*e*("Wi-Fi Schedule Error", e.getMessage());  
 Toast.*makeText*(MainActivity.this, "Ошибка при установке расписания Wi-Fi!", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 private void scheduleWifiTask(Date time, final boolean enable) {  
 TimerTask wifiTask = new TimerTask() {  
 @Override  
 public void run() {  
 wifiManager.setWifiEnabled(enable);  
 String status = enable ? "включен" : "выключен";  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 Toast.*makeText*(MainActivity.this, "Wi-Fi " + status, Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 };  
  
 long delay = time.getTime() - System.*currentTimeMillis*();  
 if (delay < 0) {  
 delay += 24 \* 60 \* 60 \* 1000;   
 }  
 timer.schedule(wifiTask, delay);  
 }  
  
 protected class ToDoTask {  
 public String Title;  
 public String Description;  
 public Date ToDoTime;  
 protected int TaskId;  
  
 public ToDoTask(int TaskId) {  
 this.TaskId = TaskId;  
 }  
  
 public void getNotification() {  
 NotificationCompat.Builder nbuilder = new NotificationCompat.Builder(getApplicationContext(), "ch-1")  
 .setContentTitle(Title)  
 .setContentText(Description)  
 .setSmallIcon(R.drawable.*ic\_launcher\_foreground*)   
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*);  
  
 NotificationManager mgr = (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 mgr.notify(TaskId, nbuilder.build());  
  
 Vibrator v = (Vibrator) getSystemService(Context.*VIBRATOR\_SERVICE*);  
 if (v != null) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 v.vibrate(VibrationEffect.*createOneShot*(400, VibrationEffect.*DEFAULT\_AMPLITUDE*));  
 } else {  
 v.vibrate(400);  
 }  
 }  
 }  
 }  
}

Листинг 1 – MainActivity.java

<RelativeLayout 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">

<LinearLayout

android:id="@+id/wifiSection"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="16dp"

android:layout\_marginTop="16dp">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="8dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Время включения Wi-Fi:"

android:layout\_gravity="center\_vertical"/>

<EditText

android:id="@+id/wifiOnTime"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:hint="ЧЧ:ММ"

android:inputType="time"

android:focusable="false"

android:clickable="true"

android:layout\_marginStart="8dp"/>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="8dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Время выключения Wi-Fi:"

android:layout\_gravity="center\_vertical"/>

<EditText

android:id="@+id/wifiOffTime"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:hint="ЧЧ:ММ"

android:inputType="time"

android:focusable="false"

android:clickable="true"

android:layout\_marginStart="8dp"/>

</LinearLayout>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/setWifiScheduleButton"

android:onClick="setWifiSchedule"

android:text="Установить расписание Wi-Fi"/>

</LinearLayout>

<LinearLayout

android:id="@+id/tasksSection"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="16dp"

android:layout\_below="@id/wifiSection">

<EditText

android:id="@+id/toDoTitle"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Заголовок задачи"

android:layout\_marginBottom="8dp"/>

<EditText

android:id="@+id/toDoDescription"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Описание задачи"

android:layout\_marginBottom="8dp"/>

<EditText

android:id="@+id/toDoTime"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Время выполнения (ЧЧ:ММ)"

android:focusable="false"

android:clickable="true"

android:inputType="time"

android:layout\_marginBottom="8dp"/>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/addTaskButton"

android:onClick="addTaskClick"

android:text="Добавить задачу"

android:layout\_marginBottom="16dp"/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Список задач:"

android:textStyle="bold"

android:layout\_marginBottom="8dp"/>

<ListView

android:id="@+id/tasksList"

android:layout\_width="match\_parent"

android:layout\_height="200dp"

