Комитет по образованию Правительства Санкт-Петербурга

**САНКТ-ПЕТЕРБРУГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ   
ТЕХНОЛОГИЙ**

**Отчет по практической работе  
МДК 01.02 «Разработка мобильных приложений»  
Разработка приложения с использованием протокола UDP для передачи данных**

Выполнил

студент группы 493:

Лукьянов Ф-И. Ш.

Преподаватель: Фомин А.В.

Санкт-Петербург 2022

Структура базы данных

База данных состоит из 2 таблиц:

1. Settings – хранит настройки приложения.
2. History – хранит историю сообщений.

ER диаграмма представлена на рисунке 1.

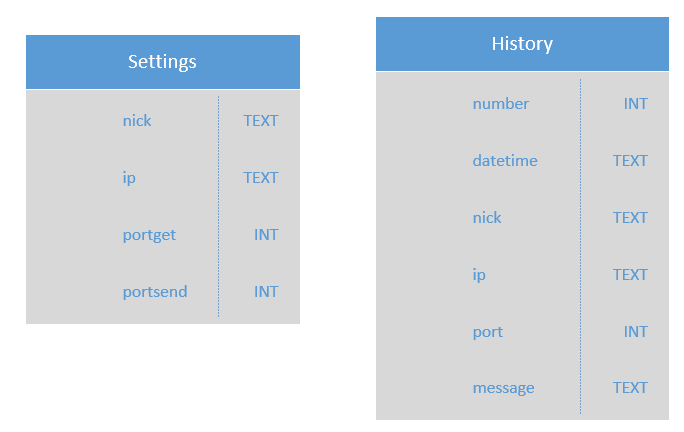


Рисунок 1 – ER диаграмма базы данных

**Таблица Settings**

Содержит сведения о настройках приложения. Таблица состоит из четырех столбцов:

1. nick – ник, отправляемый с сообщением.
2. ip – IP-адрес получателя сообщения.
3. portget – порт для получения сообщений.
4. portsend – порт для отправки сообщений.

Подробное описание столбцов представлено на рисунке 2.

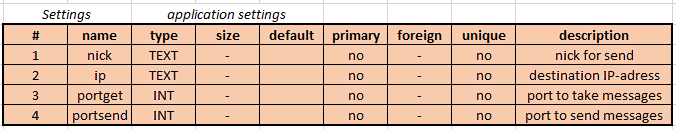


Рисунок 2 – Описание столбцов таблицы Settings

**Таблица History**

Содержит сведения о полученных и отправленных сообщениях. Таблица состоит из шести столбцов:

1. number – номер сообщения.
2. datetime – дата и время отправки/получения сообщения.
3. nick – ник отправителя.
4. ip – IP-адрес, с которого или на который было отправлено сообщение.
5. port – порт, на который было отправлено/получено сообщение.
6. message – текст сообщения.

Подробное описание столбцов представлено на рисунке 3.

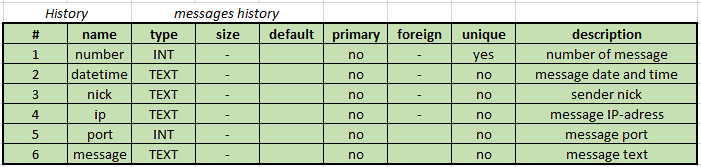


Рисунок 3 – Описание столбцов таблицы location

Интерфейс приложения

Приложение состоит из 3 форм:

1. Main Menu: стартовая форма, служит для навигации в приложении, а также для ввода и отправки сообщения, имеет историю сообщений.
2. Settings: служит для редактирования и сохранения настроек приложения.
3. Message: запускается после выбора сообщения и отображает всю информацию о сообщении.

**Форма Main Menu**

На рисунке 4 показан макет внешнего вида формы главного меню.



Рисунок 4 – Макет формы Main Menu

На рисунке 5 показан внешний вид формы главного меню в приложении.

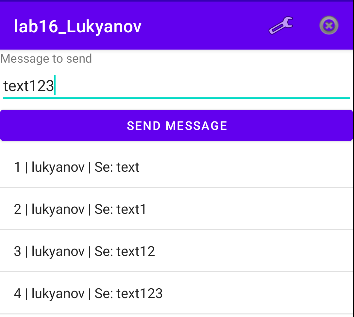


Рисунок 5 – Форма Main Menu в приложении

**Форма Settings**

На рисунке 6 показан макет внешнего вида формы настроек.

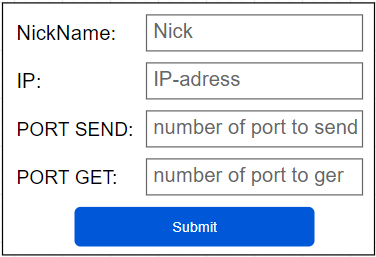


Рисунок 6 – Макет формы Settings

На рисунке 7 показан внешний вид формы настроек в приложении.

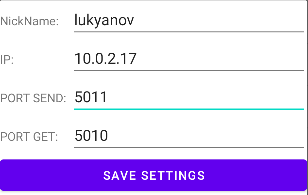


Рисунок 7 – Форма Settings в приложении

**Форма Message**

На рисунке 8 показан макет внешнего вида формы сообщения.

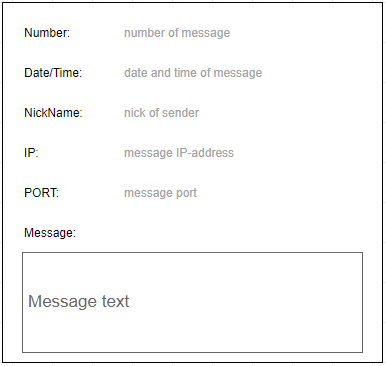


Рисунок 8 – Макет формы Message

На рисунке 9 показан внешний вид формы сообщения в приложении.

