



**Dezvoltarea Aplicaţiilor pe Telefoane Mobile**

Student: Simirea Andrei

Profesor îndrumător: Sl.dr.ing Adriana Stan

Cuprins

1. Introducere în Android…………………………………….

1.Caracteristici Android…………………………………………....

2.Avantaje Android...................................................................

3.Produse hardware care rulează Android………………………...

4.Desert Android……………………………………………………

5.Dezvoltarea Android……………………………………………..

6. Concepte specifice în Android…………………………………...

7.Arhitectura Android……………………………………………..

8.Funcţionalităţi Android....................................................................

9.Comunitatea programatorilor Android…………………………....

1. Aplicaţie –Agenda Personală………………………….......
2. Cod aplicaţie. ………………………………………….....
3. Concluzii

**I.Introducere în Android**

Android reprezintă o platform software şi un sistem de operare pentru dispositive şi telefoane mobile bazat pe nucleul Linux. A fost dezvoltat la început de compania Google, după care de un consorţiu commercial numit Open Handset Alliance. Android permite atât dezvoltatorilor cât şi utilizatorilor amatori să scrie cod gestionat şi organizat în limbajul Java, controlând dispozitivul prin intermediul bibliotecilor Java dezvoltare de Google.

La nivel de licenţă Google a lansat cea mai mare partea a codului Android sub licenţa Apache, o licenţă de tip free-software şi open source. Sub această licenţă producătorii au libertatea de a adăuga extensii proprietare, fără a le face disponibile comunităţii open source, rezervându-şi anumite drepturi.

Android prezintă un pachet complex şi functional având sistem de operare, middleware şi aplicaţii cheie pentru funcţionarea sistemului.

**1. Caracteristici Android**

* Configuraţii dispositive-Platforma se pliază uşor la configuraţii mai mari de la VGA, până la biblioteci grafice 2D, biblioteci grafice 3D bazate pe specificaţia OpenGLES 1.0
* Stocare de date-Software-ul de baze de date SQLite este folosit în scopul stocării datelor
* Conectivitate-Android suportă tehnologii de conectivitate incluzând GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth şi Wi-Fi.
* Mesagerie Instant-SMS şi SMS reprezintă formele de mesagerie instant disponibile, inclusiv conversaţii de mesaje text
* Navigatorul de web-Navigatorul de web disponibil în Android este bazat pe platforma de aplicaţii open source WebKit
* Maşina virtuală Dalvik-Sofware-ul scris în Java poate fi compilat în cod maşină Dalvik şi executat de maşina virtuala Dalvik, care este o implementare specializată de maşină virtuală concepută pentru utilizarea în dispozitivele mobile, deşi teoretic nu este o Maşină Virtuală Java standard
* Suport media-Android acceptă formate media,audio,video,imagine:MPEG-4, H.264, MP3, AAC, OGG, AMR, JPEG, PNG, GIF
* Suport hardware adiţional-Android poate utiliza camera video/ foto, touchscreen, GPS, accelerometru şi grafică accelerată 3D
* Mediu de dezvoltare-Include un emulator de dispozitive, unelte de depanare, profilare de memorie şi de performanţă, un plug-in pentru mediul de dezvoltare Eclipse
* Piaţa Android-Similar cu App Store-ul de pe iPhone, Piaţa Android este un catalog de aplicaţii care pot fii descărcate şi instalate pe hardware-ul ţintă prin comunicaţie fără fir, fără utilizarea unui PC

**2. Avantaje Android**

* Platformă de dezvoltare free şi open-source
* Arhitectură bazată pe componente inspirată de mashup-uri de Internet, în sensul că unele componente de cod pot fi reutilizabile în cadrul altor aplicaţii sau unele componente predefinite pot fi reimplementate de către dezvoltator.
* Utilizarea unor servicii “out of box”: GPS, bază de date SQL
* Management automat al ciclului de viaţă al aplicaţiei
* Grafică şi sunete de o calitate înaltă (OpenGL).
* Portabilitate: o gamă largă de hardware curente şi viitoare

**3.Produse hardware care rulează Android**

Printre produsele hardware care rulează Android putem enumera: HTC Dream, Motorola, Huawei, Archos, Lenovo, HTC, Sony Ericsson, Samsung.

