Комитет по образованию Правительства Санкт-Петербурга

**САНКТ-ПЕТЕРБУРГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОНЫХ ТЕХНОЛОГИЙ**

**Отчет по практической работе № 1**

**МДК 01.03 Разработка мобильных приложений**

**Тема: Разработка приложения с использованием протокола UDP для передачи данных**

Выполнил

студент Группы 493

сидоров антон дмитриевич

Проверила Фомин А. В.

Оценка \_\_\_\_\_\_\_\_

Санкт-Петербург 2022

**Содержание**

[1. Цели работы 3](#_Toc113830851)

[2. Макеты экранов приложения и их описание 3](#_Toc113830852)

[3. Диаграмма базы данных SQLite3 6](#_Toc113830853)

[4. Программный код 8](#_Toc113830854)

[5. Демонстрация работы приложения 22](#_Toc113830855)

[6. Вывод 30](#_Toc113830856)

# Цели работы

Разработать приложение для обмена сообщениями по протоколу UDP между двумя телефонами.

# Макеты экранов приложения и их описание

Примечания по макетам

Окна в работающем приложения могут незначительно отличаться от их макетов.

* 1. Начальное окно приложения

Данное окно появляется при открытии приложения.

Макет окна представлен на рисунке 1.

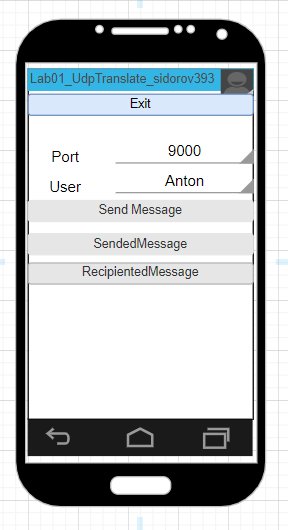


Рисунок 1 – Макет начального окна

На этом окне задаётся:

* Имя пользователя, под которым будут отправляться сообщения (*User*)
* Порт, на котором будут приниматься сообщения (*Port*)

Под логотипом имеется кнопка *Exit*, производящая выход из приложения.

Также на этом окне есть 3 кнопки:

* Ввода и отправки сообщений (*Send Message*) – Открывает окно отправки/просмотра сообщения в *режиме отправки сообщения*.
* Просмотра отправленных сообщений (*Sended Message*)
* Просмотра полученных сообщений (*Recipiended Message*)
  1. Окно отправки/просмотра сообщения

Данное окно предназначено для отправки сообщения, а также для просмотра отправленного/полученного сообщения.

Макет окна представлен на рисунке 2.

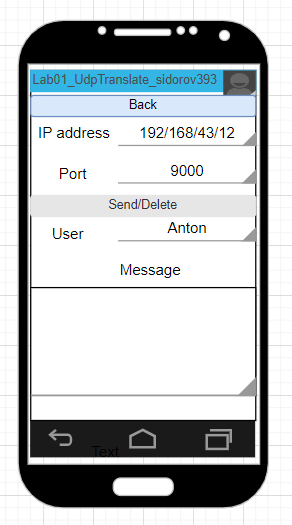


Рисунок 2 – Макет окна отправки/просмотра сообщения

Данное окно имеет 2 режима:

* Отправка сообщения
* Просмотр сообщения

Данное окно имеет кнопку *Back*, которая возвращает на начальное окно приложения.

Также, данное окно имеет 4 поля ввода:

* IP-адрес получателя (*IP address*)
* Порт, на котором получатель может принять сообщение (*Port*)
* Имя пользователя, под которым отправляется сообщение (*User*)
* Текст сообщения (*Message*)

И, наконец данное окно имеет кнопку:

* Отправки сообщения (*Send*) – Только, в *режиме отправки сообщения*
* Удаления сообщения (*Delete*) – Только в *режиме просмотра сообщения*.
  1. Окно списка отправленных/полученных приложений

Данное окно позволяет просматривать список сообщений, который приложение получает из базы данных SQLite3.

Макет окна представлен на рисунке 3.

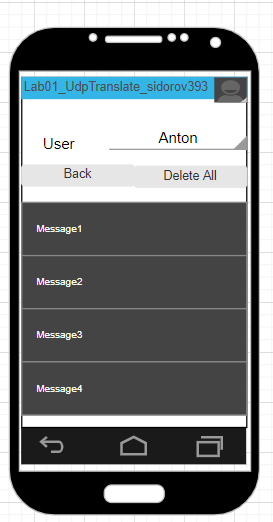


Рисунок 3 – Макет окна просмотра списка сообщений

Данное окно имеет кнопки:

* *Back*, которая возвращает на начальное окно приложения.
* *Delete All,* которая позволяет удалить все сообщения.

Также, данное окно имеет текстовое поля вывода имени пользователя (*User*).

И, наконец, данное окно имеет список сообщений. Нажатие на любое окно в этом списке происходит открытие окна отправки/просмотра сообщения в *режиме просмотра сообщения*.

# Диаграмма базы данных SQLite3

Диаграмма базы данных показана на рисунке 4.

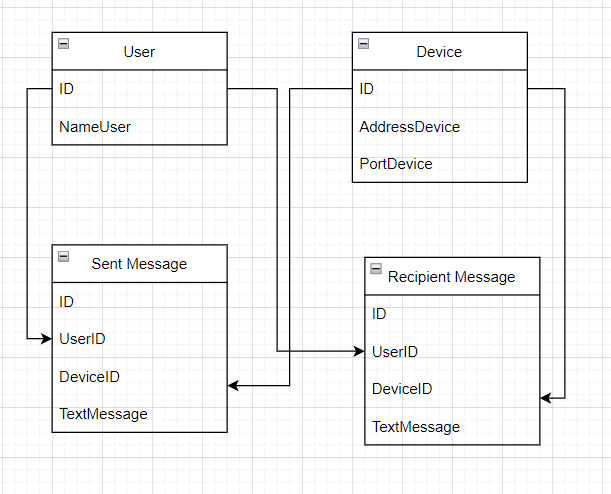


Рисунок 4 – Диаграмма базы данных в SQLite3

Примечания

Таблицы в базе данных могут незначительно отличаться своими названиями и названиями столбцов в них, от того, как это представлено на диаграмме.

