TEHNICI DE PROGRAMARE

DOCUMENTATIE: TEMA 6

-MORNING ALARM-

Stundent: Profir Alexandr

Grupa: 30227

Cuprins

1. Obiectivul temei

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

3. Proiectare (diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

4. Implementare si testare

5. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

1. Obiectivul temei

*Enunt: De proiectat si realizat o alarma destinata folosirii sub sistemul de operare Android care va permite operatii ca setarea si resetarea alarmei, alegerea zilelor in care va suna, alegerea orei de semnal de alarma si alegerea modului de resetare a ei.*

Aceasta tema are ca obiectiv crearea unei alarme din cite se observa alarma trebuie sa includa functii simple si sa furnizeze o interata comoda pentru utilizator. Imi propun sa realizez o aplicatie care va furniza o interfata grafica de introducere a datelor pentru setarea alarmei, precum va furniza si interfete grafice pentru stingerea ei, pentru inchiderea , plus la toate voi folosi tehnici pentru a da informatii utilizatorului despre starea curenta a alarmelor in genere.

Imi propun sa realizez aceasta aplicatie folosind instrumentele furnizate de interfata de dezvoltare si sa creez o interfata grafica a aplicatiei placuta si simpla, iar controlul si monitorizarea aplicatiei va fi realizat la fel intrun mod simplu.

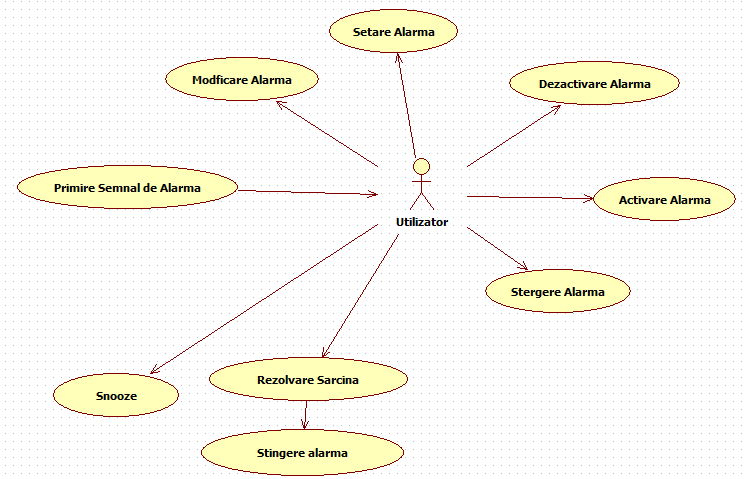
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Analiza problemei o vom face in urmatorii pasi:

1. Avem nevoie de o interfata grafica pentru a putea vedea care alarme sint deja setate si care sunt active sau inactive.
2. Avem nevoie de o interata grafica pentru afisarea ecranului in care se vor seta informatiile despre alarma adica ora la care ea va suna, melodia, modul in care va fi dezativata , in ce zile a saptaminii este activa.
3. Datorita specificului aplicatiei va trebui sa ne facem griji si asupra aspectelor ca activarea telefonului din regimul cu ecran inactiv, activarea ecranului in caz daca este stins si afisarea pe ecran a conditiei de dezactivare a alarmei chiar si in cazul in care telefonul este securizat cu lacat.
4. Deoarece va trebui sa stocam date referitor la mai multe obiecte vom lua in considerare folosirea unei baze de date pentru stocarea informatiilor, despre starea alarmelor si setarile ce lea ales utilizatorul anterior.
5. Va fi nevoie de analizat cazuri particulare de functionare ca restabilirea alarmelor dupa ce telefonul a fost stins, executarea soneriei a unei singure alarme pentru a nu irita utilizatorul, alarma nu trebuie sa impiedice utilizarea de zi cu zi a telefonului si de aceea trebuie privite cu mare grija situatiile de functionare a ei.

Pentru a deduce modelul aplicatiei date avem nevoie de analiza a scenariilor si cazurilor de utilizare.

