

Комитет по образованию Правительства Санкт-Петербурга  
**САНКТ-ПЕТЕРБУРГСКИЙ 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.

Settings	
nick	TEXT
ip	TEXT
portget	INT
portsend	INT

History	
number	INT
datetime	TEXT
nick	TEXT
ip	TEXT
port	INT
message	TEXT

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.

<i>Settings</i>		<i>application settings</i>						
#	name	type	size	default	primary	foreign	unique	description
1	nick	TEXT	-		no	-	no	nick for send
2	ip	TEXT	-		no	-	no	destination IP-address
3	portget	INT	-		no	-	no	port to take messages
4	portsend	INT	-		no	-	no	port to send messages

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.

<i>History</i>		<i>messages history</i>						
#	name	type	size	default	primary	foreign	unique	description
1	number	INT	-		no	-	yes	number of message
2	datetime	TEXT	-		no	-	no	message date and time
3	nick	TEXT	-		no	-	no	sender nick
4	ip	TEXT	-		no	-	no	message IP-address
5	port	INT	-		no		no	message port
6	message	TEXT	-		no		no	message text

Рисунок 3 – Описание столбцов таблицы location

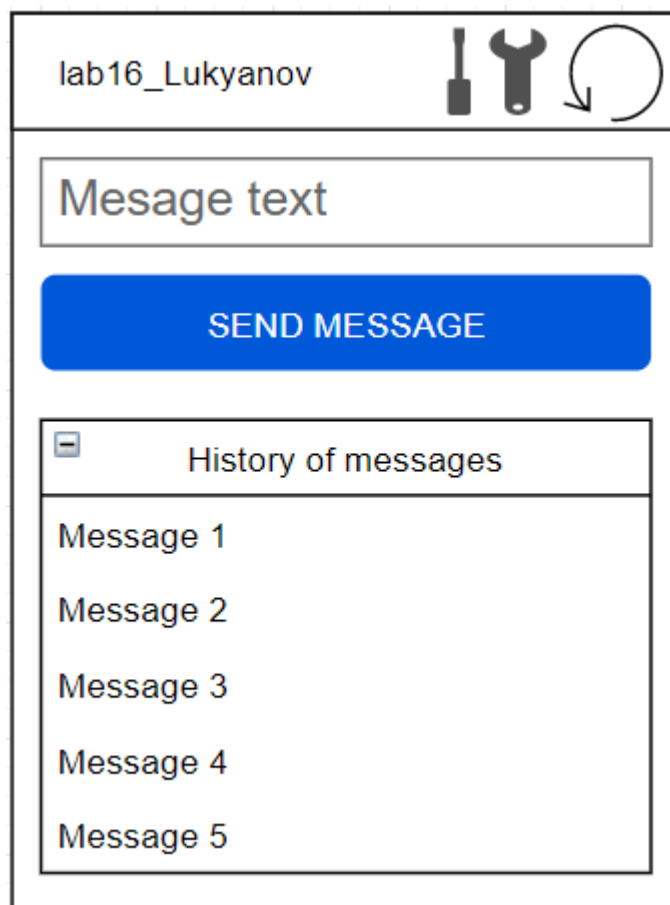
## Интерфейс приложения

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.



The mockup shows a mobile application interface. At the top is a header bar with the text 'lab16\_Lukyanov' on the left and three icons (a hammer, a wrench, and a circular arrow) on the right. Below the header is a text input field labeled 'Message text'. Underneath the input field is a blue button with the text 'SEND MESSAGE'. Below the button is a list box titled 'History of messages' with a minus icon on the left. The list contains five items: 'Message 1', 'Message 2', 'Message 3', 'Message 4', and 'Message 5'.

Рисунок 4 – Макет формы Main Menu

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

The screenshot shows a mobile application interface with a purple header bar containing the text 'lab16\_Lukyanov', a wrench icon, and a close icon. Below the header is a text input field labeled 'Message to send' containing the text 'text123'. A red underline is visible under the input field. Below the input field is a purple button labeled 'SEND MESSAGE'. Underneath the button is a list of four messages, each on a separate line with a horizontal separator:

- 1 | lukyanov | Se: text
- 2 | lukyanov | Se: text1
- 3 | lukyanov | Se: text12
- 4 | lukyanov | Se: text123

Рисунок 5 – Форма Main Menu в приложении

### Форма Settings

На рисунке 6 показан макет внешнего вида формы настроек.

The mockup shows a settings form with the following elements:

- NickName:** Input field containing 'Nick'.
- IP:** Input field containing 'IP-adress'.
- PORT SEND:** Input field containing 'number of port to send'.
- PORT GET:** Input field containing 'number of port to ger'.
- Submit:** A blue button at the bottom.

Рисунок 6 – Макет формы Settings

На рисунке 7 показан внешний вид формы настроек в приложении.

NickName:	lukyanov
IP:	10.0.2.17
PORT SEND:	5011
PORT GET:	5010

SAVE SETTINGS

Рисунок 7 – Форма Settings в приложении

## Форма Message

На рисунке 8 показан макет внешнего вида формы сообщения.

Number:	number of message
Date/Time:	date and time of message
NickName:	nick of sender
IP:	message IP-address
PORT:	message port
Message:	<div>Message text</div>

Рисунок 8 – Макет формы Message

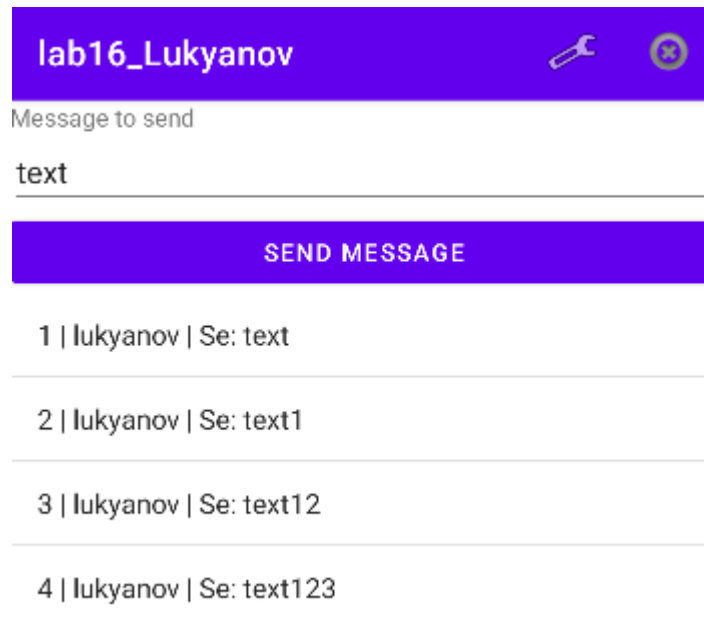
На рисунке 9 показан внешний вид формы сообщения в приложении.

Number:	4
Date/Time:	2022/09/06 19:09:44
NickName:	lukyanov
IP:	10.0.2.17
PORT:	5011
Message:	
Se:	text123

Рисунок 9 – Форма Message в приложении

## Демонстрация работы приложения

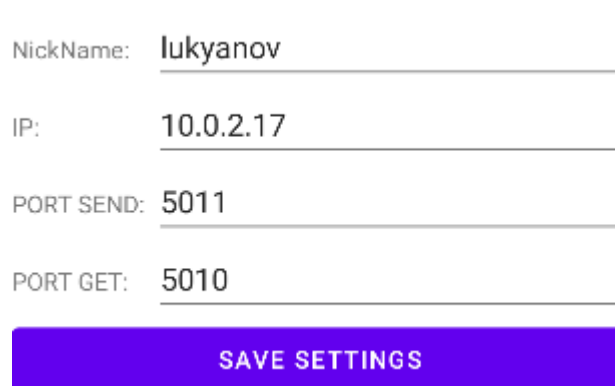
При запуске приложения отображается история сообщений, которая берётся из базы данных (рис. 10):