* 1. Таблица *User*

Данная таблица хранит информацию о пользователе:

* Идентификатор (*ID*)
* Имя (*NameUser*)
  1. Таблица *Device*

Данная таблица хранит информацию об устройствах, с которых были отправлены сообщения на данное устройство, и на которые были отправлены сообщения с данного устройства:

* Идентификатор (*ID*)
* IP-адрес (*AddressDevice*)
* Порт, с которого было отправлено сообщение, или было получено сообщение (*PortDevice*)
  1. Таблица *Sent Message*

Данная таблица хранит информацию об отправленных сообщений:

* Идентификатор (*ID*)
* Идентификатор пользователя, отправившего сообщение (*UserID*) – из таблицы *User*
* Идентификатор устройства, на которое было отправлено сообщение (*DeviceID*) – Из таблицы *Device*
* Текст сообщения (*TextMessage*)
  1. Таблица *Recipient Message*

Данная таблица хранит информацию о полученных сообщений:

* Идентификатор (*ID*)
* Идентификатор пользователя, отправившего сообщение (*UserID*) – из таблицы *User*
* Идентификатор устройства, с которого было отправлено сообщение (*DeviceID*) – Из таблицы *Device*

Текст сообщения (*TextMessage*)

# Программный код

* 1. Работа с базой данных SQLite3

**package** DB;  
  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
**import** androidx.annotation.Nullable;  
**import** java.util.ArrayList;  
  
**public class** DB **extends** SQLiteOpenHelper {  
  
 **public** DB(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory)  
 {  
 **this**(context, name, factory, 1);  
 }  
  
 **public** DB(@Nullable Context context, @Nullable String name)  
 {  
 **this**(context, name, **null**);  
 }  
  
 **public static** DB GetDB(Context context, String name)  
 {  
 **return new** DB(context, name);  
 }  
  
 **public** DB(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, **int** version) {  
 **super**(context, name, factory, version);  
 }  
  
 @Override  
 **public void** onCreate(SQLiteDatabase db) {  
 String sql = **"Create table User (ID Integer not null unique Primary key AUTOINCREMENT, NameUser TEXT);"**;  
 db.execSQL(sql);  
  
 sql = **"Create table Device (ID Integer not null unique Primary key AUTOINCREMENT, IpAddress Text, Port Text);"**;  
 db.execSQL(sql);  
  
 sql = **"Create table SentMessage (ID Integer not null unique Primary key AUTOINCREMENT, UserId int, DeviceID int, TextMessage Text);"**;  
 db.execSQL(sql);  
  
 sql = **"Create table RecipientMessage (ID Integer not null unique Primary key AUTOINCREMENT, UserId int, DeviceID int, TextMessage Text);"**;  
 db.execSQL(sql);  
 }  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** i, **int** i1) {  
  
 }  
  
 **public int** getIdAccount(String account) {  
  
 String sql = **"Select ID From User where NameUser = '"** + account + **"'"**;  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor curr = db.rawQuery(sql, **null**);  
 **try** {  
 **if** (curr.moveToFirst() == **true**) {  
 **return** curr.getInt(0);  
 } **else** {  
 **throw new** Exception();  
 }  
 }  
 **catch**(Exception e)  
 {  
 **try** {  
 curr.close();  
 }  
 **catch** (Exception ex)  
 {  
  
 }  
 String sql1 = **"Insert Into User (NameUser) Values ('"** + account + **"');"**;  
 SQLiteDatabase db1 = getWritableDatabase();  
 db1.execSQL(sql1);  
 **return** getIdAccount(account);  
 }  
 }  
  
 **public** String getIdAccount(**int** account) {  
  
 String sql = **"Select NameUser From User where id = "** + account + **";"**;  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor curr = db.rawQuery(sql, **null**);  
 **if** (curr.moveToFirst() == **true**) {  
 **return** curr.getString(0);  
 } **else** {  
 curr.close();  
 **return ""**;  
 }  
 }  
  
 **public** String getPortDevice(**int** device) {  
  
 String sql = **"Select Port From Device where ID = "** + device + **";"**;  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor curr = db.rawQuery(sql, **null**);  
 **if** (curr.moveToFirst() == **true**) {  
 **return** curr.getString(0);  
 } **else** {  
 curr.close();  
 **return ""**;  
 }  
 }  
  
 **public** String getIpAddressDevice(**int** device) {  
  
 String sql = **"Select IpAddress From Device where ID = "** + device + **";"**;  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor curr = db.rawQuery(sql, **null**);  
 **if** (curr.moveToFirst() == **true**) {  
 **return** curr.getString(0);  
 } **else** {  
 curr.close();  
 **return ""**;  
 }  
 }  
  
 **public int** getIdDevice(String ipAddress, String port) {  
  
 String sql = **"Select ID From Device where IpAddress = '"** + ipAddress + **"' and Port = '"** + port+**"'"**;  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor curr = db.rawQuery(sql, **null**);  
 **try** {  
 **if** (curr.moveToFirst() == **true**) {  
 **return** curr.getInt(0);  
 } **else** {  
 *//curr.close();* **throw new** Exception();  
 }  
 }  
 **catch** (Exception e)  
 {  
 **try** {  
 curr.close();  
 }  
 **catch** (Exception ex)  
 {  
  
 }  
 String sql1 = **"Insert Into Device (IpAddress, Port) Values ('"** + ipAddress + **"', '"** + port + **"');"**;  
 SQLiteDatabase db1 = getWritableDatabase();  
 db1.execSQL(sql1);  
 **return** getIdDevice(ipAddress, port);  
 }  
  
 }  
  
 **public int** GetRecipientMessage(String account, String ipAddress, String port, String Message)  
 {  
 **int** id\_User = getIdAccount(account);  
 **int** id\_Device = getIdDevice(ipAddress, port);  
  