Cazuri de utilizare :

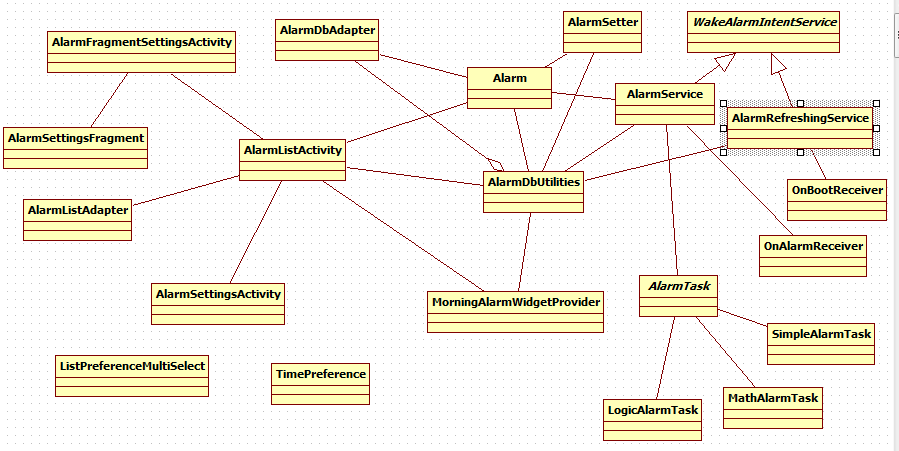


Dupa cum se poate observa din imaginea anterioara pentru a furniza o interfata grafica care corespunde cerintelor ea trebuie sa ia in considerare toate cele 9 cazuri de utilizare. Analizind cazurile de utilizare putem deduce si scenariile dupa care va fi folosita aplicatia.

Modelul aplicatiei este urmatorul: o aplicatie care va permite utilizatorului sa seteze alarme noi , precum si sa modifice, sa stearga sa activeze si sa dezactiveze alarme existente, pe de alta parte aplicatia trebuie sa instiinteze utilizatorul despre faptul ca alarma a ajuns la timpul specificat de acesta si trebuie sa ia decizia daca sa dea comanda Snooze pentru a face o pauza de 5 minute pina la urmatorul semnal sau sa rezolve sarcina pentru a dezactiva alarma. Alarma trebuie sa stocheze toate datele in baza de date, si sa se poata restabili dupa cazuri neobisnuite ca resetarea telefonului sau setarea a mai multor alarme pentru aceeasi ora.

1. Proiectare

a) Diagrama UML



b) Proiectare Clase

Dupa cum se observa din Diagrama UML am proiectat aplicatia folosind 21 de clase, rolul claselor este urmatorul:

AlarmRefreshingService : aceasta clasa este serviciul care este apelat de catre OnBootReceiver atunci cind aplicatiei i se trimite semnalul BootCompleted care este rezultatul restartarii telefonului. Rolul acestei clase este de a seta din nou toate alarmele din deoarece datorita proiectarii sistemului de operare acesta dupa ce face un restart complet a dispozitivului sterge toate alarmele setate anterior, respectiv, pentru ca aplicatia sa fie permanent functionala este necesar restabilirea tuturor alarmelor.

AlarmService : aceasta clasa este serviciul care este apelat de catre OnAlarmReceiver atunci cind aplicatiei i se transmite semnalul de alarma de catre clasa de sistem AlarmManager. Acest serviciu are rolul de a analiza ce fel de alarma este acum si care sunt proprietatile ei si trebuie sa actioneze corespunzator cazului.

AlarmSetter : aceasta clasa este responsabila pentru setarea si resetarea alarmei precum si de activarea sau deactivarea widgetului de pe ecran.

OnAlarmReceiver : este clasa care raspunde de capturarea semnalului de alarma si da startul service-ului AlarmService.

OnBootReceiver : este clasa care raspunde de capturarea semnalului de BOOT\_COMPLETED de la sistemul de operare care semnalizeaza de restartarea dispozitivului si apoi aceasta clasa da startul service-ului AlarmRefreshingService.

WakeAlarmIntentService : clasa abstracta care defineste modul in care serviciile AlarmService si AlarmRefreshingService pastreaza control asupra unui lacat care nu permite telefonului sa se stinga in timp ce se executa.

AlarmDbAdapter : clasa adaptor pentru baza de date, datorita clasei interne DatabaseHelper care extinde clasa SQLiteOpenHelper face adaptarea dintre interfata bazei de date si restul aplicatiei.