**4.Desert Android**

Toate versiunile sistemului de operare Android au denumiri de prăjituri(desert). Mai mult de atât ele sunt şi în ordine alfabetică. Dintre versiuni de Android putem enumera:

1. Android 1.0 Astro
2. Android 1.1 Bender
3. Android 1.5 Cupcake
4. Android 1.6 Donut
5. Android 2.0/2.1 Éclair
6. Android 2.2 x Froyo
7. Android 2.2.x Gingerbread
8. Android 3.x Honeycomb
9. Android 4.0.x Ice Cream Sandwich
10. Android 4.1.x Jelly Bean
11. Android 5.0 Lollipop
12. Android 6.0 Marshmallow
13. Android 7.0 Nougat
14. Android 8.0 Oreo



Figura 1–Versiunile Android exprimate prin nume de desert-uri

**5.Dezvoltarea Android**

La începutulul dezvoltării aplicaţiilor, majoritatea includeau bug-uri, lipsă de documentaţie, infrastructură de testare adecvată, lipsa unui management eficient, urmând ca după un timp lucrurile să se aşeze, să existe o linie continua de dezvoltare în cel mai profesionist sens.

Prima aplicaţie lansată de Android a fost jocul Snake. Printre telefoane la început s-a enumerat telefonul Android Dev, un dispozitiv cu SIM şi hardware neblocate, el fiind destinat dezvoltatorilor avansaţi.

Dezvoltarea în Android se poate realiza cu ajutorul mediului vizual Eclipse, utilizând SDK-ul de Android şi ADT Eclipse Plugin. Pe de altă parte se poate utiliza limbajul de programare Java, iar interfaţa grafică se realizează în XML. Aplicaţia compilată generează un fişier.apk care conţine toate xml-urile şi celelalte resurse.

**6.Concepte specifice în Android**

În cadrul platforme şi dezvoltării Android găsim o mulţime de lucruri care stau la baza realizării diferitelor aplicaţii.

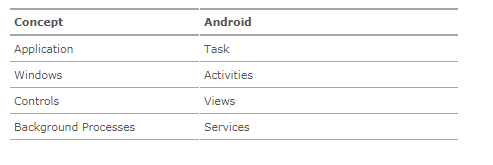


Figura 2-Concepte Android

La nivel de cadru al aplicaţiei (Application Framework) întâlnim:

* Content Providers-Permite schimbul de date între aplicaţii
* Resource Manager-Asigură accesul la resurse non-cod
* Notification Manager-Ofera tuturor aplicaţiilor posibilitatea de a afişa notificări
* Activity Manager-Gestionează ciclul de viaţă al aplicaţiilor

La nivel de blocuri(părţi de program) întâlnim următoarele:

* AndroidManifest.xml-fişier de control care înştiinţează sistemul despre cum trebuie să utilizeze elementele care alcătuiesc aplicaţia
* Activity-un obiect care are ciclul de viaţă sau secţiune de cod care furnzează o anumită funcţionalitate
* View-un obiect care “cunoaşte” cum trebuie să arate pe ecran (screen) partea grafică
* Intent-un simplu obiect de tip mesaj care reprezintă de fapt”intenţia” de a face ceva.
* Notification-reprezintă un icon care apare în cadrul barei de status pentru a înştiinţa userul asupra unui anumit eveniment care s-a petrecut într-una dintre aplicaţii.
* Service-un bloc de instrucţiuni care rulează în background.