The screenshot shows a mobile application interface. At the top, there is a blue header bar with the text "lab16\_Lukyanov" on the left, a wrench icon in the center, and a circular icon with an 'x' on the right. Below the header, there is a text input field labeled "Message to send" containing the word "text". Below the input field is a blue button labeled "SEND MESSAGE". Below the button is a list of four messages, each on a new line and separated by a horizontal line. The messages are: "1 | lukyanov | Se: text", "2 | lukyanov | Se: text1", "3 | lukyanov | Se: text12", and "4 | lukyanov | Se: text123".

Рисунок 10 – Запуск приложения

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



The screenshot shows a settings dialog box. It has four rows, each with a label on the left and a text input field on the right. The labels are "NickName:", "IP:", "PORT SEND:", and "PORT GET:". The input fields contain the values "lukyanov", "10.0.2.17", "5011", and "5010" respectively. Below the input fields is a blue button labeled "SAVE SETTINGS".

Рисунок 11 – Диалог с настройками

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

Number:	4
Date/Time:	2022/09/06 19:09:44
NickName:	lukyanov
IP:	10.0.2.17
PORT:	5011
Message:	
Se:	text123

Рисунок 12 – Информация о сообщении

Изменим настройки и текст сообщения (рис. 13):

Message to send

lab16

SEND MESSAGE

1 | lukyanov | Se: text

2 NickName: lukyanov\_493

3 IP: 10.0.2.17

4 PORT SEND: 5015

PORT GET: 5010

SAVE SETTINGS

Рисунок 13 – Изменённые настройки и текст сообщения

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

lab16\_Lukyanov

Message to send

lab16

SEND MESSAGE

1 | lukyanov | Se: text

2 | lukyanov | Se: text1

3 | lukyanov | Se: text12

4 | lukyanov | Se: text123

5 | lukyanov\_493 | Se: lab16

Рисунок 14 – История с отправленным сообщением



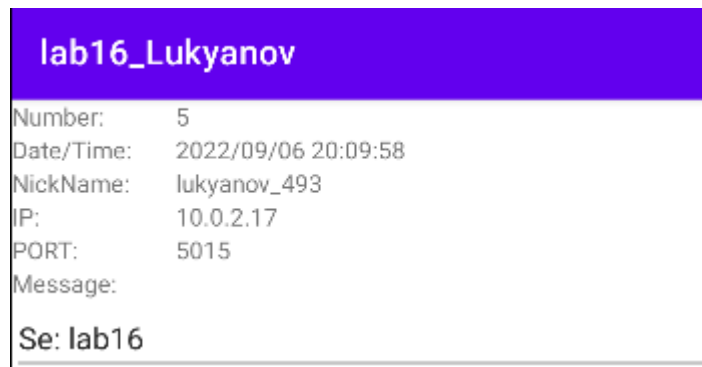


Рисунок 15 – Отправленное сообщение

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

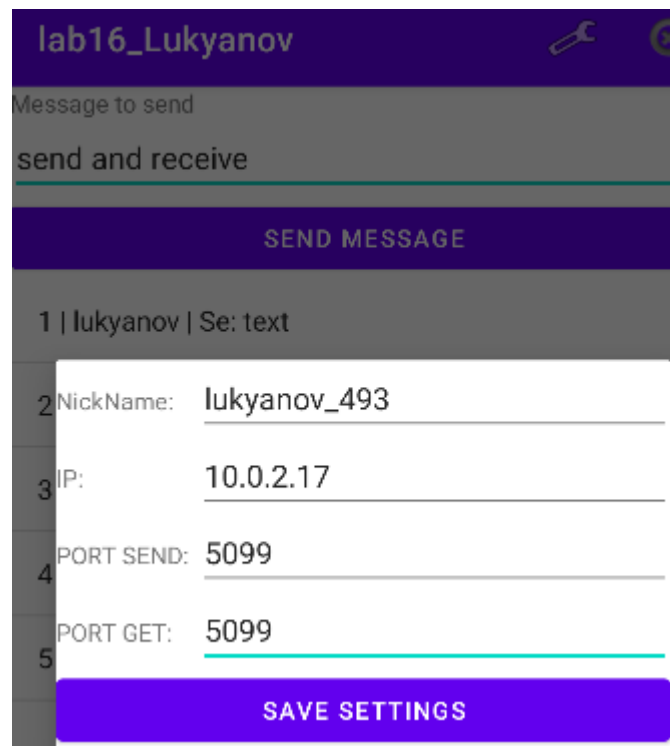


Рисунок 16 – Настройки для получения отправленного сообщения

История сообщения после отправки представлена на рисунке 17.

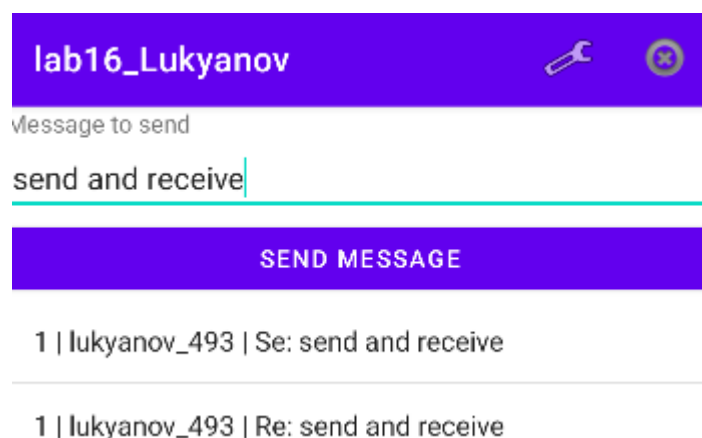


Рисунок 17 – История с отправленным и полученным сообщением

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

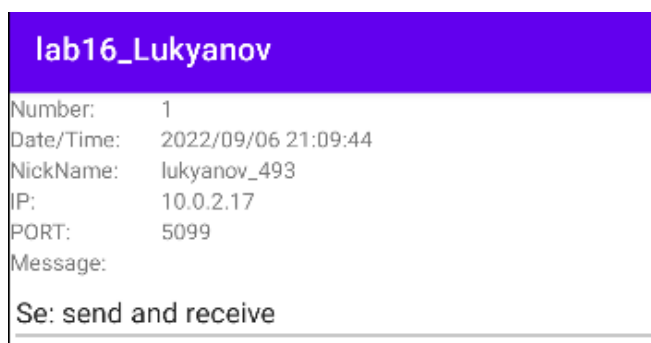


Рисунок 18 – Отправленное сообщение

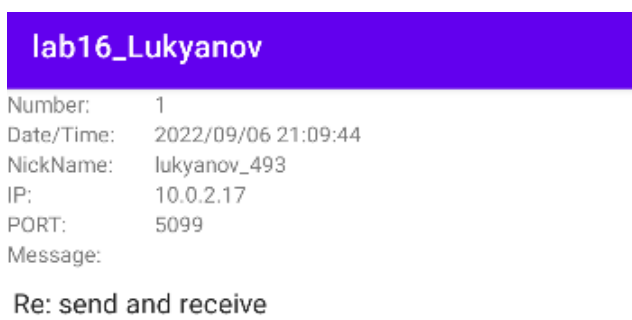


Рисунок 19 – Полученное сообщение

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

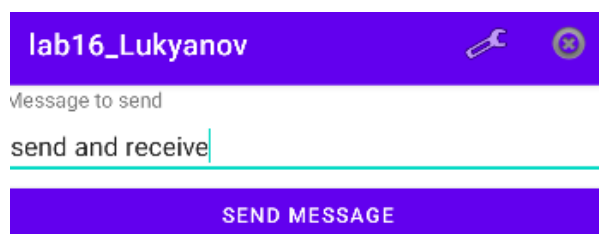


Рисунок 20 – Очищенная история

Также приложение имеет собственную иконку (рис. 21):

