Лабораторная работа №2

СОЗДАНИЕ И ОТПРАВКА SMS

**Цель работы**: получить навыки разработки программ для редактирования и отправки SMS для мобильных устройств Android.

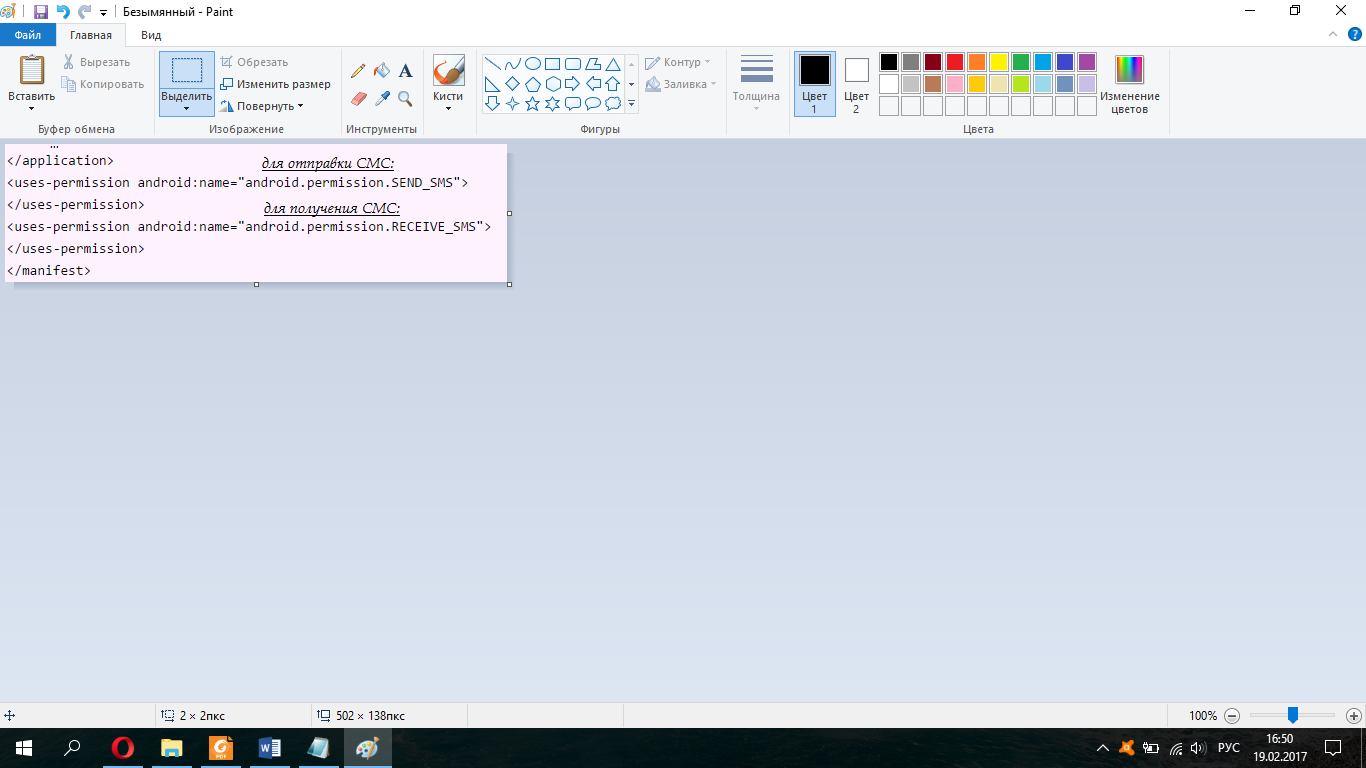
**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Рассмотрим основные классы и методы, используемые при отправке SMS, на примере. Для начала, создадим новое приложение. Для того, чтобы оно могло работать с SMS, ему необходимо дать соответствующие разрешения:

- SEND\_SMS – для отправки;

- RECEIVE\_SMS – для получения.