Ca şi tool-uri de dezvoltare ne vom orienta în jurul SDK Android. SDK Android include o varietate de tool-uri custom care ajută dezvoltatorul de aplicaţii Android. Aici putem enumera:

* Android Emulator-Un device mobil, virtual care rulează pe calculator-utilizat pentru a vizualiza design-ul, a face debug şi pentru a testa aplicaţiile run-time.
* Android Development Tools Plugin-folosit pentru Eclipse IDE
* Dalvik Debug Monitor Service(DDMS)-acest tool facilitează modul de execuţie al proceselor şi asistă procesul de debug.

Platformele actuale de dezvoltare includ calculatoare bazate pe x86 care rulează Linux, Mac OS X sau WindowsXP sau Vista. Cerinţele include JavaDevelopmentKit, Apache Ant şi Phyton 2.2.

**7.Arhitectura Android**

Software-stack-ul care defineşte Android este format din aplicaţii Java care rulează pe un framework orientat obiect, bazat si el pe Java, deasupra librăriilor de nucleu Java care rulează pe o maşină virtuală Dalvik ce conţine compilatoare JIT. Librăriile scrise în C includ managerul de suprafaţă, framework-ul OpenCore, sistemul relational de management de bază de date SQLite, OpenGl ES.2.0 3D, motorul WebKit, motorul SGL, SSL şi Bionic libc.Platforma hardware este bazată pe arhitectura ARM.

Arhitectura sistemului de operare Android cuprinde cinci secţiuni grupate pe 4 niveluri şi anume:

1. **Kernel Linux**. El conţine drivere pentru diferitele componente hardware (ecran, cameră foto, tastatură, memorie flash, dispozitive audio, antenă Wi-Fi). Kernel-ul se ocupă cu gestiunea proceselor, memoriei, perifericelor (audio/video, GPS, Wi-Fi), dispozitivelor de intrare/ieşire, reţelei şi a consumului de energie. La nivel de kernel au fost implementate anumite îmbunătăţiri:

a)***Binder***sau sistemul de comunicaţie inter-proces reprezintă mediul de comunicaţie principal dintre aplicaţii şi sistemul de operare, inclusiv funcţiile dispozitivului mobil.El se reprezintă/expune prin intermediul AIDL(Android Interface Definition Language) prin care pot fi manipulate obiecte transformate în primitive utilizate la comunicaţie propriu-zisă dintre aplicaţii şi sistemul de operare.

b)***Logger-ul*** sau sistemul de jurnalizare. El este foarte important în cazul în care trebuie realizată depanarea aplicaţiilor în special pentru a detecta anumite situaţii particulare. El este capabil să realizeze agregarea datelor provenite atât de la aplicaţia propriu-zisă cât şi de la sistemul de operare, fiind disponibile prin intermediul unor utilitare speciale.

c)***Wake locks*** reprezintă sistemul prin intermediul căruia se previne transferul sistemului de operare într-o reţea de latenţă, în care consumul de energie este redus, întrucât se blochează execuţia oricărei aplicaţii

d)***Sistemul de alarme*** oferă posibilitatea ca anumite sarcini să fie planificate la anumite momente de timp, ele putând fi utilizate, chiar dacă sistemul de operare se regăseşte într-o stare de latenţă.

e)***Viking Killer*** este un mecanism prin care sistemul de operare revendică memoria utilizată, atunci când nivelul acesteia atinge un anumit prag

f)***YAFFS2***(Yet Another Flash File System) este un sistem de fişiere adecvat pentru cipuri flash bazate pe porti NAND.

2.***Bibliotecile(user-space)*** conţin codul care oferă principalele funcţionalităţi a sistemului de operare Android, facând legătura între kernel şi aplicaţii.

3.**Motorul Android** rulează serviciile de platformă precum şi aplicaţiile care le utilizează, fiind reprezentat de:

a.***ART(Android Runtime)***-maşina virtuală Java

b.***Zygote*** reprezintă procesul care gestionează toate aplicaţiile, fiind lansat în execuţie odată cu sistemul de operare.