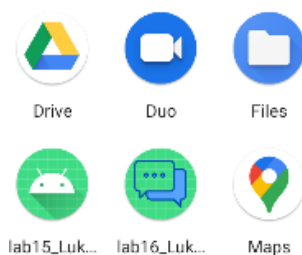


Рисунок 21 – Иконка приложения

## Описание участков кода

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

```
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, selectionArgs: null);
        if (cur.moveToFirst())
        {
            String[] settings = new String[4];
            settings[0] = cur.getString(columnIndex: 0);
            settings[1] = cur.getString(columnIndex: 1);
            settings[2] = String.valueOf(cur.getInt(columnIndex: 2));
            settings[3] = String.valueOf(cur.getInt(columnIndex: 3));
            return settings;
        }
    }
}
```

Экземпляр класса базы:

```
package com.example.lab16_lukyanov;

public final class g {
    static DB chat;
}
```

Работа с историей сообщений:

```

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 void getAllHistory(ArrayList<Message> lst)
{
    SQLiteDatabase db = getReadableDatabase();
    String sql = "SELECT * FROM History";
    Cursor cur = db.rawQuery(sql, selectionArgs: null);
    if (cur.moveToFirst())
    {
        do {
            Message mes = new Message();
            mes.number = cur.getInt( columnIndex: 0);
            mes.dateTime = cur.getString( columnIndex: 1);
            mes.nick = cur.getString( columnIndex: 2);
            mes.ip = cur.getString( columnIndex: 3);
            mes.portGet = cur.getInt( columnIndex: 4);
            mes.textMes = cur.getString( columnIndex: 5);
            lst.add(mes);
        } while (cur.moveToNext());
    }
}

public void deleteHistory()
{
    SQLiteDatabase db = getWritableDatabase();
    String sql = "DELETE FROM History";
    db.execSQL(sql);
}

```

Класс сообщения:

```

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;}
}

```

Переменные основной формы:

```

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;

```

Создание базы данных и добавление списка с историей сообщений:

```

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

    g.chat = new DB( context: this, name: "chat.db", factory: null, version: 1);

    lstHistory = findViewById(R.id.lstHistoryMes);
    adp = new ArrayAdapter<Message>( context: this, android.R.layout.simple_list_item_1, lst);
    lstHistory.setAdapter(adp);

```

Метод для обновления списка базы данных:

```

void update_list()
{
    lst.clear();
    g.chat.getAllHistory(lst);
    adp.notifyDataSetChanged();
}

```

Событие клика по элементу списка для просмотра на другой форме:

```
lstHistory.setOnItemClickListener((parent, view, position, id) -> {
    Message mes = adp.getItem(position);
    Intent i = new Intent( packageContext: this, MessageActivity.class);
    i.putExtra( name: "mes-num", mes.number);
    i.putExtra( name: "mes-dateTime", mes.dateTime);
    i.putExtra( name: "mes-ip", mes.ip);
    i.putExtra( name: "mes-nick", mes.nick);
    i.putExtra( name: "mes-port", mes.portGet);
    i.putExtra( name: "mes-text", mes.textMes);
    startActivityForResult(i, requestCode: 1);
});
```

Настройки диалога и событие кнопки для сохранения настроек:

```
//dialog settings
LayoutInflater dialogLayout = LayoutInflater.from(this);
View dialogView = dialogLayout.inflate(R.layout.dialog_settings, root: null);
AlertDialog = new AlertDialog.Builder( context: 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; //stop cycle
        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( bindaddr: null);
                first = true;
                socket.bind(local_address);
            } catch (UnknownHostException | SocketException e) {
                e.printStackTrace();
            }
        }
        run = true; //start cycle
    }
});
```

Загрузка настроек из базы данных, если они имеются:

```
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));
}
```