 *//curr.close();* String sql2 = **"Select Max(ID) From RecipientMessage"**;  
 String sql1 = **"Insert Into RecipientMessage (UserID, DeviceID, TextMessage) Values ("** + id\_User + **", "** + id\_Device + **", '"** + Message + **"');"**;  
 SQLiteDatabase db1 = getWritableDatabase();  
 db1.execSQL(sql1);  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor curr = db.rawQuery(sql2, **null**);  
 **if** (curr.moveToFirst() == **true**) {  
 **return** curr.getInt(0);  
 }  
 **else  
 return** 0;  
 }  
  
 **public void** DeleteRecipientMessage(**int** id)  
 {  
 **try** {  
 String sql = **"Delete From RecipientMessage where ID = "** + id+**";"**;  
 SQLiteDatabase db = getWritableDatabase();  
 db.execSQL(sql);  
 }  
 **catch**(Exception e)  
 {  
  
 }  
 }  
  
 **public void** DeleteSentMessage(**int** id)  
 {  
 **try** {  
 String sql = **"Delete From SentMessage where ID = "** + id+**";"**;  
 SQLiteDatabase db = getWritableDatabase();  
 db.execSQL(sql);  
 }  
 **catch**(Exception e)  
 {  
  
 }  
 }  
  
 **public int** GetSentMessage(String account, String ipAddress, String port, String Message) {  
 **int** id\_User = getIdAccount(account);  
 **int** id\_Device = getIdDevice(ipAddress, port);  
  
 *//curr.close();* String sql2 = **"Select Max(ID) From SentMessage"**;  
 String sql1 = **"Insert Into SentMessage (UserID, DeviceID, TextMessage) Values ("** + id\_User + **", "** + id\_Device + **", '"** + Message + **"');"**;  
 SQLiteDatabase db1 = getWritableDatabase();  
 db1.execSQL(sql1);  
 SQLiteDatabase db = getReadableDatabase();  
 Cursor curr = db.rawQuery(sql2, **null**);  
 **if** (curr.moveToFirst() == **true**) {  
 **return** curr.getInt(0);  
 }  
 **else  
 return** 0;  
 }  
  
 **public void** getAllSendedMessage (ArrayList<Message> lst) {  
 lst.clear();  
 SQLiteDatabase db = getReadableDatabase();  
 String sql = **"SELECT ID, TextMessage, DeviceID, UserID FROM SentMessage;"**;  
 Cursor cur = db.rawQuery(sql, **null**);  
 **int** number = 0;  
 **if** (cur.moveToFirst() == **true**) {  
 **do** {  
  
 **int** id = cur.getInt(0);  
 String message = cur.getString(1);  
 **int** device = cur.getInt(2);  
 **int** user = cur.getInt(3);  
 number++;  
 lst.add(**new** Message(id, number, message, user, device));  
 }  
 **while** (cur.moveToNext() == **true**);  
 }  
 }  
  
 **public void** getAllRecipientMessage (ArrayList<Message> lst)  
 {  
 lst.clear();  
 SQLiteDatabase db = getReadableDatabase();  
 String sql = **"SELECT ID, TextMessage, DeviceID, UserID FROM RecipientMessage;"**;  
 Cursor cur = db.rawQuery(sql,**null**);  
 **int** number = 0;  
 **if**(cur.moveToFirst() == **true**)  
 {  
 **do** {  
  
 **int** id = cur.getInt(0);  
 String message = cur.getString(1);  
 **int** device = cur.getInt(2);  
 **int** user = cur.getInt(3);  
 number++;  
 lst.add(**new** Message(id, number,message, user, device));  
 }  
 **while**(cur.moveToNext() == **true**);  
 }  
 }  
  
 **public void** DeleteAllSendMessages()  
 {  
 **try** {  
 String sql = **"Delete From SentMessage"**;  
 SQLiteDatabase db = getWritableDatabase();  
 db.execSQL(sql);  
 }  
 **catch**(Exception e)  
 {  
  
 }  
 }  
  
 **public void** DeleteAllRecipientMessages()  
 {  
 **try** {  
 String sql = **"Delete From RecipientMessage"**;  
 SQLiteDatabase db = getWritableDatabase();  
 db.execSQL(sql);  
 }  
 **catch**(Exception e)  
 {  
  
 }  
 }  
}

* 1. Начальное окно приложения

**package** com.example.lab01\_udptranslate\_sidorov393;  
  
**import** android.content.DialogInterface;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** androidx.appcompat.app.AlertDialog;  
**import** androidx.appcompat.app.AppCompatActivity;  
**public class** MainActivity **extends** AppCompatActivity {  
  
 EditText **Port**, **Name**;  
 Button **exit**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **Port** = findViewById((R.id.***PortText***));  
 **Name** = findViewById(R.id.***NameText***);  
 **exit** = findViewById(R.id.***ExitButton***);  
 }  
  
 **public void** Send\_Click(View v)  
 {  
 Intent i = **new** Intent(**this**, SendMessageWindow.**class**);  
  
 i.putExtra(**"send"**, **true**);  
 i.putExtra(**"name"**, **Name**.getText().toString());  
 i.putExtra(**"port"**, **Port**.getText().toString());  
 startActivity(i);  
 }  
  
 **public void** Exit\_Click(View v)  
 {  
 AlertDialog.Builder bld = **new** AlertDialog.Builder(**this**);  
  
 bld.setPositiveButton(**"Нет"**,  
 **new** DialogInterface.OnClickListener()  
 {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 dialog.cancel(); *// Закрываем диалоговое окно* }  
 });  
 bld.setNegativeButton(**"Да"**, **new** DialogInterface.OnClickListener(){  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 finish(); *// Закрываем Activity* }  
 });  
 AlertDialog dlg = bld.create();  
 dlg.setTitle(**"Выход из приложения"**);  
 dlg.setMessage(**"Уважаемый пользователь \n"** +  
 **"Вы действительно хотите выйти из программы \n"** +  
 **"Вы, также, можете запустить программу снова \n"** +  
 **"С уважением и любовью, Создатель программы, Сидоров Антон Дмитриевич"**);  
 dlg.show();  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu)  
 {  
 getMenuInflater().inflate(R.menu.***menu***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item)  
 {  
 **int** id = item.getItemId();  
 **switch** (id)  
 {  
 **case** R.id.***Close***: {  
 View v = **exit**;  
 Exit\_Click(v);  
 }  
 **break**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
 **public void** RunProgram(View v)  
 {  
 String text = ((Button)v).getText().toString();  
 String[] parts = text.split(**" "**);  
 String doing = parts[0];  
  