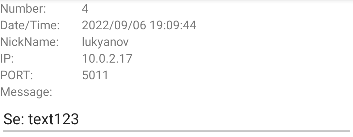


Рисунок 9 – Форма Message в приложении

Демонстрация работы приложения

При запуске приложения отображается история сообщений, которая берётся из базы данных (рис. 10):

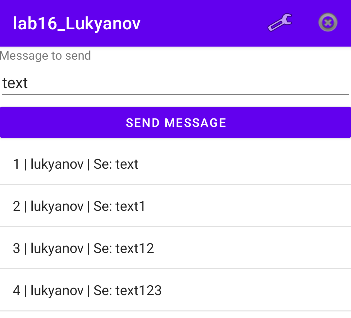


Рисунок 10 – Запуск приложения

При нажатии на кнопку в виде ключа откроется диалог с редактированием настроек, которые загружаются из базы данных (рис 11):

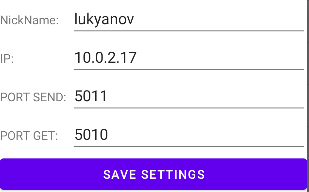


Рисунок 11 – Диалог с настройками

При нажатии на сообщение из списка запустится другая Activity с информацией о данном сообщении (рис. 12):

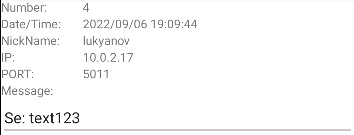


Рисунок 12 – Информация о сообщении

Изменим настройки и текст сообщения (рис. 13):

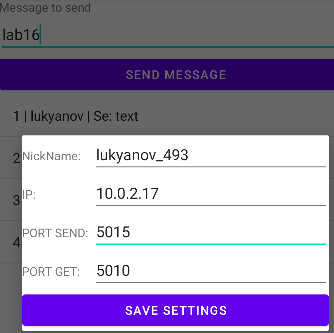


Рисунок 13 – Изменённые настройки и текст сообщения

После отправки сообщения оно добавляется в историю и в базу данных (рис 14 и рис. 15):

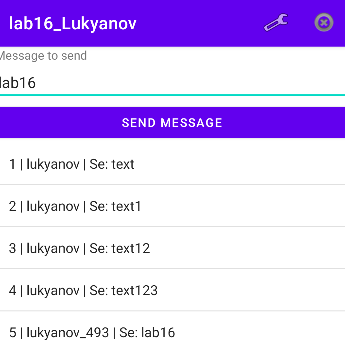


Рисунок 14 – История с отправленным сообщением

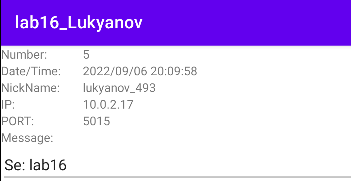


Рисунок 15 – Отправленное сообщение

Изменим настройки для получения, отправленного нами сообщения (рис. 16):

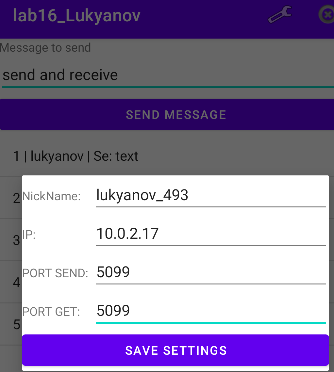


Рисунок 16 – Настройки для получения отправленного сообщения

История сообщения после отправки представлена на рисунке 17.

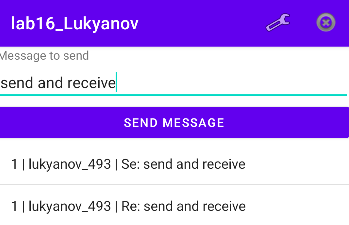


Рисунок 17 – История с отправленным и полученным сообщением

Отправленное сообщение представлено на рисунке 18, полученное на рисунке 19.

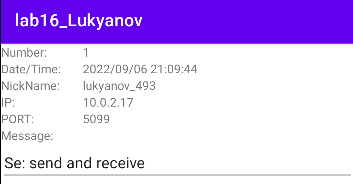


Рисунок 18 – Отправленное сообщение

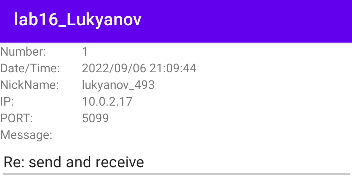


Рисунок 19 – Полученное сообщение

При нажатии на кнопку в виде крестика история очищается (рис. 20):

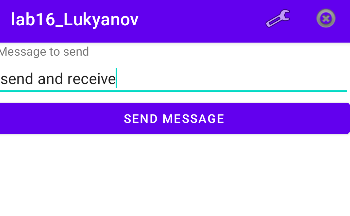


Рисунок 20 – Очищенная история

Также приложение имеет собственную иконку (рис. 21):

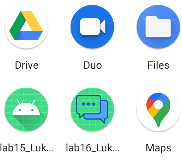
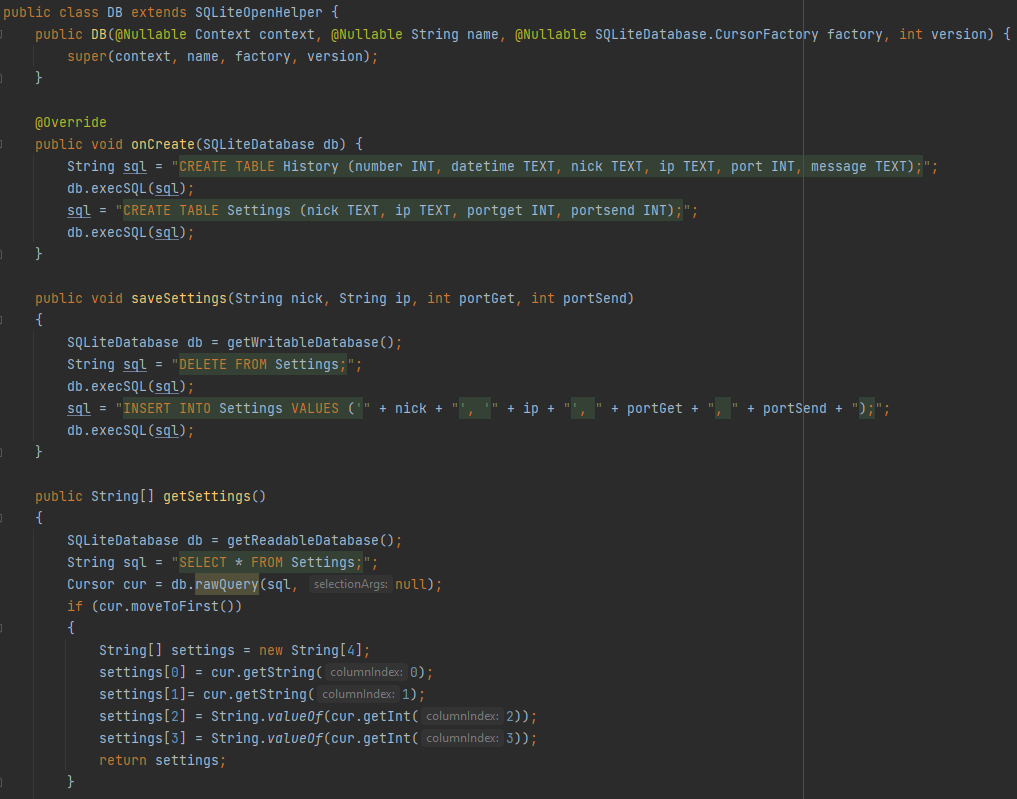