AlarmDbUtilities : aceasta clasa furnizeaza un set de functii pentru extragerea, adaugarea, stergerea, actualizarea, interogarea informatiilor din baza de date a aplicatiei.

Alarm : aceasta clasa simbolizeaza insasi obiectul alarmei care pastreaza in cimpurile sale informatii necesare pentru toate operatiile ce sunt executate asupra alarmelor.

AlarmFragmentSettingsActivity : aceasta clasa foloseste clasa AlarmSettingsFragment pentru a prezenta preferintele pentru alarma in cadrul sistemului de operare cu o versiune mai inalta sau egala cu 3.

AlarmListActivity : este meniul principal in care se pot adauga, sterge, modifica, activa si dezactiva alarmele.

AlarmListAdaper : este clasa Adapter dintre clasa ListView si clasa Alarm pentru ca in activitatea AlarmListViewActivity sa se afiseze continut corespunzator alarmei.

AlarmSettingsActivity : este clasa ce afiseaza preferinte pentru versiuni de sistem de operare Android mai mici sau egale cu 3

MorningAlarmWidgetProvider : este clasa ce controleaza continutul widgetului de pe ecranul principal si permite interactiunea cu aplicatia prin intermediul lui.

AlarmSettingsFragment : clasa ce extinde Fragment si se foloseste la afisarea preferintelor de catre clasa AlarmFragmentsSettingsActivity pentru afisearea preferintelor in cadrul sistemului de operare cu versiunea mai mare sau egala cu 3

ListPreferenceMultiSelect : se foloseste la setarea preferintelor pentru zilele in care sa functioneze alarma.

TimePreference : aceasta clasa se foloseste la afisarea de TimePicker in preferinte pentru permiterea de selectare a timpului la care va suna alarma.

AlarmTask : este o clasa abstracta care defineste comportamentul unei activitati task, adica a unei activitati ce apare pe ecran ca problema pe care trebuie sa o rezolve utilizatorul

LogicAlarmTask : este o clasa care defineste taskul logic pentru inchiderea alarmei.

MathAlarmTask : este o clasa care defineste taskul matematic pentru inchiderea alarmei.

SimpleAlarmTask : este o clasa care defineste cel mai simplu task care contine doar doua butoane snooze si cancel.

c) Pachete

In aceasta aplicatie am folosit patru pachete:

1. app.alarmmanager: AlarmRefreshingService, AlarmService, AlarmSetter, OnAlarmReceiver, OnBoot Receiver, WakeAlarmIntentService
2. app.database: AlarmDbAdapter, AlarmDbUtilities
3. app.morningalarm : Alarm, AlarmFragmentsSettingsActivity , AlarmListActivity,AlarmSettingsActivity, MorningAlarmWidgetProvider
4. app.morningalarm.preferences : AlarmSettingsFragment, ListPreferenceMultiSelect, TimePreference
5. app.tasks : AlarmTask, LogicAlarmTask, MathAlarmTask, SimpleAlarmTask.

