УДК 004

Создание клиента 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 клиентской части для десктопного сетевого чата. Приложение будет написано на чистом языке Java. Практическим результатом является клиентская часть, которая сможет передавать данные от сервера клиенту и обратно.
Ключевые слова: Чат, приложение, андроид

Creating a client 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 client side for a desktop network chat. The application will be written in pure Java language. The practical result is a client side that can transfer data from server to client and vice versa.
Keywords: Chat, app, android

Сейчас все большую популярность набирают мессенджеры и чаты для общения в сети, появляются большое количество разнообразных чатов в сети. Так же для возможности использования на работе чата между всей структурой организации имеется возможность использовать сетевые чаты, которые работают только в определенной сети. Это гораздо безопаснее, ведь данные не передаются дальше внутренней сети организации.

Цель исследования состоит в создании клиентской 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, К.А.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. А.А.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.

Клиентская часть будет написана на языке Java версии 1.8, ведь до этой версии были включена библиотека JavaFX и нет необходимости выставлять зависимость данной библиотеки как в случае с более новыми версиями Java.

После создания проекта, создадим новый класс с именем «Controller», в нем будет содержаться все, от прорисовки окон и значков, до всех возможных функция.

Для начала импортируем все библиотеки, которые необходимы будут в данном проекте (рис.1)

```
3   import javafx.application.Platform;
4   import javafx.collections.FXCollections;
5   import javafx.collections.ObservableList;
6   import javafx.fxml.FXML;
7   import javafx.fxml.Initializable;
8   import javafx.scene.control.*;
9   import javafx.scene.input.MouseEvent;
10  import javafx.scene.layout.HBox;
11  import javafx.util.Callback;
12
13  import java.io.DataInputStream;
14  import java.io.DataOutputStream;
15  import java.io.IOException;
16  import java.net.Socket;
17  import java.net.URL;
18  import java.util.ResourceBundle;
```

Рисунок 1 – Импорт библиотек

Далее создадим все переменные, которые будут использоваться в зоде написания проекта, зададим нужный нам порт, по которому клиент будет подключаться к серверу (рис.2).

```
public class Controller implements Initializable {
    @FXML
    TextArea textArea;
    @FXML
    TextField msgField;
    @FXML
    HBox loginPanel;
    @FXML
    HBox messagePanel;
    @FXML
    TextField loginField;
    @FXML
    PasswordField passFiead;
    @FXML
    ListView clientsListArea;
    private ObservableList<String> clientsObsvList;

    private Socket socket;
    private DataOutputStream out;
    private DataInputStream in;
    private String myNick;

    final String SERVER_IP = "localhost";
    final int SERVER_PORT = 8111;

```

Рисунок 2 – добавление переменных

Следующий шаг, напишем функцию, которая будет смотреть, если пользователь не авторизован, то ему доступны для видимости только 2 окна ввода логина и пароля, а если авторизовался, то эти два поля становятся для него невидимые и начинают отображаться другие два поля: поле списка клиентов и поле сообщений (рис.3)

```
public void setAutorized(boolean autorized) {
    if (autorized) {
        loginPanel.setVisible(false);
        loginPanel.setManaged(false);
        messagePanel.setVisible(true);
        messagePanel.setManaged(true);
    } else {
        loginPanel.setVisible(true);
        loginPanel.setManaged(true);
        messagePanel.setVisible(false);
        messagePanel.setManaged(false);
        myNick = "";
    }
}
```

Рисунок 3 – Функция авторизации

Далее добавляем функцию инициализации, то есть, если мы авторизовались, то функция подгружает видимые части и их стили (рис.4).

```
60      @Override
61      public void initialize(URL location, ResourceBundle resources) {
62          setAutorized(false);
63          clientsObsvList = FXCollections.observableArrayList();
64          clientsListArea.setItems(clientsObsvList);
65          clientsListArea.setCellFactory(new Callback<ListView<String>, ListCell<String>>() {
66              @Override
67              public ListCell<String> call(ListView<String> param) {
68                  return updateItem(item, empty) -> {
71                          super.updateItem(item, empty);
72                          if (!empty) {
73                              setText(item);
74                              if (item.equals(myNick)) {
75                                  setStyle("-fx-font-weight: bold;" +
76                                          " -fx-background-color: #ffead4");
77                              }
78                          } else {
79                              setGraphic(null);
80                              setText(null);
81                          }
82                  };
84              }
85          });
86      }
87
```