4.**Cadrul pentru Aplicaţii(Application Framework)** expune diferitele funcţionalităţi ale sistemul de operare Android către programatori, astfel încât aceştia să le poată utiliza în aplicaţiile lor.

5. La nivel de ***aplicaţii*** regăsim atât produsele împreună cu care este livrat dispozitivul mobil(Browser, Camera, Calculator, Clock, FM Radio, Launcher, Music Player , Phone, S Note, S Planner, Video Player, Voice Recorder), cât şi produsele instalate de pe Play Store sau cele dezvoltate de programatori.



Figura 3 –Arhitectura sistemului de operare Android

**8.Funcţionalităţi Android**

Datorită faptului că Android este un platformă open-source, producătorii au posibilitatea de a-l modifica în mod gratuit, neexistând configuraţii hardware sau software standard. Android dispune astfel de următoarele funcţionalităţi:

* Android Beam-tehnologie bazată pe NFC(Near Field Communication) care permite utilizatorilor să partajeze conţinut instant , prin apropierea dispozitivelor mobile respective
* Wi-Fi Direct-tehnologie care permite aplicaţiilor să se descopere şi să se interconecteze peste o conexiune punct-la-punct având lăţime de bandă mare
* Stocare-foloseşte SQLite, o bază de date relaţională care utilizează resurse puţine
* Conectivitate-suportă GSM/CDMA, GPRS, EDGE, 3G, BLUETOOTH(A2DP şi AVRCP), LTE, WiMAX
* **mesagerie** - atât SMS cât și MMS
* **navigare pe Internet** - bazat pe motorul open source pentru navigare WebKit impreună cu motorul JavaScript de la Chrome V8 suportând HTML5 și CSS3
* **multimedia** - suportă formatele H.263, H.264 (într-un container 3GP sau MP4), MPEG-4 SP, AMR, AMR-WB (într-un container 3GP), AAC, HE-AAC (într-un container MP4 sau 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF si BMP
* **grafică** - 2D optimizată, 3D (OpenGL ES)
* **senzori** - accelerometru, cameră foto, busolă digitală (magnetometru), senzor de proximitate, GPS / AGPS
* **multi-touch** - suportă ecrane cu posibilitate de contact în mai multe puncte concomitant
* **multi-tasking** - permite rularea de aplicații cu mai multe fire de execuție
* **GCM (Google Cloud Messaging)** - serviciu ce permite dezvoltatorilor să trimită date de dimensiuni mici către utilizatori pe dispozitive Android, fără a avea nevoie de o soluție de sincronizare proprietară
* **multi-Language** - suport pentru text unidirecțional și bidirectional
* **suport pentru aplicații Flash** (până în versiunea 4.3)
* **legătură la Internet** - suportă partajarea conexiunilor la Internet ca punct de distribuție cu fir / fără fir

**9.Comunitatea programatorilor Android**

* Google Android Training
* Stack Overflow
* Android Discuss

Cerinţe pentru dezvoltarea unei aplicaţii(cerinţe minime):

1. ***Kit-ul de dezvoltare*** pentru limbajul de programare Java
2. ***SDK-ul de Android***, prin care se descarcă definiţiile corespunzătoare unuia sau mai multor niveluri de API
3. ***Un mediu integrat de dezvoltare(IDE)***
4. Eclipse, cu plugin-ul ADT(Android Developer Tools)
5. Android Studio
6. ***Un dispozitiv pe care să ruleze aplicaţiile***
7. Un emulator care cuprinde
8. Genymotion
9. Android Virtual Device( livrat cu SDK-ul Android)

b)un telefon mobil cu sistemul de operare Android pentru care s-a dezvoltat aplicaţia

**II. Aplicaţia Agenda Personală**

În cadrul acestui proiect s-a pus accentul pe înţelegerea şi realizarea unor noţiuni specifice Android-ului. Aplicaţia numită *Agenda Personală* exemplifică anumite rubrici/părţi de proiect, ea fiind dezvoltată incluzând: Pagina de home, Contacte, Memo, Transport . Între anumite clase dintre acestea s-a folosit şi o bază de date prin care am putut extrage sau modifica anumite informaţii spre exemplu :adăugare sau ştergere de contacte, adăugare informaţii suplimentare precum adresă, număr de telefon, adresă de e-mail, etc.