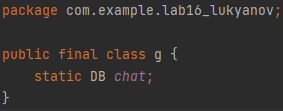
Рисунок 21 – Иконка приложения

Описание участков кода

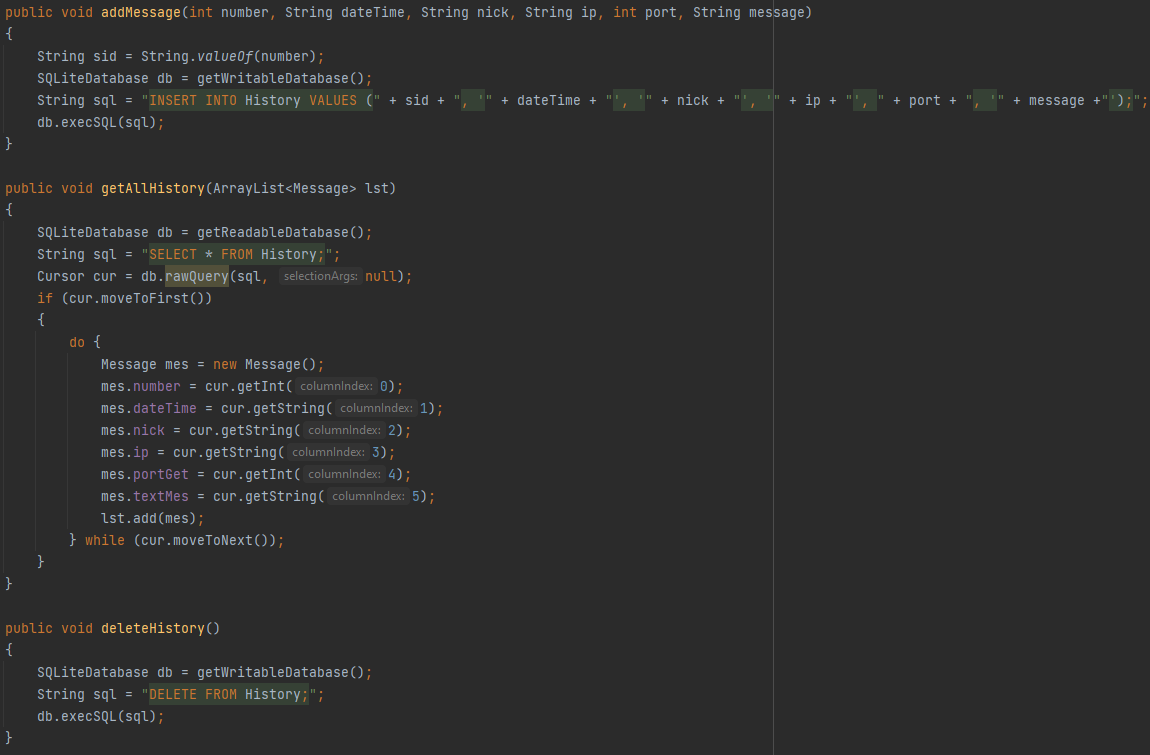
Создание базы с двумя таблицами и методы для сохранения и получения настроек приложения:



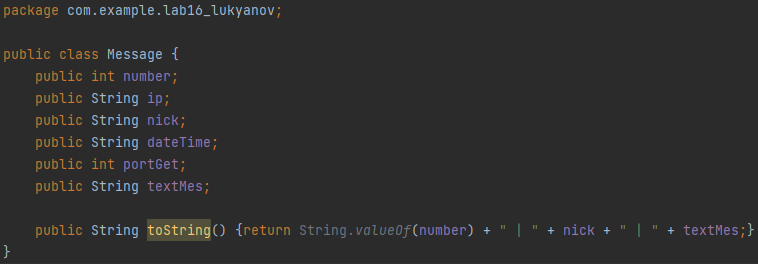
Экземпляр класса базы:



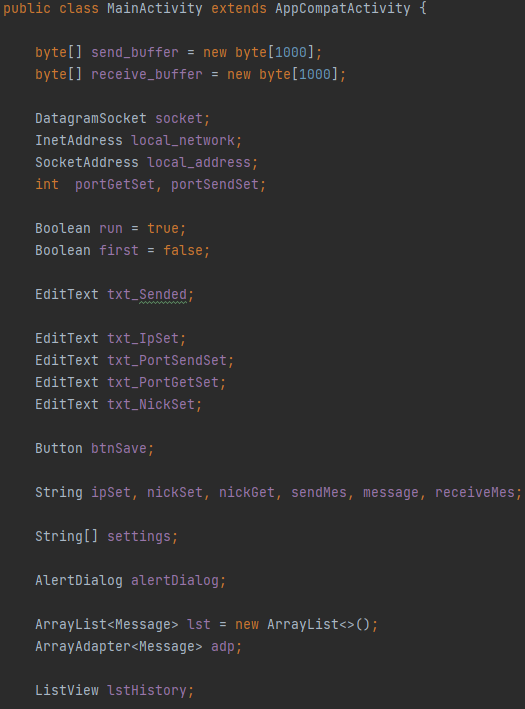
Работа с историей сообщений:



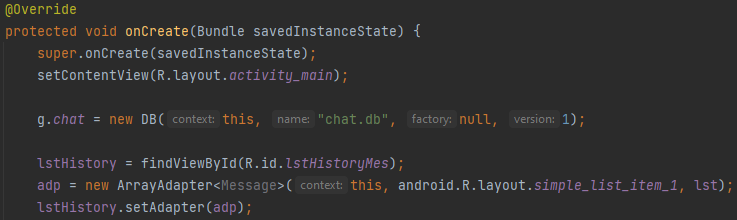
Класс сообщения:



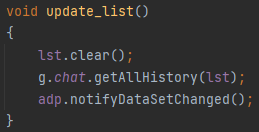
Переменные основной формы:



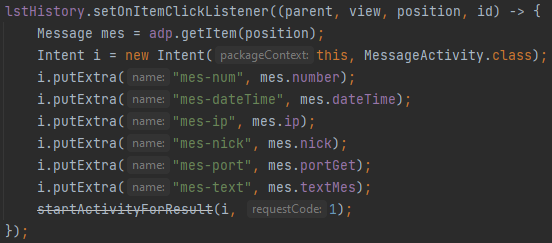
Создание базы данных и добавление списка с историей сообщений:



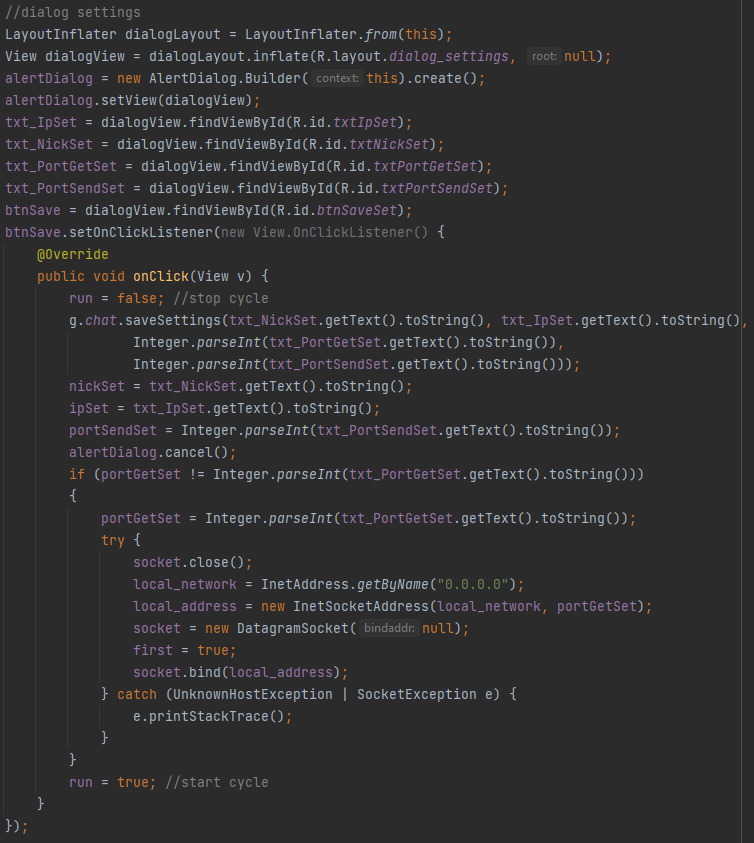
Метод для обновления списка базы данных:



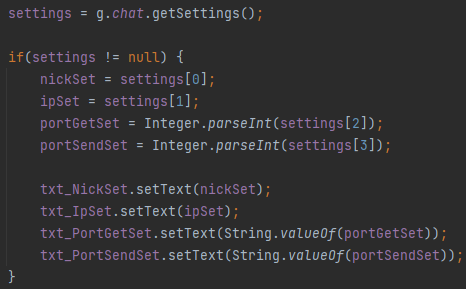
Событие клика по элементу списка для просмотра на другой форме:



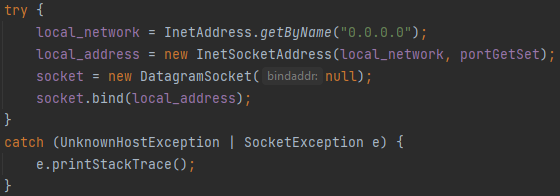
Настройки диалога и событие кнопки для сохранения настроек:



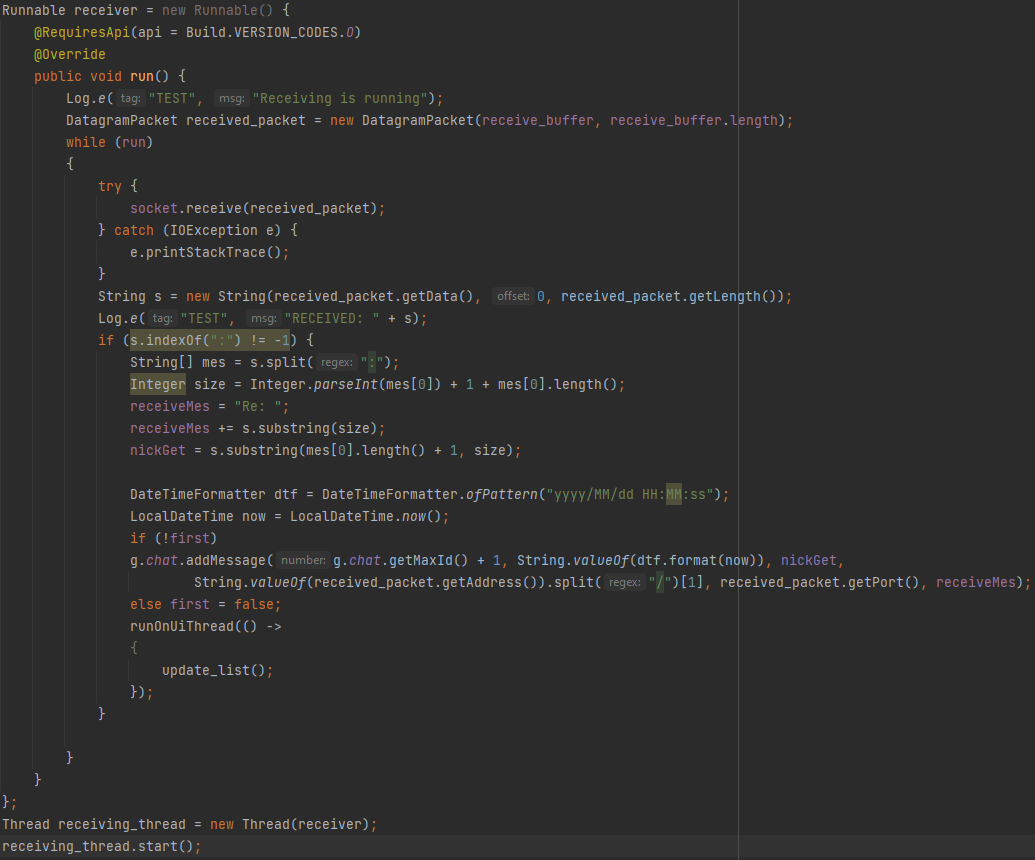
Загрузка настроек из базы данных, если они имеются:



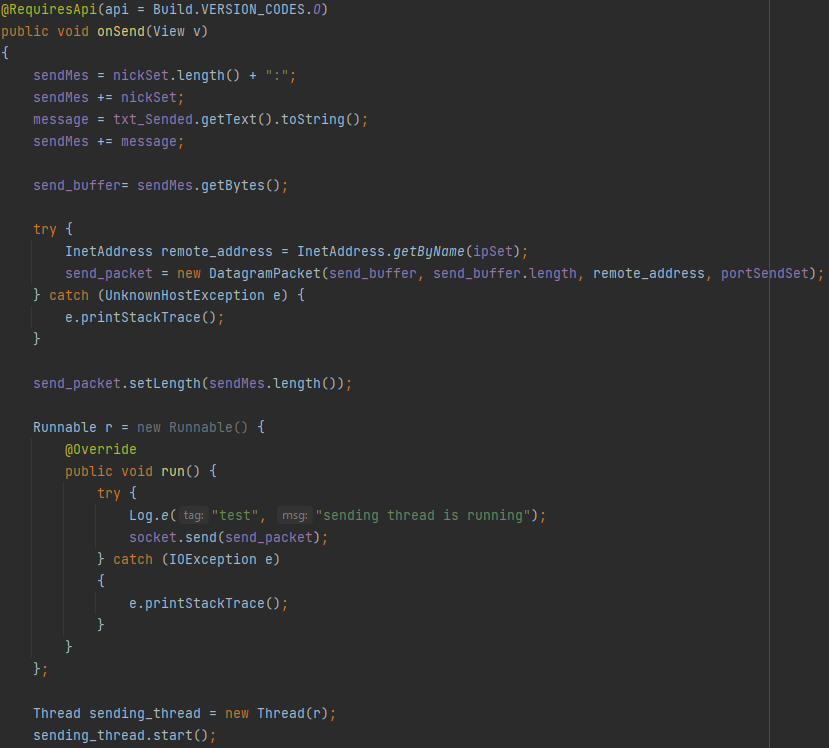
Создание сокета с указанным в настройках портом:



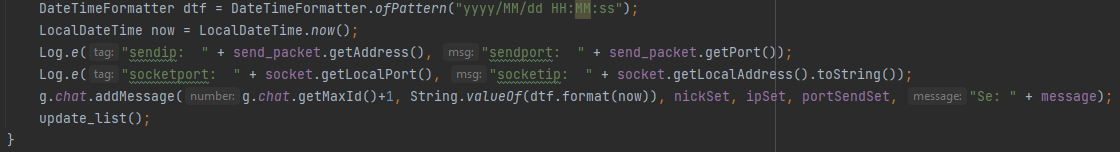
Поток для получения сообщений и сохранения в базу:



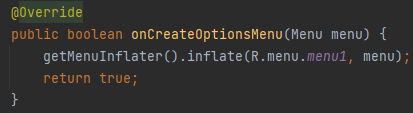
Событие клика для отправки сообщения:



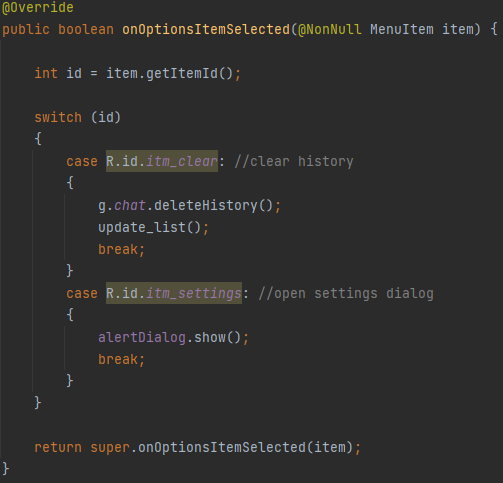
Сохранение отправленного сообщения в базу:



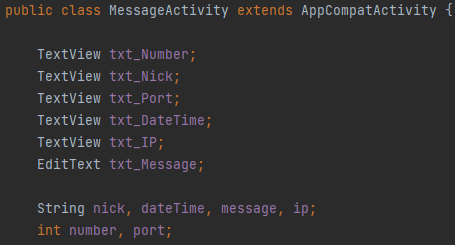
Создание меню:



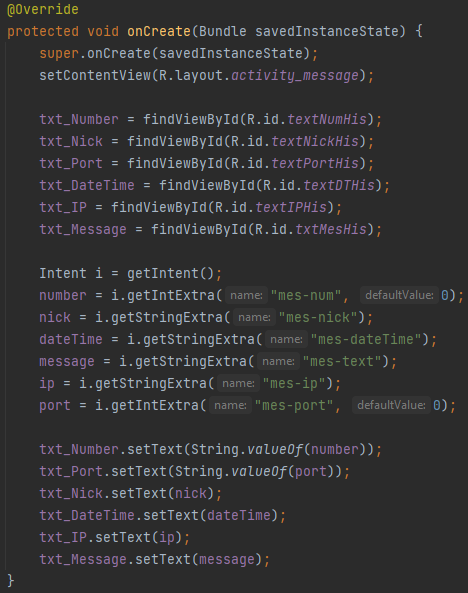
Событие для клика по элементам меню:



Переменные формы для отображения информации о сообщении:



Получение отправленной на форму информации о выбранном сообщении:



Код приложения

**Класс базы данных «DB»:**

package com.example.lab16\_lukyanov;  
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, int version) {  
 super(context, name, factory, version);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String sql = "CREATE TABLE History (number INT, datetime TEXT, nick TEXT, ip TEXT, port INT, message TEXT);";  
 db.execSQL(sql);  
 sql = "CREATE TABLE Settings (nick TEXT, ip TEXT, portget INT, portsend INT);";  
 db.execSQL(sql);  
 }  
  
 public void saveSettings(String nick, String ip, int portGet, int portSend)  
 {  
 SQLiteDatabase db = getWritableDatabase();  
 String sql = "DELETE FROM Settings;";  
 db.execSQL(sql);  
 sql = "INSERT INTO Settings VALUES ('" + nick + "', '" + ip + "', " + portGet + ", " + portSend + ");";  
 db.execSQL(sql);  
 }  
  
 public String[] getSettings()  
 {  
 SQLiteDatabase db = getReadableDatabase();  
 String sql = "SELECT \* FROM Settings;";  
 Cursor cur = db.rawQuery(sql, null);  
 if (cur.moveToFirst())  
 {  
 String[] settings = new String[4];  
 settings[0] = cur.getString(0);  
 settings[1]= cur.getString(1);  
 settings[2] = String.*valueOf*(cur.getInt(2));  
 settings[3] = String.*valueOf*(cur.getInt(3));  
 return settings;  
 }  
 else return null;  
 }  
  
 public int getMaxId()  
 {  
 SQLiteDatabase db = getReadableDatabase();  
 String sql = "SELECT MAX(number) FROM History";  
 Cursor cur = db.rawQuery(sql, null);  
 if (cur.moveToFirst()) return cur.getInt(0);  
 return -1;  
 }  
  
 public void addMessage(int number, String dateTime, String nick, String ip, int port, String message)  
 {  
 String sid = String.*valueOf*(number);  
 SQLiteDatabase db = getWritableDatabase();  
 String sql = "INSERT INTO History VALUES (" + sid + ", '" + dateTime + "', '" + nick + "', '" + ip + "', " + port + ", '" + message +"');";  
 db.execSQL(sql);  
 }  
  
 public Message getMessage(int number)  
 {  
 String sid = String.*valueOf*(number);  
 SQLiteDatabase db = getReadableDatabase();  
 String sql = "SELECT \* FROM History WHERE number = " + sid + ";";  
 Cursor cur = db.rawQuery(sql, null);  
 if (cur.moveToFirst()) {  
 Message mes = new Message();  
 mes.number = cur.getInt(0);  
 mes.dateTime = cur.getString(1);  
 mes.nick = cur.getString(2);  
 mes.ip = cur.getString(3);  
 mes.portGet = cur.getInt(4);  
 mes.textMes = cur.getString(5);  
 return mes;  
 }  
 else return null;  
 }  
  
 public void getAllHistory(ArrayList<Message> lst)  
 {  
 SQLiteDatabase db = getReadableDatabase();  
 String sql = "SELECT \* FROM History;";  
 Cursor cur = db.rawQuery(sql, null);  
 if (cur.moveToFirst())  
 {  
 do {  
 Message mes = new Message();  
 mes.number = cur.getInt(0);  
 mes.dateTime = cur.getString(1);  
 mes.nick = cur.getString(2);  
 mes.ip = cur.getString(3);  
 mes.portGet = cur.getInt(4);  
 mes.textMes = cur.getString(5);  
 lst.add(mes);  
 } while (cur.moveToNext());  
 }  
 }  
  
 public void deleteHistory()  
 {  
 SQLiteDatabase db = getWritableDatabase();  
 String sql = "DELETE FROM History;";  
 db.execSQL(sql);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
 }  
}

**Класс экземпляра базы данных «g»»:**

package com.example.lab16\_lukyanov;  
  
public final class g {  
 static DB *chat*;  
}

**Класс сообщения «Message»»:**

package com.example.lab16\_lukyanov;  
  
public class Message {  
 public int number;  
 public String ip;  
 public String nick;  
 public String dateTime;  
 public int portGet;  
 public String textMes;  
  
 public String toString() {return String.*valueOf*(number) + " | " + nick + " | " + textMes;}  
}

**Класс основной формы «MainActivity»:**

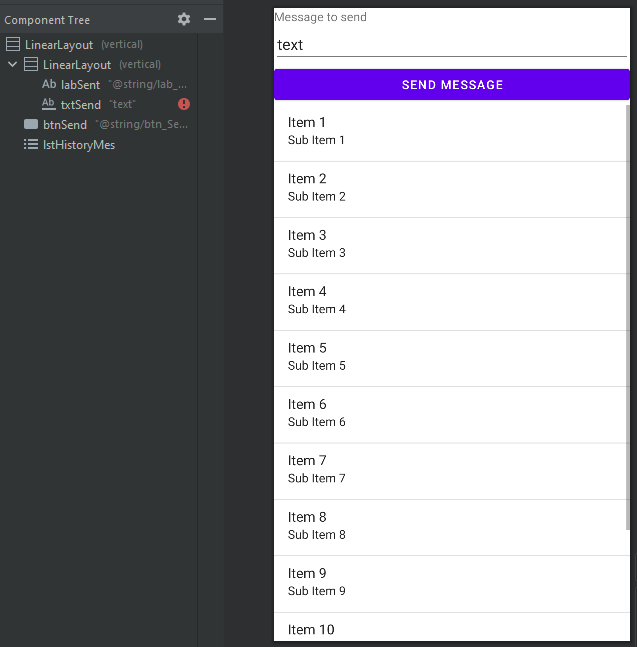
package com.example.lab16\_lukyanov;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.RequiresApi;  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Build;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ListView;  
  