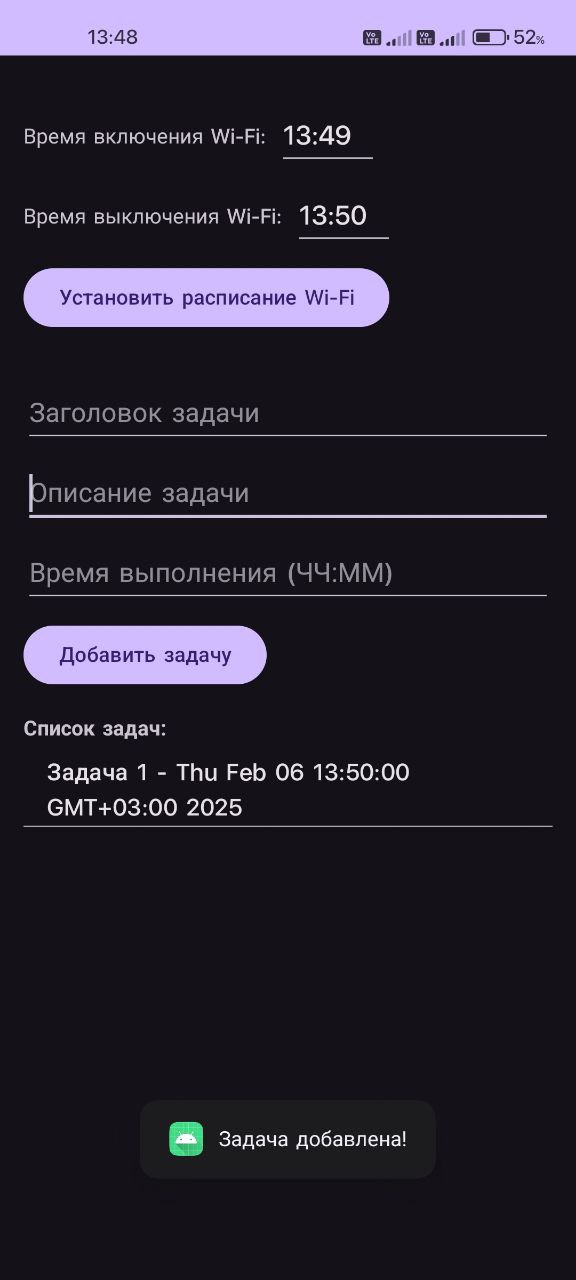
android:divider="@android:color/darker\_gray"