La deschiderea aplicaţiei s-a folosit şi un anumit sound(Splash) premergător deschiderii meniului şi agendei implicit. S-au realizat conexiuni atât între contacte, cât şi la nivel de clasă Transport sau clasă Memo.

**III. Cod aplicaţie**

O privire de ansamblu asupra claselor principale descrise în acest proiect se regăsesc mai jos. O primă clasă este Clasa Contact.

1. Clasa Contact

**package** com.example.personalagenda;  
  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.widget.Button;  
**import** android.app.Activity;  
**import** android.content.Intent;  
 *//contact dezvoltator***public class** Contact **extends** Activity **implements** OnClickListener {  
  
 Button **meniu\_principal**;  
  
 **public void** onCreate(Bundle savedInstanceState) {  
  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***contact\_dezvoltator***);  
 initButtons();  
 }  
  
 *// initializare butoane si adaugare listeneri pentru acestea* **public void** initButtons() {  
  
 **meniu\_principal** = (Button) findViewById(R.id.***contact\_dezvoltator\_b\_meniu\_principal***);  
 **meniu\_principal**.setOnClickListener(**this**);  
 }  
  
 *// se trateaza evenimentul corespunzatorul butonului* @Override  
 **public void** onClick(View v) {  
 *// activare intenturi  
 //* ***TODO Auto-generated method stub* switch** (v.getId()) {  
  
 **case** R.id.***contact\_dezvoltator\_b\_meniu\_principal***:  
 Intent i = **new** Intent(  
 **"com.example.personalagenda.MENU"**);  
 startActivity(i);  
 **break**;  
 }  
  
 }  
}

Putem observa folosirea acelor intent-uri, respectiv bundle sau Event Listener.

*Intent-urile* sunt anumite mesaje activate care deservesc cele 3 componente ale aplicaţiei şi anume : activităţi, servicii, repository.

*Repository* reprezintă locaţia unde sunt stocate toate fişierele aferente unui anumit proiect. Mesageria prin Intent –uri reprezintă o funcţionalitate pentru runtime binding între componentele unei aplicaţii. Însuşi Intent-ul , un obiect Intent, este o structură de date pasivă, care deţine o definiţie abstractă a operaţiilor ce pot fi efectuate.

Obiectul Intent este la rândul său o colecţie (Bundle) de informaţii. El conţone informaţii de interes pentru acea componentă care primeşte Intent-ul plus informaţiile de interes pentru sistemul Android.

*Bundle* reprezintă acele grupări, colecţii utilizate în general pentru transmiterea de date între diferite activităţi Android. Pe baza modelului arhitectural şi a design-ului pattern se defineşte noţiunea de Event Listener.

Orice componentă grafică poate genera evenimente(Events) atunci când utiliazatorul interacţionează prin intermediul mouse-ului aflat deasupra componentei sau când componenta deţine focus-ul, iar utilizatorul interacţionează prin intermediul tastaturii.

Un eveniment se livrează destinatarului prin transmiterea ca parametru unei metode de tratare a evenimentelor.