import java.io.IOException;  
import java.net.DatagramSocket;  
import java.net.DatagramPacket;  
import java.net.InetAddress;  
import java.net.InetSocketAddress;  
import java.net.SocketAddress;  
import java.net.SocketException;  
import java.net.UnknownHostException;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.ArrayList;  
  
public class MainActivity extends AppCompatActivity {  
  
 byte[] send\_buffer = new byte[1000];  
 byte[] receive\_buffer = new byte[1000];  
  
 DatagramSocket socket;  
 InetAddress local\_network;  
 SocketAddress local\_address;  
 int portGetSet, portSendSet;  
  
 Boolean run = true;  
 Boolean first = false;  
  
 EditText txt\_Sended;  
  
 EditText txt\_IpSet;  
 EditText txt\_PortSendSet;  
 EditText txt\_PortGetSet;  
 EditText txt\_NickSet;  
  
 Button btnSave;  
  
 String ipSet, nickSet, nickGet, sendMes, message, receiveMes;  
  
 String[] settings;  
  
 AlertDialog alertDialog;  
  
 ArrayList<Message> lst = new ArrayList<>();  
 ArrayAdapter<Message> adp;  
  
 ListView lstHistory;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 g.*chat* = new DB(this, "chat.db", null, 1);  
  
 lstHistory = findViewById(R.id.*lstHistoryMes*);  
 adp = new ArrayAdapter<Message>(this, android.R.layout.*simple\_list\_item\_1*, lst);  
 lstHistory.setAdapter(adp);  
  
 lstHistory.setOnItemClickListener((parent, view, position, id) -> {  
 Message mes = adp.getItem(position);  
 Intent i = new Intent(this, MessageActivity.class);  
 i.putExtra("mes-num", mes.number);  
 i.putExtra("mes-dateTime", mes.dateTime);  
 i.putExtra("mes-ip", mes.ip);  
 i.putExtra("mes-nick", mes.nick);  
 i.putExtra("mes-port", mes.portGet);  
 i.putExtra("mes-text", mes.textMes);  
 startActivityForResult(i, 1);  
 });  
  
 update\_list();  
  
 //dialog settings  
 LayoutInflater dialogLayout = LayoutInflater.*from*(this);  
 View dialogView = dialogLayout.inflate(R.layout.*dialog\_settings*, null);  
 alertDialog = new AlertDialog.Builder(this).create();  
 alertDialog.setView(dialogView);  
 txt\_IpSet = dialogView.findViewById(R.id.*txtIpSet*);  
 txt\_NickSet = dialogView.findViewById(R.id.*txtNickSet*);  
 txt\_PortGetSet = dialogView.findViewById(R.id.*txtPortGetSet*);  
 txt\_PortSendSet = dialogView.findViewById(R.id.*txtPortSendSet*);  
 btnSave = dialogView.findViewById(R.id.*btnSaveSet*);  
 btnSave.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 run = false;  
 g.*chat*.saveSettings(txt\_NickSet.getText().toString(), txt\_IpSet.getText().toString(), Integer.*parseInt*(txt\_PortGetSet.getText().toString()), Integer.*parseInt*(txt\_PortSendSet.getText().toString()));  
 nickSet = txt\_NickSet.getText().toString();  
 ipSet = txt\_IpSet.getText().toString();  
 portSendSet = Integer.*parseInt*(txt\_PortSendSet.getText().toString());  
 alertDialog.cancel();  
 if (portGetSet != Integer.*parseInt*(txt\_PortGetSet.getText().toString()))  
 {  
 portGetSet = Integer.*parseInt*(txt\_PortGetSet.getText().toString());  
 try {  
 socket.close();  
 local\_network = InetAddress.*getByName*("0.0.0.0");  
 local\_address = new InetSocketAddress(local\_network, portGetSet);  
 socket = new DatagramSocket(null);  
 first = true;  
 socket.bind(local\_address);  
 } catch (UnknownHostException | SocketException e) {  
 e.printStackTrace();  
 }  
 }  
 run = true;  
 }  
 });  
  
 settings = g.*chat*.getSettings();  
  
 if(settings != null) {  
 nickSet = settings[0];  
 ipSet = settings[1];  
 portGetSet = Integer.*parseInt*(settings[2]);  
 portSendSet = Integer.*parseInt*(settings[3]);  
  
 txt\_NickSet.setText(nickSet);  
 txt\_IpSet.setText(ipSet);  
 txt\_PortGetSet.setText(String.*valueOf*(portGetSet));  
 txt\_PortSendSet.setText(String.*valueOf*(portSendSet));  
 }  
  
 txt\_Sended = findViewById(R.id.*txtSend*);  
  
 try {  
 local\_network = InetAddress.*getByName*("0.0.0.0");  
 local\_address = new InetSocketAddress(local\_network, portGetSet);  
 socket = new DatagramSocket(null);  
 socket.bind(local\_address);  
 }  
 catch (UnknownHostException | SocketException e) {  
 e.printStackTrace();  
 }  
  