Рисунок 4 – Класс инициализации

Далее идет большой класс подключения к серверу, подключаясь по нужному порту, который прослушивает в свое время сервер, клиент отдает ответ серверу о подключении, далее открывается окно с вводом данных и если ввести правильные данные, на которые сервер ответит как «/authok», то пускает нас дальше, иначе выдает ошибку авторизации. Так же если простоять на окне авторизации более 2 минут бездействий, то сервер перестает нас слушать и придется подключаться заново. Добавим порядки для вывода сообщений на каждой новой строчке и сделаем исключение с ошибками для более стабильной работы клиента (рис.5-6)

```
88    public void connect() {
89        try {
90            socket = new Socket(SERVER_IP, SERVER_PORT);
91            in = new DataInputStream(socket.getInputStream());
92            out = new DataOutputStream(socket.getOutputStream());
93
94            Thread t = new Thread(() -> {
95                try {
96                    while (true) {
97                        String s = in.readUTF();
98                        if (s.startsWith("/")) {
99                            if (s.startsWith("/authok")) {
100                               setAutorized(true);
101                               myNick = s.split( regex: "\\s")[1];
102                               textArea.appendText("успешная авторизация\n");
103                               break;
104                           }
105                           if (s.startsWith("/timeout")) {
106                               Platform.runLater(() -> showAlert("Соединение закрыто по таймауту"));
107                           }
108                           continue;
109                       }
110                       textArea.appendText(s + "\n");
111                   }
112                   while (true) {
113                       String msg = in.readUTF();
114                       if (msg.startsWith("/clients")) {
115                           String[] clientsString = msg.substring(9).split( regex: " ");
```

Рисунок 5 – класс «Connect»

```
116                           Platform.runLater(() -> {
117                               clientsObsvList.clear();
118                               clientsObsvList.addAll(clientsString);
119                           });
120                           continue;
121                       }
122                       textArea.appendText(msg + "\n");
123                   }
124               } catch (IOException e) {
125                   showAlert("Соединение с сервером разорвано.");
126               } finally {
127                   setAutorized(false);
128                   try {
129                       socket.close();
130                   } catch (IOException e) {
131                       e.printStackTrace();
132                   }
133               }
134           });
135           t.setDaemon(true);
136           t.start();
137       } catch (IOException e) {
138           showAlert("Не удалось подключиться к серверу. Проверьте сетевое соединение.");
139       }
140   }
141
```

Рисунок 6 – класс «Connect»

Следующий класс - это авторизация, в нем будет получение введенных данных клиентом и отправка их на сервер, на сервере они проверяются и отсылается ответ об успешной, либо не успешной авторизации (рис.7).

```
142     public void authorization() {
143         if (!loginField.getText().isEmpty() && !passFiead.getText().isEmpty()) {
144             if (socket == null || socket.isClosed()) connect();
145             try {
146                 out.writeUTF( str: "/auth " +
147                         loginField.getText() + " " +
148                         passFiead.getText());
149                 loginField.clear();
150                 passFiead.clear();
151             } catch (IOException e) {
152                 e.printStackTrace();
153             }
154         } else {
155             showAlert("Неполные данные для авторизации!");
156         }
157     }
158
```

Рисунок 7 – Класс авторизации

И последние классы, это класс отправки сообщений, который вызывается при написании сообщения, он очищает строку ввода после отправки и делает на ней фокус курсора. Класс «showAlert» выводит всплывающее окно после непредвидимой ошибки, когда нет подключения к серверу, неправильных данных и т.д. И последний класс, который позволяет двойным щелчком мыши по никнейму клиента ввести в строку ввода «/w nickname …» для быстрого ввода личного сообщения, это упрощает написание таких сообщений (рис.8).

```
159     public void sendMsg() {
160         try {
161             out.writeUTF(msgField.getText());
162             msgField.clear();
163             msgField.requestFocus();
164         } catch (IOException e) {
165             e.printStackTrace();
166         }
167     }
168
169     public void showAlert(String message) {
170         Platform.runLater(() -> {
171             Alert alert = new Alert(Alert.AlertType.INFORMATION);
172             alert.setTitle("Ой! Проблемка нарисавалася!");
173             alert.setHeaderText(null);
174             alert.setContentText(message);
175             alert.showAndWait();
176         });
177     }
178
179 @   public void clientChoise(MouseEvent event) {
180         if (event.getClickCount() == 2) {
181             msgField.setText("/w " + clientsListArea.getSelectionModel().getSelectedItem() + " ");
182             msgField.requestFocus();
183             msgField.selectEnd();
184         }
185     }
186 }
```

Рисунок 8 – Помогающие классы

Теперь осталось добавить класс «Main» и прописать в нем включение класса контроллера, для запуска клиентской части (рис.9).

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource( name: "sample.fxml"));
        primaryStage.setTitle("JavaFX project Client");
        primaryStage.setScene(new Scene(root,  width: 550,  height: 800));
        primaryStage.show();
    }


    public static void main(String[] args) { launch(args); }
}
```

Рисунок 9 – Класс «Main»

Теперь при запуске сервера, а потом клиента у нас будет открываться окно авторизации (рис.10)

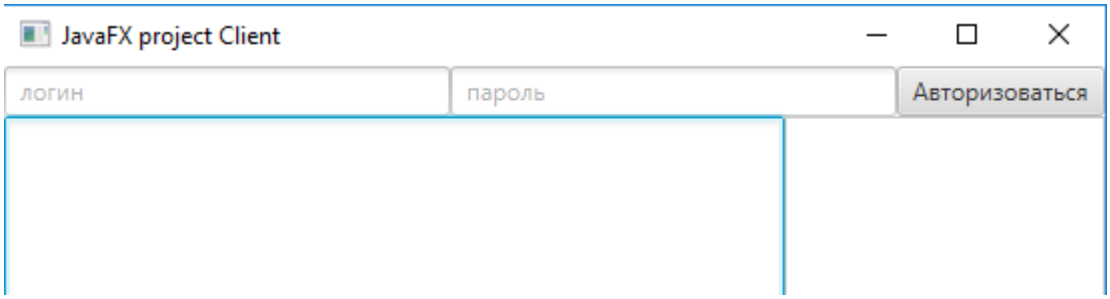

Рисунок 10 – Окно авторизации

После ввода правильных логина и пароля, открывается окно чата, где справа написаны никнеймы подключенных клиентов, а слева окно чата (рис.11)

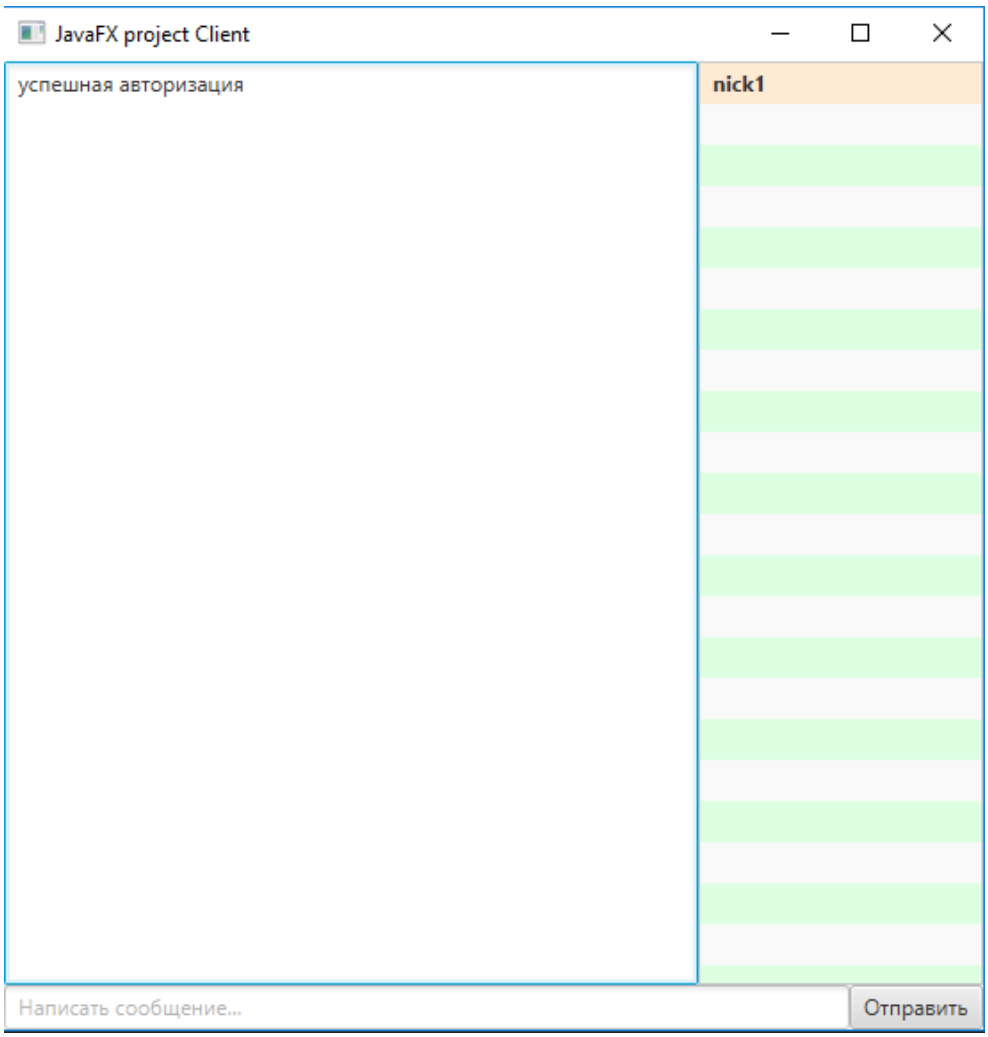

Рисунок 11 – Окно чата