 Intent i = **new** Intent(**this**, SentReciveMessages.**class**);  
  
 *//i.putExtra("send", true);* i.putExtra(**"doing"**, doing);  
 i.putExtra(**"text"**, text);  
 i.putExtra(**"port"**, **Port**.getText().toString());  
 i.putExtra(**"user"**, **Name**.getText().toString());  
 startActivity(i);  
 }  
  
}

* 1. Окно отправки/просмотра сообщения

**package** com.example.lab01\_udptranslate\_sidorov393;  
  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** android.widget.TextView;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** java.io.IOException;  
**import** java.net.DatagramPacket;  
**import** java.net.DatagramSocket;  
**import** java.net.InetAddress;  
**import** java.net.InetSocketAddress;  
**import** java.net.SocketAddress;  
**import** java.net.SocketException;  
**import** java.net.UnknownHostException;  
**import** java.util.ArrayList;  
**import** DB.DB;  
**import** DB.Message;  
  
**public class** SentReciveMessages **extends** AppCompatActivity {  
  
 TextView **doingText**, **doing**;  
 ListView **lstctl**;  
  
 ArrayList<Message> **lst** = **new** ArrayList<>();  
 ArrayAdapter<Message> **adp**;  
 DB **db**;  
 Boolean **send**;  
 String **sent**;  
 **int port**;  
 **byte**[] **recive\_buffer** = **new byte**[200];  
 TextView **user**;  
  
 **public void** get\_Message()  
 {  
  
 **lst**.clear();  
 **if**(**send**)  
 **db**.getAllSendedMessage(**lst**);  
 **else  
 db**.getAllRecipientMessage(**lst**);  
  
 **adp**.notifyDataSetChanged();  
 }  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_sent\_recive\_messages***);  
 **doingText** = findViewById(R.id.***DoingText***);  
 **doing** = findViewById(R.id.***Doing***);  
 **lstctl** = findViewById(R.id.***MessageList***);  
  
 **adp** = **new** ArrayAdapter<Message>(**this**, android.R.layout.***simple\_list\_item\_1***, **lst**);  
 **lstctl**.setAdapter(**adp**);  
 **db** = DB.*GetDB*(GetContext(), **"Messages.db"**);  
 **user** = findViewById(R.id.***UserText***);  
  
 **lstctl**.setOnItemClickListener((parent,view,position, id) -> {  
 Message message = **adp**.getItem(position);  
  
 Intent i = **new** Intent(**this**, SendMessageWindow.**class**);  
  
 i.putExtra(**"send"**, **false**);  
  
 String name = **db**.getIdAccount(message.**userID**);  
 String port = **db**.getPortDevice(message.**deviceID**);  
 String ip = **db**.getIpAddressDevice(message.**deviceID**);  
 **int** id1 = message.**id**;  
 i.putExtra(**"name1"**, name);  
 i.putExtra(**"port"**, port);  
 i.putExtra(**"ip"**, ip);  
 i.putExtra(**"id"**, id1);  
 i.putExtra(**"text"**, message.**Message**);  
 i.putExtra(**"sent"**, **send**);  
 startActivityForResult(i, 666);  
  
 });  
  
 Intent i = getIntent();  
 **doingText**.setText(i.getStringExtra(**"text"**));  
 **doing**.setText(i.getStringExtra(**"doing"**));  
 **send** = (**doing**.getText().toString().compareTo(**"Sent"**) == 0);  
 **sent** = **doing**.getText().toString();  
 **port** = Integer.*valueOf*(i.getStringExtra(**"port"**));  
 **user**.setText(i.getStringExtra(**"user"**));  
  
 get\_Message();  
 **if**(!**send**)  
 Recive();  
 }  
  
 Thread **ReciveThread**;  
 DatagramSocket **socket**;  
   
 **public void** Recive()  
 {  
  
 **try** {  
 InetAddress local\_network = InetAddress.*getByName*(**"0.0.0.0"**);  
 SocketAddress local\_address = **new** InetSocketAddress(local\_network, **port**);  
 **socket** = **new** DatagramSocket(**null**);  
 **socket**.bind(local\_address);  
 **socket**.setBroadcast(**true**);  
 }  
 **catch** (UnknownHostException | SocketException e)  
 {  
 e.printStackTrace();  
 }  
  
 Runnable reciver = **new** Runnable() {  
 @Override  
 **public void** run() {  
  
 Log.*e*(**"TEST"**, **"RECIVING THREAD IS RUNNING"**);  
  
 DatagramPacket recived\_packet = **new** DatagramPacket(**recive\_buffer**, **recive\_buffer**.**length**);  
 **while**(**true**)  
 {  
 **try** {  
 **socket**.receive(recived\_packet);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 **return**;  
 }  
 String s = **new** String(**recive\_buffer**, 0, recived\_packet.getLength());  
 Log.*e*(**"TEST"**, **"RECIVED: "**+s);  
  
 s = s.trim();  
  
 String[] parts = s.split(**" "**);  
 **if**(parts[0].compareTo(**"hello"**) == 0)  
 {  
 String name = parts[1];  
 String text = **""**;  
 **for**(**int** i = 2; i< parts.**length**; i++)  
 {  
 text += parts[i]+**" "**;  
 }  
  
 String ipAddess = **"0.0.0.0"**;  
 **int** port = **socket**.getPort();  
 String finalText = text;  
 runOnUiThread(() ->  
 {  
 **int** id = **db**.GetRecipientMessage(name, ipAddess, String.*valueOf*(port), finalText);  
 get\_Message();  
 });  
 }  
  
 }  
 }  
 };  
 **ReciveThread** = **new** Thread(reciver);  
 **ReciveThread**.start();  
  
 }  
  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
  
  
 get\_Message();  
  