1. Clasa Memo

**package** com.example.personalagenda;  
  
**import** android.os.Bundle;  
**import** android.app.Activity;  
**import** android.content.Intent;  
**import** android.view.Menu;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.widget.AdapterView;  
**import** android.widget.AdapterView.OnItemClickListener;  
**import** android.widget.Button;  
**import** android.widget.ListView;  
  
**public class** Memo **extends** Activity **implements** OnClickListener {  
  
 **private** ListView **menuLV**;  
 Button **meniuPrincipal**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***memo\_main***);  
  
 *// Listener* **meniuPrincipal** = (Button)findViewById(R.id.***memo\_main\_meniu\_b\_principal***);  
 **meniuPrincipal**.setOnClickListener(**this**);  
 **menuLV** = (ListView) findViewById(R.id.***memo\_main\_lv\_opt***);  
 **menuLV**.setOnItemClickListener(**new** OnItemClickListener() {  
  
 @Override  
 **public void** onItemClick(AdapterView<?> adapterView, View listView, **int** selectedItemId,  
 **long** arg4) {  
 *//* ***TODO Auto-generated method stub*** mainListViewOnItemClickHandler(adapterView, listView, selectedItemId, arg4);  
 }  
 });  
 }  
  
 *// handler click* **private void** mainListViewOnItemClickHandler(AdapterView<?> adapterView, View listView, **int** selectedItemId,**long** arg4) {  
 *//selectare item* String selectedItem = (String) **menuLV**.getItemAtPosition(selectedItemId);  
   
 **if**(selectedItem.equals(getString(R.string.***memo\_main\_lv\_vezi\_grupuri***))){  
 Intent intentVeziGrupuri = **new** Intent(**"com.example.personalagenda.MemoViewGroups"**);  
 startActivity(intentVeziGrupuri);  
 }  
 **if**(selectedItem.equals(getString(R.string.***memo\_main\_lv\_vezi\_memouri***))){  
 Intent intentVeziMemouri = **new** Intent(**"com.example.personalagenda.MemoViewMemos"**);  
 startActivity(intentVeziMemouri);  
 }  
 }  
   
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.***main***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public void** onClick(View v) {  
 *//* ***TODO Auto-generated method stub* switch** (v.getId()) {  
  
 *//in cazul butonului meniu principal, ne intoarcem in meniul anterior* **case** R.id.***memo\_main\_meniu\_b\_principal***:  
 Intent i = **new** Intent(**"com.example.personalagenda.MENU"**);  
 startActivity(i);  
 **break**;  
   
 }  
  
}  
}

Noutatea în cazul de faţă este ***TODO Auto-generated method stub***. Această metodă este generată automat de Eclipse odată cu crearea claselor.

***TODO*** reprezintă unul dintre cele mai multe cuvinte cheie prestabilite care indică sarcini care vor fi efectuate în Eclipse. Acest lucru poate fi observat astfel: Window->Preferinţe->Java->Compiler-> Sarcina Tag-uri(vizualizare cuvinte cheie).

***Stub*** reprezintă clasa care implementează interfaţa de la distanţă într-un mod pe care utilizatorul îl poate folosi local. Clasa Stub se ocupă de date marshalling sau unmarshalling precum şi de trimiterea sau primirea unui serviciu de la distanţă. Termenul de stub este folosit pentru a descrie această funcţionalitate şi în alte metode precum RPC(com, Java Remoting).

1. Splash

Se poate stabili un sunet care să fie premergător deschiderii sau parcurgerii claselor din aplicaţie.

**package** com.example.personalagenda;  
  
**import** android.app.Activity;  
**import** android.content.Intent;  
**import** android.media.MediaPlayer;  
**import** android.os.Bundle;  
  
*/\* Ecran ce contine logoul aplicatiei si adauga un sunet  
 \*/***public class** Splash **extends** Activity{  
   
 MediaPlayer **ourSong**; *//sunet* @Override  
 **protected void** onCreate(Bundle splash) {  
 *//* ***TODO Auto-generated method stub* super**.onCreate(splash);  
 setContentView(R.layout.***splash***); *// setare layout corespunzator* **ourSong** = MediaPlayer.create(Splash.**this**, R.raw.***music***); *//sunet* **ourSong**.start(); *//start sunet  
   
 //se defineste un Thread pentru a permite rularea Splash-ului 5 secunde* Thread timer = **new** Thread(){  
 **public void** run(){  
 **try**{  
 *sleep*(2000);  
 } **catch** (InterruptedException e){  
 e.printStackTrace();  
 } **finally** {  
 Intent openMainActivtity = **new** Intent(**"com.example.personalagenda.MENU"**);  
 startActivity(openMainActivtity);  
 }  
 }  
 };  
 timer.start();  
 }  
  
 @Override  
 **protected void** onPause() {  
 *//* ***TODO Auto-generated method stub* super**.onPause();  
 **ourSong**.release(); *//oprire muzica* finish(); *//incheiere activitate* }

d) Clasa Transport

Această clasă este exemplificată având mai multe linii de transport. În exemplu de faţă linia 35.

**package** com.example.personalagenda;  
  
**import** android.app.Dialog;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.widget.AdapterView;  
**import** android.widget.AdapterView.OnItemSelectedListener;  
**import** android.widget.Button;  
**import** android.widget.ImageView;  
**import** android.widget.RadioButton;  
**import** android.widget.Spinner;  
**import** android.widget.TextView;  
  
*// clasa creeare , modificare Memo***public class** TransportUTCN **extends** MemoGeneralKeepLayout **implements** OnClickListener {  
  
 Spinner **spinnerLinii**;  
 Button **meniuPrincipal**;  
 TextView **perechea1wd**, **perechea2wd**, **perechea1wS**,**perechea1wD**, **perechea2wS**, **perechea2wD**, **distanta**, **durata**, **sursa1**, **sursa2**, **destinatie1**, **destinatie2**;  
 TransportUTCNHandlerSQL **entry** = **new** TransportUTCNHandlerSQL(TransportUTCN.**this**);  
 ImageView **imagineTip**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***transport\_utcluj***);  
 initButoane();   
 **meniuPrincipal**.setOnClickListener(**this**);  
 initDB();  
   
 *//se selecteaza ruta* **spinnerLinii**.setOnItemSelectedListener(**new** OnItemSelectedListener() {  
  
 @Override  
 **public void** onItemSelected(AdapterView<?> arg0, View arg1,  
 **int** arg2, **long** arg3) {  
 String opt = **spinnerLinii**.getSelectedItem().toString();  
   
 *//* ***TODO Auto-generated method stub* if** (opt.equals(**"Linia"**)) {  
   
 **imagineTip**.setImageResource(R.drawable.***ic\_launcher***);  
 **distanta**.setText(**"Distanta"**);  
 **durata**.setText(**"Durata"**);  
 **sursa1**.setText(**"Capat 1"**);  
 **destinatie1**.setText(**"Capat 2"**);  
 **sursa2**.setText(**"Capat 1"**);  
 **destinatie2**.setText(**"Capat 2"**);  
 **perechea1wd**.setText(**""**);  
 **perechea1wS**.setText(**""**);  
 **perechea1wD**.setText(**""**);  
 **perechea2wd**.setText(**""**);  
 **perechea2wS**.setText(**""**);  
 **perechea2wD**.setText(**""**);  
 }  
 **if** (opt.equals(**"35"**)) {  
 **entry**.open();  
 String type35 = **entry**.getBusType(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***);  
 String source35 = **entry**.getsSource(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***);  
 String dest35 = **entry**.getDestination(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***);  
 String duration35 = **entry**.getDuration(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***);  
 String distance35 = **entry**.getDistance(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***);  
 String hours35wd1 = **entry**.getWDHours1(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***).replaceAll(**" "**, **"\n-------------------\n"**).replaceAll(**"[.]"**, **":"**).replaceAll(**"[,]"**,**":"**);  
 String[] hours35w1 = **entry**.getWHours1(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***).split(**"[!]"**);  
 String p1ws = hours35w1[0].replaceAll(**" "**, **"\n--------\n"**).replaceAll(**"[.]"**, **":"**).replaceAll(**"[,]"**,**":"**);  
 String p1wd = hours35w1[1].replaceAll(**" "**, **"\n--------\n"**).replaceAll(**"[.]"**, **":"**).replaceAll(**"[,]"**,**":"**);  
 String hours35wd2 = **entry**.getWDHours2(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***).replaceAll(**" "**, **"\n-------------------\n"**).replaceAll(**"[.]"**, **":"**).replaceAll(**"[,]"**,**":"**);  
 String[] hours35w2 = **entry**.getWHours2(**"35"**, TransportUTCNHandlerSQL.***KEY\_LINE***).split(**"[!]"**);  
 String p2ws = hours35w2[0].replaceAll(**" "**, **"\n--------\n"**).replaceAll(**"[.]"**, **":"**).replaceAll(**"[,]"**,**":"**);  
 String p2wd = hours35w2[1].replaceAll(**" "**, **"\n--------\n"**).replaceAll(**"[.]"**, **":"**).replaceAll(**"[,]"**,**":"**);  
 **entry**.close();  
   
