《网络编程技术》

大作业

**项目名称： 基于java实现的网盘**

**姓 名： 易艳钟**

**学 号： 2017109238**

**姓 名： 魏琳**

**学 号： 2017109238**

**姓 名： 张小静**

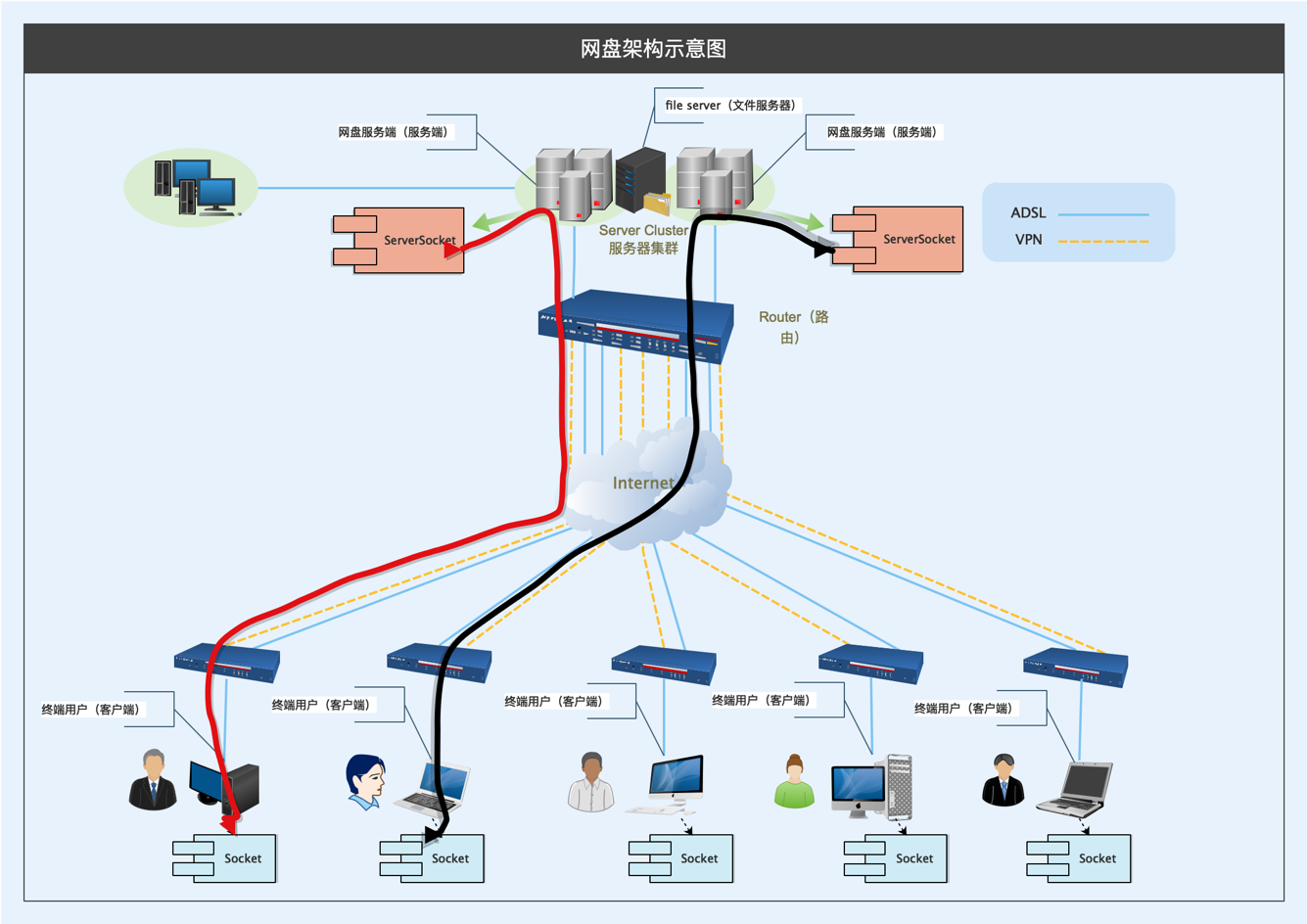
**学 号： 2017109238**

**班 级： 计科1711**

1. **项目目的**
2. 掌握基于TCP/IP协议的Socket接口编写网络应用程序的方法。重点掌握TCP网络应用程序架构的设计思想，并能根据项目需要选择使用TCP或UDP来进行开发。另外针对目前广大客户所使用的平台，本课程中主要注重实用，大部分程序在windows/Linux平台中进行。
3. 能够根据实验结果分析是否达到了实验目的和要求，掌握网络原理和协议，对Socket编程的基本开发流程要比较熟悉，能设计出网络传输程序的基本架构，熟悉套接字格式的使用规范，对网络传输程序的可靠性和健壮性能进行基本的控制。
4. 理解网络编程的实际意义，掌握基本的java语言中socket的使用，io的技术运用，线程的组合使用，以及javaFx或者swing图形界面的基本编程。
5. **功能要求**
6. 用户可以注册成网盘用户（注册从客户端ui上操作，数据存储在服务器数据库）（使用jdbc机制实现即可）。（必做）
7. 用户在客户端打开网盘客户端程序，登陆账号，浏览自己网盘空间里的文件（使用jdbc机制实现即可）。（必做）
8. 用户可以将本地文件上传到网盘服务器上存储。（必做）
9. 用户可以从网盘客户端下载之前存储到网盘服务器上的文件到本地。（必做）
10. 用户可以在网盘中新建文件夹，对文件进行分类管理（选做）。
11. 用户上传文件时可以实现极速秒传功能（选做）。
12. 用户下载文件可以实现断点续功能（选做）。
13. 可以根据用户的会员级别实行限速措施（选做）。
14. **主要技术**
15. socket，io，tcp/udp，集合，线程，java面向对象等基本知识。
16. 数据库sql知识，jdbc基本知识
17. javaFX/swing等图形界面设计的知识。
18. **网络底层架构设计**

整个底层网络结构采用java的tcp连接机制实现。

客户端的ui底层持有一个socket对象，服务器机器上运行的服务端开启一个serversoket对象，服务端设计成多线程模式，接受多人连接，可以同时对多个客户端提供上传和下载服务。客户端如果有多任务下载，可以客户端也开启多个线程支持多任务下载。



1. **功能细节要求:**

具体功能点：

1. 新用户可以打开客户端的窗口，在客户端窗口上填写用户资料注册成网盘用户。服务器接收到注册信息后，将新用户的信息写入数据库存储。注册的账号要做基本的验证和判断重复功能（可以采用序列化存储，或者采用关系型数据库存储）
2. 注册后的用户可以使用注册的账户名和密码在客户端的窗口上登陆操作，登陆成功后，进入网盘主界面，显示当前用户的网盘的空间里的内容。可以对网盘文件进行删除操作。
3. 用户可以点击上传按钮打开文件选择框，选择本地的文件，上传到服务器上存储该文件。上传过程中要有实时进度条提示，还有预估时间显示。
4. 用户可以在网盘主窗口中看到自己上传的保存在服务器端的文件列表，可以选择某个具体的文件点击右键菜单下载该文件。下载过程中同样要有进度条显示进度，同时显示预估时间。
5. 用户上传文件时可以考虑提供极速秒传功能（极速秒传功能其实就是上传之前获取文件的hash值（hash值是每个文件的唯一id，文件一样hash值一摸一样），将hash值提交给服务器，服务器先检查服务器上是否有这个文件，如果有，则不执行io动作，仅仅是将当前用户上传的文件指向服务器上已经有的一个文件地址，然后提示上传成功，并在用户的网盘空间里记录上传了这个文件）。（选做）
6. 用户下载文件过程中，如果临时退出，暂停，后续再次操作下载时，可以在之前的下载进度基础下继续下载文件，而不是重新下载文件，也就是实现断点续传功能。（这块可以参照使用java中的RandomAcessFile类来实现对文件的特殊操作）。（选做）
7. 提供用户可以在网盘空间里新建文件夹对网盘文件进行分类管理的功能，可以新建文件夹，删除文件夹，移动文件到指定文件夹等操作。（选做）
8. 可以给用户分登陆，提供充值功能，充值后变成vip，然后vip用户下载通道快一些，普通用户下载速度慢一些。（选做）
9. **运行截图:**

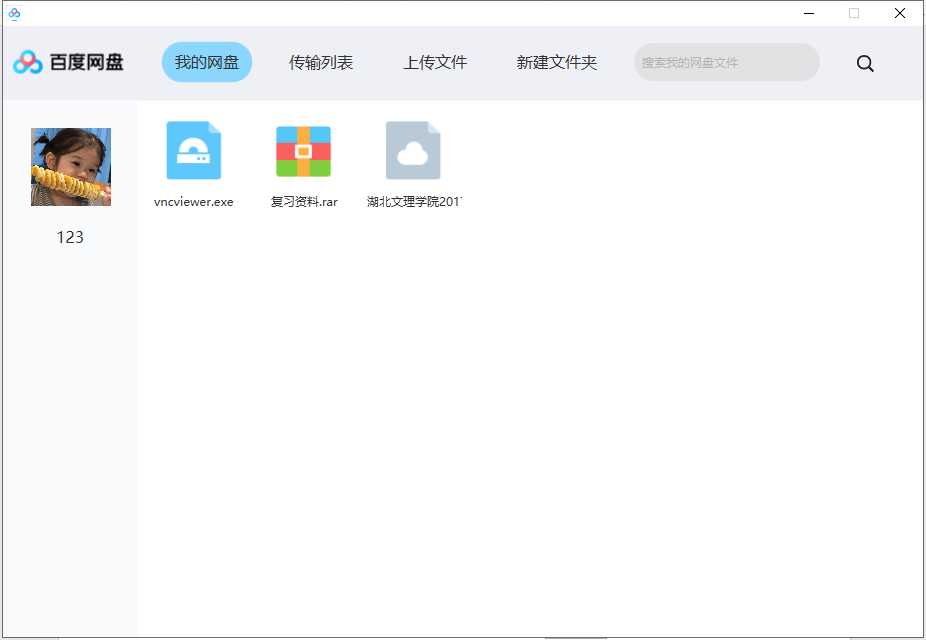
登录界面：



注册界面：



主界面：



1. **关键代码：**

登录实现：

1.首先是页面处理获取到用户输入的信息

2.将表单获取到的信息封装传给服务端

3.服务端接受请求查询数据库然后反馈一个登录是否成功的信息

服务端接收客户端的连接，并进行相应操作

public Server(){  
 //接受客户端的连接  
 try {  
 while (true) {  
 Socket client = serverSocket.accept();  
 out = new ObjectOutputStream(client.getOutputStream());  
 in = new ObjectInputStream(client.getInputStream());  
 dao = new UserDAOImpl();  
 tfdao = new TransferDAOImpl();  
 System.*out*.println(client.getInetAddress().getHostAddress()+"连接进来了");  
 ReceivedMessage rm = new ReceivedMessage(in,out);  
 rm.start();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

登录页面数据获取及数据处理

@FXML  
public void login(){  
 String usernameInput = username.getText().trim();  
 String passwordInput = password.getText().trim();  
 try {  
 Socket client = new Socket("localhost",8123);  
 ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());  
 ObjectInputStream in = new ObjectInputStream(client.getInputStream());  
 //封装用户登录的消息  
 Message loginMessage = new Message();  
 user = new User(usernameInput,passwordInput);  
 loginMessage.setFrom(user);  
 loginMessage.setType(MessageType.*LOGIN*);  
 //写给服务器登录消息  
 out.writeObject(loginMessage);  
 out.flush();  
 //读取服务器返回的消息  
 Message loginResult = (Message) in.readObject();  
 user = loginResult.getFrom();  
 if (user==null){  
 Alert a = new Alert(Alert.AlertType.*ERROR*);  
 a.setContentText("登陆失败，用户名或密码错误！");  
 a.show();  
 }else{  
 //跳转到主页面，并隐藏当前登录页面  
 Main main = new Main();  
 main.start(new Stage());  
 loginBtn.getScene().getWindow().hide();  
 }  
 //关闭连接  
 out.close();  
 in.close();  
 client.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

登录的数据库操作：

*/\*\*  
 \* 用户登录，判断用户名密码是否正确  
 \** ***@param*** *username  
 \** ***@param*** *password  
 \** ***@return*** *\*/*@Override  
public User login(String username, String password) {  
 User user = null;  
 ResultSetHandler<User> bh = new BeanHandler<>(User.class);  
 QueryRunner runner = new QueryRunner(ConnectionPool.*getDataSource*());  
 try {  
 user = runner.query("select \* from USER where username=? and password=?",bh,username,password);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return user;  
}

服务器接收到登录请求的操作：

case *LOGIN*:{  
 System.*out*.println("服务器收到用户登录的消息");  
 User LoginUser = dao.login(message.getFrom().getUsername(),message.getFrom().getPassword());  
 message.setFrom(LoginUser);  
 out.writeObject(message);  
 out.flush();  
 break;  
}

注册功能实现：

1.首先是页面处理获取到用户输入的信息

2.将表单获取到的信息封装传给服务端

3.服务端接受请求查询数据库然后反馈一个注册是否成功的信息

页面上信息获取以及表单验证封装注册信息：

//点击注册按钮，提交用户填写的数据  
@FXML  
public void register() {  
 //获取用户输入的数据  
 String usernameInput = username.getText().trim();  
 String passwordInput = password.getText().trim();  
 String confirmPasswordInput = confirmPassword.getText().trim();  
 String sexInput = male.isSelected()?"男":"女";  
 String ageInput = age.getValue().toString();  
 headControl = (HeadControl) ControlCollections.*controls*.get(HeadControl.class);  
 this.url = headControl.getUrl();  
 String imageInput = url;  
 String telInput = tel.getText().trim();  
 String emailInput = email.getText().trim();  
 //表单验证  
 Toast toast = new Toast(registerBtn.getScene().getWindow());  
 Toast.Level level = Toast.Level.*values*()[0]; //提示级别  
 if(usernameInput.equals("")){  
 toast.show(level, 1000, "用户名不能为空！");  
 username.requestFocus();  
 }else if(passwordInput.equals("")){  
 toast.show(level, 1000, "密码不能为空！");  
 password.requestFocus();  
 }else if(confirmPasswordInput.equals("")){  
 toast.show(level, 1000, "请再次输入密码！");  
 confirmPassword.requestFocus();  
 }else if(!passwordInput.equals(confirmPasswordInput)){  
 toast.show(level, 1000, "两次密码不一致！");  
 confirmPassword.requestFocus();  
 }else {  
 try {  
 Socket client = new Socket("localhost", 8123);  
 ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());  
 ObjectInputStream in = new ObjectInputStream(client.getInputStream());  
 //封装用户注册的消息  
 Message registerMessage = new Message();  
 User user = new User(usernameInput, passwordInput, sexInput, ageInput, telInput, emailInput,imageInput);  
 registerMessage.setFrom(user);  
 registerMessage.setType(MessageType.*REGISTER*);  
 //写给服务器注册消息  
 out.writeObject(registerMessage);  
 out.flush();  
 //读取服务器返回的消息  
 Message registerResult = (Message) in.readObject();  
 user = registerResult.getFrom();  
 if (user==null){  
 Alert a = new Alert(Alert.AlertType.*ERROR*);  
 a.setContentText("注册失败，该用户名已存在！");  
 a.show();  
 }else{  
 //跳转到主页面，并隐藏当前登录页面  
 Login login = new Login();  
 login.start(new Stage());  
 registerBtn.getScene().getWindow().hide();  
 }  
 //关闭连接  
 out.close();  
 in.close();  
 client.close();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
}

