**ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ УНИВЕРЗИТЕТА У БЕОГРАДУ**


# АНДРОИД АПЛИКАЦИЈА ЗА ПРОРАЧУН ПОЗНАТИХ МОДЕЛА ИЗ ТЕОРИЈЕ СЕРВИСНИХ СИСТЕМА

– Дипломски рад –

Кандидат:

Стефан Радуновић 2009/297

Ментор:

доц. др. Зоран Чича

Београд, Септембар 2014.

# САДРЖАЈ

**САДРЖАЈ** ..... **2**

**1. УВОД** ..... **3**

**2. АНДРОИД АПЛИКАЦИЈЕ** ..... **4**

2.1. Развојни систем ..... 4
2.2. Eclipse IDE развојно окружење ..... 5

**3. СТРУКТУРА КОДА АПЛИКАЦИЈЕ** ..... **11**

3.1. Прорачуни ..... 11
3.2. Цртање једноструких и вишеструких графика ..... 23

**4. УПУТСТВО ЗА КОРИШЋЕЊЕ АПЛИКАЦИЈЕ** ..... **27**

4.1. Опште упутство ..... 27
4.2. *M/M/m/m/L сервисни систем* ..... 27
4.3. *M/G/1* сервисни систем ..... 30
4.4. *M/M/m* сервисни систем ..... 31
4.5. *M/M/m/m* сервисни систем ..... 32

**5. ЗАКЉУЧАК** ..... **34**

**ЛИТЕРАТУРА** ..... **35**

# 1.Увод

Теорија сервисних система проучава рад система у смислу чекања, кашњења и загушења. Широко је примељива, и налази примену у техници, економији, организацији, индустрији и многим другим научним и привредним областима. У телекомуникацијама је веома значајна, и користи се за моделовање телекомуникационог система или његовог дела. Модел сервисног система је математичка представа реалног система, уз помоћ које се могу прорачунати важне карактеристике система, као што су време задржавања унутар система, вероватноћа блокаде, број корисника у систему у неком тренутку итд[9][10].

Андроид апликације су намењене Андроид *(Android)* оперативном систему за мобилне уређаје, првенствено за уређаје са тачскрином, као што су паметни телефони и таблет рачунари, као и специјализоване корисничке интерфејсе за телевизоре *(Android TV),* аутомобиле *(Android Auto)* и ручне сатове *(Android Wear)*. Уређаји са андроид оперативним системом су све више распрострањени, због својих малих и релативно малих димензија, и веома великих могућности. Апликације се програмирају у *Java* програмском језику, уз помоћ *SDK (Android Software Development Kit)*[2].

У овом раду ће бити реализован калкулатор за прорачуне модела сервисних система за потребе андроид корисника. Биће обрађена 4 најпознатија модела сервисних система: *M/M/m, M/M/m/m, M/M/m/m/l* и *M/G/1*. Опције, омогућене кориснику апликације, ће се састојати од прорачуна важних и специфичних вредности као и исцртавања графика и вишеструких графика, за вредности које уноси корисник. Поред приложеног програмског решења, остатак рада ће се састојати од 4 поглавља:

- поглавља у ком ће бити описане основе програмирања андроид апликација као и искуства аутора овог рада и његове препоруке
- поглавља у ком ће подробно бити описана и појашњена сама апликација и њен код
- поглавља које ће представљати упуство за коришћење апликације
- поглавља које даје завршна разматрања аутора, као и смернице са за будуће унапређење реализоване апликације

# 2. АНДРОИД АПЛИКАЦИЈЕ

У овом поглављу биће дате основне информације о Андроид оперативном систему, Андроид апликацијама и софтверу потребном за њихово програмирање.

## 2.1. Развојни систем

Андроид оперативни систем је базиран на Линукс језгру и развија га компанија *Google*. Кориснички интерфејс је дизајниран првенствено за тачскрин уређаје. Пошто су уређаји, за које је развијен Андроид, обично напајани батеријом, Андроид управља *RAM* меморијом на минимуму потрошње струје. Апликација која није тренутно у употреби је аутоматски губи своју алоцирану меморију, иако је апликација и даље укључена, тј. ради у позадини док опет не буде активирана.

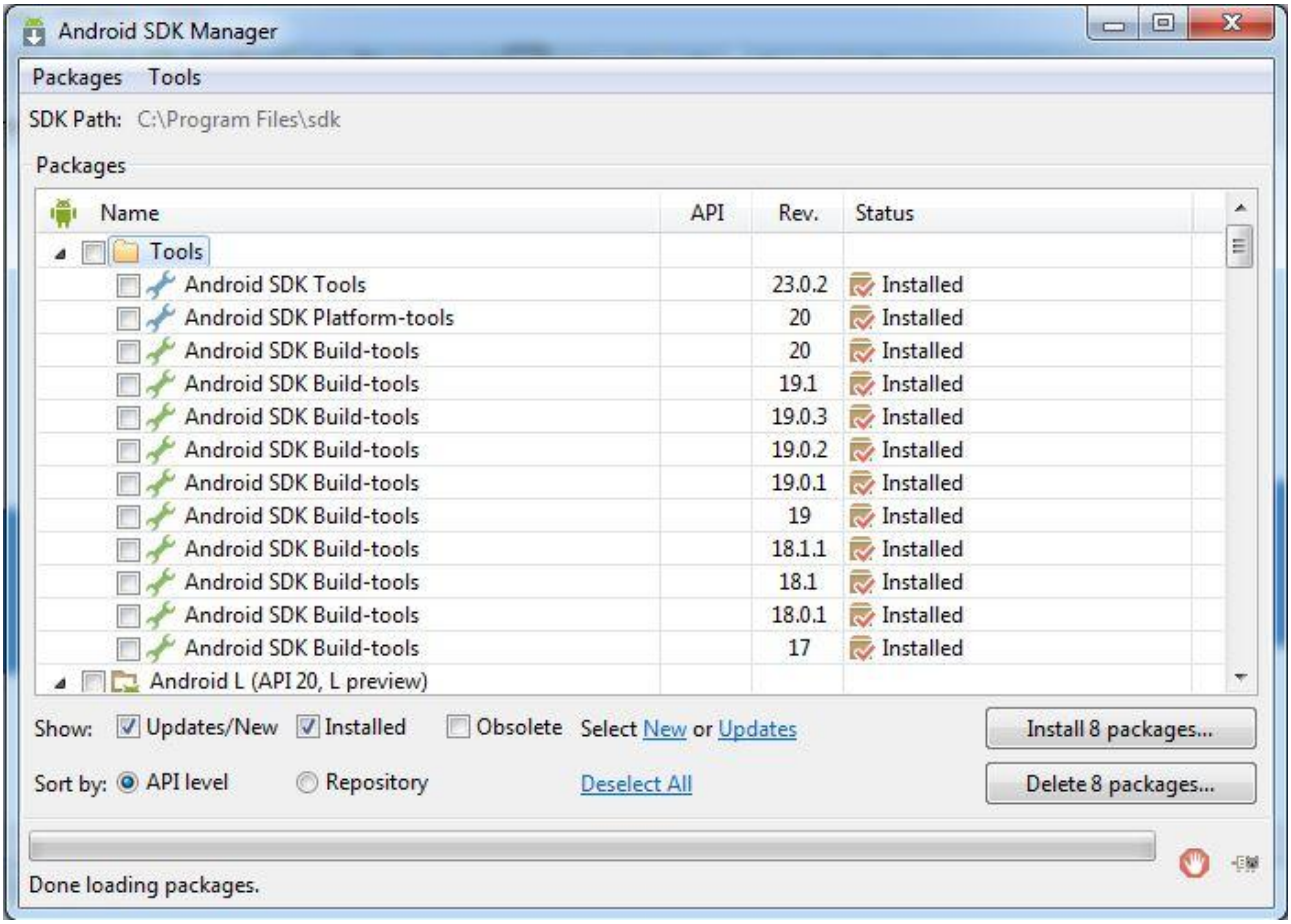
Главна хардверска платформа за Андроид је *32-bit ARMv7* архитектура. Најчешћи процесори на андроид платформама, као што су телефони, су произвођача *Intel*. За новије Андроид уређаје са Андроид 4.4 верзијом оперативног система препоручује се најмање 512MB *RAM*[3].

Апликације за Андроид се раде у *Java* програмском језику користећи *Software Development Kit (SDK)*, али постоје и други развојни алати. Андроид *SDK* садржи дебагер, библиотеке и емулатор Андроид мобилног уређаја[4].

На веб адреси http://developer.android.com/index.html могуће је наћи корисне информације у вези програмирања Андроид апликација, алате потребне за њихово програмирање као и туторијале. На адреси http://developer.android.com/sdk/index.html могуће је бесплатно преузети *SDK* надограђен на *Eclipse IDE (Integrated Development Environment).* Такође је могуће *Eclipse IDE* бесплатно преузети директно са веб сајта проивођача тог развојног окружења (https://www.eclipse.org/) а затим га надоградити *SDK* библиотеком преузетом са веб странице http://developer.android.com/sdk/installing/index.html?pkg=tools.