 **if**(type35.equals(**"autobuz"**)){  
 **imagineTip**.setImageResource(R.drawable.***transport\_autobuz***);  
 }  
 **else  
 imagineTip**.setImageResource(R.drawable.***ic\_launcher***);  
 **distanta**.setText(distance35);  
 **durata**.setText(duration35);  
 **sursa1**.setText(source35);  
 **destinatie1**.setText(dest35);  
 **sursa2**.setText(dest35);  
 **destinatie2**.setText(source35);  
 **perechea1wd**.setText(hours35wd1);  
 **perechea1wS**.setText(p1ws);  
 **perechea1wD**.setText(p1wd);  
 **perechea2wd**.setText(hours35wd2);  
 **perechea2wS**.setText(p2ws);  
 **perechea2wD**.setText(p2wd);  
  
  
 }

În continuare putem observa clasele care au dus la realizarea aplicaţiei.

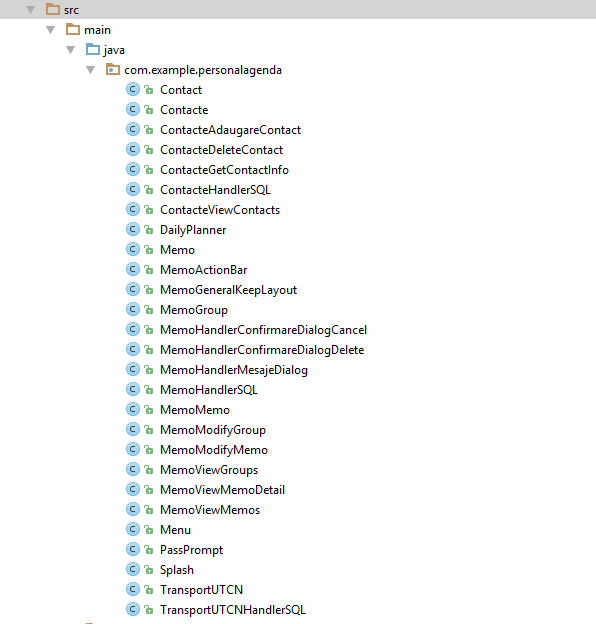


Figura 4-Clase aplicaţiei Agenda Personală

**IV.Concluzii**

Android rămâne una dintre cele mai utile şi performante platforme pentru dezvoltarea aplicaţiilor externalizând serviciile pentru crearea unor soft-uri inteligente în crearea aplicaţiilor mobile, care duc la simplificarea şi bunăstarea utilizatorilor atât din punct de vedere personal cât şi professional. Numărul de aplicaţii dezvoltate pentru Android a fost şi este în continua creşteare, acest lucru fiind unul îmbucurător atât pentru dezvoltatori cât şi pentru utilizatori. Rivalitatea cu Apple va rămâne în continuare, acest lucru reprezentând pentru comunitatea Android, un studiu continuu de dezvoltare software.