Откройте файл AndroidManifest.xml и добавьте в нем следующее разрешение:



Теперь открываем файл activity\_main.xml и создаем интерфейс нашего приложения:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Enter the phone number of recipient"

/>

<EditText

android:id="@+id/txtPhoneNo"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Message"

/>

<EditText

android:id="@+id/txtMessage"

android:layout\_width="fill\_parent"

android:layout\_height="150px"

android:gravity="top"

/>

<Button

android:id="@+id/btnSendSMS"

android:layout\_width="fill\_parent"

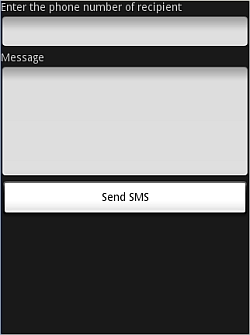
android:layout\_height="wrap\_content"

android:text="Send SMS"

/>

</LinearLayout>

В результате приложение будет выглядеть примерно следующим образом:



**Отправка СМС**

Перейдем к редактированию кода. В классе MainActivity.java напишем обработчик нажатия на кнопку. После щелчка мы должны проверить введен ли номер телефона и текстовое сообщение. Если все в порядке, мы вызываем функцию sendSMS(), которая отправляет сообщение.

**Public class** SMS **extends** Activity

{

// Объявляем переменные

Button btnSendSMS;

EditText txtPhoneNo;

EditText txtMessage;

@Override

**public** void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

  // Связываемся с элементами интерфейса

btnSendSMS=(Button) findViewById(R.id.btnSendSMS);

txtPhoneNo=(EditText) findViewById(R.id.txtPhoneNo);

txtMessage=(EditText) findViewById(R.id.txtMessage);

btnSendSMS.setOnClickListener(**new** View.OnClickListener()

{

**public** void onClick(View v)

{

String phoneNo= txtPhoneNo.getText().toString();

String message= txtMessage.getText().toString();

If (phoneNo.length()>0&amp;&amp; message.length()>0)

sendSMS(phoneNo, message); // Отправка сообщения

else

Toast.makeText(getBaseContext(),

"Please enter both phone number and message.",

Toast.LENGTH\_SHORT).show(); // Не заполнены поля

}

});

}

}

Напишем реализацию функции sendSMS():

**private** void sendSMS(String phoneNumber, String message)

{

PendingIntent pi= PendingIntent.getActivity(this,0,

**new** Intent(this, SMS.**class**),0);

SmsManager sms= SmsManager.getDefault();

sms.sendTextMessage(phoneNumber,**null**, message,pi,**null**);

}

Сам процесс отправки сообщения выполняется с помощью SmsManager, который мы вызываем при нажатии кнопки "Отправить", передаем в этот менеджер данные о введенных номере и текст сообщении, преобразовывая их в строки, и задаем отправку сообщения с помощью команды sendTextMessage. Если телефон или текст сообщения не заданы, сообщаем об этом всплывающим Toast сообщением.

В отличии от других классов, мы не можем получить экземпляр SmsManager напрямую. Для решения этой задачи необходимо вызвать статический метод getDefault(), который возвращает требуемый объект класса SmsManager. Метод sendTextMessage() отправляет sms сообщение с помощью PendingIntent. Объект PendingIntent используется для определения activity, которая будет вызвана после отправки сообщения. В приведенном выше коде объект pi ссылается на ту же activity из которой он вызывается, то есть после отправки SMS сообщения activity не поменяется.

Для отправки SMS сообщений можно также воспользоваться встроенным в Android стандартным приложением, дав ему задание через Intent:

Intent sendIntent=**new** Intent(Intent.ACTION\_VIEW);

sendIntent.putExtra("sms\_body","Content of the SMS goes here...");

sendIntent.setType("vnd.android-dir/mms-sms");

startActivity(sendIntent);

**Отладка и тестирование**

Для отслеживания состояния процесса отправки сообщения вам понадобится два объекта PendingIntent и два объекта BroadcastReceiver. Когда SMS сообщение будет отправлено, сгенерируется первое событие onReceive для BroadcastReceiver. Именно здесь можно будет проверить статус процесса отправки. Объект PendingIntent используется для мониторинга доставки. Когда сообщение будет успешно доставлено, сгенерируется еще одно событие onReceive для BroadcastReceiver.

Теперь можно провести тестирование приложения. При отладке на компьютере можно посылать сообщения от одного эмулятора другому. Для этого просто запустите два эмулятора (в папке android SDK зайдите в каталог Tools Emulator.exe). В качестве номера, куда вы отправляете SMS, нужно указать номер порта эмулятора. Его можно увидеть в заголовке окна эмулятора в круглых скобках.