d) Structuri de Date

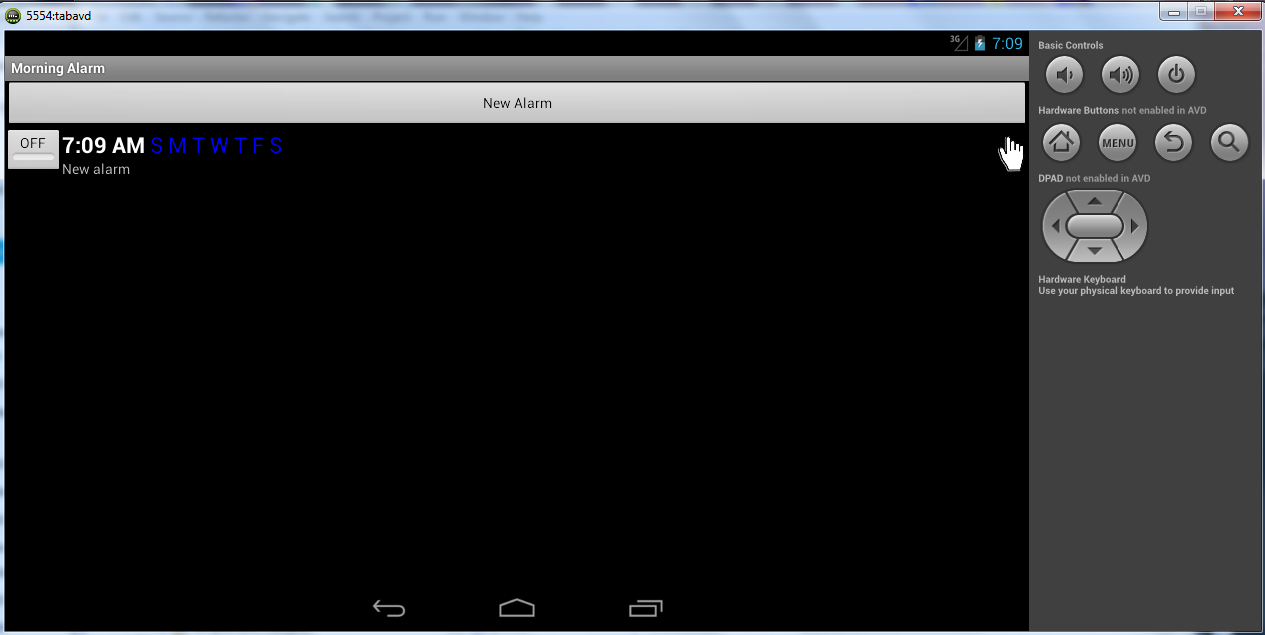
Structurile de date folosite pentru aplicatia data au fost ArrayList-urile standarte din Java ce s-au folosit pentru pastrarea datelor despre Alarme

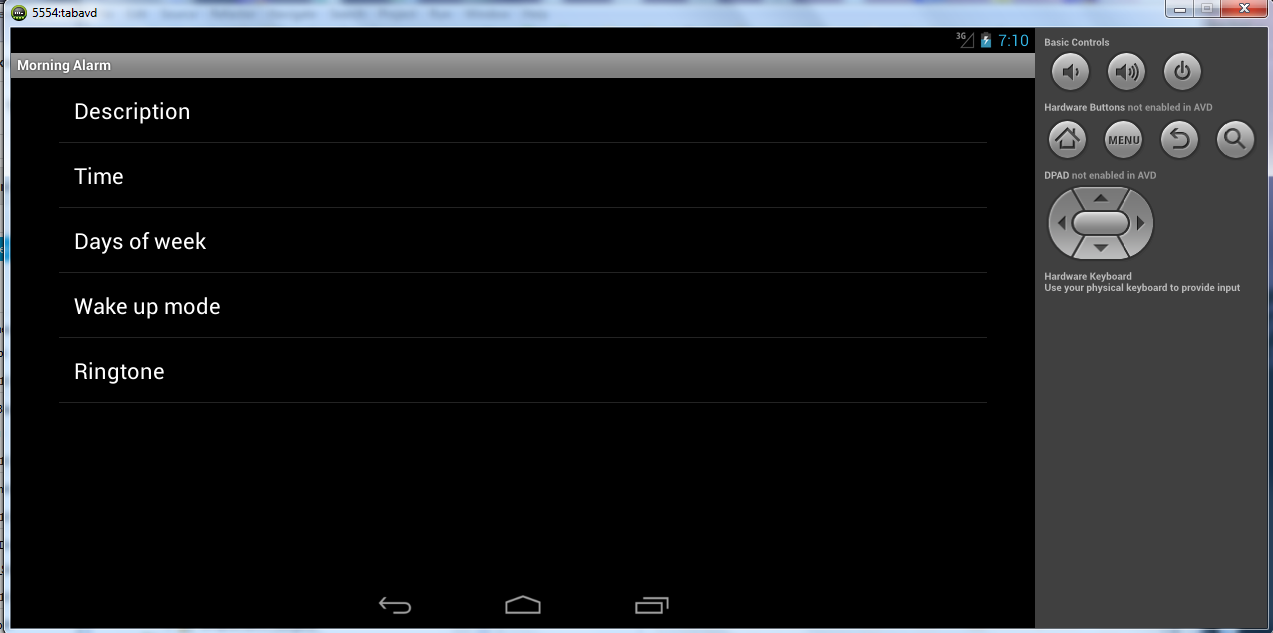
e) Interfete

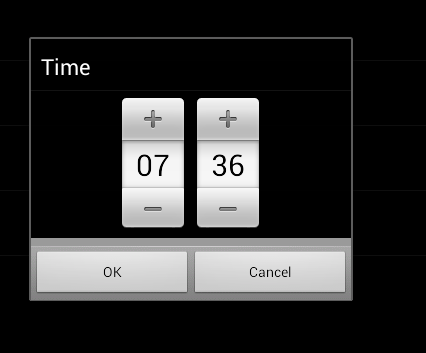
In aceasta tema am creat interfete noi, cele folosite fiind cele propuse de sistem.