查询数据库

*/\*\*  
 \* 用户注册，向数据库插入一条数据  
 \** ***@param*** *user  
 \** ***@return*** *\*/*@Override  
public boolean registerUser(User user) {  
 boolean flag = false;  
 QueryRunner runner = new QueryRunner(ConnectionPool.*getDataSource*());  
 try {  
 int count = runner.update("insert into user values(?,?,?,?,?,?,?)",  
 user.getUsername(),user.getPassword(),user.getSex(),user.getAge(),user.getTel(),user.getEmail(),user.getImages());  
 flag = count>0?true:false;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return flag;  
}

服务端接收来自客户端的请求并且反馈一个注册消息

//注册消息  
case *REGISTER*:{  
 System.*out*.println("服务器收到用户注册消息");  
 boolean result = dao.registerUser(message.getFrom());  
 //插入失败写回空对象  
 if(!result) message.setFrom(null);  
 out.writeObject(message);  
 out.flush();  
 break;  
}

上传头像：

页面上对选中的头像进行获取：

//图片事件的处理  
private void selectImage(ImageView head){  
 head.setOnMouseClicked(new EventHandler<MouseEvent>() {  
 @Override  
 public void handle(MouseEvent event) {  
 SelectImage = head.getImage();  
 url = (String) head.getUserData();  
 pane.getScene().getWindow().hide();  
 }  
 });  
}

主界面显示已有文件：

用户登录成功获取用户名，并查找该用户的所有文件：

@Override  
public void initialize(URL location, ResourceBundle resources) {  
 put();  
 scrollPane.setBackground(new Background(new BackgroundFill(Color.*WHITE*,null,null)));  
 //获取登录的用户  
 LoginControl loginControl = (LoginControl) ControlCollections.*controls*.get(LoginControl.class);  
 this.user = loginControl.getUser();  
 images.setImage(new Image(user.getImages()));  
 username.setText(user.getUsername());  
  
 this.transferControl = new TransferControl();  
  
 //初始化界面时，加载所有的文件  
 try {  
 //建立Socket连接，将当前用户的文件列表从服务器读取过来  
 Socket client = new Socket("localhost",8123);  
 ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());  
 ObjectInputStream in = new ObjectInputStream(client.getInputStream());  
 //封装更新界面的消息  
 Message updateMessage = new Message(user,MessageType.*UPDATE*,null,0);  
 out.writeObject(updateMessage);  
 out.flush();  
 //读取服务器回复的结果消息  
 Message updateResult = (Message)in.readObject();  
 this.fileList = updateResult.getFileList();  
 //取到集合后再实例化文件控制器  
 this.fileControl = new FileControl();  
 myNetDisk();  
 //关闭连接  
 in.close();  
 out.close();  
 client.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

更新主界面登陆初始化以及上传后页面更新

//更新主界面的方法，用于登录时初始化界面以及上传文件时更新界面  
public void updateUI(LinkedList<File> fileList){  
 double x = 0;  
 double y = 0;  
 scrollContent.getChildren().clear();  
 if(fileList!=null && fileList.size()>0) {  
 for (File f : fileList) {  
 String suffix = f.getName().split("\\.")[f.getName().split("\\.").length-1];  
 //存放文件图片  
 ImageView fileImage = null;  
 try {  
 fileImage = new ImageView(new Image("images/" + suffix + ".png"));  
 }catch (Exception e){  
 fileImage = new ImageView(new Image("images/other.png"));  
 }  
 fileImage.setFitWidth(78);  
 fileImage.setFitHeight(78);  
 //存放文件名  
 TextField fileName = new TextField(f.getName());  
 fileName.setStyle("-fx-background-color: white; -fx-alignment: center");  
 fileName.setEditable(false);  
 //垂直布局  
 VBox vBox = new VBox();  
 vBox.setPrefSize(110,120);  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.getChildren().addAll(fileImage, fileName);  
 vBox.setLayoutX(x);  
 vBox.setLayoutY(y);  
 if(x < 660){  
 x += 110;  
 }else{  
 x=0;  
 y += 120;  
 }  
 scrollContent.getChildren().add(vBox);  
 click(vBox); //点击事件  
 }  
 }  
}