 **super**.onActivityResult(requestCode, resultCode, data);  
 }  
  
 **public void** Back\_Click(View v)  
 {  
 **try** {  
 **ReciveThread**.stop();  
 }  
 **catch** (Exception e)  
 {  
  
 }  
  
 **try** {  
 **socket**.close();  
 }  
 **catch** (Exception e)  
 {  
  
 }  
  
 finish();  
 }  
  
 **public void** Delete(View v)  
 {  
 **if**(**send**)  
 **db**.DeleteAllSendMessages();  
 **else  
 db**.DeleteAllRecipientMessages();  
  
 get\_Message();  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu)  
 {  
 getMenuInflater().inflate(R.menu.***menu***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item)  
 {  
 **int** id = item.getItemId();  
 **switch** (id)  
 {  
 **case** R.id.***Close***:  
 {  
 finish();  
 }  
 **break**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
 **public** Context GetContext()  
 {**return this**;}  
  
}

* 1. Окно просмотра списка сообщений

**package** com.example.lab01\_udptranslate\_sidorov393;  
  
**import** android.app.Activity;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** java.io.IOException;  
**import** java.net.DatagramPacket;  
**import** java.net.DatagramSocket;  
**import** java.net.InetAddress;  
**import** java.net.InetSocketAddress;  
**import** java.net.SocketAddress;  
**import** java.net.UnknownHostException;  
**import** java.nio.charset.StandardCharsets;  
**import** java.util.ArrayList;  
**import** DB.DB;  
  
**public class** SendMessageWindow **extends** AppCompatActivity {  
  
 **byte**[] send\_buffer = **new byte**[6];  
 Button SendDelete;  
 **boolean** send, sent;  
 EditText name, ipAddress, port, Message;  
 String port1, name1;  
  
 DatagramSocket socket;  
 DatagramPacket sent\_Packet;  
  
  
 **int** id = 0;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_send\_message\_window);  
  
 SendDelete = findViewById(R.id.SendDeleteButton);  
  
 Intent i = getIntent();  
 send = i.getBooleanExtra(**"send"**, **true**);  
 **name** = findViewById(R.id.***UserNameText***);  
 **name**.setText(i.getStringExtra(**"name"**));  
  
 **ipAddress** = findViewById(R.id.***ipAddresstext***);  
 **port** = findViewById(R.id.***portText***);  
 **port1** = i.getStringExtra(**"port"**);  
 **Message** = findViewById(R.id.***MessageText***);  
  
 **if**(**send**)  
 {  
  
 **SendDelete**.setText(**"Send message"**);  
 }  
 **else** {  
 String ip = i.getStringExtra(**"ip"**);  
 **SendDelete**.setText(**"Delete message"**);  
 **ipAddress**.setEnabled(**false**);  
 **port**.setEnabled(**false**);  
 **port**.setText(**port1**);  
 **ipAddress**.setText(ip);  
 **id**= i.getIntExtra(**"id"**, 0);  
 String message = i.getStringExtra(**"text"**);  
 **name**.setText(i.getStringExtra(**"name1"**));  
 **Message**.setText(message);  
 **sent** = i.getBooleanExtra(**"sent"**,**true**);  
  
 }  
 }  
  
 **public void** Back\_Click(View v)  
 {  
 finish();  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu)  
 {  
 getMenuInflater().inflate(R.menu.menu, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item)  
 {  
 **int** id = item.getItemId();  
 **switch** (id)  
 {  
 **case** R.id.Close:  
 {  
 finish();  
 }  
 **break**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
 **public** Context GetContext()  
 {**return this**;}  
  
 **public** Activity GetActicivity()  
 {**return this**;}  
  
  
 **public void** SendDelete\_Click(View v)  
 {  
 InetAddress local\_network;  
 SocketAddress local\_Address;  
 **if**(send)  
 {  
  
 **try** {  
  
 local\_network = InetAddress.getByName(**"0.0.0.0"**);  
 local\_Address = **new** InetSocketAddress(local\_network, Integer.valueOf(port1));  
  
 socket = **new** DatagramSocket(**null**);  
 socket.bind(local\_Address);  
 socket.setBroadcast(**true**);  
  
 String ip = ipAddress.getText().toString();  
 **int** port = Integer.valueOf(**this**.port.getText().toString());  
  
 send\_buffer[0] = **'h'**;  
 send\_buffer[1] = **'e'**;  
 send\_buffer[2] = **'l'**;  
 send\_buffer[3] = **'l'**;  
 send\_buffer[4] = **'o'**;  
 send\_buffer[5] = **' '**;  
  
 ArrayList<Byte> send\_Bytes = **new** ArrayList<Byte>();  
 **for**(**int** i = 0; i< send\_buffer.length;i++)  
 {  
 send\_Bytes.add(send\_buffer[i]);  
  
 }  
  
 String name = **this**.name.getText().toString();  
 **byte**[] text = name.getBytes(StandardCharsets.UTF\_8);  
 **for**(**int** i = 0; i< text.length; i++)  
 {  
 send\_Bytes.add(text[i]);  
 }  
 **byte** space = **' '**;  
 send\_Bytes.add(space);  
  
 String message = Message.getText().toString();  
  
 text = message.getBytes(StandardCharsets.UTF\_8);  
 **for**(**int** i = 0; i< text.length; i++)  
 {  
 send\_Bytes.add(text[i]);  
 }  
  
 send\_buffer = **new byte**[200];  
  
 **int** j = 0;  
 **for**(**int** i = 0; i< send\_Bytes.size(); i++)  
 {  
 send\_buffer[i] = send\_Bytes.get(i);  
 j++;  
 }  
 **for**(**int** i = j; i< send\_buffer.length; i++)  
 {  
 send\_buffer[i] = **' '**;  
 }  
  
 InetAddress remote\_address = InetAddress.getByName(ip);  
 sent\_Packet = **new** DatagramPacket(send\_buffer, send\_buffer.length, remote\_address, port);  
  
 **final** Boolean[] no = {**false**};  
 Runnable sender = **new** Runnable() {  
 @Override  
 **public void** run() {  
 Log.e(**"TEST"**, **"SENDING THREAD IS RUNNING"**);  
 **try** {  
  
 socket.send(sent\_Packet);  
  
 }  
 **catch**(IOException e)  
 {  
 e.printStackTrace();  
 no[0] = **true**;  
  