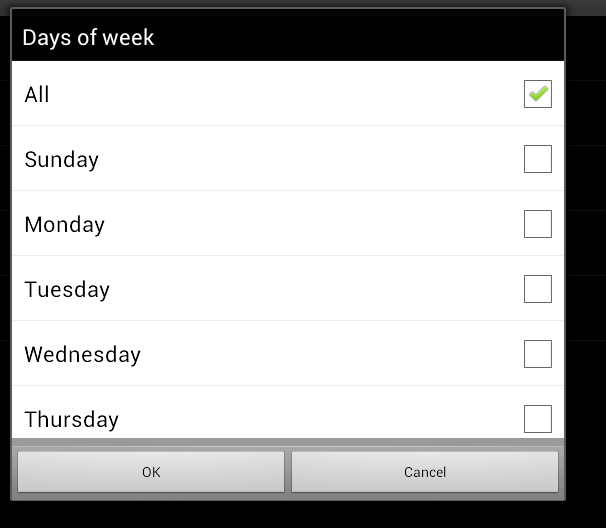
f) Interfata utilizator

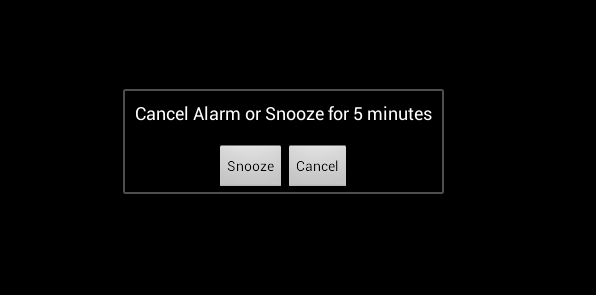
Interfata utilizator a fost realizato in felul urmator:

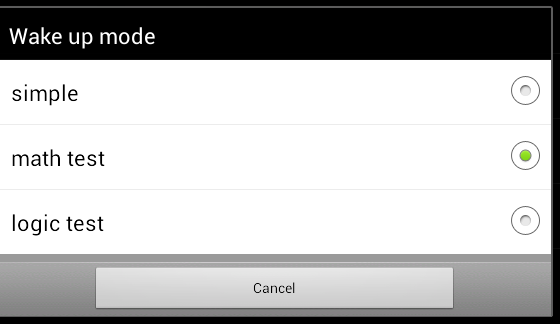


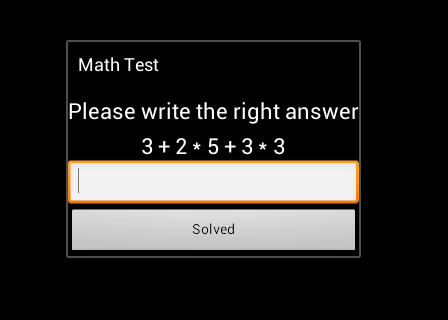












4. Implementare si testare

Clasa AlarmRefreshingService :

* public void doAlarmWork(Intent intent) : seteaza din nou toate alarmele

Clasa AlarmService :

* synchronized void doAlarmWork(Intent intent) : este apelat cind a fost primit un semnal de alarma , metoda primeste prin extras parametrul ID dupa care cauta alarma in baza de date si afla daca trebuie sa fie apelata activitatea de afisare sau trebuie resetata alarma pe ziua urmatoare.