По преузимању *Eclipse IDE* верзије *Luna* са сајта произвођача, и распакивања *rar* фајла, потребно је преузети и инсталирати *Java SE Development Kit* са веб сајта компаније *Oracle* http://www.oracle.com/index.html како би било могуће покренути *Eclipse*. Сад је потребно инсталирати *ADT (Android Deweloper Tools)* селектовањем на *Help > Install New Software,* кликом на *Add* и уписивањем *https:/dl-ssl.google.com/android/eclipse/* у поље *Location*[8]. Пошто већ постоји инсталирано развојно окружење, на веб страници за преузимање *SDK* бира се опција *Get the SDK for an existing IDE* и преузимање се врши одатле. Како би се *Eclipse* развојно окружење повезало са *Android SDK Manager* алатом, потребно је у дијалогу који се отвара кад се покрене *Eclipse* после инсталације *ADT* алата, унети адресу на којој се на хард диску налази *sdk* директоријум. Пошто је то урађено, покреће се *Android SDK Manager* и иконе за њега и *Android Virtual Device Manager* сад се налазе под опцијом *Windows*. Сад је потребно инсталирати алате за програмирање Андроид апликација. Препоручујемо инсталацију свих алата које нуди *Android SDK Manager* у одељку

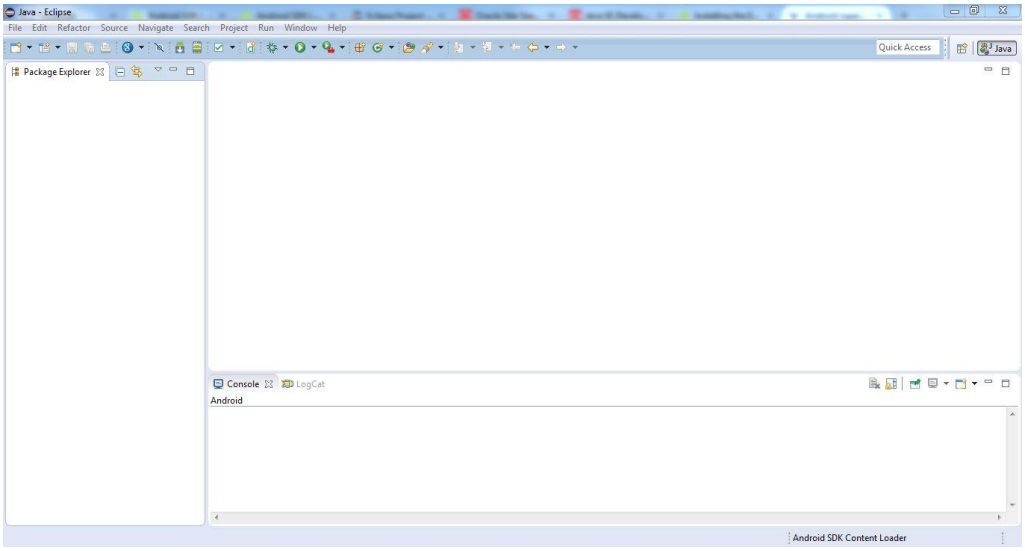
*Tools* као и у осталим одељцима. По инсталацији, развојно окружење је спремно за развој Андроид апликација.



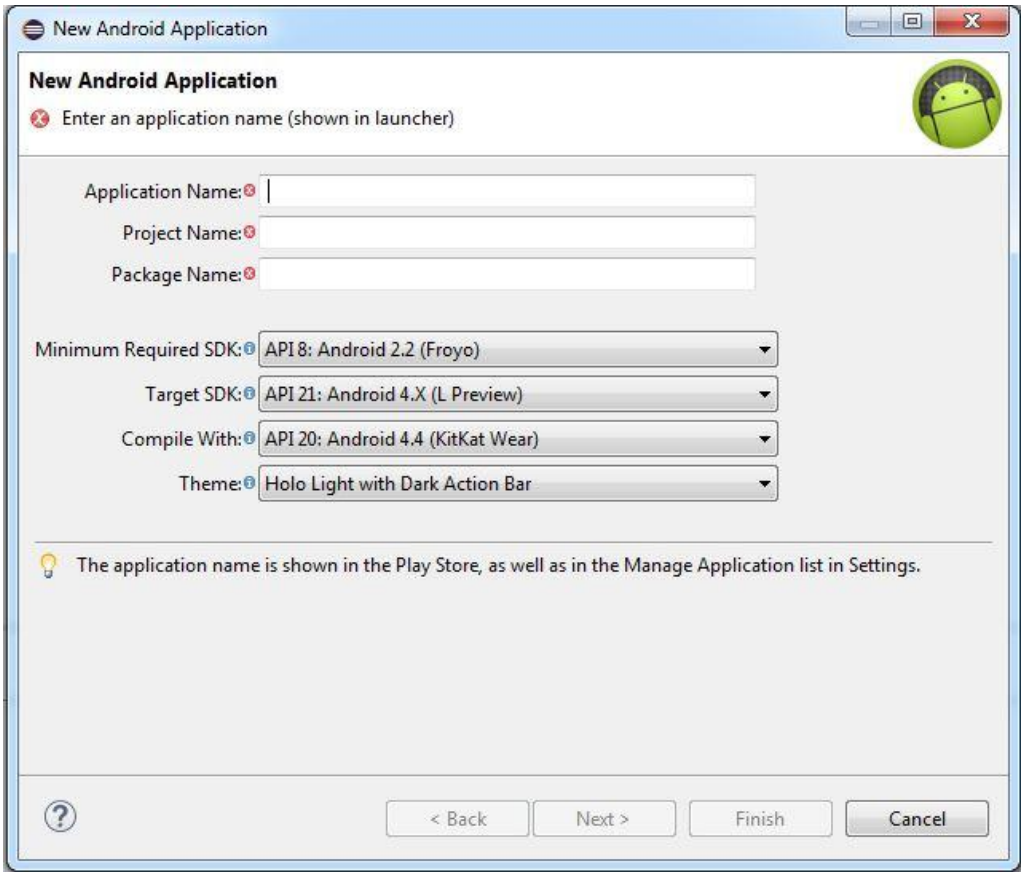

***Слика 2.1.1 Прозор Android SDK Manager алата***

## 2.2. Eclipse IDE развојно окружење

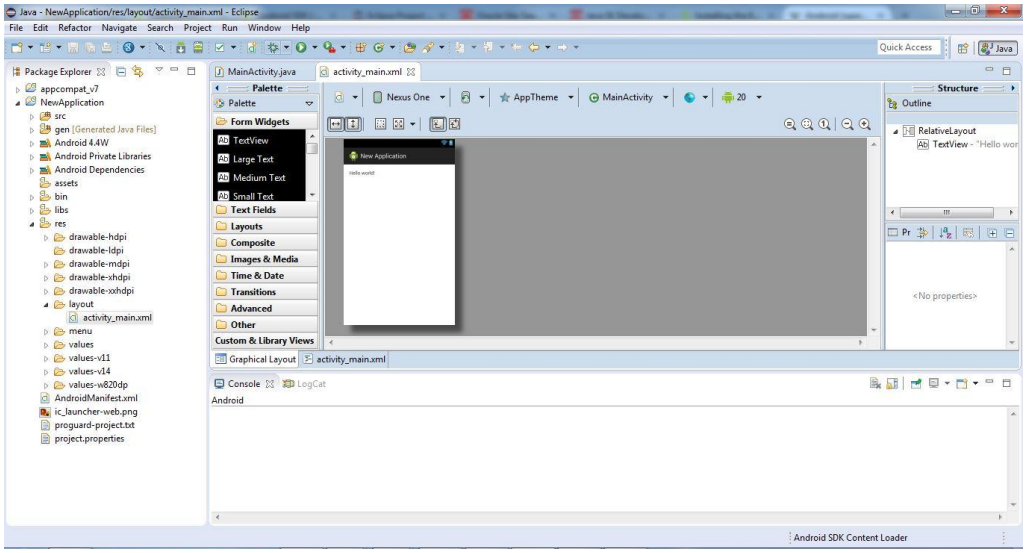
Изглед тек инсталираног *Eclipse IDE* приказан је на слици 2.2.1. Селектовањем *File > New > Android Application Project* започиње се рад на новој Андроид апликацији. При првом покретању, тражи се избор фолдера у ком ће се вршити упис и читање апликација које корисник прави. Изглед прозора за прављење нове Андроид апликације дат је на слици 2.2.2. Име апликације уноси се у поље *Application Name*, кликће се на дугме *Next* на овом и наредним прозорима и на крајњем се кликће на дугме *Finish*. Свака нова апликација креће као једноставна апликација која исписује *Hello world!* поруку. Изглед развојног окружења по прављењу нове Андроид апликације, приказан је на слици 2.2.3.




*Слика 2.2.1. Тек инсталиран Eclipse IDE*




*Слика 2.2.2 Прозор за прављење нове Андроид апликације*



***Слика 2.2.3 Изглед развојог окружења по прављењу нове Андроид апликације***

Аутоматски су направљени *activity_main.xml* и *MainActivity.java* фајлови и њихови кодови биће приказани у наредном тексту.

Код *xml* фајла:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.newapplication.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Следећи код представља маргине. У овом коду се позивају вредности из фајла *dimens.xml* из фолдера res/values који се може наћи у *Package Values* прозору. У овом фајлу се налазе вредности за величине које корисник жели стандардизовати и касније позивати из овог фајла. Такође је могуће унети димензије директним уносом вредности у овај код.

```
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
```

Осим *dimens.xml*, у истом фолдеру могу се наћи и *strings.xml* и *styles.xml*, чија је сврха иста, специфицирање дизајна интерфејса апликације. У фајлу *strings.xml* налазе се стрингови које корисник жели често користити унутар апликације, а у *styles.xml* се налазе боје за текстове, позадине и слично. Као што је било могуће унети директним уносом у код вредност за

маргине и било које друге димензије, тако је могућ директан унос у код активитија за стилове и стрингове.