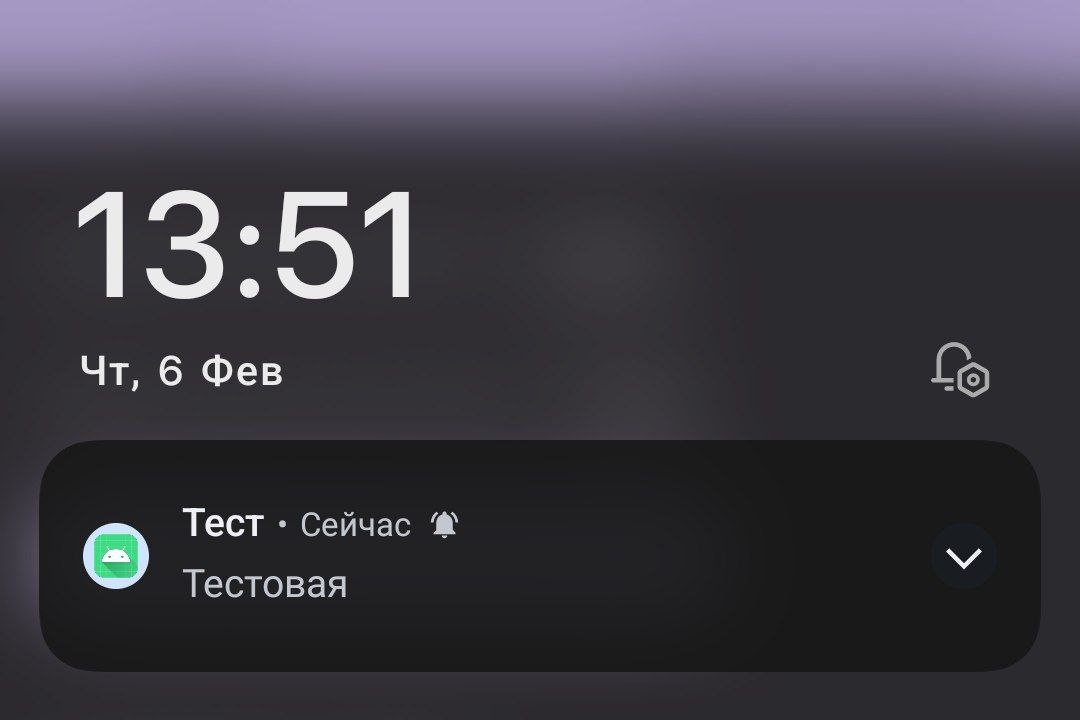
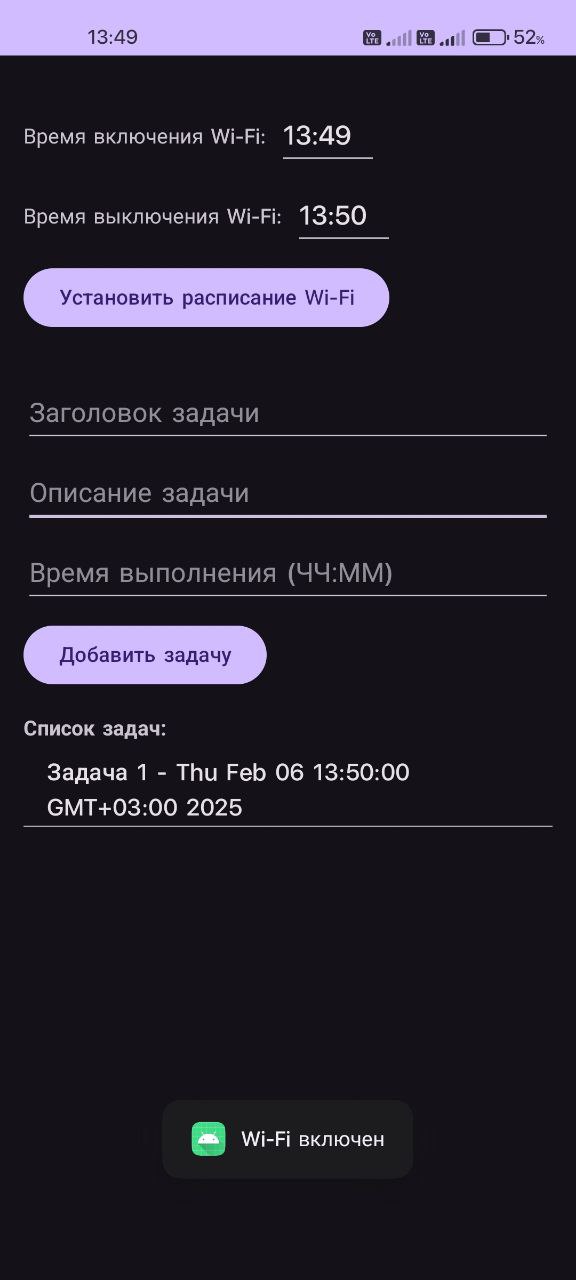
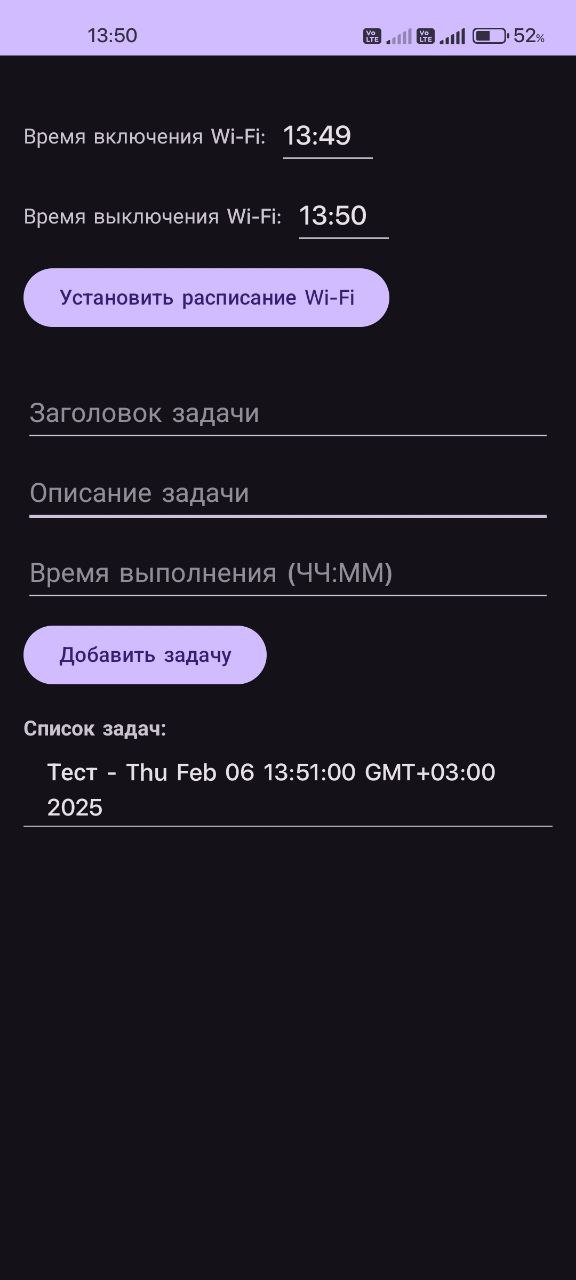
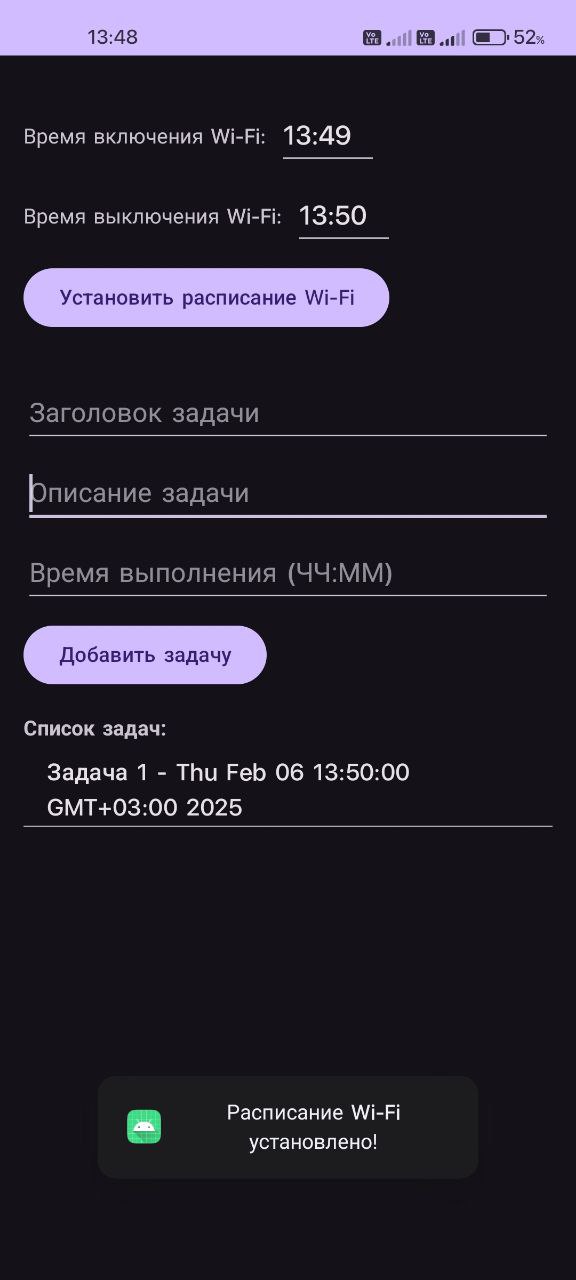
android:dividerHeight="1dp"/>

</LinearLayout>

</RelativeLayout>

Листинг 2 – activity\_main.xml





Вывод: В ходе выполнения работы, мы изучили работу разрешений в приложениях андроид, путем создания различных задач и уведомлений.