 }  
 **catch** (Exception e)  
 {  
 e.printStackTrace();  
 no[0] = **true**;  
  
 }  
  
 **try** {  
  
 socket.close();  
 }  
 **catch** (Exception e)  
 {  
  
 }  
 **if**(no[0])  
 {  
 GetActicivity().runOnUiThread(() -> {**int** n = Integer.valueOf((**"abc"**));});*//Вызвать исключение в случае неудачи* }  
  
 }  
 };  
  
 Thread sending\_thread = **new** Thread(sender);  
 sending\_thread.start();;  
  
 Thread now = Thread.currentThread();  
 *//now.join();* DB.GetDB(GetContext(), **"Messages.db"**).GetSentMessage(name,ip, String.valueOf(port), message);  
 }  
 **catch** (UnknownHostException e)  
 {  
 e.printStackTrace();  
 **socket**.close();  
  
 **return**;  
 }  
 **catch** (Exception e)  
 {  
  
 e.printStackTrace();  
 **socket**.close();  
 **return**;  
 }  
  
 }  
 **else** {  
 **if**(**sent**)  
 {  
 DB.*GetDB*(GetContext(), **"Messages.db"**).DeleteSentMessage(**id**);  
 }  
 **else** {  
 DB.*GetDB*(GetContext(), **"Messages.db"**).DeleteRecipientMessage(**id**);  
 }  
 }  
  
 Back\_Click(v);  
 }  
  
}

# Демонстрация работы приложения

Данное приложение предназначено для обмена сообщениями между двумя устройствами, поэтому будут использоваться 2 телефона:

1. *BQ-5731L\_08* с операционной системой *Android 9* и IP-адресом *192.168.0.9*.
2. *Redmi 4A* версии *MIUI Global 10.2.3 Стабильная 10.2.3.0 (NCCMIXM)* с операционной системой *Android 7.1.2 N2G47H*.
   1. Отправка сообщений с телефона 1 на телефон 2

На телефоне 1 открыто окно отправки сообщения, как показано на рисунке 5.

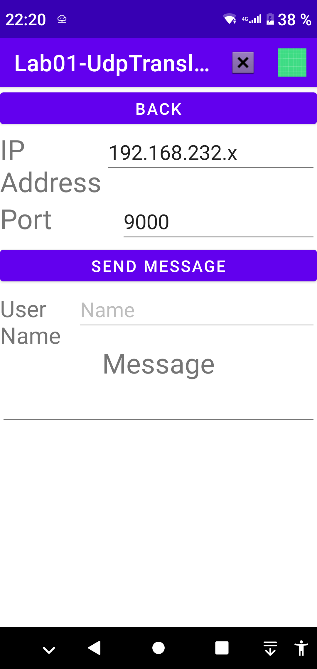


Рисунок 5 – Запущенное приложение на телефоне 1.

На телефоне 2 открыто окно списка полученных сообщений, как показано на рисунке 6.

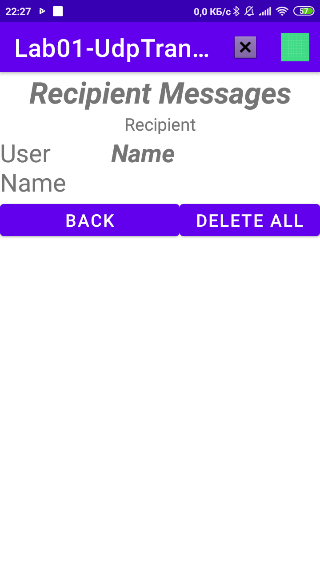


Рисунок 6 – Запущенное приложение на телефоне 2.

Введём текст сообщения и IP-адрес 2-ого телефона (для отправки туда), как показано на рисунке 7.

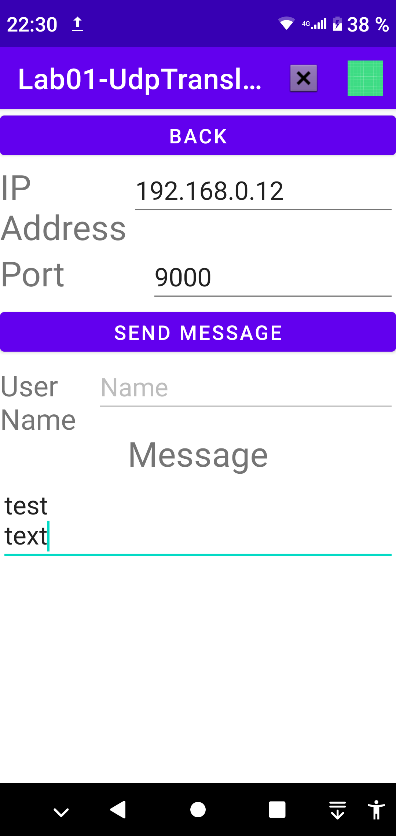


Рисунок 7 – Данные отправляемого сообщения

После нажатия кнопки *send message*, сообщение было отправлено на телефон 2, как показано на рисунке 8.

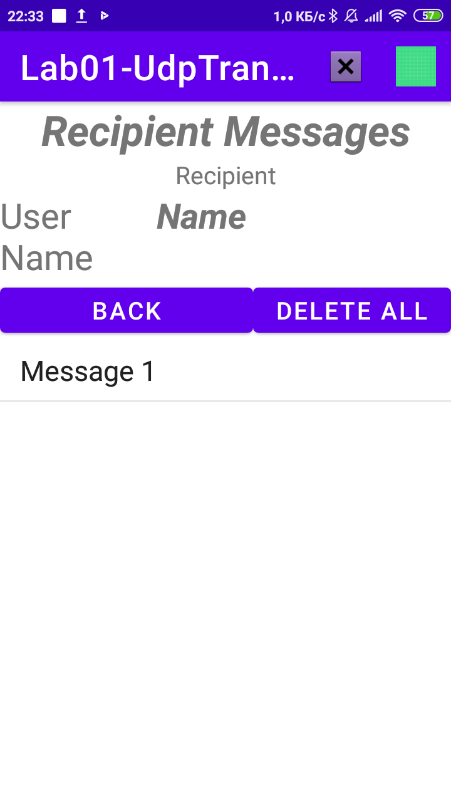


Рисунок 8 – Полученное сообщение

Открытое сообщение показано на рисунке 9.

