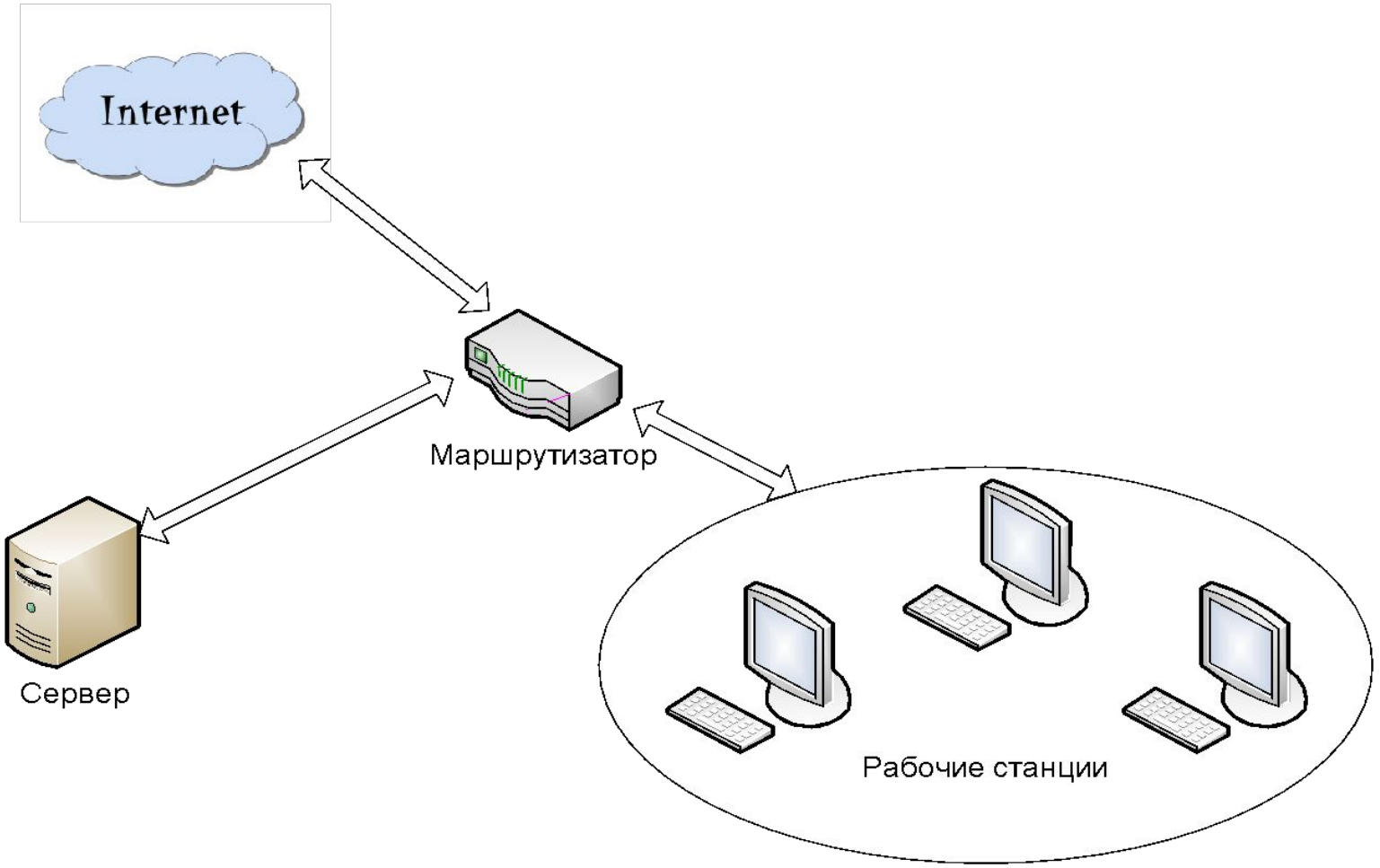
Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Филиал ФГБОУ ВПО «ЮУрГУ» (НИУ) в г. Златоусте
Факультет Техники и технологии
Кафедра «Математика и вычислительная техника»

Выпускная квалификационная работа

Программное обеспечение мониторинга сетевого оборудования с помощью мобильного устройства