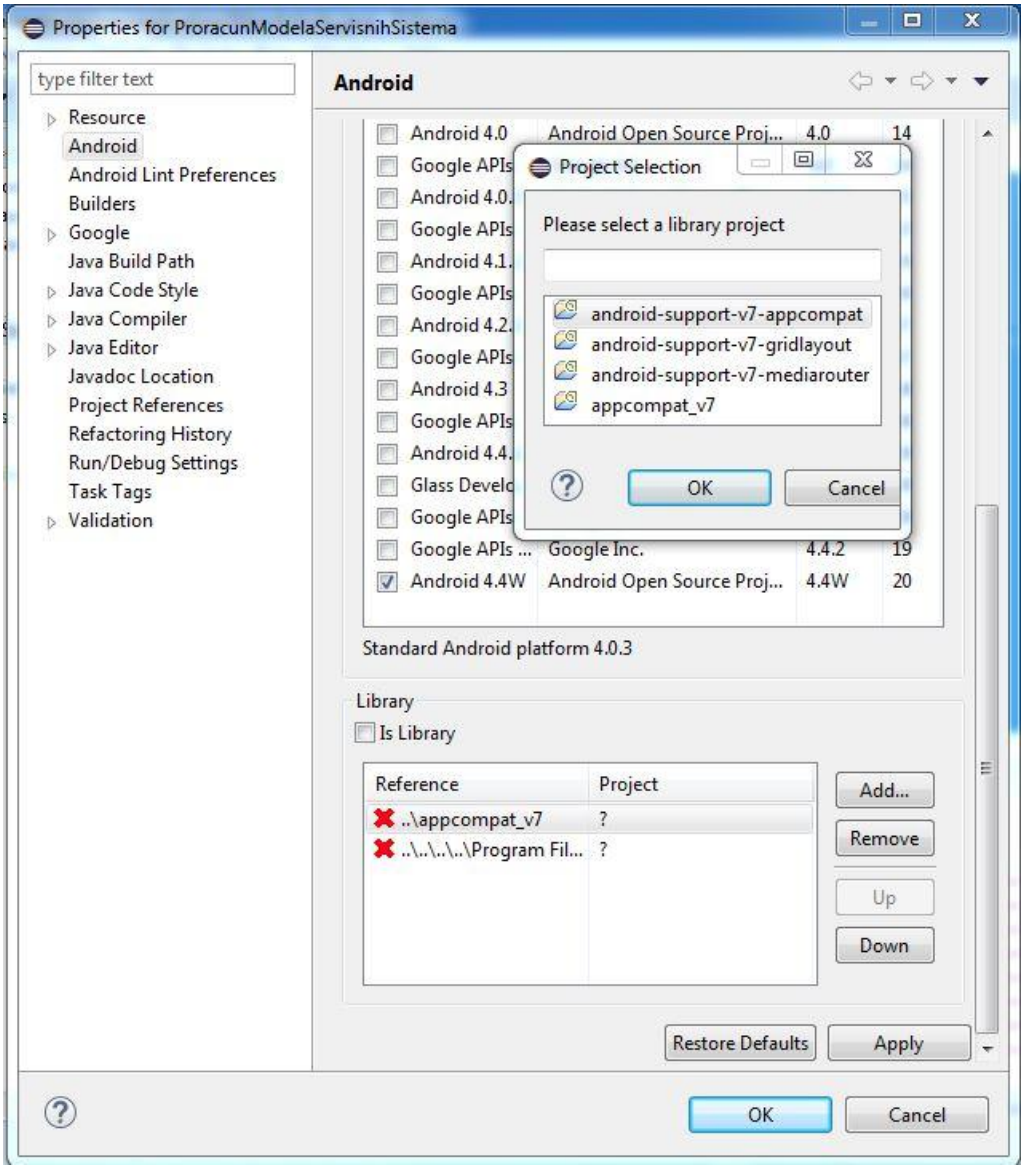
Код *Java* фајла:

```
package com.example.newapplication;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		if (id == R.id.action_settings) {
			return true;
		}
		return super.onOptionsItemSelected(item);
	}
}
```

*MainActivity* класа се аутоматски генерише приликом прављења новог активитија, и унутар ње корисник програмира своју апликацију.

Импортовање већ постојећег пројекта се врши десним кликом у прозору *Package Explorer*, и селектовањем опције *Import*. У прозору који се отвара потребно је изабрати опцију *Existing Android Code into Workspace* и унети адресу пројекта. По уношењу пројекта приложеног у овом раду, потребно га је повезати са библиотекама које тај пројект користи. Те библиотеке се налазе у фолдеру *sdk* на адреси *extras/android/support/v7/*. Потребно је унети библиотеку *appcompat_v7*, на исти начин као што се уноси већ постојећи пројект, што је објашњено у претходном тексту. Кад је то обављено, потребно је селектовати десним кликом у прозору *Package Explorer* пројект који је приложен у овом раду, и изабрати опцију *Properties*. На прозору који се отвара потребно је изабрати са леве стране опцију *Android* и на дну прозора под делом *Library* селектовати *..\appcompat_v7* и кликнути на дугме *Add*. У прозору који се затим отвара, потребно је селектовати *android-support-v7-appcompat* и кликнути на дугме *OK*. Пошто је то обављено, потребно је кликнути на дугме OK на прозору

*Properties*, и повезивање са библиотекама је сад успешно обављено. Изглед прозора *Properties* је приказан на слици 2.2.4.




***Слика 2.2.4 Прозор Properties и прозор који се отвара после клика на дугме Add***

Још једна важна ставка је покретање апликације, на емулатору или реалном уређају. Покретање на реалном уређају се показало као далеко брже од покретања на емулатору, због веома дугог времена потребног за покретање самог емулатора. У случају реалног уређаја, пре употребе, потребно је инсталирати *OEM (Original Equipment Manufacturer)* драјвере. Комплетна инсталација је објашњена на адреси http://developer.android.com/tools/extras/oem-usb.html. У случају емулатора, потребно је селектовати опцију *Android Virtual Device (AVD) Manager*. У том прозору могуће је креирати жељну врсту уређаја који се жели емулирати, или клонирати већ дефинисане уређаје на табу *Device Definitions*. Апликација се покреће селектовањем стрелице поред дугмета *Run* и избором опције *Run As > Android Application.* На прозору који се отвара потребно је изабрати жељен уређај, реалан или емулиран, и кликнути на дугме *OK*.

За програмирање Андроид апликација потребно је основно познавање програмских језика као што су *C* и *Java.* Рад у *xml* делу пројекта је лако разумљив, и олакшан широком палетом објеката који се, осим директним уносом кода, могу унети и избором из палете објеката и постављањем на симулацију екрана мобилног уређаја. Скренуо бих пажњу на ситуацију иницијализовања промењљиве типа *double* у *Java* програмском језику. У случају иницијализовања целобројним бројем, програмски језик га неће аутоматски пребацити у децималан, већ ће га посматрати као промењљиву типа *integer* и доделити му опсег тог типа бројева.

# 3. Структура кода апликације

У овом поглављу је објашњена структура кода апликације. Састоји се од *Java* фајлова који обављају функције апликације, и њима придружених *xml* фајлова који праве кориснички интерфејс апликације.

Апликација се састоји из четири одвојена дела, у зависности од сервисног система који тај део апликације покрива. Надаље, унутар сваког сервисног система, апликација је подељена на три до пет делова, од којих последња два чине алгоритам за цртање једноструких графика и алгоритам за цртање вишеструких графика, док преостали чине карактеристичне прорачуне унутар сервисног система, којих има од једног до три, у зависности од система. Сви прорачуни, унутар делова за прорачуне, као и делова за цртање једноструких и вишеструких графика, узети су из Анекса А скрипти за предмет "Комутациони системи" на Електротехничком Факултету у Београду[1].

## 3.1. Прорачуни

Андроид апликација се састоји од активитија *(Activity)*. Активити представља прозор интерфејса, који обавља одређену функцију (активност). Сваки активити има свој *xml* фајл који прави кориснички интерфејс прозора, дугмиће *(Button)*, просторе за унос података *(EditText)*, падајуће меније *(Spinner)*, лабеле за испис текстова *(TextView)*, листе података *(ListView)*, кућице за чекирање *(CheckBox)* и остало. Java фајл активитија је придружен *xml* фајлу, и обавља функције које се дешавају пошто корисник обави инпут на интерфејсу.


***Слика 3.1.1-2 Интерфејс активитија activity_prvi_sistem_racun_multi_graph.xml***

Фајлови којима се у имену појављује "четврти систем" представљају фајлове M/M/m/m сервисног система. Ако се појављује "трећи систем" ради се о фајловима M/M/m сервисног

система. "Други систем" представља фајлове M/G/1 сервисног система, док "први систем" представља M/M/m/m/l.