Создание сокета с указанным в настройках портом:

```
try {
    local_network = InetAddress.getByName("0.0.0.0");
    local_address = new InetSocketAddress(local_network, portGetSet);
    socket = new DatagramSocket(bindaddr: null);
    socket.bind(local_address);
}
catch (UnknownHostException | SocketException e) {
    e.printStackTrace();
}
```

Поток для получения сообщений и сохранения в базу:

```

Runnable receiver = new Runnable() {
    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    public void run() {
        Log.e( tag: "TEST", msg: "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(), offset: 0, received_packet.getLength());
            Log.e( tag: "TEST", msg: "RECEIVED: " + s);
            if (s.indexOf(":") != -1) {
                String[] mes = s.split( regex: ":" );
                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.sendMessage( number: g.chat.getMaxId() + 1, String.valueOf(dtf.format(now)), nickGet,
                        String.valueOf(received_packet.getAddress().split( regex: "/" )[1], received_packet.getPort(), receiveMes);
                else first = false;
                runOnUiThread() ->
                {
                    update_list();
                });
            }
        }
    }
};

Thread receiving_thread = new Thread(receiver);
receiving_thread.start();

```

Событие клика для отправки сообщения:



```

@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.getByIp(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( tag: "test", msg: "sending thread is running");
                socket.send(send_packet);
            } catch (IOException e)
            {
                e.printStackTrace();
            }
        }
    };

    Thread sending_thread = new Thread(r);
    sending_thread.start();
}

```

Сохранение отправленного сообщения в базу:

```

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
LocalDateTime now = LocalDateTime.now();
Log.e( tag: "sendip: " + send_packet.getAddress(), msg: "sendport: " + send_packet.getPort());
Log.e( tag: "socketport: " + socket.getLocalPort(), msg: "socketip: " + socket.getLocalAddress().toString());
g.chat.addMessage( number: g.chat.getMaxId()+1, String.valueOf(dtf.format(now)), nickSet, ipSet, portSendSet, message: "Se: " + message);
update_list();
}

```

Создание меню:

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu1, menu);
    return true;
}

```

Событие для клика по элементам меню:

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {

    int id = item.getItemId();

    switch (id)
    {
        case R.id.itm_clear: //clear history
        {
            g.chat.deleteHistory();
            update_list();
            break;
        }
        case R.id.itm_settings: //open settings dialog
        {
            alertDialog.show();
            break;
        }
    }

    return super.onOptionsItemSelected(item);
}
```

Переменные формы для отображения информации о сообщении:

```
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;
```

Получение отправленной на форму информации о выбранном сообщении:

```

@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( name: "mes-num", defaultValue: 0);
    nick = i.getStringExtra( name: "mes-nick");
    dateTime = i.getStringExtra( name: "mes-dateTime");
    message = i.getStringExtra( name: "mes-text");
    ip = i.getStringExtra( name: "mes-ip");
    port = i.getIntExtra( name: "mes-port", defaultValue: 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);
}

```

## 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.AdapterView;
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.getByIp(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.sendMessage(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);

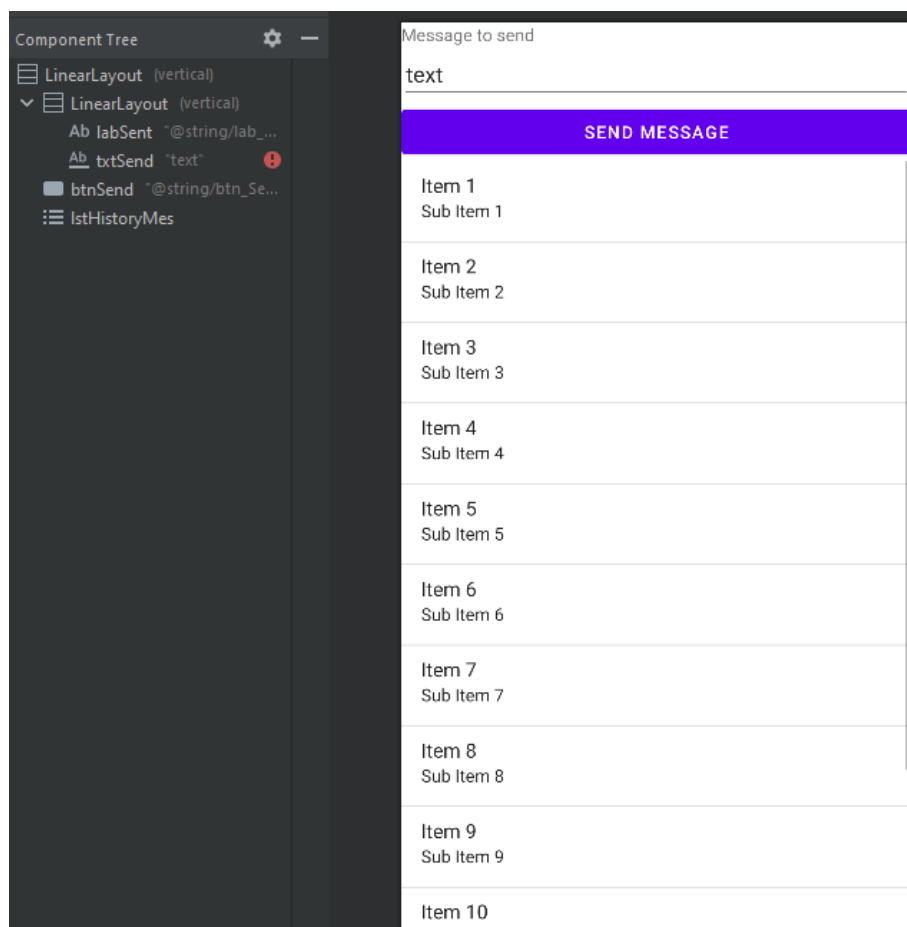
        Intent i = getIntent();
        number = i.getIntExtra("mes-num", 0);
        nick = i.getStringExtra("mes-nick");
        dateTime = i.getStringExtra("mes-dateTime");
        message = i.getStringExtra("mes-text");
        ip = i.getStringExtra("mes-ip");
        port = i.getIntExtra("mes-port", 0);