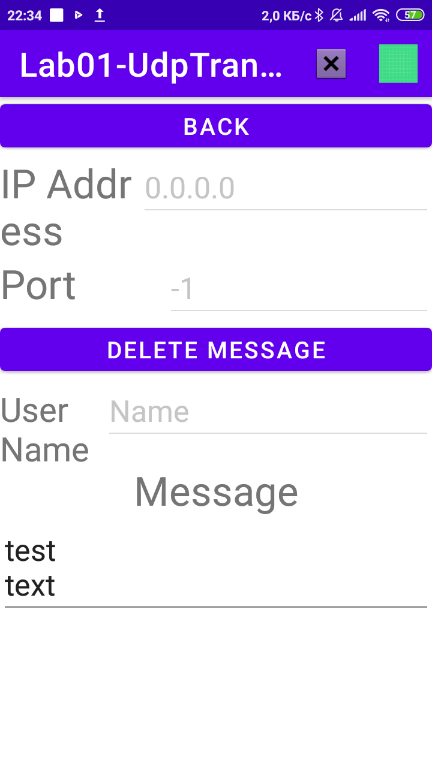


Рисунок 9 – Полученное сообщение

Теперь, на телефоне 1 сменим имя пользователя, как показано на рисунке 10.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – Смена имени пользователя

Данные нового отправляемого сообщения показаны на рисунке 12.

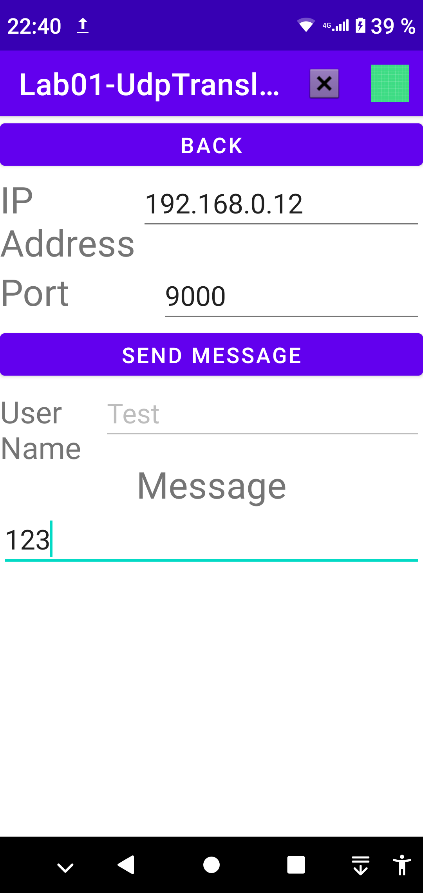


Рисунок 12 – Данные отправляемого сообщения

После отправки данного сообщения, на телефоне 2, уже стало 2 полученных сообщения, как показано на рисунке 13.

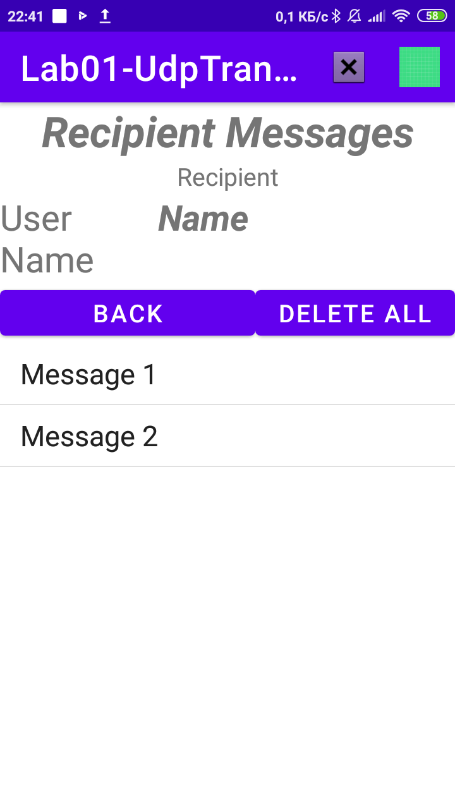


Рисунок 13 – Список полученных сообщений

Открытое, это новое полученное сообщение показано на рисунке 14.

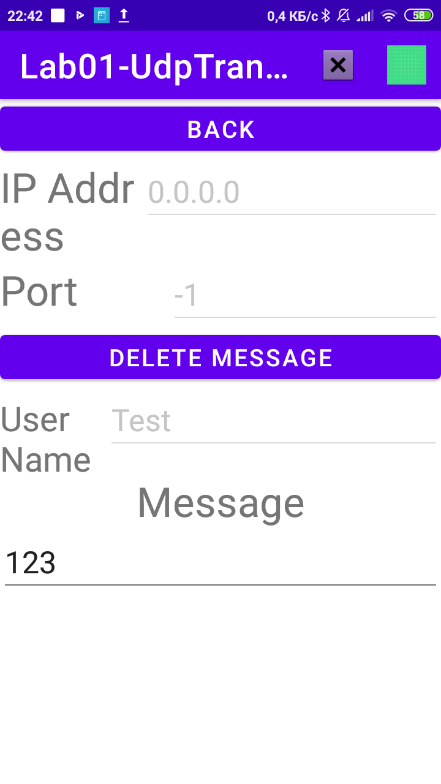


Рисунок 14 – Данные полученного сообщения

* 1. Отправка сообщений с телефона 2 на телефон 1

На телефоне 2 открыто окно отправки сообщения, как показано на рисунке 15.