Код *xml* фајла за активити са слике 3.1.1-2:

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:background="@drawable/c3"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

tools:context="com.drugradun.proracunmodelaservisnihsistema.PrviSistemRacunMultiGraph"
>

      <TableLayout
            android:id="@+id/layout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:orientation="vertical" >

            <TextView
                android:id="@+id/textView4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:textSize="18sp" />

            <TableRow
                android:id="@+id/tableRow1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" >

                <EditText
                    android:id="@+id/editText3"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:ems="10"
                    android:inputType="numberDecimal" />

             <CheckBox
                 android:id="@+id/checkBox1"
                 android:layout_width="wrap_content"
                    android:layout_height="wrap_content"/>

            </TableRow>

            <TableRow
                android:id="@+id/tableRow2"
                android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

 <CheckBox
     android:id="@+id/checkBox2"
     android:layout_width="wrap_content"
       android:layout_height="wrap_content"/>

</TableRow>

<TableRow
    android:id="@+id/tableRow3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

 <CheckBox
     android:id="@+id/checkBox3"
     android:layout_width="wrap_content"
       android:layout_height="wrap_content"/>

</TableRow>

<TableRow
    android:id="@+id/tableRow4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

 <CheckBox
     android:id="@+id/checkBox4"
     android:layout_width="wrap_content"
       android:layout_height="wrap_content"/>

</TableRow>

<TableRow
    android:id="@+id/tableRow5"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

 <CheckBox
     android:id="@+id/checkBox5"
     android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

</TableRow>

 <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:textSize="18sp" />

<TableRow
    android:id="@+id/tableRow6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

</TableRow>

<TableRow
    android:id="@+id/tableRow7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

</TableRow>

<TableRow
    android:id="@+id/tableRow8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditText
```

```
        android:id="@+id/editText10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

</TableRow>

<TableRow
    android:id="@+id/tableRow9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

</TableRow>

<TableRow
    android:id="@+id/tableRow10"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal" />

</TableRow>

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="@string/donja_granica"
    android:textSize="18sp" />

<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="numberDecimal" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="@string/gornja_granica"
```

```
            android:textSize="18sp" />

        <EditText
            android:id="@+id/editText2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="numberDecimal" />

        <Button
            android:id="@+id/button1_4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="@string/calc" />

    </TableLayout>
</ScrollView>
```

Део за прорачуне, као и делови за цртање једноструких и вишеструких графика састоје се од серије узастопних активитија у којима се прикупљају кориснички подаци, и крајњег активитија који врши прорачун и исписује резултат.

Код *Java* фајла за активити са слике 3.1.1-2:

```
package com.drugradun.proracunmodelaservisnihsistema;

import android.support.v7.app.ActionBarActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;

public class PrviSistemRacunMultiGraph extends ActionBarActivity {
	private CheckBox check1, check2, check3, check4, check5;
	private Button btnSubmit;
	int sgn1=0, sgn2=0, sgn3=0, sgn4=0, sgn5=0;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_prvi_sistem_racun_multi_graph);
		addListenerOnButton();
		addListenerOnChk1();
		addListenerOnChk2();
		addListenerOnChk3();
		addListenerOnChk4();
		addListenerOnChk5();
		addTextOnTextView();
	}

	public void addTextOnTextView(){
		Intent intent2 = getIntent();
```

```
        String[] myStrings = intent2.getStringArrayExtra("strings");
        int a=Integer.parseInt(myStrings[2]);
        if (a==0){
               TextView textView1 = (TextView) findViewById(R.id.textView4);
               textView1.setText("Uneti do 5 vrednosti za m");
               TextView textView2 = (TextView) findViewById(R.id.textView1);
               textView2.setText("Uneti do 5 vrednosti za l");
        }
        if (a==1){
               TextView textView1 = (TextView) findViewById(R.id.textView4);
               textView1.setText("Uneti do 5 vrednosti za A");
               TextView textView2 = (TextView) findViewById(R.id.textView1);
               textView2.setText("Uneti do 5 vrednosti za l");
        }
        if (a==2){
               TextView textView1 = (TextView) findViewById(R.id.textView4);
               textView1.setText("Uneti do 5 vrednosti za A");
               TextView textView2 = (TextView) findViewById(R.id.textView1);
               textView2.setText("Uneti do 5 vrednosti za m");
        }
}
 public void addListenerOnChk1() {
              check1 = (CheckBox) findViewById(R.id.checkBox1);
              check1.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                     if (((CheckBox) v).isChecked()) {
                            sgn1=1;
                     }
                     else{
                            sgn1=0;
                     }
                }
              });

        }
  public void addListenerOnChk2() {
              check2 = (CheckBox) findViewById(R.id.checkBox2);
              check2.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                     if (((CheckBox) v).isChecked()) {
                            sgn2=1;
                     }
                     else{
                            sgn2=0;
                     }
                }
              });

        }
  public void addListenerOnChk3() {
              check3 = (CheckBox) findViewById(R.id.checkBox3);
              check3.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                     if (((CheckBox) v).isChecked()) {
```

```
                        sgn3=1;
                    }
                    else{
                        sgn3=0;
                    }
                }
            });

        }
    public void addListenerOnChk4() {
            check4 = (CheckBox) findViewById(R.id.checkBox4);
            check4.setOnClickListener(new OnClickListener() {
              @Override
              public void onClick(View v) {
                    if (((CheckBox) v).isChecked()) {
                        sgn4=1;
                    }
                    else{
                        sgn4=0;
                    }
              }
            });

        }
    public void addListenerOnChk5() {
            check5 = (CheckBox) findViewById(R.id.checkBox5);
            check5.setOnClickListener(new OnClickListener() {
              @Override
              public void onClick(View v) {
                    if (((CheckBox) v).isChecked()) {
                        sgn5=1;
                    }
                    else{
                        sgn5=0;
                    }
              }
            });

        }
private void addListenerOnButton() {
      btnSubmit = (Button) findViewById(R.id.button1_4);
      btnSubmit.setOnClickListener(new OnClickListener() {

              @Override
              public void onClick(View v) {
                      EditText editText1 = (EditText)
                      findViewById(R.id.editText1);
                      String message1 = editText1.getText().toString();
                      EditText editText2 = (EditText)
                      findViewById(R.id.editText2);
                      String message2 = editText2.getText().toString();
                      String message4 = "0";
                      String message5 = "0";
                      String message7 = "0";
                      String message8 = "0";
                      String message10 = "0";
                      String message11 = "0";
```

```
String message13 = "0";
String message14 = "0";
String message16 = "0";
String message17 = "0";
String message3 = Integer.toString(sgn1);
if(sgn1==1){
       EditText editText3 = (EditText)
       findViewById(R.id.editText3);
       message4 = editText3.getText().toString();
        EditText editText8 = (EditText)
        findViewById(R.id.editText8);
       message5 = editText8.getText().toString();
}
String message6 = Integer.toString(sgn2);
if(sgn2==1){
       EditText editText4 = (EditText)
       findViewById(R.id.editText4);
       message7 = editText4.getText().toString();
       EditText editText5 = (EditText)
       findViewById(R.id.editText9);
       message8 = editText5.getText().toString();
}
String message9 = Integer.toString(sgn3);
if(sgn3==1){
       EditText editText6 = (EditText)
       findViewById(R.id.editText5);
       message10 = editText6.getText().toString();
       EditText editText7 = (EditText)
       findViewById(R.id.editText10);
       message11 = editText7.getText().toString();
}
String message12 = Integer.toString(sgn4);
if(sgn4==1){
       EditText editText8 = (EditText)
       findViewById(R.id.editText6);
       message13 = editText8.getText().toString();
       EditText editText9 = (EditText)
       findViewById(R.id.editText11);
       message14 = editText9.getText().toString();
}
String message15 = Integer.toString(sgn5);
if(sgn5==1){
       EditText editText10 = (EditText)
       findViewById(R.id.editText7);
       message16 = editText10.getText().toString();
       EditText editText11 = (EditText)
       findViewById(R.id.editText12);
       message17 = editText11.getText().toString();
}

 Intent intent1 = getIntent();
String[] myStrings =
intent1.getStringArrayExtra("strings");
Intent intent = new
Intent(PrviSistemRacunMultiGraph.this,
PrviSistemMultiGraphCrt.class);
```

```
                                String[] myStringArray = new String[]{myStrings[0],
                                myStrings[1], myStrings[2], myStrings[3], message1,
                                message2, message3, message4, message5, message6,
                                message7, message8, message9, message10, message11,
                                message12, message13, message14, message15, message16,
                                message17};
                                intent.putExtra("strings", myStringArray);
                                startActivity(intent);


                               }

                        });

        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.prvi_sistem_racun_multi_graph, menu);
                return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
                // Handle action bar item clicks here. The action bar will
                // automatically handle clicks on the Home/Up button, so long
                // as you specify a parent activity in AndroidManifest.xml.
                int id = item.getItemId();
                if (id == R.id.action_settings) {
                        return true;
                }
                return super.onOptionsItemSelected(item);
        }
}
```

Крајњи прорачуни и исписивање се врше у фајловима:

- CetvrtiSistemRacun1.java
- CetvrtiSistemRacun2.java
- CetvrtiSistemRacun3.java
- DrugiSistemRacun11.java
- DrugiSistemRacun12.java
- TreciSistemRacun1.java
- TreciSistemRacun2.java
- PrviSistemRacun.java

Како би се пратили кориснички уноси у објектима, унутар *protected void onCreate(Bundle savedInstanceState)* уводи се:

```
addListenerOnButton();
addListenerOnChk1();
```

Код *addListenerOnButton();* прати да ли је корисник притиснуо дугме, и после притиска на дугме обавља операције које су додељене дугмету. У овом случају, приликом

притиска на дугме иде се у следећи активити. То обавља код који се налази унутар *private void addListenerOnButton()*. Обавља се ишчитавање података које унео корисник, и претвара се у низ стрингова, који се уписују у *intent*. *Intent* је класа која представља везу међу различитим активитијима апликације. Убацивањем у *intent*, подаци прикупљени у једном активитију постају расположиви и осталим активитијима апикације. То се обавља следећим кодом:

```
String[] myStringArray = new String[]{myStrings[0], myStrings[1], myStrings[2],
myStrings[3], message1, message2, message3, message4, message5, message6,
message7, message8, message9, message10, message11, message12, message13,
message14, message15, message16, message17};
intent.putExtra("strings", myStringArray);
startActivity(intent);
```

Код *addListenerOnChk1()* прати да ли је корисник чекирао чекбокс и обавља функције придружене чекбоксу. У овом случају, променљивој *sgn1* која је придружена првом чекбоксу, додељује се вредност 1 ако је чекирана и 0 ако није. То се обавља следећим кодом:

```
public void addListenerOnChk1() {
	check1 = (CheckBox) findViewById(R.id.checkBox1);
	check1.setOnClickListener(new OnClickListener() {
	@Override
	public void onClick(View v) {
		if (((CheckBox) v).isChecked()) {
			sgn1=1;
			}
			else{
				sgn1=0;
			}
		}
	});

}
```

Код *addTextOnTextView()* служи за упис предефинисаних вредности у простор за унос података, а те вредности је могуће после променити. Ово је коришћено за приказивање дифолт вредности за опсег параметра *x* осе. Код који ово обавља је:

```
public void addTextOnTextView(){
	Intent intent2 = getIntent();
	String[] myStrings = intent2.getStringArrayExtra("strings");
	int a=Integer.parseInt(myStrings[2]);
	if (a==0){
		TextView textView1 = (TextView) findViewById(R.id.textView4);
		textView1.setText("Uneti do 5 vrednosti za m");
		TextView textView2 = (TextView) findViewById(R.id.textView1);
		textView2.setText("Uneti do 5 vrednosti za l");
	}
	if (a==1){
		TextView textView1 = (TextView) findViewById(R.id.textView4);
		textView1.setText("Uneti do 5 vrednosti za A");
		TextView textView2 = (TextView) findViewById(R.id.textView1);
		textView2.setText("Uneti do 5 vrednosti za l");
	}
	if (a==2){
		TextView textView1 = (TextView) findViewById(R.id.textView4);
		textView1.setText("Uneti do 5 vrednosti za A");
```

```
            TextView textView2 = (TextView) findViewById(R.id.textView1);
            textView2.setText("Uneti do 5 vrednosti za m");
        }
    }
```

У наредном тексту, биће приказани делови кода фајла PrviSistemRacun.java фајла и наведена њихова намена. Овим делом кода се ишчитавају подаци прослеђени из претходних активитија.

```
Intent intent = getIntent();
String[] myStrings = intent.getStringArrayExtra("strings");
double lambda = Double.parseDouble(myStrings[0]);
double mi = Double.parseDouble(myStrings[1]);
int m = Integer.parseInt(myStrings[2]);
int l = Integer.parseInt(myStrings[3]);
int sgn = Integer.parseInt(myStrings[4]);
```

Део кода који прорачунава вредност параметра $p_0$ за систем M/M/m/m/l.

```
double factl=1.0;
for (int j=1; j<=l; j++){
      factl=factl*j;
}

double sum=0.0;
for (int i=0; i<=m; i++){

      double factn=1.0;
      for (int j=1; j<=i; j++){
            factn=factn*j;
      }
      double factln=1.0;
      for (int j=1; j<=(l-i); j++){
            factln=factln*j;
      }
      sum=sum+factl/(factn*factln)*Math.pow(r,i);
}

double p0=1/sum;
```

Резултати се уписују у *ListView*, имплементран у придруженом *xml* фајлу горе наведеног *Java* фајла, овим кодом.

```
String[] values = new String[2*m+10];

values[0]="λ: "+Double.toString(lambda);
if(sgn==0){
      values[1]="μ: "+Double.toString(mi);
}
else {
      values[1]="1/μ: "+Double.toString(1/mi);
}
values[2]="m: "+Integer.toString(m);
values[3]="l: "+Integer.toString(l);
values[4]="A: "+Double.toString(A);
values[5]="As: "+Double.toString(As);
values[6]="B: "+Double.toString(B);
values[7]="Pl: "+Double.toString(Pl);
for (int i=8; i<m+9; i++){
```

```
        values[i]="p"+Integer.toString(i-8)+": "+ Double.toString(pArray[i-8]);
        }
 for (int i=m+9; i<2*m+10; i++){
        values[i]="q"+Integer.toString(i-m-9)+": "+ Double.toString(qArray[i-m-
        9]);
        }

ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, android.R.id.text1,
values);
listView.setAdapter(adapter);
```

У падајуће меније могуће је унети податке преко *Java* као и *xml* фајла. У овом раду је коришћен унос преко *Java* фајла. Пример кода за унос података на падајући мени из *Proracun1CetvrtiSistem.java*:

```
private void addItemsOnSpinner1() {
        spinner1 = (Spinner) findViewById(R.id.spinner1);
        List<String> list = new ArrayList<String>();
        list.add("μ");
        list.add("1/μ");
        ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
              android.R.layout.simple_spinner_item, list);

        dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
        wn_item);
        spinner1.setAdapter(dataAdapter);

        }
```

Овај код прави падајући мени са излистаним $\mu$ и $1/\mu$. По притиску на дугме, ишчитава се избор у падајућем менију помоћу следећег кода:

```
String str=String.valueOf(spinner1.getSelectedItem());
double pom=Double.parseDouble(message2);
if (str.equals("1/μ")){
        pom=1/pom;
        message2=Double.toString(pom);
        sgn=1;
        }
```

Помоћна променљивом *sgn1* прати се избор из падајућег менија, који ће бити 1 за $1/\mu$ и 0 за $\mu$.

## 3.2. Цртање једноструких и вишеструких графика

Делови за цртање једноструких и вишеструких графика, апликације, имају идентичан начин прикупљања корисничких параметара као у делу за прорачун. Разлика је што, осим прорачуна, крајњи активитији врше и исцртавање графика за прорачунате вредности. Цртање графика се обавља имплементацијом *Android GraphView*, апликације за цртање графика[5].

Прорачун се врши унутар кориснички унетог опсега параметра са $x$ осе, резолуцијом 100 (у 101 тачки), у случају параметара који не морају бити целобројни. У случају параметара који морају бити целобројни (број сервера, број корисника итд.), тачке су све целобројне вредности унутар опсега. Прорачунате вредности се чувају у низовима (у случају једноструких графика) и матрицама (у случају вишеструких график). Вредности из низова и

матрица се на крају графички исцртавају. Различити прорачуни и исцртавања се врше у зависности од кориснички изабраних параметара за *x* и *y* осе.

Исцртавање графика се врши у следећим *Java* фајловима:

- CetvrtiSistemGraphCrt.java
- CetvrtiSistemMultiGraphCrt.java
- DrugiSistemGraphCrt.java
- DrugiSistemMultiGraphCrt.java
- PrviSistemGraphCrt.java
- PrviSistemMultiGraphCrt.java
- TreciSistemGraphCrt.java
- TreciSistemMultiGraphCrt.java

Део кода из фајла CetvrtiSistemMultiGraphCrt.java који служи за исцртавање вишеструких графика у случају кад је параметар $A$ на $x$ оси, а параметар $A_s$ на $y$ оси.

```
int num = 101;
GraphViewData[] data1 = new GraphViewData[num];
GraphViewData[] data2 = new GraphViewData[num];
GraphViewData[] data3 = new GraphViewData[num];
GraphViewData[] data4 = new GraphViewData[num];
GraphViewData[] data5 = new GraphViewData[num];
for (int i=0; i<num; i++) {
	data1[i] = new GraphViewData(i, i);
	data2[i] = new GraphViewData(i, i);
	data3[i] = new GraphViewData(i, i);
	data4[i] = new GraphViewData(i, i);
	data5[i] = new GraphViewData(i, i);
	}
for (int i=0; i<num; i++) {
	int pom=0;
	if (sgn3==1){
		data1[i] = new GraphViewData(AArray[i], AsArray[i][pom]);
		pom=pom+1;
		}
	if (sgn4==1){
	data2[i] = new GraphViewData(AArray[i], AsArray[i][pom]);
	pom=pom+1;
	}
if (sgn5==1){
	data3[i] = new GraphViewData(AArray[i], AsArray[i][pom]);
	pom=pom+1;
	}
if (sgn6==1){
	data4[i] = new GraphViewData(AArray[i], AsArray[i][pom]);
	pom=pom+1;
	}
if (sgn7==1){
	data5[i] = new GraphViewData(AArray[i], AsArray[i][pom]);
	pom=pom+1;
	}
}
GraphView graphView = new LineGraphView(this, "y osa: As | x osa: A");
```

```
if(sgn3==1){
	GraphViewSeries graph1 = new GraphViewSeries("y osa: As1 | x osa: A", new
	GraphViewSeriesStyle(Color.RED, 3), data1);
	graphView.addSeries(graph1);
	}
if(sgn4==1){
	GraphViewSeries graph2 = new GraphViewSeries("y osa: As2 | x osa: A", new
	GraphViewSeriesStyle(Color.BLUE, 3), data2);
	graphView.addSeries(graph2);
	}
if(sgn5==1){
	GraphViewSeries graph3 = new GraphViewSeries("y osa: As3 | x osa: A", new
	GraphViewSeriesStyle(Color.GREEN, 3), data3);
	graphView.addSeries(graph3);
	}
if(sgn6==1){
	GraphViewSeries graph4 = new GraphViewSeries("y osa: As4 | x osa: A", new
	GraphViewSeriesStyle(Color.GRAY, 3), data4);
	graphView.addSeries(graph4);
	}
if(sgn7==1){
	GraphViewSeries graph5 = new GraphViewSeries("y osa: As5 | x osa: A", new
	GraphViewSeriesStyle(Color.BLACK, 3), data5);
	graphView.addSeries(graph5);
	}
graphView.setViewPort(donja, gornja-donja);
graphView.setScalable(true);
graphView.getGraphViewStyle().setTextSize(getResources().getDimension(R.dimen.pie
_segment_label_font_size));
graphView.getGraphViewStyle().setNumHorizontalLabels(5);
graphView.getGraphViewStyle().setNumVerticalLabels(4);
graphView.setShowLegend(true);
graphView.setLegendAlign(LegendAlign.BOTTOM);
graphView.setLegendWidth(200);
LinearLayout layout =(LinearLayout) findViewById (R.id.subLayout);
layout.addView(graphView);
```

*GraphViewData[]* је посебна врста података које користи *Android GraphView* апликација. Ти подаци представљају вредности на *x* и *y* осама за сваку тачку. Код за упис података у промењљиву *GraphViewData[]* типа изгледа овако.