Выполнил:
студент группы ФТТ-551
Ярушин М.Ю.
Руководитель :
ассистент **Манатин А.А.**

Существующее решение



Постановка задачи

Цель: удаленный мониторинг сетевого оборудования с помощью мобильных устройств

Объект : клиент-серверная система

Предмет: программный продукт, реализующий просмотр состояния сетевых устройств

Сравнение программ-аналогов

Мониторинг серверов «PING»

- Позволяет получать список IP для указанного хоста
- Позволяет проверять маршрут до выбранного хоста с помощью стандартной команды traceroute

Net Look

- При загрузке программы мы сразу видим личную сетевую информацию
- Возможность просканировать сеть и найти все подключенные устройства

WirelessNetView

- Проверка соединения каждые 10 секунд
- Большой функционал

Задачи

- изучить методы создания TCP клиент-серверного приложения
- рассмотреть методы для обращения к терминальной оболочке системы
- изучить методы проверки соединения с определенным сетевым ресурсом
- изучить методы программной отправки сообщения на почту
- спроектировать и реализовать программу мониторинга сетевого оборудования и удаленного управления им с помощью мобильного устройства

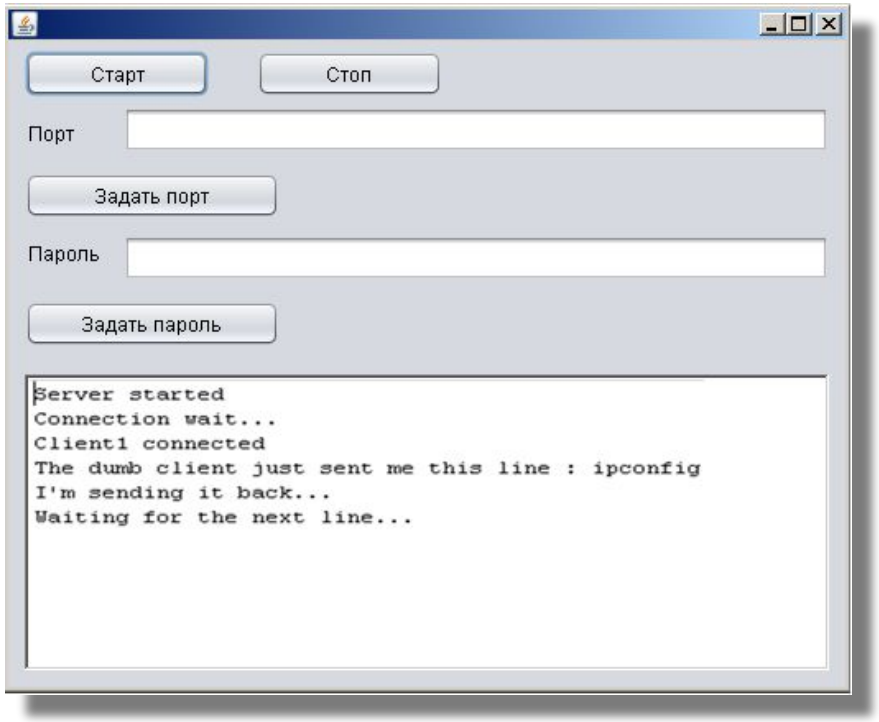
Выбор среды разработки

Преимущества использования Eclipse:

- Платформа для разработки расширений
- Бесплатность среды разработки
- Широкий спектр инструментов для процесса разработки приложений
- Высокая производительность полученного программного продукта
- Платформо-независимый продукт

Сервер

- Server – базовый класс
- Control – класс создающий сокет и ожидающий подключение клиентов
- ClientThread – класс для работы с клиентами в другом потоке
- cmdMain – класс, реализующий отправку команды оболочке cmd



Класс Control

```
private void StartServer(){
        try {
            ServerSocket ss=new ServerSocket(PORT);
             while(!shutdown){
                 Socket incoming=ss.accept();
                 System.out.println("Client"+numClient+
                "connected");
                 ClientThread client=new ClientThread(incoming,
                numClient);
                 clientList.add(client);
                 Thread t=new Thread(client);
                 numClient++;
                 t.start();
            }
        }catch (IOException ex) {
            System.out.println("Server internal error
               "+ex.getMessage());
        }
    }
```