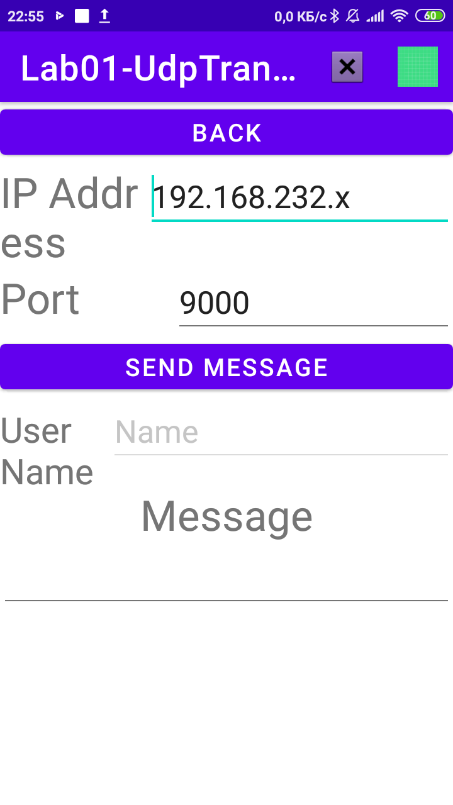


Рисунок 15 – Запущенное приложение на телефоне 2.

На телефоне 1 открыто окно списка полученных сообщений, как показано на рисунке 16.

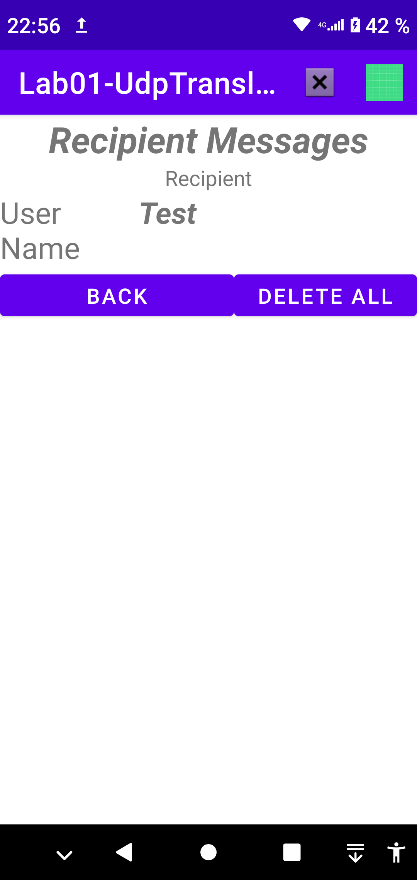


Рисунок 16 – Запущенное приложение на телефоне 1.

Введём данные отправляемого сообщения, как показано на рисунке 17.

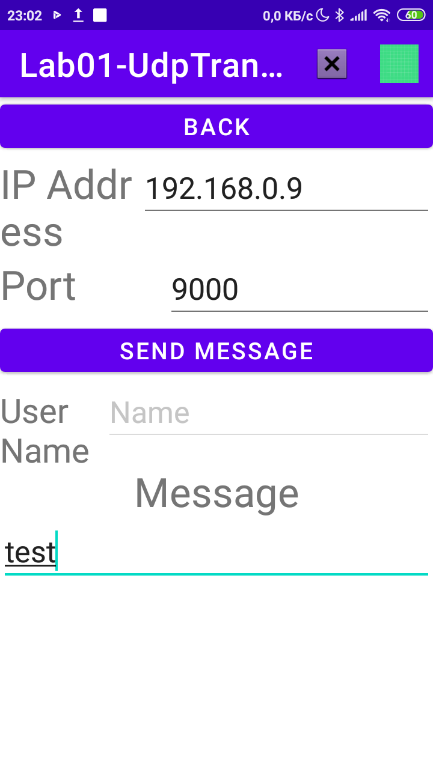


Рисунок 17 – Данные отправляемого сообщения

После отправки сообщения, оно отобразится на телефоне 1, как показано на рисунке 18.

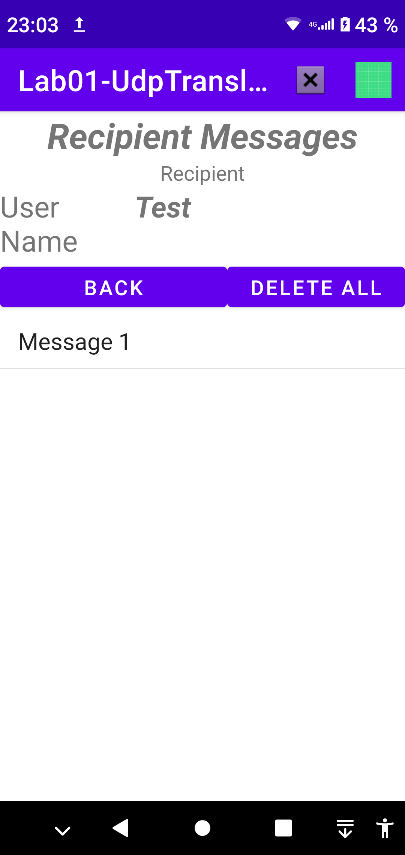


Рисунок 18 – Полученное сообщение

Открытое это полученное сообщение показано на рисунке 19.

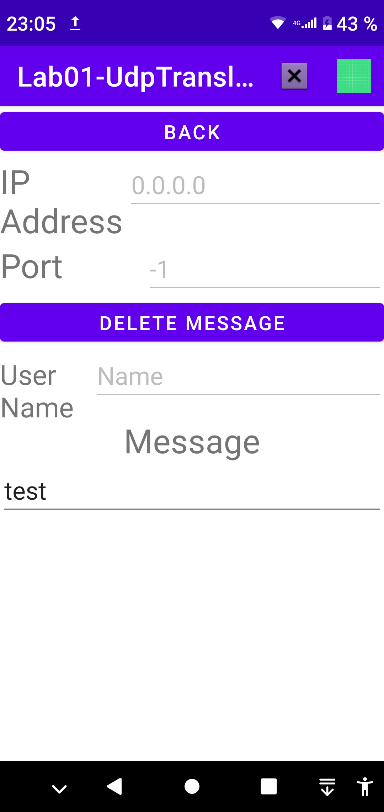


Рисунок 19 – Полученное сообщение

Изменим порт, на котором телефон 1 будет принимать сообщения, как показано на рисунке 20.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20 – смена номера порта

# Вывод

Освоено программирование обмена данными по протоколу *UDP*.