```
GraphViewData[] data1 = new GraphViewData[num];
.
.
.
for (int i=0; i<num; i++) {
	data1[i] = new GraphViewData(i, i);
	.
	.
	.
	}
```

У наведеном коду се уносе безначајни подаци, пошто се у том тренутку још не зна број криви које корисник жели исцртати. У следећем делу кода се испитује да ли корисник жели исцртати криву са редним бројем 1, и у случају потврдног одговора, уносе се корисни подаци добијен у прорачуну.

```
if(sgn3==1){
	GraphViewSeries graph1 = new GraphViewSeries("y osa: As1 | x osa: A", new
	GraphViewSeriesStyle(Color.RED, 3), data1);
	graphView.addSeries(graph1);
	}
```

Пре уношења информација типа *GraphViewData[]*, мора се направити сам график. То се ради помоћу следећег кода.

```
GraphView graphView = new LineGraphView(this, "y osa: As | x osa: A");
```

На тај начин се прави сам график, и даје му се име, у овом случају то је "y osa: As | x osa: A". Име је могуће исписати изнад самог графика. Пошто апликација за цртање графика не омогућава исписивање имена променљивих које су представљене на осама, име графика је у овој апликацији искоришћено у ту сврху. Следећи код одређује део графика који ће бити приказан.

```
graphView.setViewPort(donja, gornja-donja);
```

Приказана је легенда и омогућено је умањивање и увећавање криви приказаних на графику и то је омогућено следећим кодом.

```
graphView.setScalable(true);
graphView.setShowLegend(true);
```

На самом крају, график се исцртава на екрану уређаја следећим кодом.

```
LinearLayout layout =(LinearLayout) findViewById (R.id.subLayout);
layout.addView(graphView);
```

# 4. УПУТСТВО ЗА КОРИШЋЕЊЕ АПЛИКАЦИЈЕ

У овом поглављу биће објашњен и приближен начин коришћења апликације кориснику. Биће приказани примери рада апликације за одређене унесене вредности, и приказани резултати, графици и вишеструки графици.

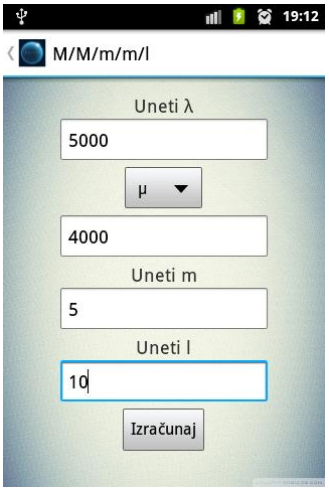
## 4.1. Опште упутство

Апликација се може покренути на емулатору или на реалном мобилном уређају који је прикључен на рачунар. У оба случаја, програм се покреће опцијом *Run as* > *Application Project* и избором жељеног начина покретања апликације.

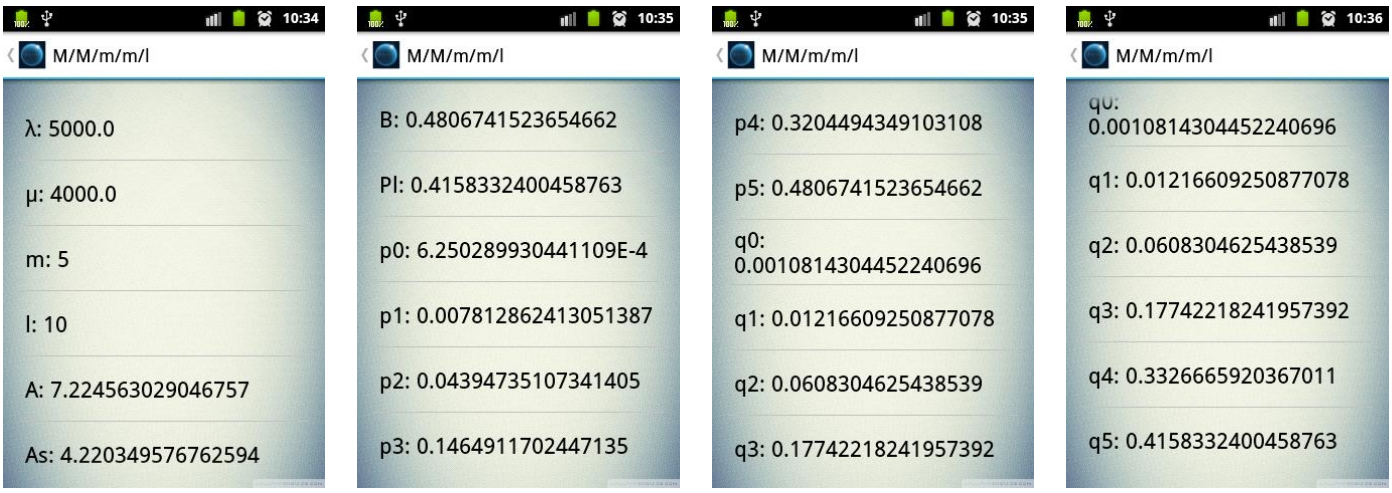
На почетном интерфејсу апликације врши се избор сервисног система за који су потребни прорачуни. Испод дугмића за четири сервисна система налази се дугме за упутство, у ком су рашчишћене недоумице у вези уноса параметара и коришћења функција апликације. Као што је у уводу већ наведено, апликација покрива прорачуне за *M/M/m, M/M/m/m, M/M/m/m/l* и *M/G/1* сервисне системе. Графици имају омогућено увећавање и умањивање стављањем два прста на график и њиховим ширењем и скупљањем, респективно.

## 4.2. *M/M/m/m/l сервисни систем*

*M/M/m/m/l* је први понуђен на главном интерфејсу. Обезбеђен је један тип прорачуна за овај систем, који за унето $\lambda$ (број корисника у јединици времена), $\mu$ (број обрађених корисника у серверу у јединици времена), $m$ (број сервисера) и $l$ (број потенцијалних корисника) израчунава $A$ (понуђен саобраћај), $A_s$ (интензитет оствареног саобраћаја), $B$ (вероватноћу блокаде), $P_L$ (вероватноћу губитка корисника), $p_0$ - $p_m$ (вероватноће попуњености система) и $q_0$ - $q_m$ (вероватноће стања система које затекне корисник по уласку у систем). Обезбеђена је опција уношења параметра $\mu$ и у облику $1/\mu$. Одабир се врши из падајућег менија.




***Слика 4.2.1 Интерфејс за унос параметара***

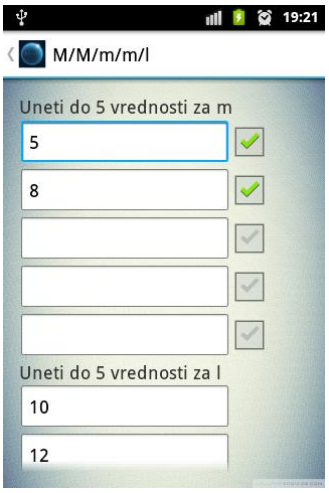



***Слика 4.2.2-5 Резултати за унесене параметре λ=5000, μ=4000, m=5 l=10***

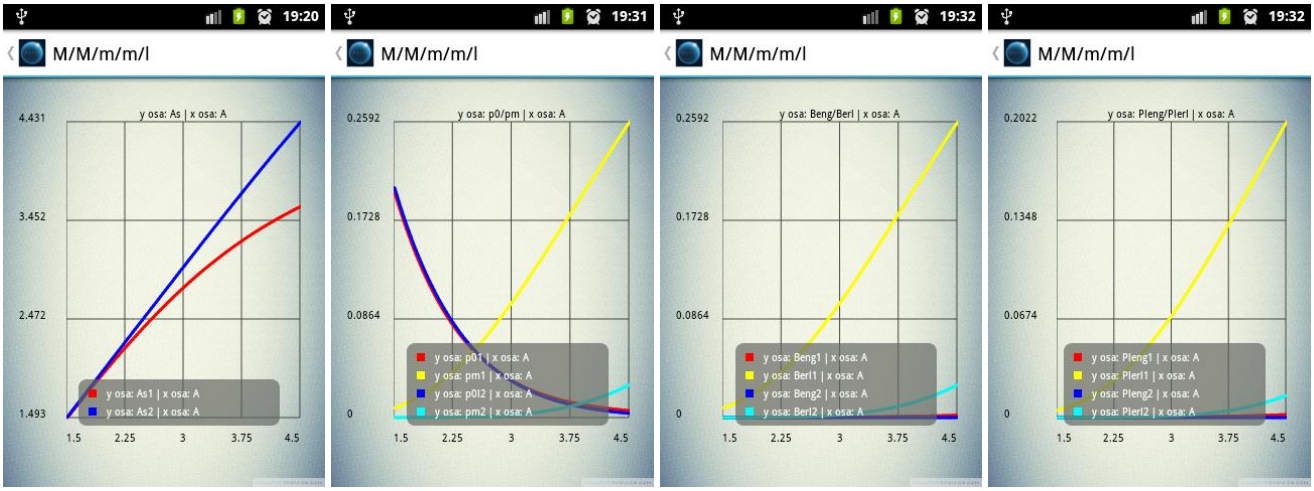
Цртање једноструких графика обезбеђује кориснику избор од три промењљиве, *A*, *m* и *l* које жели ставити на *x* осу. На *y* осу, могуће је ставити промењљиве $A_s$, $p_0$, $p_m$, $q_0$, $q_m$, $P_L$ и *B*. Такође постоји опција истовременог цртања парова $P_L$ и *B*, $p_0$ и $p_m$ као и $q_0$ и $q_m$. Такође постоји опција цртања упоредног графика за парове *B* према Енгсетовом *(Engset)* и Ерланговом *(Erlang)* моделу, и $P_L$ према иста два модела. Апликација нуди дифолтни опсег параметра са *x* осе.


***Слика 4.2.6-17 Поједини графици за параметре из претходног примера, и дифолтни опсег***

Опција цртања вишеструких графика омогућава цртање до пет различитих криви (или парова криви). По одабиру ове опције, уноса заједничких параметара за све криве, и одабира параметра који ће бити на *x* оси (*A*, *m* и *l*), долази се до интерфејса на ком се уносе карактеристичне вредности за сваку од пет криви а то су преостала два параметра која нису изабрана за *x* осу. Потребно је унети пар параметара и чекирати чекбокс који се налази поред места уноса једног од тих параметара. На дну истог интерфејса уносе се границе за одабран параметар са *x* осе парова криви.



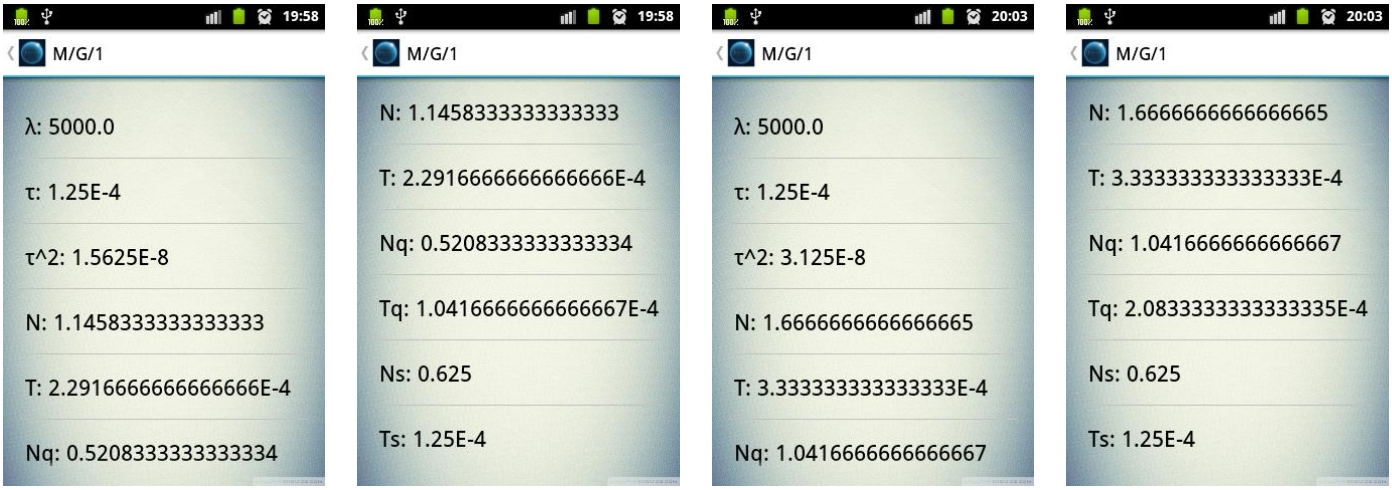

***Слика 4.2.18 Интерфејс за унос парова карактеристичних параметара за криве***




***Слика 4.2.19-22 Неки графици за λ=5000, μ=4000 и A на x оси, за унете парове параметара са слике 4.18***

## 4.3. *M/G/1* сервисни систем

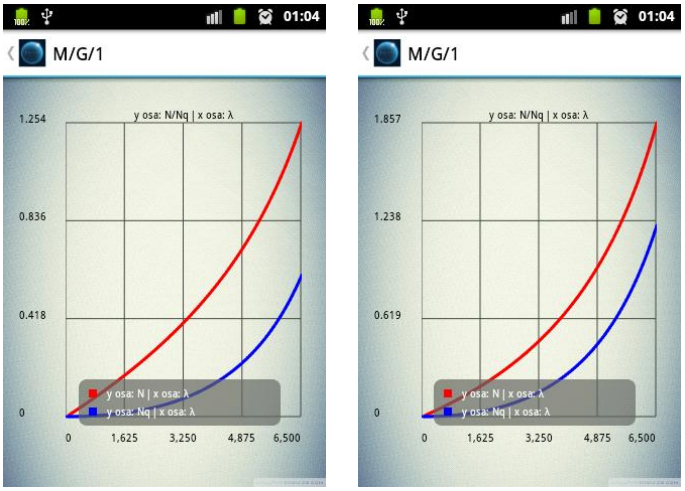
*M/G/1* је други понуђен сервисни систем на главном интерфејсу. Обезбеђен је један тип прорачуна за овај систем. Параметри за унос су $\lambda$ и $\tau$ док се тип уноса $\tau^2$ бира из падајућег менија. Одабиром, из падајућег менија, опције *"unos τ^2"* унос $\tau^2$ се врши уносом од стране корисника. Одабиром опције *"direktan proračun τ^2"*, $\tau^2$ апликација одређује сама према детеминистичкој или експоненцијалној расподели, у зависности од избора корисника на следећем интерфејсу.




***за детерминистичку расподелу*** ***за експоненцијалну расподелу***

***Слика 4.3.1-4 За унете параметре λ=5000, τ=0,000125 и изабран директан прорачун $\tau^2$***

У опцији цртања једноструких графика, уносе се параметри $\lambda$ и $\tau$ и из падајућег менија бира се расподела по којој се прорачунава $\tau^2$. У падајућем менију дате су опције исцртавања пара криви $N/N_q$ и $T/T_q$. На $x$ оси могу бити параметри $\lambda$ или $\tau$, у зависности од избора из падајућег менија за параметре осе. Апликација сама поставља дифолтни опсег за $x$ осу, као и у претходном моделу.

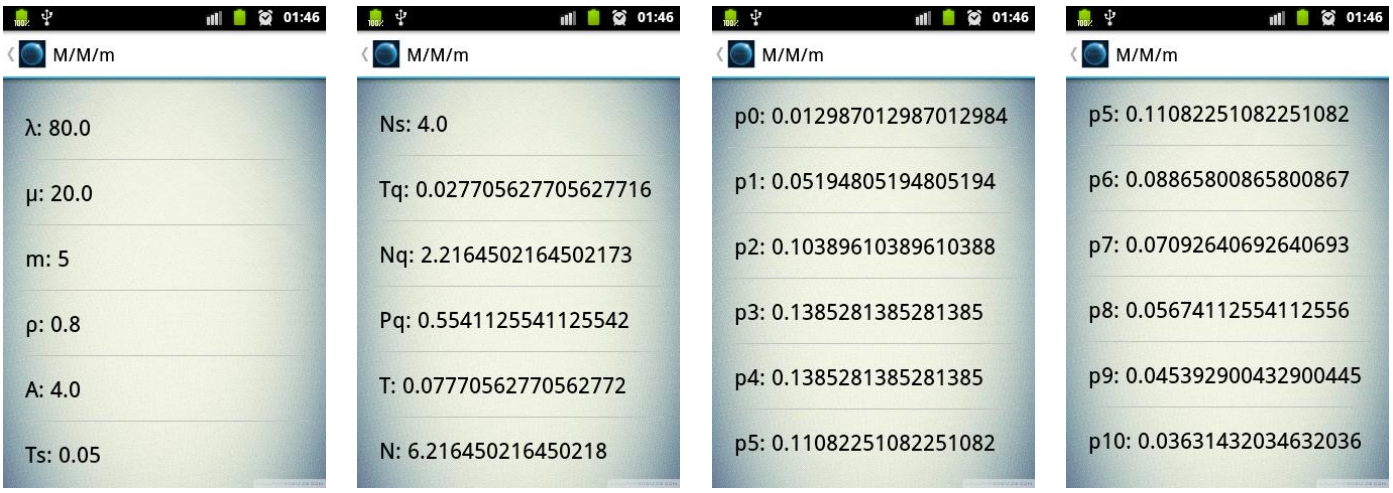



***Слика 4.3.5-6 графици за λ=6500 τ=0.0001 за детерминистичку и експоненцијалну расподелу, респективно***

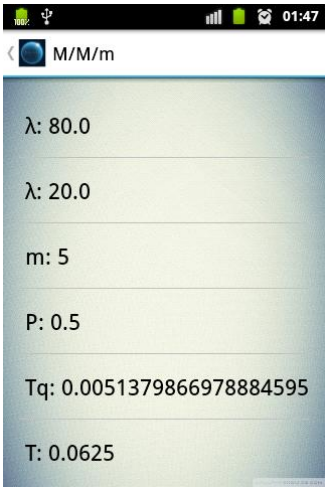
Опција цртања вишеструких графика исцртава на истом графику криве којима је парметар τ2 израчунат експоненцијалном и детерминистичком расподелу.

## 4.4. *M/M/m* сервисни систем

*M/M/m* систем омогућава две врсте прорачуна, први који за улазне параметре $\lambda$, $\mu$, и $m$ израчунава $A$, $N$, $N_Q$, $N_S$, $T$, $T_S$, $T_Q$, $P_Q$ и $\rho$, као и вероватноће $p_0 - p_{2m}$. Апликација прати да ли је систем стабилан, тј. да ли је параметар $\rho$ мањи од један. Други прорачун за исте параметре и додатни, $T_Q$, $T$ или $P$, прорачунава вероватноћу, $P$, како ће корисник чекати мање или једнако од задатог времена у случају додатних параметара $T_Q$ или $T$, док за додатни параметар $P$ прорачунава гранична времена чекања и задржавања, $T_Q$ и $T$.




***Слика 4.4.1-4 Резултат за унето λ=80 λ=20 m=5***

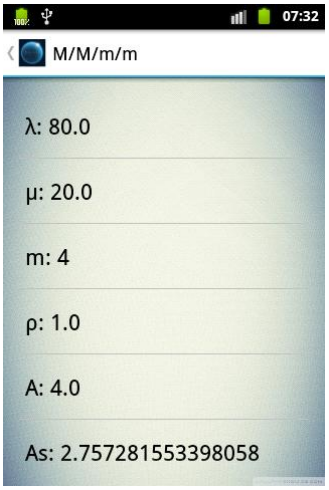


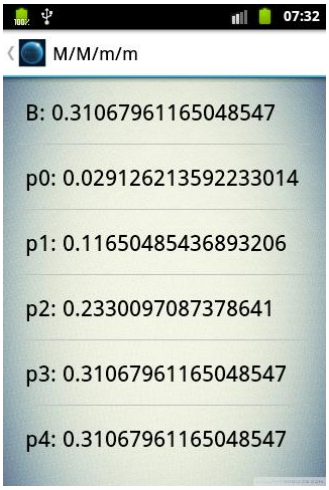

***Слика 4.4.5 Резултат за λ=80 λ=20 m=5 P=0.5***

Опције цртања једноструких и вешеструких графика се користи слично као у претходна два система.

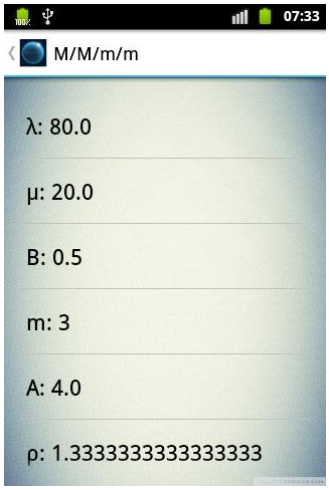
## 4.5. *M/M/m/m* сервисни систем

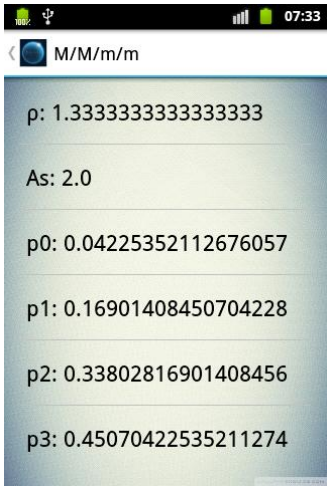
*M/M/m/m* систем има омогућена три прорачуна. Први прорачун за унете вредности $\lambda$, $\mu$, и $m$ израчунава $A$, $A_S$, $B$, $\rho$, као и вероватноће $p_0 - p_m$. Други прорачун за улазне параметре $\lambda$, $\mu$, и $B$, одређује број сервисера који задовољава услов блокаде, $B$, као и параметре $A$, $A_S$, $\rho$, и вероватноће $p_0 - p_m$. Трећи прорачун има улазне параметре $A$, $B$, и $m$, од којих се два задају, а трећи прорачунава из два задата.



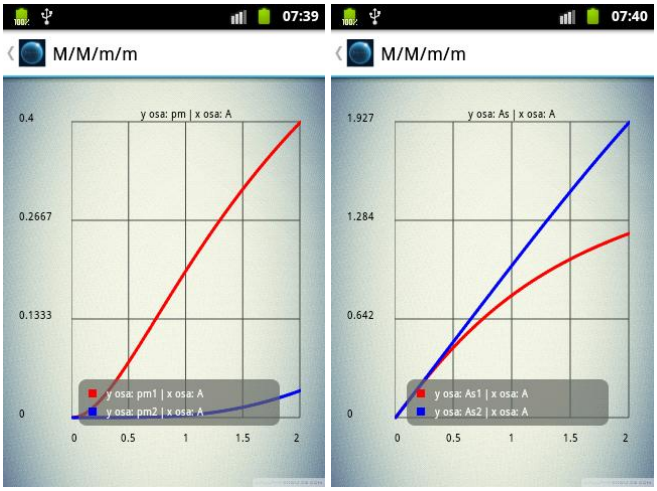

***Слика 4.5.1-2 Резултат за први прорачун за λ=80 μ=20 m=4***



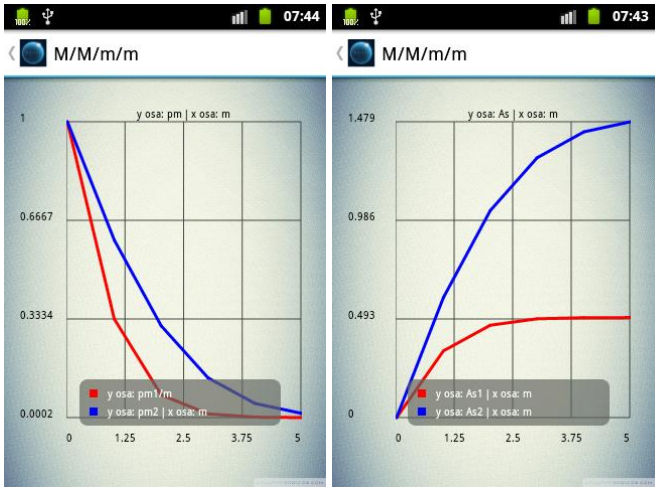




***Слика 4.5.3-4 Резултат за други прорачун за λ=80 μ=20 B=0.5***

Цртање вишеструких и једноструких графика се користи слично као у претходним системима.




*Слика 4.5.5-6 График за A=[0,2] $m_1$=2 $m_2$=5*




*Слика 4.5.7-8 График за m=[0,5] $A_1$=0.5 $A_2$=1.5*

# 5. Закључак

Апликација је калкулатор који обавља прорачуне за све познате моделе сервисних система, исцртава графике зависности као и упоредне графике за различите улазне параметре. Не захтева интернет конекцију за рад, једноставна је за коришћење и не захтева велику количину расположиве меморије ни јак ни напредан Андроид уређај (прављена је минимално за Андроид 2.2).

Функција за цртање графика је једноставна и лака за имплементирати, али има и доста мана, првенствено немогућност постављања лабела за јединице које су представљене осама, немогућност тачног исцртавања линија које не почињу на истим тачкама *x* осе. Апликација не онемогућава невалидне уносе од стране корисника, већ почиње од претпоставке како су унесене вредности смислене, тако да може доћи до пуцања и бесмислених резултата у таквим случајевима. Апликација спорије ради у случајевима кад се резултат мора израчунати итеративно, због великог броја итерација које су потребне за прихватљиво тачан резултат. Код није оптимизован, и вероватно би могао бити доста редукован без губитка функција, са вероватним побољшањем ефикасности.

Основна надоградња би требала бити увођење контроле корисничког уноса и онемогућавање невалидних уноса и самим тим спречавање било каквог случаја који може довести до пуцања апликације. Оптимизација кода би такође била веома логична надоградња апликације. И на крају, имплементирање боље апликације за исцртавање графика би заокружило апликацију као комплетан и користан програм сваком ко има потребе за оваквим прорачунима.

# Литература


[1] http://telekomunikacije.etf.rs/predmeti/te4ks/docs/KS/KS_Prilog_A.pdf

[2] http://developer.android.com/about/index.html

[3] http://web.archive.org/web/20140222153131/http://grail.cba.csuohio.edu/~matos/notes/cis-493/lecture-notes/Android-Chapter03-Life-Cycle.pdf

[4] https://developer.android.com/guide/components/fundamentals.html

[5] http://android-graphview.org/

[6] http://developer.android.com/training/index.html

[7] http://stackoverflow.com/

[8] http://developer.android.com/sdk/installing/installing-adt.html

[9] http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-203j-logistical-and-transportation-planning-methods-fall-2004/lecture-notes/qlec1.pdf

[10] www.emse.fr/~xie/SJTU/Ch6**Queue**.ppt