Класс ClientThread 1

```
  public void run() {
          while(s!=null){
                  try {
                      sin = s.getInputStream();
                      sout = s.getOutputStream();
                      new DataInputStream(sin);
                      out = new DataOutputStream(sout);
                      if(in.readUTF()== pas){
                         while(true&&!shutdown){
                             line = in.readUTF();
                             System.out.println("The dumb client just
sent                        me this line : " + line);
                             System.out.println("I'm sending it
back...");
                             try {
                cmd = new String(cmdMain.cmdMain(line));
             } catch (InterruptedException e) {
               e.printStackTrace();
           }
                             ...
```

Класс ClientThread 2

```
…
            out.writeUTF(cmd);
            out.flush();
          System.out.println("Waiting for the next line...");
               System.out.println();}}
            else break;
          } catch (IOException ex) {
          System.out.println("Error initialization clients
           streams:"+ex.getMessage());
          }finally{
          try {
           sin.close();
           shutdown=true;
           s.close();
           s=null;
           Control.objControl.ShutdownClient(this);
           System.out.println("Client disconnect");
        } catch (IOException ex) {
               System.out.println("Client thread error:"+ex.getMessage());}
      }
    }
  }
```

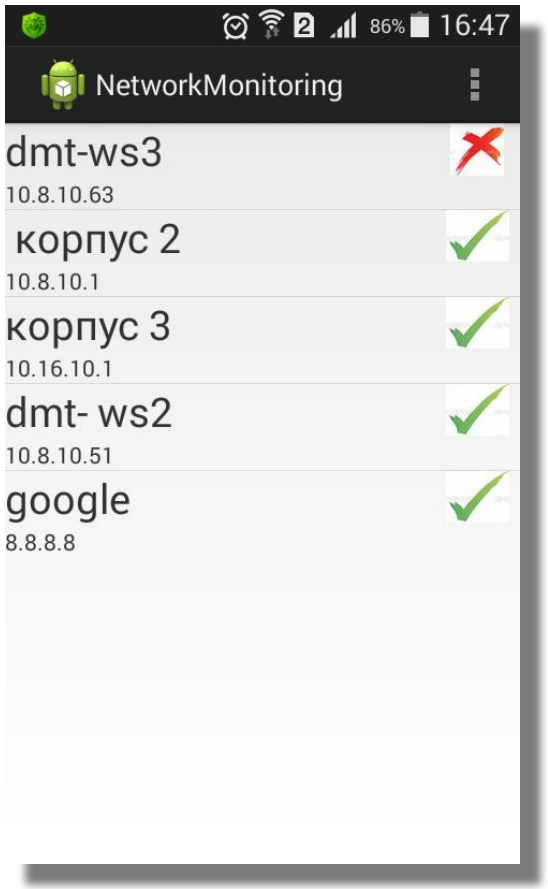
Класс cmdMain 1

```
public static String cmdMain(String cmd) throws InterruptedException{

    obj = new cmdMain();

    command = new String[]{"cmd.exe","/c","start "+ cmd +"
>code1.txt"};



    output = obj.executeCommand(command);

return output;
 }
…
```

Класс cmdMain 2

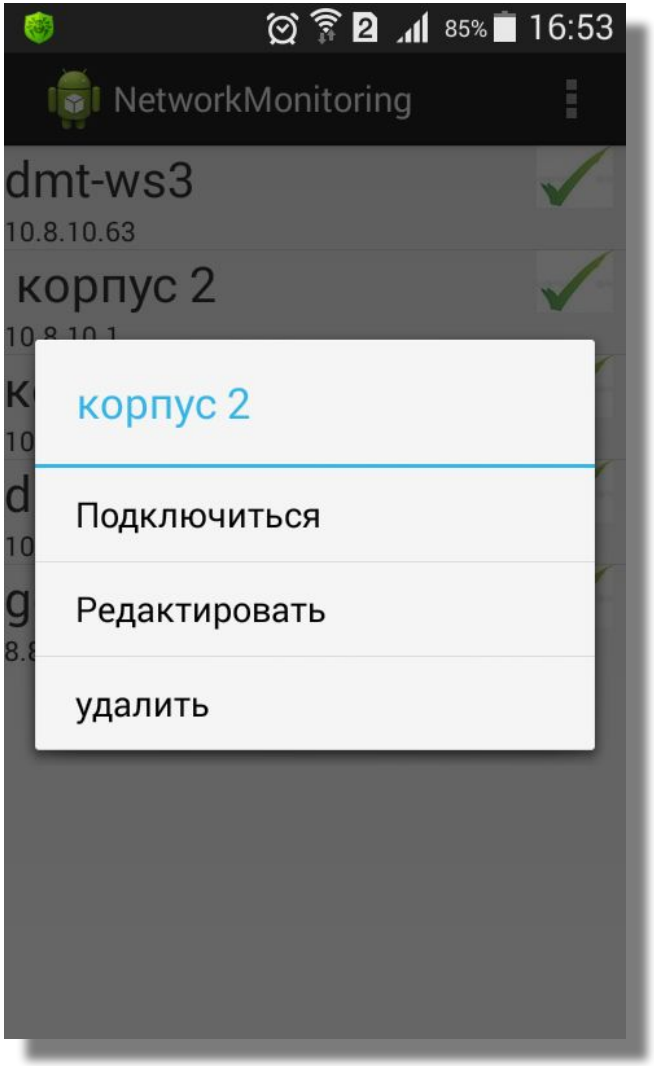
```
…
private String executeCommand(String[] command) throws InterruptedException{
 Runtime rt = Runtime.getRuntime();
   try{
    pr = rt.exec(command);
    pr.waitFor();
    file = new File("code1.txt");
    fis = new FileInputStream(file);
    isr = new InputStreamReader(fis, "cp1251");
    red = new BufferedReader(isr);
    String line = "";
    while((line = red.readLine()) != null){
        output.append(line + "\n");
    }
      }catch (IOException e){
    e.printStackTrace();
   }
   return output.toString();
 }
```

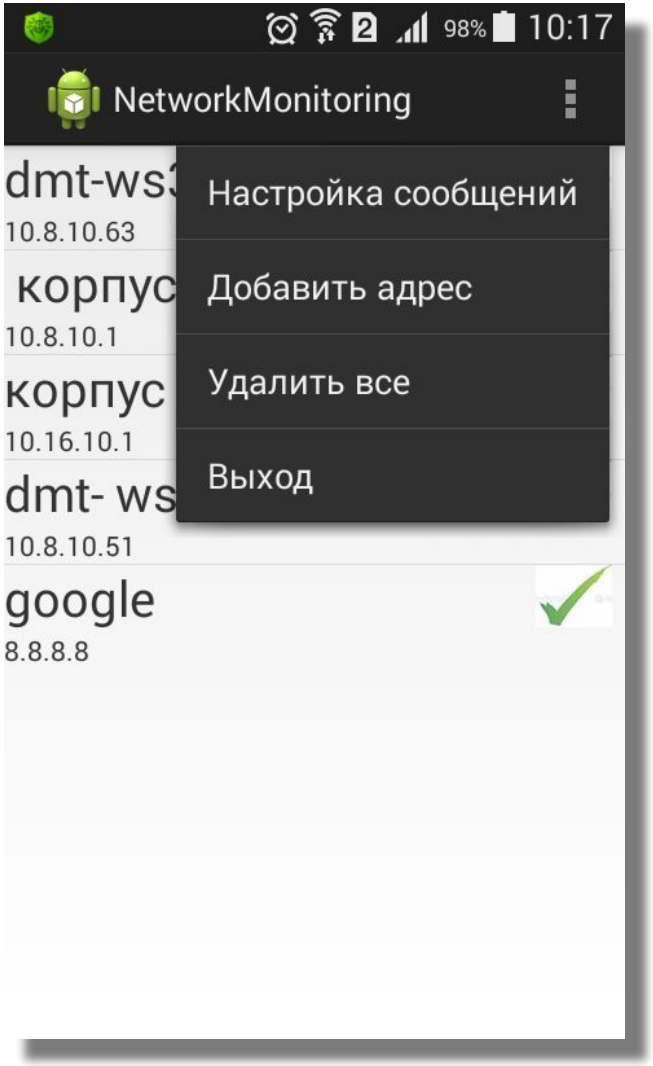
Мобильный клиент




```
Runtime runtime = Runtime.getRuntime();
    try{
      String pingCmd = "/system/bin/ping -w 1 -c 1"
             + ex;
      Process  mIpAddrProcess =
             runtime.exec(pingCmd);
      int mExitValue =mIpAddrProcess.waitFor();
      if(mExitValue==0){
                return true;
     }
     …
    }
```

Меню программы

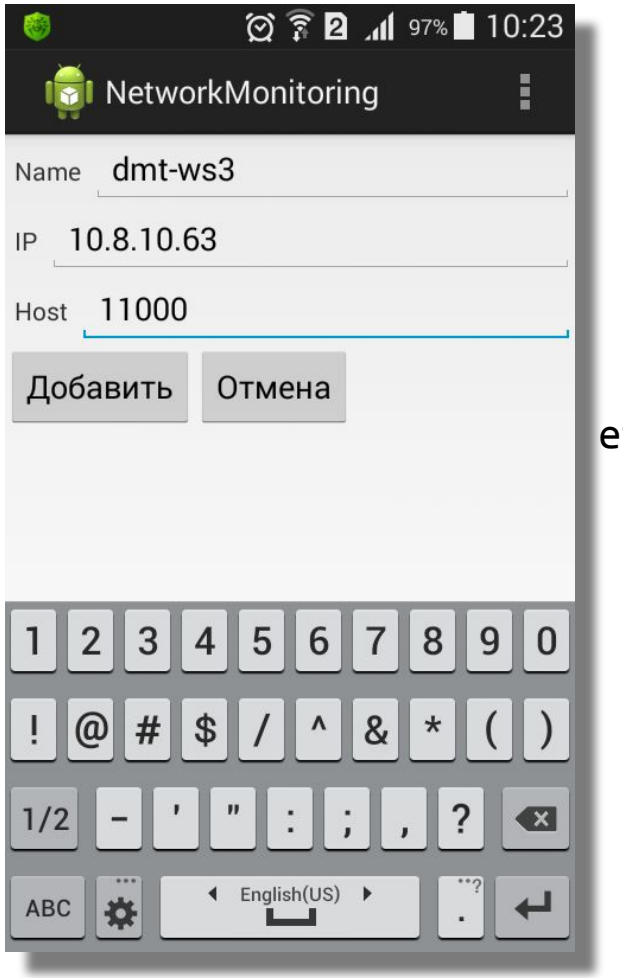




Хранение информации сетевых устройств

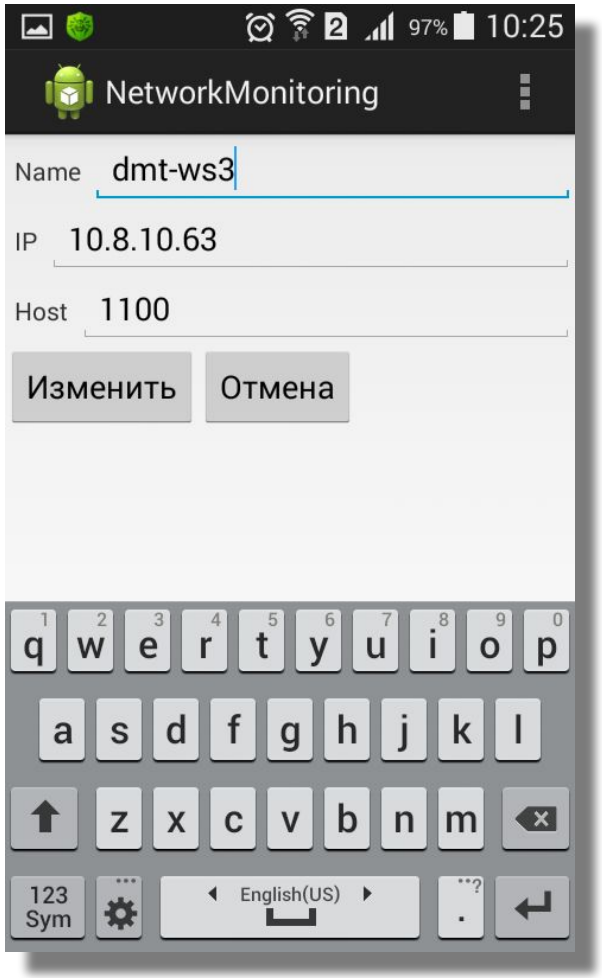
```
public class ipDataBase extends SQLiteOpenHelper implements
                    BaseColumns {

private static final String DATABASE_NAME = "mydatabase.db";
private static final String DATABASE_TABLE = "contact2";

public static final String IP_NAME_COLUMN = "ip_name";
public static final String IP_COLUMN = "ip_address";
public static final String IP_PORT = "ip_port";


private static final String DATABASE_CREATE_SCRIPT = "create table "
+ DATABASE_TABLE + " (" + BaseColumns._ID
+ " integer primary key autoincrement, " + IP_NAME_COLUMN
+ " text not null, " " + IP_COLUMN
+ " text not null, " + IP_PORT + " text not null);";
@Override
public void onCreate(SQLiteDatabase db) {
    // TODO Auto-generated method stub
    db.execSQL(DATABASE_CREATE_SCRIPT);
    }
}
```

Запись в базу данных



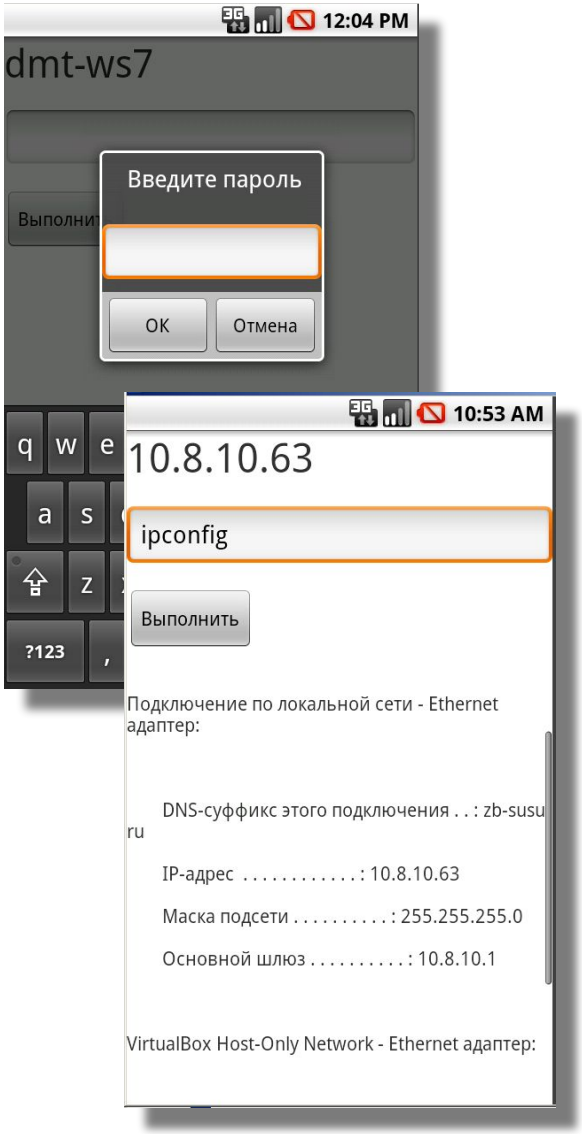

```
@Override
public void onClick(View v) {
 sdb = dbHelper.getWritableDatabase();
 switch (v.getId()) {
 case R.id.btnAdd:
 dbHelper = new ipDataBase(this, "mydatabase.db",
       null, 1);
 ContentValues newValues = new ContentValues();
 newValues.put(dbHelper.IP_NAME_COLUMN,
etName.getText().toString());
 newValues.put(dbHelper.IP_COLUMN, etIP.getText().
       toString()); sdb.insert
       ("contact2", null, newValues)
 etName.getText().clear();
 etIP.getText().clear();}
 break;
…
 }
}
```

Редактирование данных



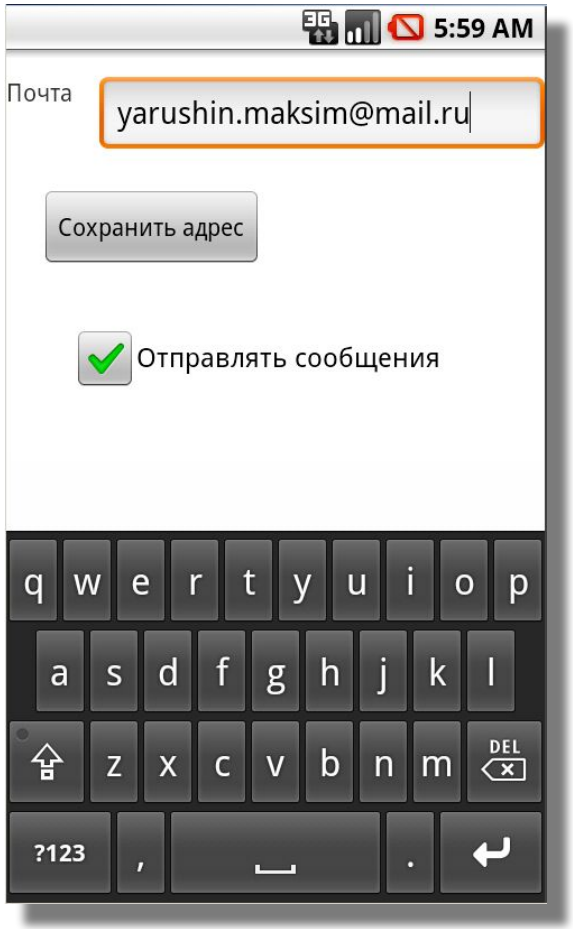

```
public void onClick(View v) {
  sdb = dbHelper.getWritableDatabase();
  dbHelper = new ipDataBase(this, "mydatabase.db",
                  null, 1);
  ContentValues cv = new ContentValues();
  switch (v.getId()) {
       case R.id.btnAdd:
          if (id.equalsIgnoreCase("")) {
                  break;
            }
          cv.put(dbHelper.NAME_COLUMN, title);
          cv.put(dbHelper.IP_COLUMN, ip);
          cv.put(dbHelper.HOST_COLUMN, port);
          int updCount = sdb.update("contact", cv,
         "_id = ?",new String[] { id });
          Intent list1= new Intent(this,
            MainActivity.class);
  startActivity(list1);
  finish();
      break;
...
}
```

Отправка команды серверу




```
class TT extends Thread{
@SuppressWarnings("resource")
@Override
  public void run() {
  super.run();
  try
  {
   InetAddress ipAddress
=InetAddress.getByName(ip);
   client = new Socket(ipAddress, port);
   in = client.getInputStream();
   out = client.getOutputStream();
   dos = new DataOutputStream(out);
   dis = new DataInputStream(in); }
   catch(Exception ex){
   ex.printStackTrace();
   }
  }
 }
}
```

Настройка передачи оповещений




```
public void SimpleEmail (String text){
    if(chak.callOnClick() == true){
     final Intent emailIntent = new

Intent(android.content.Intent.ACTION_SEND);
     emailIntent.setType("plain/text");
     emailIntent.putExtra(android.content.
                 Intent.EXTRA_EMAIL,new String[]
                 {address.getText().toString() });
     emailIntent.putExtra(android.content.
                Intent.EXTRA_SUBJECT,"
МОНИТОРИНГ");
     emailIntent.putExtra(android.content.
                 Intent.EXTRA_TEXT,"Связь с "
                 + text.toString() + " потеряна ");
     mailActivity.this.startActivity(Intent.
                 createChooser(emailIntent,null));
      }
      try {
        mMediaPlayer.start();
        Thread.sleep(1000);
        mMediaPlayer.stop();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
}
```

Заключение

- изучены методы создания TCP клиент-серверного приложения
- рассмотрены методы для обращения к терминальной оболочке системы
- изучены методы проверки соединения с определенным сетевым ресурсом
- изучены методы программной отправки сообщения на почту
- спроектирована и реализована программа мониторинга сетевого оборудования и удаленного управления им с помощью мобильного устройства