**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ …………………………………………………………………………. 4

1. АНАЛИЗ ИСХОДНЫХ ДАННЫХ …………………………………………………. 5
   1. Анализ предметной области …………………………………………………. 5
   2. Специальное и общесистемное программное обеспечение .…………………….5
2. ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ ………………………………………….. 6
   1. Пользовательский интерфейс ………………………………………………….. 6
   2. Функциональное обеспечение ………………………………………………… 7
   3. Структурное обеспечение ………………………………………………… 7
   4. Алгоритмическое обеспечение ………………………………………………… 8
   5. Архитектурное обеспечение ………………………………………………… 8
   6. Информационные структуры ………………………………………………… 8
3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ………………………………………………… 9
   1. Детальная реализация функциональных частей программного обеспечения 9
      1. Реализация функциональной части ………………………………….. 9
      2. Реализация пользовательского интерфейса ………………………….. 12
   2. Сопроводительная документация программного обеспечения …………... 21
      1. Минимальные аппаратные и системные требования …………………….. 21
      2. Разработка описания программы ………………………………….. 21
      3. Разработка руководства пользователя ………………………………….. 22
   3. Анализ программного обеспечения …………………………………………... 23
      1. Анализ исходного кода приложения …………………………………... 23
      2. Анализ производительности приложения …………………………… 24
   4. Тестирование программного обеспечения …………………………………… 25
      1. Разработка модульных тестов …………………………………………… 25
      2. Результаты интегрированного тестирования …………………………… 26

ЗАКЛЮЧЕНИЕ …………………………………………………………………………... 27

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ ………………………………….. 28

ПРИЛОЖЕНИЯ ………………………………………………………………………….. 29

# ВВЕДЕНИЕ

По условиям данного курсового проекта необходимо было разработать клиент-серверное приложение «Аудио конференция».

Аудио конференция – программное обеспечение, устанавливаемое на компьютере пользователей и предназначенное для голосового общения с другими пользователями. Программа должна обеспечивать передачу аудио информации с микрофона другим клиентам, а так же информацию о самих клиентах участвующих в разговоре.

На данный момент существует большое количество аналогичных программ для голосового общения. Примерами таких программ являются: TeamSpeak**,** Skype**,** RaidCall, Venriloи другие. В данном курсовом проекте реализована самая необходимая часть для реализации голосового общения между пользователями.

Подобное приложение востребовано, так как использовать телефон для междугородных звонков не экономно, а так же общение происходит только между двумя людьми. В век компьютерных технологий можно решить эту проблему - имея компьютер, наушники, микрофон и стабильное подключение к сети Интернет можно избежать денежных затрат и общаться с несколькими людьми сразу. Так же такую программу можно использовать геймерам, людям, которые играют в онлайн-игры командами. В командной игре важна координация каких либо действий каждого игрока, а использовать текстовый чат в рамках игры неудобно и медленно, поэтому идеальным вариантом становится голосовая связь.

В ходе данной курсовой работы я создал приложение, позволяющее общаться группе людей используя микрофон и динамики.

# 1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ

## 1.1 Анализ предметной области

В данной курсовой работе необходимо разработать клиент-серверное приложение «Аудио конференция».

Пользователи должны иметь возможность слышать друг друга, а так же видеть список участников аудио конференции. В любой момент пользователь может отключиться от аудио конференции без каких либо проблем.

## 1.2 Специальное и общесистемное программное обеспечение

При написании данного курсового проекта использовалось следующее стороннее программное обеспечение и инструментарий:

* Язык программирования, используемый при написании данного курсового проекта – Java.
* NetBeans IDE 8.0.2 – среда разработки программы;
* Microsoft Office Word 2007 – средство для написания пояснительной записки, описания программы;
* Enterprise Architect 12.0 – проектирование программы и составление диаграмм.

# 2 ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ

## 2.1 Пользовательский интерфейс

При разработке данного курсового проекта, был создан простой и интуитивно понятный графический интерфейс, позволяющий пользователю максимально эффективно работать с клиентской частью. Для серверной части был реализован консольный интерфейс, так как в графическом интерфейсе нет нужды.

Клиентское приложение имеет поля для ввода IP-адреса и порта для подключения к серверной части приложения. Кнопки «Connect» и «Disconnect», подключение и отключение, соответственно. Список пользователей, подключенных к серверу.

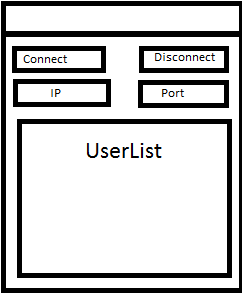


Рисунок 2.1 – Графический интерфейс клиента.

## 2.2 Функциональное обеспечение

Функциональное обеспечение сервера и клиента продемонстрировано на диаграмме вариантов использования (см. Приложение А).

Пользователь клиентского приложения может указать адрес и порт сервера. Подключение к серверу происходит автоматически.

## 2.3 Структурное обеспечение

Серверное приложение содержит класс RequestHandler, в котором обрабатываются и записываются данные о подключаемых клиентах, а так же происходит синхронизация пользователей друг с другом, класс UsersList, в котором происходит добавление пользователей в созданный список. Инициализация протокола TCP/IP осуществляющаяся средствами java.net.ServerSocket, происходит в классе Server.

Клиентское приложение содержит объект класса ConnectionManager, который отсылает сигналы классу MicrophoneReader считывать данные с микрофона и передавать его серверу. Класс GUI содержит интерфейс программы.

Диаграмма классов представлена в Приложении Б.

## 2.4 Алгоритмическое обеспечение

При запуске серверного приложения, сервер создает поток сервиса, который обеспечивает прием данных о клиентах, и переходит в режим ожидания подключения новых клиентов. При подключении нового клиента, сервер выводит сообщение о подключении клиента и его имя, и снова перейдет в ждущий режим.

При запуске клиентского приложения, загружается файл конфигурации, который устанавливает поля IP адреса и порта. Потом клиент создает отдельный поток для связи с сервером и передачи звука. Подключение к серверу происходит при нажатии определенной кнопки.

Диаграммы последовательности и активности представлены в Приложениях В-Г.

## 2.5 Архитектурное обеспечение

Данное приложение планируется реализовать на операционной системе Windows 10 x64/x86, в среде NetBeans IDE 8.0.2. Для связи клиента и сервера, используется протокол TCP/IP. Так же для работы клиента и сервера, необходимы файлы библиотек Java.

Диаграмма развертывания представлена в приложении Д.

## 2.6 Информационные структуры

Пользователь не может самостоятельно изменить протокол передачи данных. Для передачи данных используется протокол TCP/IP и UDP.

Пользователь самостоятельно может изменить порт и адрес подключения.

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## 3.1 Детальная реализация функциональных частей программного обеспечения

### 3.1.1 Реализация функциональных частей

Методы, реализованные в серверном приложении, представлены в листинге 3.1.1.

Листинг 3.1.1 – Методы сервера.

|  |
| --- |
| 1. public void run(); 2. synchronized public int getIndex(); 3. private static void addUser(Socket user) |

Исходный код, реализующий работу сервера с клиентом, представлен в листинге 3.1.2.

Листинг 3.1.2 – Работа сервера с клиентом.

|  |
| --- |
| 1. public class RequestHandler extends Thread { 2. private Socket socket; 3. private String ip; 4. private int port; 5. public RequestHandler(Socket socket) { 6. this.socket = socket; 7. ip = socket.getInetAddress().toString(); 8. port = socket.getPort(); 9. } 10. @Override 11. public void run() { 12. synchronized (UsersList.users) { 13. DataOutputStream dataOutputStreamCurrent = null; 14. try { 15. dataOutputStreamCurrent = new DataOutputStream(socket.getOutputStream()); 16. } catch (IOException e) { 17. e.printStackTrace(); 18. } 19. for (int i = 0; i < UsersList.users.size(); i++) { 20. DataOutputStream dataOutputStreamOther = null; 21. try { 22. dataOutputStreamOther = new DataOutputStream(UsersList.users.get(i).getOutputStream()); 23. } catch (IOException e) { 24. e.printStackTrace(); 25. } 26. try { 27. dataOutputStreamOther.writeUTF("ADD"); 28. dataOutputStreamOther.writeUTF(ip); 29. dataOutputStreamOther.writeInt(port); 30. dataOutputStreamOther.flush(); 31. dataOutputStreamCurrent.writeUTF("ADD"); 32. dataOutputStreamCurrent.writeUTF(UsersList.users.get(i).getInetAddress().toString()); 33. dataOutputStreamCurrent.writeInt(UsersList.users.get(i).getPort()); 34. dataOutputStreamCurrent.flush(); 35. } catch (IOException e) { 36. e.printStackTrace(); 37. } 38. } 39. UsersList.addUser(socket); 40. } 41. DataInputStream dataInputStream = null; 42. try { 43. dataInputStream = new DataInputStream(socket.getInputStream()); 44. String request = dataInputStream.readUTF(); 45. if (request.compareToIgnoreCase("DELETE") == 0) { 46. synchronized (UsersList.users) { 47. UsersList.users.remove(getIndex()); 48. for (int i = 0; i < UsersList.users.size(); i++) { 49. DataOutputStream dataOutputStreamOther = null; 50. try { 51. dataOutputStreamOther = new DataOutputStream(UsersList.users.get(i).getOutputStream()); 52. } catch (IOException e) { 53. e.printStackTrace(); 54. } 55. try { 56. dataOutputStreamOther.writeUTF("DELETE"); 57. dataOutputStreamOther.writeUTF(ip); 58. dataOutputStreamOther.writeInt(port); 59. dataOutputStreamOther.flush(); 60. } catch (IOException e) { 61. e.printStackTrace(); 62. } 63. } 64. } 65. } 66. } catch (IOException e) { 67. e.printStackTrace(); 68. }finally { 69. try { 70. socket.close(); 71. } catch (IOException e) { 72. e.printStackTrace(); 73. } 74. } 75. } 76. synchronized public int getIndex(){ 77. int index = 0; 78. for (int i = 0; i < UsersList.users.size(); i++) { 79. String localIp = UsersList.users.get(i).getInetAddress().toString(); 80. int localPort = UsersList.users.get(i).getPort(); 81. if (localIp.equals(ip) && localPort == port) { 82. index = i; 83. return index; 84. } 85. } 86. return index; 87. } 88. } |

Исходный код, реализующий инициализацию сервиса и подключение новых клиентов, представлен в листинге 3.1.3.

Листинг 3.1.3 –Подключение клиентов.

|  |
| --- |
| 1. public class Server { 2. public Server(){ 3. ServerSocket serverSocket = null; 4. UsersList usersList = null; 5. try { 6. usersList = new UsersList(); 7. serverSocket = new ServerSocket(6969); 8. while (true) { 9. if (UsersList.users.size() != 4) { 10. Socket socket = serverSocket.accept(); 11. new RequestHandler(socket).start(); 12. } 13. try { 14. Thread.sleep(200); 15. } catch (InterruptedException e) { 16. e.printStackTrace(); 17. } 18. } 19. } catch (IOException e) { 20. e.printStackTrace(); 21. } 22. } 23. } 24. } |

Функции, реализованные в клиентском приложении, представлены в листинге 3.1.4.

Листинг 3.1.4 – Функции клиента.

|  |
| --- |
| 1. public int getIndex() 2. public void closeConnection() 3. public String getIp() 4. public int getPort() 5. public void run() 6. public void closeSocket() |

Исходный код, реализующий отправку звука с клиента, представлен в листинге 3.1.5.

Листинг 3.1.5 – Отправка звука с клиента.

|  |
| --- |
| 1. public void run() { 2. try { 3. datagramSocket = new DatagramSocket(); 4. DatagramPacket datagramPacket; 5. System.out.println("UDP send socket preper"); 6. while (!finishFlag) { 7. synchronized (ConnectionManager.data) { 8. ConnectionManager.senderNotReady++; 9. datagramPacket = new DatagramPacket(ConnectionManager.data, ConnectionManager.data.length, inetAddress, port); 10. datagramSocket.send(datagramPacket); 11. System.out.println("datagram sendet"); 12. ConnectionManager.senderNotReady--; 13. } 14. try { 15. Thread.sleep(150); 16. } catch (InterruptedException e) { 17. e.printStackTrace(); 18. } 19. } |

Исходный код, реализующий подключение к серверу, представлен в листинге 3.1.6.

Листинг 3.1.6 – подключение клиента к серверу.

|  |
| --- |
| 1. public ConnectionManager(){ 2. mr = new MicrophoneReader(); 3. mr.start(); 4. Socket socket = null; 5. try { 6. socket = new Socket(InetAddress.getByName("192.168.43.56"), 6969); 7. new ServerConnection(socket).start(); 8. } catch (IOException e1) { 9. e1.printStackTrace(); 10. } 11. } |

### 3.1.2 Реализация пользовательского интерфейса

Исходный код, реализующий графическое отображение окна клиента, представлен в листинге 3.1.7, а само окно – на рисунке 3.1

Листинг 3.1.7 – Внешний вид окна клиента.

|  |
| --- |
| 1. import java.awt.\*; 2. import javax.swing.\*; 3. import java.awt.event.ActionEvent; 4. import java.awt.event.ActionListener; 5. import java.io.\*; 6. import java.util.Scanner; 7. public class Main extends JFrame{ 8. private JButton connect, disconnect; 9. private JTextField host, port; 10. private JList<String>stringJList; 11. private JLabel jLabel; 12. private JFormattedTextField jFormattedTextField; 13. public Main() throws FileNotFoundException { 14. super("Client"); 15. setSize(250, 250); 16. setResizable(false); 17. setLayout(new FlowLayout()); 18. connect = new JButton("Connect!"); 19. disconnect = new JButton("Disconnect!"); 20. host = new JTextField(); 21. port = new JTextField(); 22. host.setEditable(true); 23. host.setText("IP "); 24. port.setText("Port "); 25. stringJList = new JList<String>(); 26. add(connect); 27. add(disconnect); 28. add(host); 29. add(port); 30. add(stringJList); 31. connect.addActionListener(new Listener()); 32. disconnect.addActionListener(new Listener()); 33. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 34. setVisible(true); 35. } 36. public class Listener implements ActionListener { 37. public void actionPerformed(ActionEvent e) { 38. if (e.getSource() == connect) { 39. } 40. if (e.getSource() == disconnect) { 41. } 42. } 43. } 44. public static void main(String[] args) throws FileNotFoundException { 45. new Main(); 46. } 47. } |
|  |

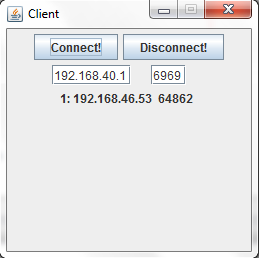


Рисунок 3.1 – Окно клиента.

## 3.2 Сопроводительная документация программного обеспечения

### 3.2.1 Минимальные аппаратные и системные требования

Минимальные системные требования:

* Операционная система: Windows 7 x32;
* Процессор: 2.00 Ghz Intel Pentium CPU;
* Оперативная память: 2 GB;
* Разрешение экрана: 1024x768.

**3.2.2 Разработка описания программы**

Для запуска серверной части приложения необходимо запустить HumanRoarServer.jar. Клиенту в свою очередь необходимо запустить приложение HumanRoarClient.jar.

Пользователю клиентской части, после запуска необходимо ввести значение для полей IP-адрес и порт компьютера, представляющего серверную часть. Сделать это можно в главном окне приложения.

Серверная часть запустится автоматически.

**3.2.3 Руководство пользователя**

Работа с Серверным приложением:

1. После запуска программы появится главное окно.

2. Как только клиент подключится – его порт и IP-адрес появятся в консоли.

3. Отключение сервера происходит при вводе в консоль “Exit”.

Работа с Клиентским приложением:

1. После запуска программы появится главное окно приложения.

2. Для подключения требуется ввести в поля “IP” и “Port” нужные значения и нажать кнопку “Connect!”.

3. Для отключения от сервера нужно нажать кнопку “Disconnect!”.

4. Для закрытия программы необходимо нажать на крестик в верхнем правом углу.

## 3.3 Анализ программного обеспечения

### 3.3.1 Анализ исходного кода приложения

Анализ исходного кода проводился с помощью встроенной утилиты в среду разработки NetBeans IDE 8.0.2., которую можно вызвать с помощью выбора “Выполнить” – “Тестировать проект”.

## Анализ исходного кода сервера до исправления ошибок представлен на рисунке 3.3.

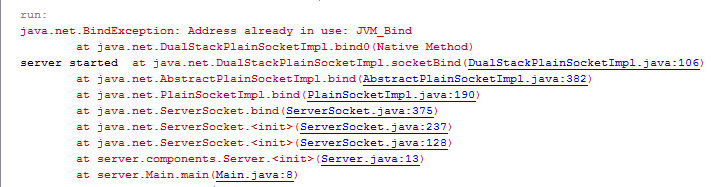
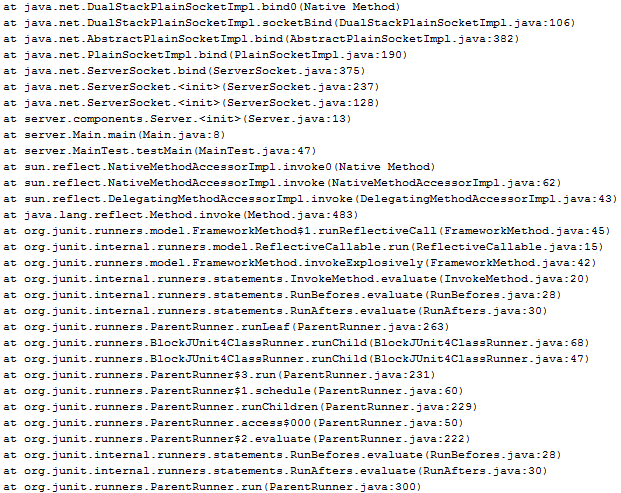


Рис. 3.3 – Анализ исходного кода проекта до исправления

Анализ исходного кода проекта после исправления ошибок представлен на рисунке 3.4.



Рисунок 3.4 – Анализ исходного кода сервера после редактирования

### 3.3.2 Анализ производительности приложения

Для анализа производительности серверного приложения и клиентского приложения был использован стандартный анализатор входящий в NetBeans IDE 8.0.2.

График анализа производительности сервера представлен на рисунке 3.5.

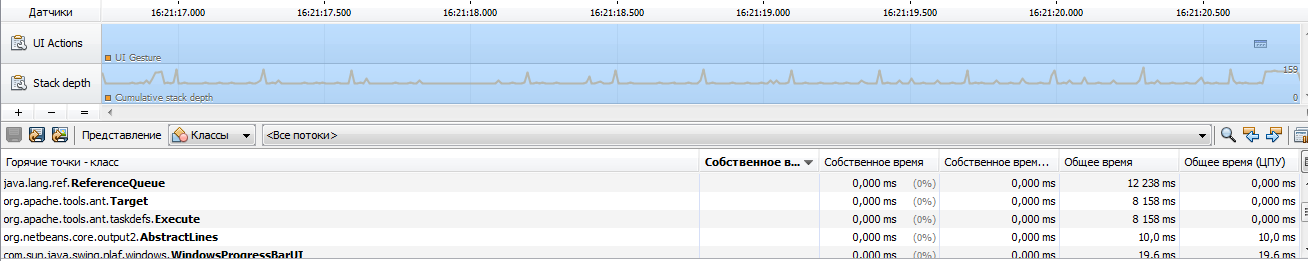


Рисунок 3.5 – График производительности сервера.

График анализа производительности клиента представлен на рисунке 3.6.

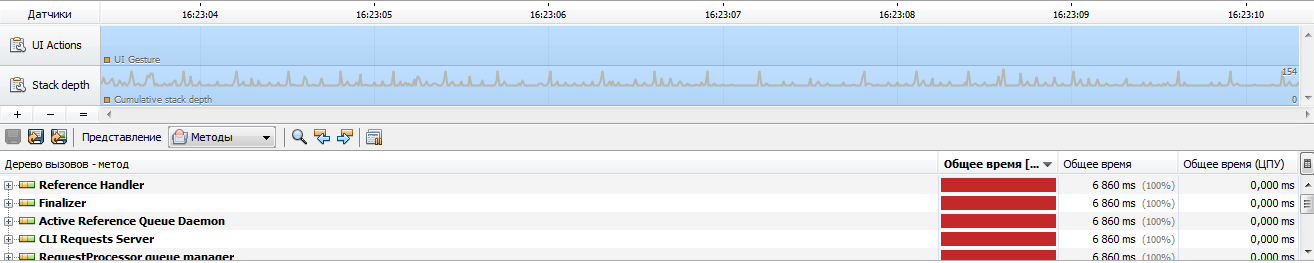


Рисунок 3.6 – График производительности клиента.

## 3.4 Тестирование программного обеспечения

### 3.4.1 Разработка модульных тестов

Модульное тестирование проводилось по мере выполнения данного проекта, тем не менее, и в конечной версии программы сохранились некоторые из тестов.

При завершении работы сервера пользователю выводится сообщение об отключении сервера.

### 3.4.2 Разработка и результаты интегрированного тестирования

Интегрированные тесты и их результаты приведены в таблицах 3.1 и 3.2 соответственно.

Таблица 3.1 – Тесты

|  |  |  |
| --- | --- | --- |
| Цель теста | Действие | Результат |
| 1. Запуск приложения на разных версиях ОС Windows | Программа запущенна успешно | Выполнено |
| 1. Установка соединения между двумя объектами по локальной сети | Соединение установлено | Выполнено |
| 1. Запуск нескольких экземпляров проекта на одном компьютере | Программа успешно запущенна | Выполнено |
| 1. Подключение к серверу клиента | Ожидание подключения | Подключен |
| 1. Отключение от сервера клиента | Удаление из списка подключенных | Выполнено |
| 1. Закрытие сервера | Неожиданное отключение | Клиент отключился и перешел в режим ожидания |

Таблица 3.2 – Результаты тестов

|  |  |
| --- | --- |
| Номер теста | Результат |
| 1 | Пройден |
| 2 | Пройден |
| 3 | Пройден |
| 4 | Пройден |
| 5 | Пройден |
| 6 | Пройден |

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было реализовано клиент-серверное приложение для аудио конференции. Был разработан простой и понятный интерфейс серверного приложения, достаточный для его запуска. Разработанный клиентский интерфейс дает возможность пользователю без проблем пользоваться приложением. Неопытный пользователь сразу сможет освоить интуитивно понятный интерфейс и сразу может начать общаться с другими пользователями данного приложения. Также были получены навыки по написанию приложений в среде разработки Java.

Тестирование программы прошло успешно, выявленные неполадки были исправлены. Программа показала себя устойчивой к непредвиденным сбоям, таким как выключение сервера или клиента.

В результате выполнения данного курсового проекта мной были получены навыки и опыт работы с языком программирования Java, а так же подключаемыми плагинами JFrame и easyUML. Был повышен общий уровень знаний в данной предметной области.

# 

# 

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Липаев, В. В. Проектирование программных систем /В. В. Липаев. – М. : Высш. шк., 1996.
2. Форум программистов и сисадминов CyberForum.ru // Электронный ресурс: URL: http://www.cyberforum.ru//
3. Cайт вопросов и ответов для программистов // Электронный ресурс: URL: http://ru.stackoverflow.com/

## Приложение А

**(обязательное)**

**Диаграмма вариантов использования**

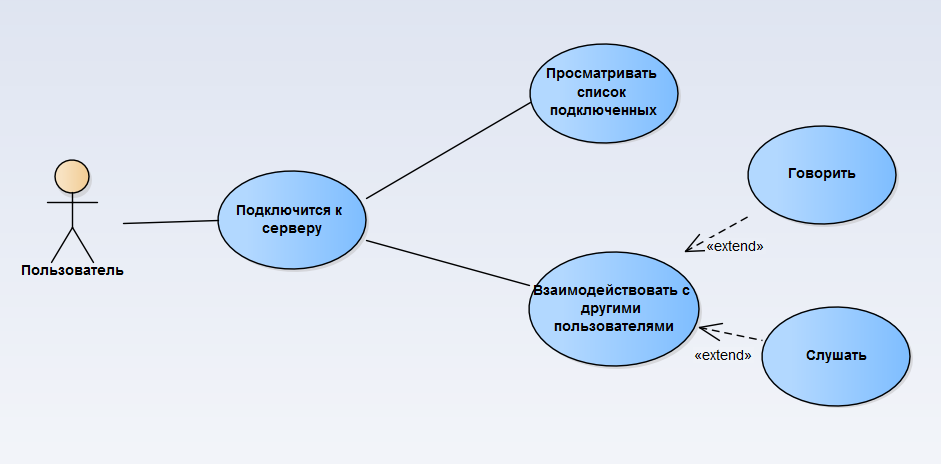
****

Рисунок A.1 – Диаграмма вариантов использования

## Приложение Б

**(обязательное)**

**Диаграмма классов**

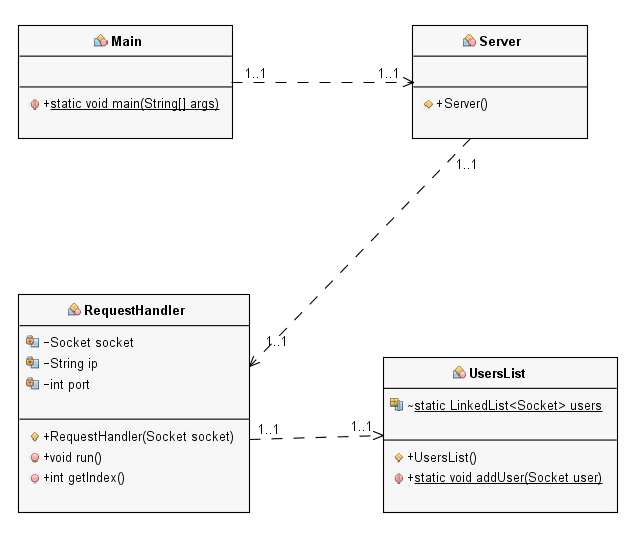
****

Рисунок Б.1 – Диаграмма классов сервера

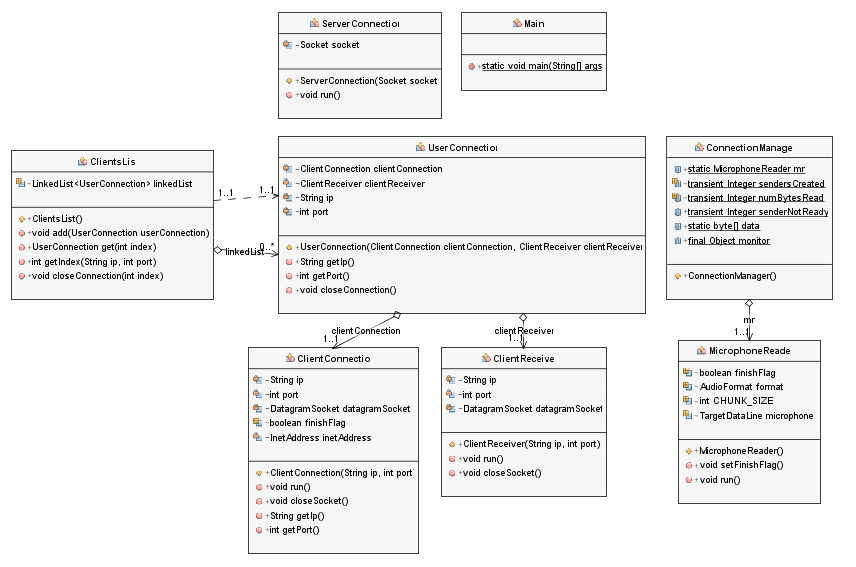


Рисунок Б.2 – Диаграмма классов клиента

**Приложение В**

**(обязательное)**

**Диаграмма последовательности**

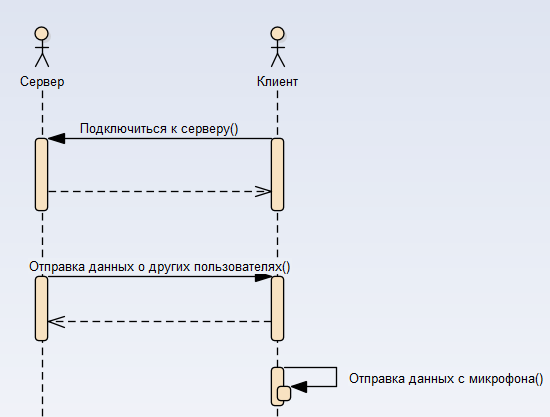
****

Рисунок В – Диаграмма последовательности

## Приложение Г

**(обязательное)**

**Диаграмма активности**

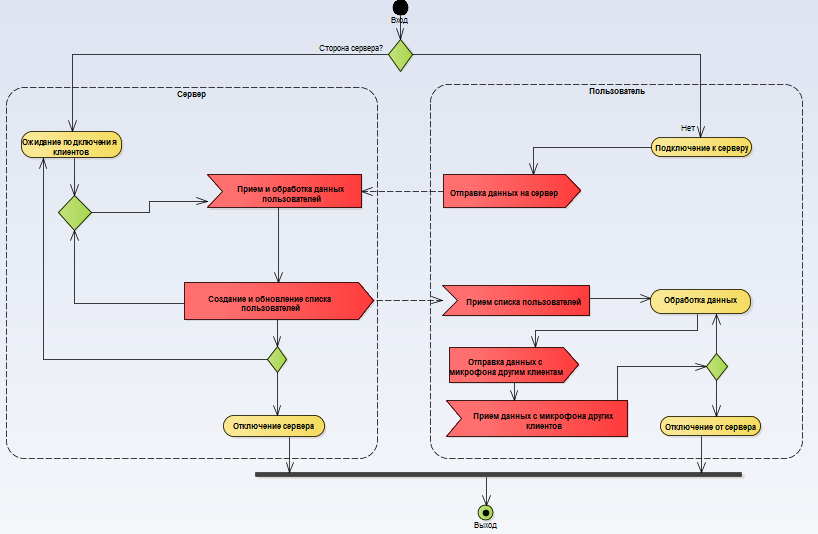
****

Рисунок Г.1 – Диаграмма активности

## Приложение Д

**(обязательное)**

**Диаграмма развертывания**

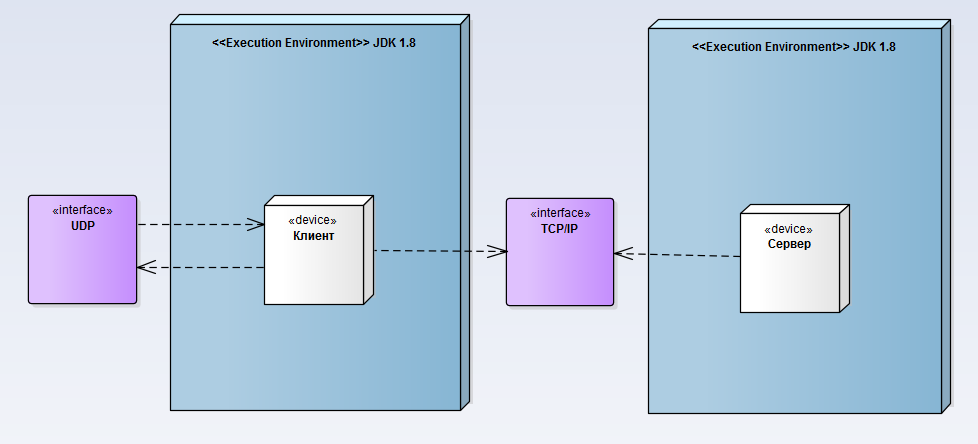
****

Рисунок Д – Диаграмма развертывания