Clasa AlarmSetter :

* public void refreshAllAlarms() : face refresh la toate alarmele, le seteaza din nou
* public void setAlarm(Alarm alarm) : seteaza alarma alarm
* private void getAlarmUpToDate(Alarm alarm) : verifica daca alarma este trecuta de ora curenta atuni o muta pe ziua viitoare
* public void removeAlarm(String alarmId) : sterge alarma cu id-ul alarmId din sistem

Clasa OnAlarmReceiver:

* public void onReceive(Context context, Intent intent) : este apelat atunci cind un semnal de alarma a fost primit, primeste id-ul alarmei si il transmite mai departe la service-ul AlarmService ca sa faca actiunea necesara

Clasa OnBootReceiver:

* public void onReceive(Context context, Intent intent) : metoda este apelata atunc cind telefonul a fost resetat la primirea semnalului apeleaza service-ul alarmRefreshingService pentru a seta si actualiza toate alarmele

Clasa WakeAlarmIntentService:

* public static void acquireStaticLock(Context context) : pune lacat la context
* synchronized private static PowerManager.WakeLock getLock(Context context): primeste lacat ca aplicatia sa fie in regim wake
* final protected void onHandleIntent(Intent intent) : executa metoda doAlarmWork si elibereaza lacatul

Clasa AlarmDbAdapter:

* public static AlarmDbAdapter getInstance(Context context) : metoda singleton
* public AlarmDbAdapter open() throws SQLException : deschide DataBaseHelper si primeste acces la baza de date
* public void close() : inchide database helper
* public long createAlarm() : introduce alarma in baza de date
* public long deleteAlarm(Alarm alarm) : sterge alarma din baza de date
* public long deletAll() : sterge toate alarmele din baza de date
* public Cursor fetchAllAlarms() : returneaza cursor ce contine toate alarmele
* public Cursor fetchAlarm(String rowId) throws SQLException : returneaza cursor ce contine alarma cu id rowId
* public Cursor fetchNewAlarm() throws SQLException : returneaza alarma cu descrierea de alarma noua
* public Cursor fetchEnabledAlarms() throws SQLException : returneaza alarma cu descrierea de alarmele activate
* public long updateAlarm(Alarm alarm) : face update la alarma

Clasa AlarmDbUtilities:

* public static final ArrayList<Alarm> fetchCursor(Cursor c) : returneaza arrayList cu toate alarmele din cursor
* public static final ArrayList<Alarm> fetchAllAlarms(Context context) : returneaza toate alarmele din baza de date
* public static final Alarm fetchAlarm(Context context, String alarmId) : returneaza Alarma cu id-ul alarmId din baza de date
* public static final Alarm fetchNewAlarm(Context context) : returneaza din baza de date alarma nou creata
* public static final void deleteAlarm(Context context, Alarm alarm) : sterge din baza de date alarma
* public static final void deleteAll(Context context) : sterge toate alarmele din baza de date
* public static final void updateAlarm(Context context, Alarm alarm) : face update la alarma in baza de date
* public static final ArrayList<Alarm> fetchEnabledAlarms(Context context) : returneaza ArrayList cu toate alarmele care sunt activate

Clasa Alarm:

* public long getTime() : returneaza timpul la care va suna alarma
* public void setTime(long time) : seteaza timpul la care va suna alarma
* public String getWakeUpMode() : returneaza tipul de task
* public void setWakeUpMode(String wakeUpMode) : seteaza modul de trezire
* public String getRingtone() : returneaza ringtonul
* public void setRingtone(String ringtone) : seteaza ringtonul
* public String getDaysOfWeek() : returneaza zilele in care va functiona alarma
* public void setDaysOfWeek(String daysOfWeek) : seteaza zilele in care va functiona alarma
* public int isEnabled() : returneaza daca este activata
* public void setEnabled(int enabled) : seteaza daca alarma este activata
* public String getId() : returneaza id-ul alarmei
* public void setId(String id) : seteaza id-ul alarmei
* public String getDescription() : returneaza descrierea alarmei
* public void setDescription(String description) : seteaza descrierea alarmei

5. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

Efectuind aceasta tema am invatat sa cum sa folosesc eficient paradigmele OOP, si cum sa creez aplicatii stabile si functionale pentru sistemul de operare Android. M-am convins ca este o tehnologie avansata si foarte vasta care merita sa fie studiata si invatata.

Rezolvind sarcinile puse in tema curenta am invatat cum trebuie de lucrat in mod eficient, ordinea si modul in care trebuie analizata problema si cautata solutia.

Dezvoltari ulterioare:

1. Se mai pot adauga si alte moduri de trezire, adica setare de noi taskuri.

Bibliografie

1. http://developer.android.com/