 Runnable receiver = new Runnable() {  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 @Override  
 public void run() {  
 Log.*e*("TEST", "Receiving is running");  
 DatagramPacket received\_packet = new DatagramPacket(receive\_buffer, receive\_buffer.length);  
 while (run)  
 {  
 try {  
 socket.receive(received\_packet);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 String s = new String(received\_packet.getData(), 0, received\_packet.getLength());  
 Log.*e*("TEST", "RECEIVED: " + s);  
 if (s.indexOf(":") != -1) {  
 String[] mes = s.split(":");  
 Integer size = Integer.*parseInt*(mes[0]) + 1 + mes[0].length();  
 receiveMes = "Re: ";  
 receiveMes += s.substring(size);  
 nickGet = s.substring(mes[0].length() + 1, size);  
  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:MM:ss");  
 LocalDateTime now = LocalDateTime.*now*();  
 if (!first)  
 g.*chat*.addMessage(g.*chat*.getMaxId() + 1, String.*valueOf*(dtf.format(now)), nickGet, String.*valueOf*(received\_packet.getAddress()).split("/")[1], received\_packet.getPort(), receiveMes);  
 else first = false;  
 runOnUiThread(() ->  
 {  
 update\_list();  
 });  
 }  
  
 }  
 }  
 };  
  
 Thread receiving\_thread = new Thread(receiver);  
 receiving\_thread.start();  
  
  
 }  
  
  
 DatagramPacket send\_packet;  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 public void onSend(View v)  
 {  
 sendMes = nickSet.length() + ":";  
 sendMes += nickSet;  
 message = txt\_Sended.getText().toString();  
 sendMes += message;  
  
 send\_buffer= sendMes.getBytes();  
  
 try {  
 InetAddress remote\_address = InetAddress.*getByName*(ipSet);  
 send\_packet = new DatagramPacket(send\_buffer, send\_buffer.length, remote\_address, portSendSet);  
 } catch (UnknownHostException e) {  
 e.printStackTrace();  
 }  
  
 send\_packet.setLength(sendMes.length());  
  
 Runnable r = new Runnable() {  
 @Override  
 public void run() {  
 try {  
 Log.*e*("test", "sending thread is running");  
 socket.send(send\_packet);  
 } catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
 };  
  
 Thread sending\_thread = new Thread(r);  
 sending\_thread.start();  
  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:MM:ss");  
 LocalDateTime now = LocalDateTime.*now*();  
 Log.*e*("sendip: " + send\_packet.getAddress(), "sendport: " + send\_packet.getPort());  
 Log.*e*("socketport: " + socket.getLocalPort(), "socketip: " + socket.getLocalAddress().toString());  
 g.*chat*.addMessage(g.*chat*.getMaxId()+1, String.*valueOf*(dtf.format(now)), nickSet, ipSet, portSendSet, "Se: " + message);  
 update\_list();  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*menu1*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
  
 int id = item.getItemId();  
  
 switch (id)  
 {  
 case R.id.*itm\_clear*:  
 {  
 g.*chat*.deleteHistory();  
 update\_list();  
 break;  
 }  
 case R.id.*itm\_settings*:  
 {  
 alertDialog.show();  
 break;  
 }  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
  
 void update\_list()  
 {  
 lst.clear();  
 g.*chat*.getAllHistory(lst);  
 adp.notifyDataSetChanged();  
 }  
}

**Класс дополнительной формы «MessageActivity»:**

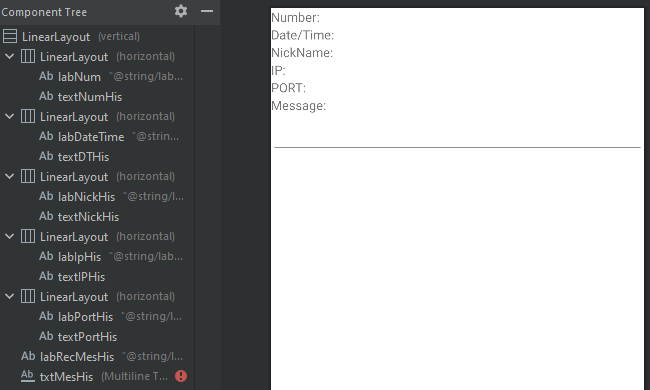
package com.example.lab16\_lukyanov;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.widget.EditText;  
import android.widget.TextView;  
  
public class MessageActivity extends AppCompatActivity {  
  
 TextView txt\_Number;  
 TextView txt\_Nick;  
 TextView txt\_Port;  
 TextView txt\_DateTime;  
 TextView txt\_IP;  
 EditText txt\_Message;  
  
 String nick, dateTime, message, ip;  
 int number, port;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_message*);  
  
 txt\_Number = findViewById(R.id.*textNumHis*);  
 txt\_Nick = findViewById(R.id.*textNickHis*);  
 txt\_Port = findViewById(R.id.*textPortHis*);  
 txt\_DateTime = findViewById(R.id.*textDTHis*);  
 txt\_IP = findViewById(R.id.*textIPHis*);  
 txt\_Message = findViewById(R.id.*txtMesHis*);  
  
 Intent i = getIntent();  
 number = i.getIntExtra("mes-num", 0);  
 nick = i.getStringExtra("mes-nick");  
 dateTime = i.getStringExtra("mes-dateTime");  
 message = i.getStringExtra("mes-text");  
 ip = i.getStringExtra("mes-ip");  
 port = i.getIntExtra("mes-port", 0);  
  
 txt\_Number.setText(String.*valueOf*(number));  
 txt\_Port.setText(String.*valueOf*(port));  
 txt\_Nick.setText(nick);  
 txt\_DateTime.setText(dateTime);  
 txt\_IP.setText(ip);  
 txt\_Message.setText(message);  
 }  
}

Формы приложения в окне редактора

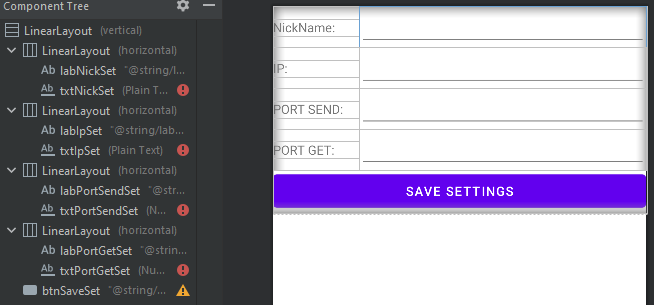
Основная форма:



Дополнительная форма:



Диалог с настройками:



Меню приложения:

