УДК 004

# Создание сервера для десктопного сетевого чата

*Семченко Регина Викторовна*
*Приамурский государственный университет имени Шолом-Алейхема*
*Студент*

*Еровлев Павел Андреевич*
*Приамурский государственный университет имени Шолом-Алейхема*
*Студент*

**Аннотация**
В данной статье рассмотрена возможность создания серверной части для десктопного сетевого чата. Приложение будет написано на чистом языке Java с использованием базы данных, где будут храниться логины и пароли пользователей. Практическим результатом является серверная часть, которая сможет передавать данные от сервера клиенту и обратно
**Ключевые слова:** Чат, приложение, андроид

# Creating a server for desktop network chat

*Semchenko Regina Viktorovna*
*Sholom-Aleichem Priamursky State University*
*Student*

*Erovlev Pavel Andreevich*
*Sholom-Aleichem Priamursky State University*
*Student*

**Abstract**
This article discusses the possibility of creating a server side for a desktop network chat. The application will be written in pure Java language using a database where usernames and passwords will be stored. The practical result is a server side that can transfer data from server to client and vice versa.
**Keywords:** Chat, app, android

Общение в сети становится все более популярным, люди реже выбираются на встречи, ведь куда проще обсудить всю работу сидя дома перед компьютером.

Цель статьи состоит в создании серверной части для возможности передачи данных между сервером и клиентом.

Исследованиями в области разработки мобильных приложений занимались многие российские и зарубежные исследователи. А.С.Винокуров, Р.И. Баженов [1] рассмотрели разработку приложений для мобильных

устройств. D.Y. Bichkovski, F.N.Abu-Abed, A.R. Khabarov, K.A.Karelskaya [2] исследовали возможность информирования студентов с помощью андроид приложения. К.В. Аксенов [3] рассмотрел современные средства разработки мобильных приложений. В.Ю. Ким [4] изучал особенности дизайна интерфейса пользователя для приложений. Романов и др. [5] описали разработку мобильного приложения для управления документами из облачных хранилищ. E.W.T. Ngaia, A.Gunasekaran [6] рассмотрели методы разработки мобильных бизнес приложений.

Приложение будет написано на языке Java, который легко справляется с написанием приложений, как на компьютеры, так и на мобильные устройства.

Для начала создадим проект в IntelliJIdea, и добавим новый класс с названием «Server», в нем будут описаны классы авторизации клиентов, открытие сервера под определенным портом, прослушивание портов клиента и отключение от прослушки, а так же отправление сообщений от сервера. Были добавлены исключения, позволяющие переопределить ошибки, для того, чтобы сервер не «рухнул», а просто выводилась ошибка и сервер был в рабочем состоянии (рис.1-3).

```
9   public class Server {
10      private Vector<ClientHandler> clientList;
11      private AuthService authService;
12
13      public AuthService getAuthService() { return authService; }
16
17      public Server(int port) {
18          try (ServerSocket serverSocket = new ServerSocket(port)) {
19              clientList = new Vector<>();
20
21              authService = new AuthService();
22              authService.connect();
23
24              System.out.println("Server started...Waiting for clients");
25              while (true) {
26                  Socket socket = serverSocket.accept();
27                  System.out.println("Client connected" +
28                          socket.getInetAddress() + ":" +
29                          socket.getPort() + ":" +
30                          socket.getLocalPort());
31                  new ClientHandler( server: this, socket);
32              }
33          } catch (IOException e) {
34              System.err.println("Запрашиваемый порт занят");
35          } catch (ClassNotFoundException | SQLException e) {
36              System.out.println("Не удалось запустить сервис авторизации");
```

Рисунок 1 – класс «Server»

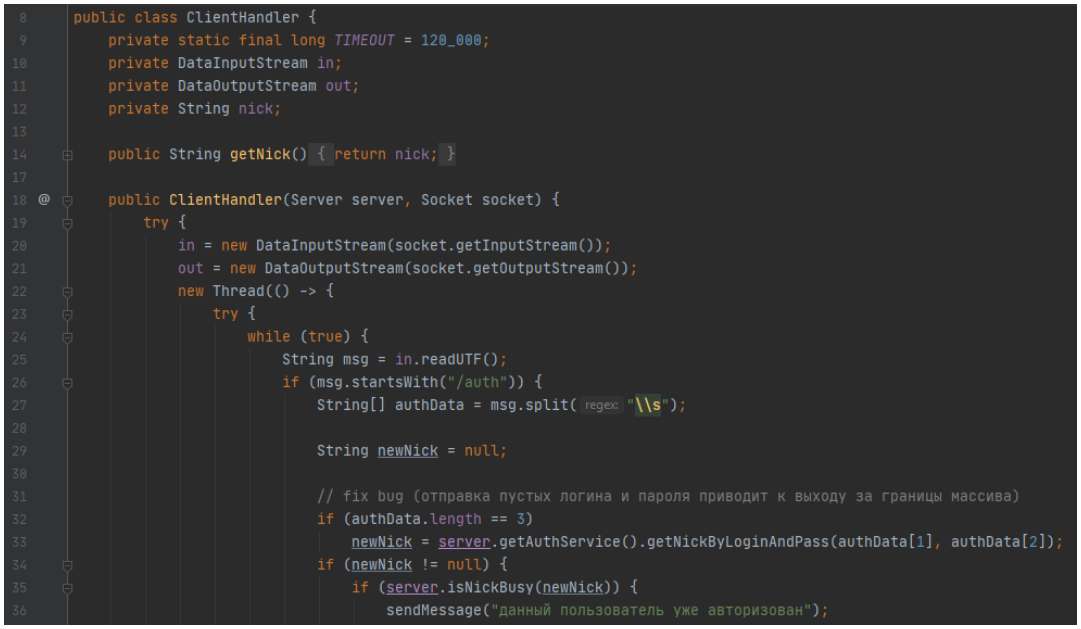
```
38                  assert authService != null;
39                  authService.disconnect();
40              }
41          }
42
43          public void subscribe(ClientHandler clientHandler) {
44              clientList.add(clientHandler);
45              broadcastClientListSender();
46          }
47
48          public void unsubscribe(ClientHandler clientHandler) {
49              clientList.remove(clientHandler);
50              broadcastClientListSender();
51          }
52
53          public boolean isNickBusy(String nick) {
54              for (ClientHandler ch : clientList)
55                  if (ch.getNick().equals(nick)) return true;
56
57              return false;
58          }
59
60          public void broadcastSender(String message) {
61              for (ClientHandler ch :
62                      clientList) {
63                  ch.sendMessage(message);
64              }
65          }
```

Рисунок 2 – класс «Server»

```
67      public void privateSender(ClientHandler sender, String destNnick, String message) {
68          for (ClientHandler ch :
69                  clientList) {
70              if (destNnick.equals(ch.getNick())) {
71                  ch.sendMessage(sender.getNick() + " private: " + message);
72                  sender.sendMessage("private to " + destNnick + ": " + message);
73                  return;
74              }
75          }
76          sender.sendMessage("клиент с ником " + destNnick + " не найден");
77      }
78
79      private void broadcastClientListSender() {
80          StringBuilder sb = new StringBuilder("/clients ");
81          for (ClientHandler ch :
82                  clientList) {
83              sb.append(ch.getNick()).append(" ");
84          }
85          String clientsListMsg = sb.toString();
86          for (ClientHandler ch :
87                  clientList) {
88              ch.sendMessage(clientsListMsg);
89          }
90      }
91  }
```

Рисунок 3 – класс «Server»

Следующим шагом добавляем класс «ClientHandler» в нем будут описаны методы отключения от сервера после простоя в 2 минуты, а так же функция авторизации, которая будет считывать введенные данные пользователя и сверять их с данными в базе данных и если есть ошибка, то выдавать окно с ошибкой о неправильно введенных данных. Так же если данный пользователь уже авторизован, то выдавать ошибку авторизации. Добавлены специальные символы, при использовании которых происходят определенные действия, если клиент ввел «/end», то сервер отключает прослушивание данного клиента, если «/w» то данный спец символ позволяет отправить личное сообщение. В конце добавлена функция отправки сообщений в общий чат (рис.4-7).



```
public class ClientHandler {
    private static final long TIMEOUT = 120_000;
    private DataInputStream in;
    private DataOutputStream out;
    private String nick;

    public String getNick() { return nick; }

    public ClientHandler(Server server, Socket socket) {
        try {
            in = new DataInputStream(socket.getInputStream());
            out = new DataOutputStream(socket.getOutputStream());
            new Thread(() -> {
                try {
                    while (true) {
                        String msg = in.readUTF();
                        if (msg.startsWith("/auth")) {
                            String[] authData = msg.split( regex: "\\s");

                            String newNick = null;

                            // fix bug (отправка пустых логина и пароля приводит к выходу за границы массива)
                            if (authData.length == 3)
                                newNick = server.getAuthService().getNickByLoginAndPass(authData[1], authData[2]);
                            if (newNick != null) {
                                if (server.isNickBusy(newNick)) {
                                    sendMessage("данный пользователь уже авторизован");
```

Рисунок 4 – класс «ClientHandler»

```
                                    continue;
                                }
                                nick = newNick;
                                sendMessage("/authok " + nick);
                                server.subscribe( clientHandler: this);
                                break;
                            } else {
                                sendMessage("Неверный логин/пароль");
                            }
                        }
                    }
                    while (true) {
                        String msg = in.readUTF();
                        System.out.println(nick + ": " + msg);

                        if (msg.startsWith("/")) {
                            if (msg.equals("/end")) break;
                            if (msg.startsWith("/w")) {
                                String[] data = msg.split( regex: "\\s", limit: 3);
                                server.privateSender( sender: this, data[1], data[2]);
                            }
                            continue;
                        }
                        server.broadcastSender(nick + ": " + msg);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
```

Рисунок 5 – класс «ClientHandler»

```
            } finally {
                server.unsubscribe( clientHandler: this);
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
        new Thread(() -> {
            try {
                Thread.sleep(TIMEOUT);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            if(nick == null){
                try {
                    sendMessage("/timeout");
                    socket.close();
                    in.close();
                    out.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                System.out.println("timeout");
            }
        }).start();
```

Рисунок 6 – класс «ClientHandler»

```
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public void sendMessage(String message) {
            try {
                out.writeUTF(message);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

Рисунок 7 – класс «ClientHandler»

Далее создаем класс «AuthServer», в нем напишем функции подключения к базе данных, добавим функцию получения никнейма по логину и паролю и так же функцию отключения от сервера (рис.8-9).

```java
public class AuthService {
    private Connection connection;
    private Statement statement;

    public void connect() throws ClassNotFoundException, SQLException{
        Class.forName("org.sqlite.JDBC");
        // как подключить базу, расположение которой отличается от корня проекта
        connection = DriverManager.getConnection( url: "jdbc:sqlite:JavaFX_chat.db");
        statement = connection.createStatement();
    }

    public String getNickByLoginAndPass(String login, String pass){
        try {
            ResultSet rs = statement.executeQuery( sql: "SELECT nick FROM users where login='"+ login +
                    "' and password='"+ pass +"';");
            while (rs.next()){
                return rs.getString( columnLabel: "nick");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }

    public void disconnect(){
        try {
            connection.close();
```

Рисунок 8 – класс «AuthServer»

```java
    public void disconnect(){
        try {
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Рисунок 9 – класс «AuthServer»

На этом создание сервера закончено и осталось лишь написать класс, который будет включать все эти функции (рис.10) и протестировать сервер (рис.11).

```java
package ru.geekbrains.java2.lesson_07.javaFXChat.server;

public class ServerMain {
    public static void main(String[] args) { new Server( port: 8111); }
}
```

Рисунок 10 – Создание класса Main

```
ServerMain
C:\Users\79644\.jdks\corretto-1.8.0_252\bin\java.exe ...
Server started...Waiting for clients
```

Рисунок 11 – Работа сервера

После запуска сервера нам выдается сообщение, о том, что сервер запущен и ожидает подключения клиентов. После подключения клиентов будут, выводятся данные об их количество и о том кто отключился и подключился.

В данной статье была рассмотрена реализация сервера для сетевого чата на чистом языке Java. Готовым результатом является рабочий сервер для подключения клиентов и передачи между ними сообщений.

**Библиографический список**

1. Винокуров А.С., Баженов Р.И. Разработка мобильного приложения информационного сайта для абитуриентов и первокурсников университета // Современные научные исследования и инновации. 2015. № 7-2 (51). С. 54-62
2. Бычковский Д.Ю., Абу-Абед Ф.Н., Хабаров А.Р., Карельская К.А. Разработка мобильного приложения онлайн-радио // Программные продукты и системы. 2016. №2 (114). С. 185-194
3. Аксенов К.В. Обзор современных средств для разработки мобильных приложений // Новые информационные технологии в автоматизированных системах. 2014. №17. С. 508-513
4. Ким В.Ю. Особенности разработки дизайна пользовательского интерфейса для мобильного приложения // Новые информационные технологии в автоматизированных системах. 2015. №18. С. 479-481
5. Романов А.А., Панченко Е.А., Винокуров И.В. Разработка мобильного приложения для управления документами из облачных хранилищ // Символ науки. 2016. №3. С. 84-87
6. Ngaia E.W.T., Gunasekaran A. A review for mobile commerce research and applications // Decision Support Systems. 2007. №43 (1). С. 3–15.