服务器端查询数据库列出所有文件：

//更新界面的消息  
case *UPDATE*: {  
 System.*out*.println("服务器收到更新界面的消息");  
 //遍历用户目录下的文件  
 File[] files = userDir.listFiles();  
 LinkedList<File> fileList = new LinkedList<>();  
 for(File f:files){  
 fileList.add(f);  
 }  
 Message updateResult = new Message();  
 updateResult.setFileList(fileList); //文件列表传给客户端  
 out.writeObject(updateResult);  
 out.flush();  
 break;  
}

上传功能：

1.主页面点击长传按钮，弹出文件选择框。

2.根据选择的文件，封装成一个上传信息，传给服务端

3.服务端接收上传请求，通过通道流读取文件信息，再通过字节流从内存写向磁盘

4.并且向传输列表插入一条记录

//上传文件的方法  
@FXML  
public void uploadFile(){  
 clearBtnBackground();  
 uploadBtn.setStyle("-fx-background-color: rgba(96,204,255,0.7); -fx-background-radius: 20");  
 //建立短连接，上传或下载时才连接  
 Socket client = null;  
 ObjectOutputStream out = null;  
 //1. 弹出文件选择器，让用户选择文件  
 FileChooser fc = new FileChooser();  
 File selectFile = fc.showOpenDialog(scrollPane.getScene().getWindow());  
 if(selectFile!=null) {  
 //2. 在执行上传之前先建立底层的socket连接  
 try {  
 client = new Socket("localhost",8123);  
 out = new ObjectOutputStream(client.getOutputStream());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 //3. 上传之前先封装一个消息对象  
 Message uploadMessage = new Message(user, MessageType.*UPLOAD*,  
 selectFile.getAbsolutePath()+" "+selectFile.getName(), selectFile.length());  
 //4. 向服务器传递上传文件的消息  
 try {  
 out.writeObject(uploadMessage);  
 out.flush();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 //5. 消息对象发送给服务器后，就开始上传文件数据  
 try {  
 FileInputStream fileIn = new FileInputStream(selectFile);  
 byte[] bs = new byte[1024];  
 int length = -1;  
 //通过字节流从本地磁盘读到内存，再通过通道流写给服务器  
 while ((length = fileIn.read(bs)) != -1) {  
 out.write(bs, 0, length);  
 out.flush();  
 }  
 Alert a = new Alert(Alert.AlertType.*INFORMATION*);  
 a.setContentText("文件上传成功");  
 a.show();  
 //更新界面，新增一个文件  
 fileList.add(selectFile);  
 myNetDisk();  
 fileIn.close();  
 out.close();  
 client.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }else {  
 System.*out*.println("上传失败，未选择文件");  
 }  
}

服务端处理：

//上传文件的消息  
case *UPLOAD*: {  
 System.*out*.println("服务器收到一条上传文件的消息");  
 //通过通道流读取文件信息，再通过字节流从内存写向磁盘  
 FileOutputStream fileOut = new FileOutputStream(userDir + "/" + message.getFileName().split(" ")[1]);  
 byte[] bs = new byte[1024];  
 int length = -1;  
 while ((length = in.read(bs)) != -1) {  
 fileOut.write(bs,0,length);  
 fileOut.flush();  
 }  
 fileOut.close();  
 in.close();  
 //向传输记录表插入一条数据  
 String time = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date()); //获取当前时间  
 Transfer record = new Transfer(0,message.getFrom().getUsername(),message.getFileName().split(" ")[1],  
 message.getFileLength(),message.getFileName().split(" ")[0],time,"上传");  
 tfdao.addRecode(record);  
 break;  
}

数据库处理：

*/\*\*  
 \* 向传输记录表中添加一条记录  
 \** ***@param*** *t  
 \** ***@return*** *\*/*@Override  
public boolean addRecode(Transfer t) {  
 boolean flag = false;  
 QueryRunner runner = new QueryRunner(ConnectionPool.*getDataSource*());  
 try {  
 int count = runner.update("insert into transfer values(?,?,?,?,?,?,?)",  
 0,t.getUsername(),t.getFilename(),t.getFileLength(),t.getFilepath(),t.getTime(),t.getOperating());  
 flag = count>0?true:false;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return flag;  
}

文件的删除下载重命名：

*/\*\*  
 \* 每个文件的点击事件，点击获取文件信息，弹出菜单，进行其他操作  
 \*/* public void click(VBox vBox){  
 vBox.setOnMouseClicked(new EventHandler<MouseEvent>() {  
 @Override  
 public void handle(MouseEvent event) {  
 //获取点击的文件名  
 String fileName = ((TextField)vBox.getChildren().get(1)).getText();  
 if(event.getButton() == MouseButton.*SECONDARY* && event.getClickCount()==1){  
 //右键单击，弹出菜单  
 ContextMenu rightMenu = new ContextMenu();  
 MenuItem item1 = new MenuItem("下载");  
 MenuItem item2 = new MenuItem("删除");  
 MenuItem item3 = new MenuItem("重命名");  
 rightMenu.getItems().addAll(item1,item2,item3);  
 rightMenu.show(vBox, Side.*BOTTOM*,50,-50);  
 //点击下载，调用下载文件的方法  
 item1.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 downloadFile(fileName);  
 }  
 });  
 //点击删除，调用删除文件的方法  
 item2.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 deleteFile(fileName);  
 }  
 });  
 //点击重命名，调用重命名文件的方法  
 item3.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 TextField fileName = ((TextField)vBox.getChildren().get(1));  
 renameFile(fileName);  
 }  
 });  
 }  
 if(event.getButton() == MouseButton.*PRIMARY* && event.getClickCount()==2){  
 //双击打开文件  
// openFile(fileName);  
 }  
 }  
 });  
 }

文件下载：

//下载文件的方法  
public void downloadFile(String fileName){  
 //在执行下载之前先建立底层的socket连接  
 try {  
 Socket client = new Socket("localhost",8123);  
 ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());  
 ObjectInputStream in = new ObjectInputStream(client.getInputStream());  
 //用户选择文件下载路径 目录选择器  
 DirectoryChooser dc = new DirectoryChooser();  
 File saveDir = dc.showDialog(scrollContent.getScene().getWindow());  
 if(!saveDir.exists()) saveDir.mkdir();  
 //文件的下载路径  
 String filePath = saveDir + "\\" + fileName;  
 //向服务器发送下载消息  
 Message downloadMessage = new Message(user, MessageType.*DOWNLOAD*,filePath+" "+fileName,0);  
 out.writeObject(downloadMessage);  
 out.flush();  
 //通过通道流读取文件信息，再通过字节流从内存写向磁盘  
 FileOutputStream fileOut = new FileOutputStream(saveDir + "/" + fileName);  
 byte[] bs = new byte[1024];  
 int length = -1;  
 while ((length = in.read(bs)) != -1) {  
 fileOut.write(bs,0,length);  
 fileOut.flush();  
 }  
 Alert a = new Alert(Alert.AlertType.*INFORMATION*);  
 a.setContentText("文件下载成功");  
 a.show();  
 //关闭流  
 fileOut.close();  
 out.close();  
 in.close();  
 client.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

字节流从磁盘读取文件写向内存，再通过通道流写向客户端

//下载文件的消息  
case *DOWNLOAD*: {  
 System.*out*.println("服务器收到一条下载文件的消息");  
 //字节流从磁盘读取文件写向内存，再通过通道流写向客户端  
 File file = new File(userDir +"/" +message.getFileName().split(" ")[1]); //目录+文件名  
 FileInputStream fileIn = new FileInputStream(file);  
 byte[] bs = new byte[1024];  
 int length = -1;  
 while((length = fileIn.read(bs)) != -1) {  
 out.write(bs,0,length);  
 out.flush();  
 }  
 fileIn.close();  
 out.close(); //关闭流！！  
 //向传输记录表中插入一条数据  
 String time = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date()); //获取当前时间  
 Transfer record = new Transfer(0,message.getFrom().getUsername(),message.getFileName().split(" ")[1],  
 file.length(),message.getFileName().split(" ")[0],time,"下载");  
 tfdao.addRecode(record);  
 break;  
}

文件删除：

//删除文件的方法  
public void deleteFile(String fileName){  
 try {  
 Socket client = new Socket("localhost",8123);  
 ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());  
 Message deleteMessage = new Message(user,MessageType.*DELETE*,fileName,0);  
 out.writeObject(deleteMessage);  
 out.flush();  
 //删除成功后更新UI界面  
 System.*out*.println("之前"+fileList.size());  
 for (int i=0; i<fileList.size(); i++){  
 if(fileList.get(i).getName().equals(fileName)){  
 fileList.remove(fileList.get(i));  
 break;  
 }  
 }  
 System.*out*.println("后面"+fileList.size());  
 updateUI(fileList);  
 Alert a = new Alert(Alert.AlertType.*INFORMATION*);  
 a.setContentText("文件删除成功");  
 a.show();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

//删除文件的消息  
case *DELETE*:{  
 System.*out*.println("服务器收到一条删除文件的消息");  
 File file = new File(userDir+"/"+message.getFileName());  
 file.delete();  
 break;  
}

传输列表删除一条记录

*/\*\*  
 \* 删除一条记录  
 \** ***@param*** *id  
 \** ***@return*** *\*/*@Override  
public boolean deleteRecode(long id) {  
 boolean flag = false;  
 QueryRunner runner = new QueryRunner(ConnectionPool.*getDataSource*());  
 try {  
 int count = runner.update("delete from transfer where id=?",id);  
 flag = count>0?true:false;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return flag;  
}

文件重命名：

//重命名文件的方法  
public void renameFile(TextField editName){  
 editName.setEditable(true);  
 String fullName = editName.getText(); //文件全名  
 String suffix = fullName.substring(fullName.lastIndexOf(".")); //后缀  
 String before = fullName.substring(0,fullName.lastIndexOf(".")); //文件名  
 //只显示文件名，不允许修改后缀  
 editName.setText(before);  
 editName.requestFocus();  
 editName.setOnKeyReleased(new EventHandler<KeyEvent>() {  
 @Override  
 public void handle(KeyEvent event) {  
 //按下回车键确定修改  
 if(event.getCode() == KeyCode.*ENTER*) {  
 String after = editName.getText();  
 if (after.equals("")) {  
 Alert a = new Alert(Alert.AlertType.*INFORMATION*);  
 a.setContentText("文件名不能为空");  
 a.show();  
 } else {  
 editName.setEditable(false); //修改后不可编辑  
 editName.setText(after + suffix);  
 try {  
 Socket client = new Socket("localhost", 8123);  
 ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());  
 String result = fullName + " " + after + suffix;  
 Message renameMessage = new Message(user, MessageType.*RENAME*, result, 0);  
 out.writeObject(renameMessage);  
 out.flush();  
 Alert a = new Alert(Alert.AlertType.*INFORMATION*);  
 a.setContentText("文件重命名成功");  
 a.show();  
 //关闭流  
 out.close();  
 client.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 //更新文件列表，搜索时能搜到更新后的文件  
 for (int i=0; i<fileList.size(); i++){  
 if(fileList.get(i).getName().equals(fullName)){  
 fileList.remove(fileList.get(i));  
 fileList.add(new File(after+suffix));  
 break;  
 }  
 }  
 }  
 }  
 }  
 });  
}

//重命名文件的消息  
case *RENAME*:{  
 System.*out*.println("服务器收到一条重命名文件的消息");  
 String beforeName = message.getFileName().split(" ")[0];  
 String afterName = message.getFileName().split(" ")[1];  
 File beforeFile = new File(userDir + "/" + beforeName);  
 File afterFile = new File(userDir+ "/" +afterName);  
 beforeFile.renameTo(afterFile);  
 break;  
}

文件搜索：

//点击搜索按钮搜索  
@FXML  
public void searchFile1(){  
 String fileName = searchText.getText();  
 if (!fileName.equals("")){  
 LinkedList<File> list = new LinkedList<>();  
 for (int i=0;i<fileList.size();i++){  
 if (fileList.get(i).getName().contains(fileName)){  
 list.add(fileList.get(i));  
 }  
 }  
 fileControl.updateUI(list);  
 }else {  
 fileControl.updateUI(fileList);  
 }  
}  
  
//回车搜索  
@FXML  
public void searchFile2(){  
 searchText.setOnKeyReleased(new EventHandler<KeyEvent>() {  
 @Override  
 public void handle(KeyEvent event) {  
 if(event.getCode() == KeyCode.*ENTER*){ //回车搜索  
 searchFile1();  
 }  
 }  
 });  
}

文件传输列表：

加载所有的传输列表：

//加载所有传输记录  
public void loadRecord(){  
 //初始化时读取所有的传输记录  
 try {  
 Socket client = new Socket("localhost",8123);  
 ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());  
 ObjectInputStream in = new ObjectInputStream(client.getInputStream());  
 Message transferMessage = new Message(user,MessageType.*TRANSFER*,"",0);  
 out.writeObject(transferMessage);  
 out.flush();  
 Message transferResult = (Message)in.readObject();  
 recordList = transferResult.getRecordList();  
 updateRecord(recordList);  
 //关闭连接  
 in.close();  
 out.close();  
 client.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

数据库查询出所有的传输记录：

//查询所有记录  
@Override  
public List<Transfer> getAllRecords(String username) {  
 QueryRunner runner = new QueryRunner(ConnectionPool.*getDataSource*());  
 List<Transfer> list = null;  
 try {  
 list = runner.query("select \* from transfer where username=?", new BeanListHandler<Transfer>(Transfer.class),username);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return list;  
}

//获取传输记录的方法  
case *TRANSFER*:{  
 System.*out*.println("服务器收到一条获取传输记录的消息");  
 List<Transfer> list = tfdao.getAllRecords(message.getFrom().getUsername());  
 LinkedList<Transfer> recordList = new LinkedList<>();  
 for (Transfer t:list){  
 recordList.add(t);  
 }  
 message.setRecordList(recordList);  
 out.writeObject(message);  
 out.flush();  
 break;  
}

添加或删除一条记录时页面的更新：

//更新传输列表的方法，用于初始化时以及上传或下载文件时更新传输列表  
public void updateRecord(LinkedList<Transfer> recordList) {  
 double y = 0;  
 scrollContent.getChildren().clear();  
 if(recordList!=null && recordList.size()>0){  
 for(Transfer t : recordList){  
 //文件图片  
 String suffix = t.getFilename().split("\\.")[t.getFilename().split("\\.").length-1];  
 ImageView fileImage = null;  
 try {  
 fileImage = new ImageView(new Image("images/" + suffix + ".png"));  
 }catch (Exception e){  
 fileImage = new ImageView(new Image("images/other.png"));  
 }  
 fileImage.setFitWidth(61);  
 fileImage.setFitHeight(65);  
 //文件名以及文件大小  
 VBox v = new VBox(5);  
 v.setAlignment(Pos.*TOP\_LEFT*);  
 v.setPadding(new Insets(7,0,5,10));  
 v.setPrefSize(336,66);  
 Label fileName = new Label(t.getFilename());  
 FileSizeUtil f = new FileSizeUtil();  
 Label fileLength = new Label(f.formatFileSize(t.getFileLength())); //转换文件大小  
 fileLength.setStyle("-fx-text-fill: #7c7a7a");  
 v.getChildren().addAll(fileName,fileLength);  
  
 //时间、操作类型以及按钮  
 HBox h = new HBox(22);  
 h.setAlignment(Pos.*CENTER*);  
 h.setPrefSize(397,66);  
 DateUtil d = new DateUtil();  
 Label time = new Label(d.getDate(t.getTime()));  
 time.setPrefWidth(90);  
 time.setStyle("-fx-text-fill: #7c7a7a");  
 Label type = new Label(t.getOperating()+"完成");  
 type.setStyle("-fx-text-fill: #7c7a7a");  
  
 //打开文件按钮  
 ImageView openPic = new ImageView(new Image("images/open.png"));  
 Button openBtn = new Button("",openPic);  
 openBtn.setStyle("-fx-background-color: white");  
 openBtn.setPrefSize(30,30);  
 openBtn.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 openFile(t.getFilepath());  
 }  
 });  
  
 //删除按钮  
 ImageView deletePic = new ImageView(new Image("images/delete.png"));  
 Button deleteBtn = new Button("",deletePic);  
 deleteBtn.setStyle("-fx-background-color: white");  
 deleteBtn.setPrefSize(30,30);  
 deleteBtn.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 deleteRecord(t.getId());  
 }  
 });  
 h.getChildren().addAll(time,type,openBtn,deleteBtn);  
 //存放每条完整的传输记录  
 HBox hBox = new HBox();  
 hBox.setPrefSize(788,66);  
 hBox.getChildren().addAll(fileImage,v,h);  
 hBox.setLayoutY(y);  
 y += 66;  
 scrollContent.getChildren().add(hBox);  
 }  
 }  
}

传输记录删除：

//删除文件传输记录的消息  
 case *DELETERECORD*:{  
 System.*out*.println("服务器收到一条删除传输记录的消息");  
 if(tfdao.deleteRecode(message.getFileLength())){  
 message.setFileName("true");  
 }else{  
 message.setFileName("false");  
 }  
 out.writeObject(message);  
 out.flush();  
 break;  
 }  
 default:{  
 break;  
 }  
}

*/\*\*  
 \* 删除一条记录  
 \** ***@param*** *id  
 \** ***@return*** *\*/*@Override  
public boolean deleteRecode(long id) {  
 boolean flag = false;  
 QueryRunner runner = new QueryRunner(ConnectionPool.*getDataSource*());  
 try {  
 int count = runner.update("delete from transfer where id=?",id);  
 flag = count>0?true:false;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return flag;  
}

文件本地打开：

//打开本地文件  
private void openFile(String filePath){  
 System.*out*.println("点击了打开文件的方法");  
 String path=filePath.replace("\\", "\\\\");  
 File file=new File(path);  
 Alert a = new Alert(Alert.AlertType.*INFORMATION*);  
 if (file.exists()) {  
 @SuppressWarnings("unused")  
 Process p;  
 try {  
 p = Runtime.*getRuntime*().exec("cmd /C " + path);  
 } catch (Exception e) {  
 a.setTitle("失败");  
 a.setContentText("无法打开文件");  
 a.show();  
 }  
 }else{  
 a.setTitle("失败");  
 a.setContentText("文件不存在或已被移动");  
 a.show();  
 }  
}

1. **项目总结：**

本次项目是对网络编程课程的一个综合考察，我们主要完成了用javafx，网络通讯实现的一个简单的百度网盘项目。在本次项目完成的过程中，我们首先把项目的整体框架打起来，例如首先对数据库进行了设计，把项目要用到的界面先写出来，然后对界面上的按钮进行无验证功能的简单条件。然后我们再通过数据库进行校验完善功能代码。在写这个项目的过程中，功能代码其实比较清晰和明确，但是由于对javafx知识的不够熟练导致页面渲染的时候遇到了一些问题，最终通过百度和请教其他同学得到了解决。本次项目我们小组三个成员均参与其中，最后这个版本时综合了我们三个人中写的最好最完善。此外，我们组还将继续研究断电续传以及极速秒传。