        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);
    }
}

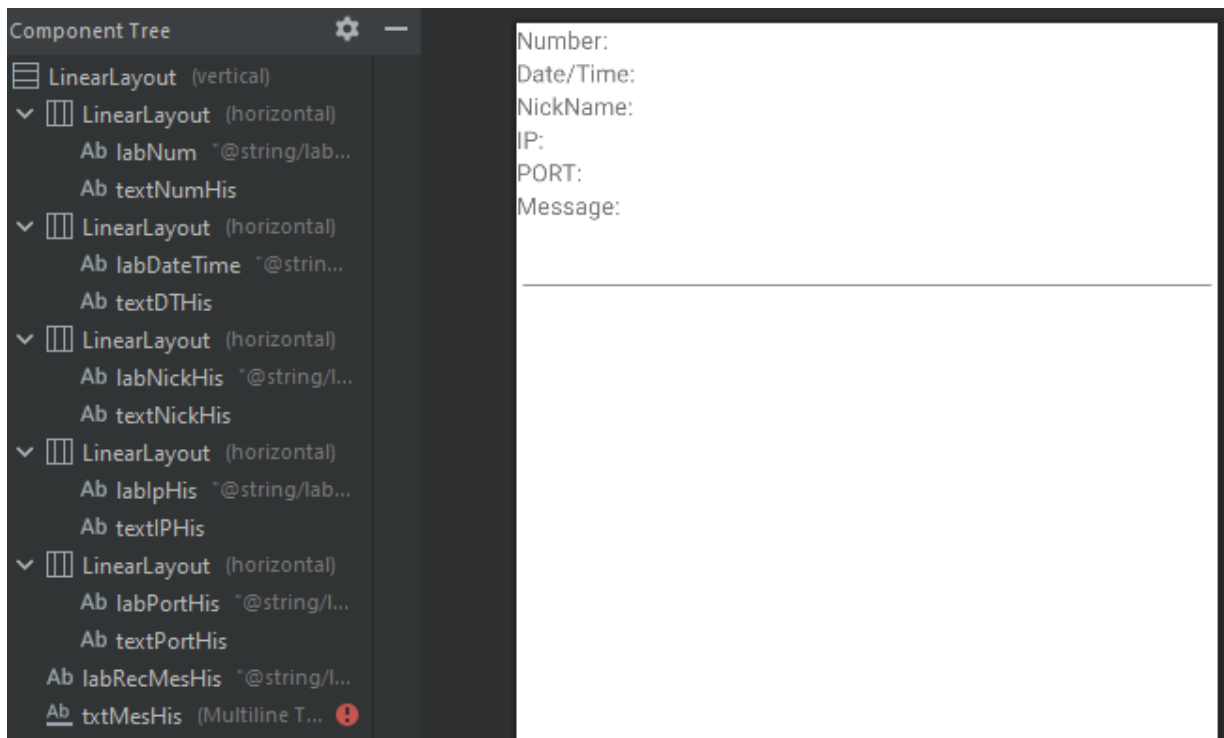
```

## Формы приложения в окне редактора

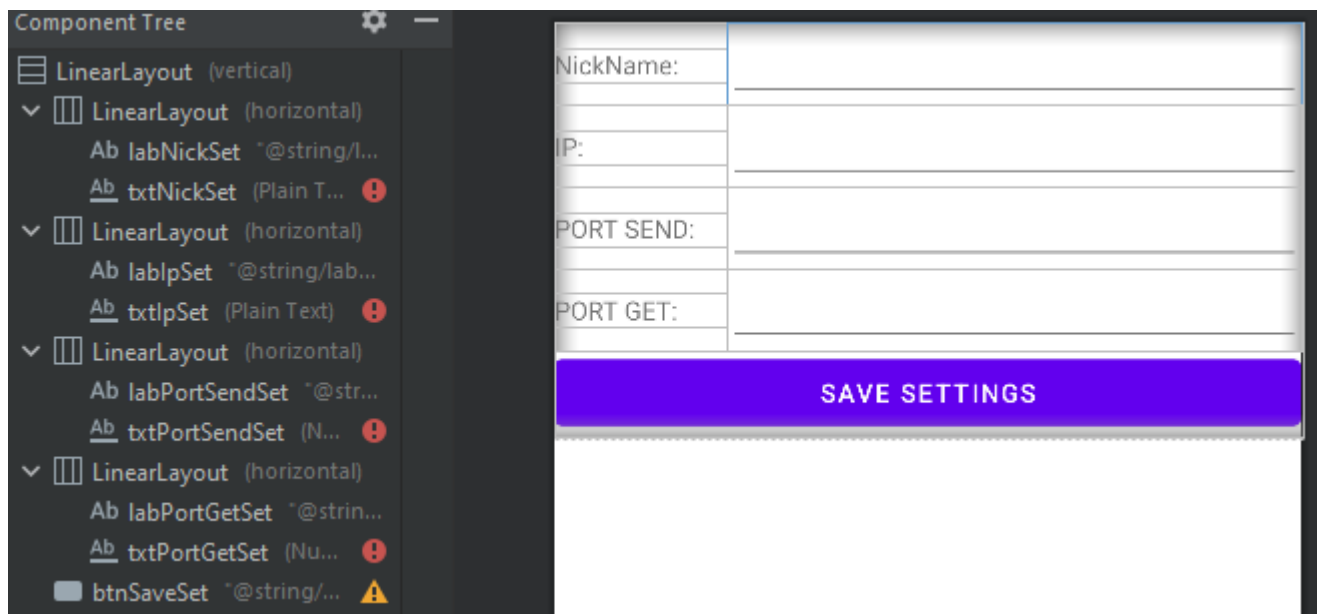
Основная форма:



Дополнительная форма:



Диалог с настройками:



Меню приложения:

