# 软件设计文档

## 技术选型理由

本次开发中我们设计并实现了一个使用在Android操作系统上的聊天程序.

选择Android操作系统作为软件的使用环境是因为Android操作系统适用范围极广, 开发使用Java语言, 易于学习, 并且可参考的资料更多. Android操作系统是一个基于Linux内核的开放源代码移动操作系统. 2017年3月, Android 全球网络流量和设备超越 Microsoft Windows, 正式成为全球第一大操作系统.

本次开发使用的编程语言为Java语言.

## 架构设计

本软件要实现的功能为用户登录, 查询所有在线用户, 添加好友, 移除好友, 发送个人消息, 发送群组消息, 用户退出.

整个系统被分为客户端和服务器端. 客户端运行在Android操作系统上, 服务器端负责处理来自客户端的信息和请求, 并将信息发送给指令中指定的接收端.

服务器端分为四个部分, 分别为Client, ClientManager, HandlerThread, 以及MainThread.

1. public class Client {
2. String name;
3. Socket socket;
4. OutputStreamWriter streamWriter;
5. BufferedWriter bufferedWriter;
6. public Client(String n, Socket s) {
7. name = n;
8. socket = s;
9. try {
10. streamWriter = new OutputStreamWriter(socket.getOutputStream(), "utf-8");
11. bufferedWriter = new BufferedWriter(streamWriter);
12. } catch (IOException e) {
13. e.printStackTrace();
14. }
15. }
16. public void sendMessage(String message) {
17. try {
18. bufferedWriter.write(message);
19. bufferedWriter.newLine();
20. bufferedWriter.flush();
21. } catch (IOException e) {
22. e.printStackTrace();
23. }
24. }
25. }

Client中包含一个基本的sendMessage函数, 用于向BufferedWriter发送指定信息.

ClientManager负责在服务器端对来自客户端的信息进行分类加工. addClient函数负责添加客户端. removeClient函数用于关闭客户端. getAllClient函数显示所有在线用户. clientAddFriend函数对来自客户端的信息(用户名)进行处理(添加前缀’2’), 将处理后的信息使用send Message函数发送. clientRemoveFriend函数对来自客户端的信息(用户名)进行处理(添加前缀’3’), 将处理后的信息使用send Message函数发送. sendPersonalMessage函数对来自客户端的信息(对话消息)进行处理(添加前缀’4’), 将处理后的信息使用send Message函数发送. sendGroupMessage函数对来自客户端的信息(群发消息)进行处理(添加前缀’5’), 将处理后的信息使用send Message函数发送.

1. public boolean addClient(String clientname, Socket socket) {
2. boolean isexisted = false;
3. for (Client client: clients) {
4. if (client.name.equals(clientname)) {
5. isexisted = true;
6. break;
7. }
8. }
9. if (isexisted) {
10. return false;
11. } else {
12. Client client = new Client(clientname, socket);
13. clients.add(client);
14. return true;
15. }
16. }
17. public void removeClient(String clientname) {
18. Client client = null;
19. for (Client temp: clients) {
20. if (temp.name.equals(clientname)) {
21. client = temp;
22. break;
23. }
24. }
25. if (client != null) {
26. clients.remove(client);
27. System.out.println(client.name + " off");
28. }
29. }
30. public String getAllClient() {
31. String result = "";
32. for (Client client: clients) {
33. result += client.name + " ";
34. }
35. return result;
36. }
37. public void clientAddFriend(String clientname, String friendname) {
38. for (Client client: clients) {
39. if (client.name.equals(clientname)) {
40. String message = "2 " + friendname;
41. client.sendMessage(message);
42. break;
43. }
44. }
45. }
46. public void clientRemoveFriend(String clientname, String friendname) {
47. for (Client client: clients) {
48. if (client.name.equals(clientname)) {
49. String message = "3 " + friendname;
50. client.sendMessage(message);
51. break;
52. }
53. }
54. }
55. public void sendPersonalMessage(String sender, String receiver, String words) {
56. for (Client client: clients) {
57. if (client.name.equals(receiver)) {
58. String message = "4 " + sender + " " + words;
59. client.sendMessage(message);
60. break;
61. }
62. }
63. }
64. public void sendGroupMessage(String sender, String words) {
65. String message = "5 " + sender + " " + words;
66. for (Client client: clients) {
67. client.sendMessage(message);
68. }
69. }

HandlerThread函数对处理后的信息进行分析并依此执行相应函数, 分别对’1’, 2’, 3’, 4’, ‘5’, ‘6’等不同前缀做出相应处理, 执行绑定用户, 查询在线用户, 添加好友, 删除好友, 发送私人信息, 发送群组信息等功能.

1. if (tag == '0') {
2. boolean issucceed = manager.addClient(content, socket);
3. String feedback = null;
4. if (issucceed) {
5. clientname = content;
6. feedback = "0 succeed";
7. System.out.println(clientname + " on");
8. } else {
9. feedback = "0 fail";
10. }
11. sendMessage(feedback);
12. continue;
13. }
14. if (tag == '1') {
15. System.out.println(clientname + " query");
16. String result = manager.getAllClient();
17. String feedback = "1 " + result;
18. sendMessage(feedback);
19. continue;
20. }
21. if (tag == '2') {
22. String[] temp = content.split(" ");
23. manager.clientAddFriend(temp[1], temp[0]);
24. continue;
25. }
26. if (tag == '3') {
27. String[] temp = content.split(" ");
28. manager.clientRemoveFriend(temp[1], temp[0]);
29. continue;
30. }
31. if (tag == '4') {
32. int index1 = content.indexOf(" ");
33. String sender = content.substring(0, index1);
34. String temp = content.substring(index1 + 1);
35. int index2 = temp.indexOf(" ");
36. String receiver = temp.substring(0, index2);
37. String words = temp.substring(index2 + 1);
38. manager.sendPersonalMessage(sender, receiver, words);
39. continue;
40. }
41. if (tag == '5') {
42. int index = content.indexOf(" ");
43. String sender = content.substring(0, index);
44. String words = content.substring(index + 1);
45. manager.sendGroupMessage(sender, words);
46. continue;
47. }
48. if (tag == '6') {
49. closeConnection();
50. manager.removeClient(clientname);
51. break;
52. }

## 模块划分

程序分为Client和Server两个部分.

Server分为四个模块, 分别为 Client, ClientManager, HandlerThread, MainThread.

Client分为Chat, ChattingRecord, FriendList等模块.

## 软件设计技术

本次开发中我们使用到了Structure Programming和Object-Oriented Programming中的多种技术, 有效提高了代码质量和可读性.

### Structure Programming

在开发本软件的过程中, 我们使用了结构化编程的思想. 这样的方式大大提高了整个工程的结构性, 使得源代码可读性更强, 也更不容易出现错误.

If-then语句, 体现结构性:

1. if (tag.equals("2")) {
2. friendListHelper.insert(content);
3. return;
4. }
5. if (tag.equals("3")) {
6. friendListHelper.delete(content);
7. return;
8. }
9. if (tag.equals("4")) {
10. int index = content.indexOf(" ");
11. String sender = content.substring(0, index);
12. String words = content.substring(index + 1);
13. if (sender.equals(friendname)) {
14. adapter4.addRecord(sender, words);
15. } else {
16. ChatRecordHelper helper = new ChatRecordHelper(Chat.this, sender, null, 1);
17. helper.insert(sender, words);
18. }
19. if (chatingListHelper.insert(sender, sender, words) < 0) {
20. chatingListHelper.update(sender, sender, words);
21. }
22. return;
23. }
24. if (tag.equals("5")) {
25. int index = content.indexOf(" ");
26. String sender = content.substring(0, index);
27. String words = content.substring(index + 1);
28. if (friendname.equals("群聊")) {
29. adapter4.addRecord(sender, words);
30. } else {
31. ChatRecordHelper helper = new ChatRecordHelper(Chat.this, "群聊", null, 1);
32. helper.insert(sender, words);
33. }
34. if (chatingListHelper.insert("群聊", sender, words) < 0) {
35. chatingListHelper.update("群聊", sender, words);
36. }
37. }

### Object-Oriented Programming

实现每一个功能的函数均定义在一个恰当的类当中, Helper类对Java中的Helper类进行拓展, 实现函数的具体功能.

1. public class ChatingListHelper extends SQLiteOpenHelper
2. public long insert(String name, String speaker, String message) {
3. SQLiteDatabase database = getWritableDatabase();
4. ContentValues values = new ContentValues();
5. values.put("name", name);
6. values.put("speaker", speaker);
7. values.put("message", message);
8. long flag = database.insert(tablename, null, values);
9. database.close();
10. return flag;
11. }
12. public void delete(String name) {
13. SQLiteDatabase database = getWritableDatabase();
14. String[] args = {
15. name
16. };
17. database.delete(tablename, "name = ?", args);
18. database.close();
19. }
20. public void update(String name, String speaker, String message) {
21. SQLiteDatabase database = getWritableDatabase();
22. String[] args = {
23. name
24. };
25. ContentValues values = new ContentValues();
26. values.put("name", name);
27. values.put("speaker", speaker);
28. values.put("message", message);
29. database.update(tablename, values, "name = ?", args);
30. database.close();
31. }
32. public Cursor query() {
33. SQLiteDatabase database = getReadableDatabase();
34. String rawquery = "select \* from " + tablename;
35. return database.rawQuery(rawquery, null);
36. }

### Design Patterns

开发中用到了设计模式中结构式模式里的适配器模式, 即将某个类的接口转换成客户端期望的另一个接口表示. 适配器模式可以消除由于接口不匹配所造成的类兼容性问题.

1. public class Myadapter2 extends BaseAdapter {
2. public FriendListHelper helper;
3. public List < String > list;
4. private Context context;
5. Myadapter2(Context pcontext, FriendListHelper phelper) {
6. helper = phelper;
7. list = new ArrayList < > ();
8. context = pcontext;
9. updateSelf();
10. }
11. public void updateSelf() {
12. list.clear();
13. Cursor cursor = helper.query();
14. while (cursor.moveToNext()) {
15. int index = cursor.getColumnIndex("name");
16. String name = cursor.getString(index);
17. list.add(name);
18. }
19. cursor.close();
20. notifyDataSetChanged();
21. }
22. public void addFriend(String friendname) {
23. for (String name: list) {
24. if (name.equals(friendname)) return;
25. }
26. list.add(friendname);
27. helper.insert(friendname);
28. notifyDataSetChanged();
29. }
30. public void removeFriend(String friendname) {
31. list.remove(friendname);
32. helper.delete(friendname);
33. notifyDataSetChanged();
34. }@
35. Override public int getCount() {
36. if (list == null) return 0;
37. return list.size();
38. }@
39. Override public long getItemId(int i) {
40. return i;
41. }@
42. Override public Object getItem(int i) {
43. if (list == null) return null;
44. return list.get(i);
45. }@
46. Override public View getView(int i, View view, ViewGroup viewgroup) {
47. if (view == null) view = LayoutInflater.from(context).inflate(R.layout.item2, viewgroup, false);
48. String username = list.get(i);
49. Button icon = (Button) view.findViewById(R.id.usericon);
50. TextView name = (TextView) view.findViewById(R.id.username);
51. String firstletter = "" + username.charAt(0);
52. icon.setText(firstletter);
53. name.setText(username);
54. return view;
55. }
56. }