Теперь для полной проверки чата открываем 3 клиента сразу и начинаем переписку (рис.12).

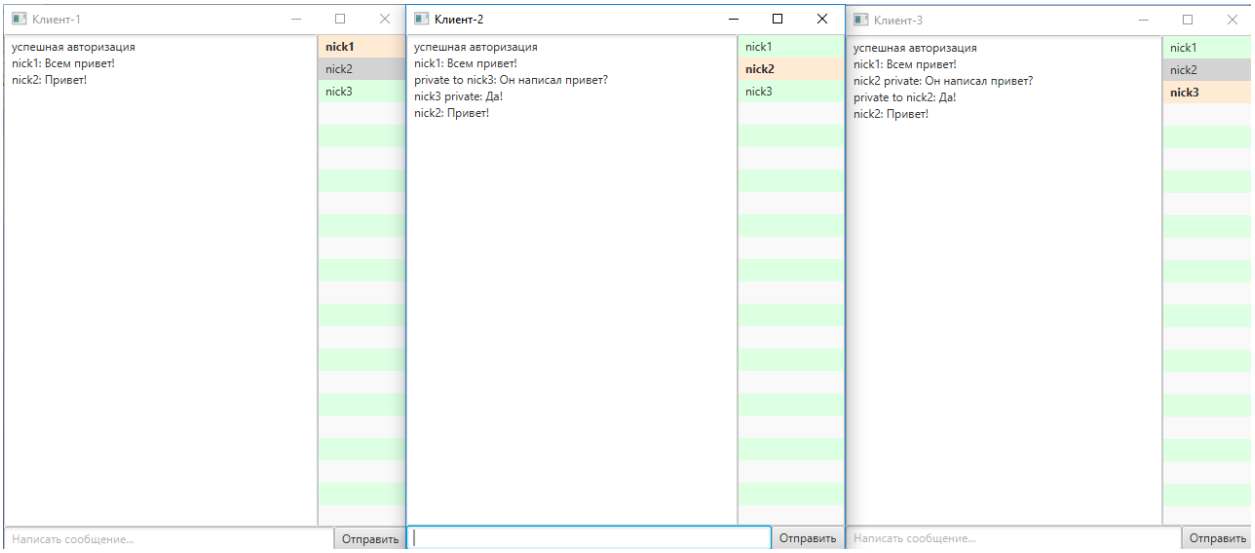

Рисунок 12 – Проверка чата

Как видно из рисунка, то клиент-1 написал в общий чат сообщение, которое отображается у всех клиентов, но клиент 2 отправил личное сообщение клиенту-3 и получил на него ответ, о том что оно личное можно судить по тому, что у клиента-1 ничего не отобразилось, после чего клиент 2 написал в общий чат и у всех отобразилось сообщение.

После всех тестов мы получили рабочее приложение позволяющее совершать обмен сообщениями между пользователей одной сети..

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 Постулат. 2019. №5 ISSN 2414-4487 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.