**Работа с телефонной книгой. Получение списка контактов**

Для получения доступа к списку контактов необходимо прописать соответствующее разрешение:

<uses-permission android:name=”android.permission.READ\_CONTACTS”/>

Для примера выведем полученный список контактов в виде списка «Имя - телефон». Для этого в файле activity\_main.xml создадим элемент TextView:

<**LinearLayout** xmlns:android="schemas.android.com/apk/res/android"

xmlns:tools="schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity" >

<**ScrollView**

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<**TextView**

android:id="@+id/tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

</**ScrollView**>

</**LinearLayout**>

Теперь идем в MainActivity.java. Здесь мы инициализируем свой TextView, создаем и описываем метод getContacts(), в котором получаем из контактных данных устройства имя и номер контактов, преобразовываем их в строки, полученные значения имени и номера привязываем к соответствующим полям "Имя: " и "Номер: ", а напоследок отображаем все в элементе TextView:

**public** **class** MainActivity **extends** Activity {

*//Объявляем использование элемента TextView:*

**public** **TextView** contacts;

**@Override**

**public** **void** onCreate(**Bundle** *savedInstanceState*) {

super.onCreate(savedInstanceState);

setContentView(**R**.layout.activity\_main);

*//Связываемся с нашим элементом TextView:*

contacts = (**TextView**) findViewById(**R**.id.tv);

*//Метод получения контактных данных*

getContacts();

}

*//Описываем метод:*

**public** **void** getContacts() {

**String** phoneNumber = null;

*//Связываемся с контактными данными и берем с них значения id контакта, имени контакта и его номера:*

**Uri** CONTENT\_URI = **ContactsContract**.**Contacts**.CONTENT\_URI;

**String** \_ID = **ContactsContract**.**Contacts**.\_ID;

**String** DISPLAY\_NAME = **ContactsContract**.**Contacts**.DISPLAY\_NAME;

**String** HAS\_PHONE\_NUMBER = **ContactsContract**.**Contacts**.HAS\_PHONE\_NUMBER;

**Uri** **PhoneCONTENT\_URI** = **ContactsContract**.**CommonDataKinds**.**Phone**.CONTENT\_URI;

**String** **Phone\_CONTACT\_ID** = **ContactsContract**.**CommonDataKinds**.**Phone**.CONTACT\_ID;

**String** NUMBER = **ContactsContract**.**CommonDataKinds**.**Phone**.NUMBER;

**StringBuffer** output = new **StringBuffer**();

**ContentResolver** contentResolver = getContentResolver();

**Cursor** cursor = contentResolver.query(CONTENT\_URI, null,null, null, null);

*//Запускаем цикл обработчик для каждого контакта:*

if (cursor.getCount() > 0) {

*//Если значение имени и номера контакта больше 0 (то есть они существуют) выбираем*

*//их значения в приложение привязываем с соответствующие поля "Имя" и "Номер":*

while (cursor.moveToNext()) {

**String** contact\_id = cursor.getString(cursor.getColumnIndex( \_ID ));

**String** name = cursor.getString(cursor.getColumnIndex(DISPLAY\_NAME));

**int** hasPhoneNumber = **Integer**.parseInt(cursor.getString(cursor.getColumnIndex(HAS\_PHONE\_NUMBER)));

*//Получаем имя:*

if (hasPhoneNumber > 0) {

output.append("\n Имя: " + name);

**Cursor** phoneCursor = contentResolver.query(**PhoneCONTENT\_URI**, null,

**Phone\_CONTACT\_ID** + " = ?", new **String**[] { contact\_id }, null);

*//и соответствующий ему номер:*

while (phoneCursor.moveToNext()) {

phoneNumber = phoneCursor.getString(phoneCursor.getColumnIndex(NUMBER));

output.append("\n Телефон: " + phoneNumber);

}

}

output.append("\n");

}

*//Полученные данные отображаем с созданном элементе TextView:*

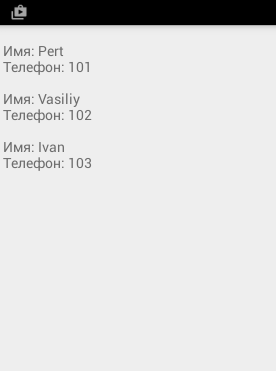
contacts.setText(output);

}

}

}

Результат получения списка контактов показан на рисунке ниже:



**Работа с телефонной книгой. Добавление контакта**

Для начала нам надо установить соответствующее разрешение в файле манифеста:

<uses-permission android:name=”android.permission.WRITE\_CONTACTS”/>

В Android контакты распределяются по трем таблицам: contacts, raw contacts и data. Нам надо добавить новый контакт в две последние таблицы. Данные контакта представляют объект ContentValues, который состоит из ключей и их значений, то есть объект словаря. После его создания происходит добавление в него пары элементов:

contactValues.put(RawContacts.ACCOUNT\_NAME, newContact);

contactValues.put(RawContacts.ACCOUNT\_TYPE, newContact);

Здесь устанавливается название и тип контакта. В качестве ключей выставляются значения RawContacts.ACCOUNT\_NAME и RawContacts.ACCOUNT\_TYPE, а в качестве их значения – текст из текстового поля, полученный от пользователя. Далее этот объект добавляется в таблицу RawContacts с помощью метода insert():

Uri newUri = getContentResolver().insert(RawContacts.CONTENT\_URI, contactValues);

Метод insert() возвращает URI – ссылку на добавленный объект в таблице, у которого мы можем получить id. Затем после очистки мы подготавливаем объект для добавления в таблицу Data, вновь наполняя его данными:

contactValues.put(Data.RAW\_CONTACT\_ID, rawContactsId);

contactValues.put(Data.MIMETYPE, StructuredName.CONTENT\_ITEM\_TYPE);

contactValues.put(StructuredName.DISPLAY\_NAME, newContact);

И опять добавление производит метод insert():

getContentResolver().insert(Data.CONTENT\_URI, contactValues);

Для добавления контакта в телефонную книгу можно использованный сформированный ранее интерфейс, добавив в него кнопку и связав ее обработчик с приведенными методами.

**Вопросы для самопроверки**

1. Какие разрешения могут потребоваться для создания SMS мессенджера?
2. Для чего нужен класс SmsManager? Каковы особенности создания его экземпляра?
3. Что вам известно про класс Toast? Какой метод класса отвечает за появление всплывающего сообщения?
4. Для чего могут быть использованы объекты PendingIntent и BroadcastReceiver?
5. Какое событие сигнализирует о доставке сообщения?
6. Какие таблицы контактов есть в Android? Какие методы и типы данных используются для добавления в них записей?

**Дополнительные материалы**

1. Основы Contacts API в Android – [Электронный ресурс] – <https://habrahabr.ru/post/130148/> (дата обращения – 19.02.2017).
2. Android – Sending SMS – [Электронный ресурс] - <https://www.tutorialspoint.com/android/android_sending_sms.htm> (дата обращения – 19.02.2017).
3. Android Send And Receive SMS – [Электронный ресурс] - <http://codetheory.in/android-sms/> (дата обращения – 19.02.2017).
4. Способы отправки SMS эмулятору Android в Windows – [Электронный ресурс] – <https://habrahabr.ru/post/151080/> (дата обращения – 19.02.2017).

**Задание**

Предоставить пользователю возможность вводить SMS. Адрес получателя выбирается из списка (в дальнейшем контактной книги). Если телефона получателя SMS нет в списке, дать возможность пользователю его добавить. Также предоставить возможность пользователю отправить SMS без сохранения номера телефона получателя. Список последних отправленных десяти SMS должен быть сохранён. Кроме этого организовать работу с шаблонами SMS (создание, просмотр и вставка шаблона в текст SMS).

**Варианты заданий**

1. В списке последних отправленных сообщений должна отображаться дата отправки.
2. Пользователь должен иметь возможность отправить сообщение нескольким (до 3-х) контактам.
3. Сохранять неотправленное (но не пустое) сообщение как черновик при выходе из приложения.
4. Позволить пользователю вставлять в сообщение смайл из предложенных.
5. В списке просматриваемых контактов должна отображаться дополнительная известная информация, если есть (например, группа, электронная почта, день рождения или прочее на выбор).
6. При ошибке отправки сообщения предложить пользователю изменить номер получателя.
7. В списке последних отправленных сообщений должно отображаться время отправки сообщения.
8. Позволить пользователю вставлять в текст сообщения телефон из списка контактов.
9. Пользователь должен иметь возможность добавить тему сообщения.
10. Позволить пользователю сортировать последние отправленные сообщения по имени контакта.