Politécnico de Coimbra

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Sistemas de Distribuídos

Ano Letivo 2024/25

Projeto Final – Servidor de partilha de ficheiros

Elaborado em: 28/06/2025

Rafael Ventura a2022131397

Índice

1 Lista de Figuras

2 Lista de Tabelas

3 Introdução

O presente relatório descreve o desenvolvimento do projeto final da unidade curricular, Sistemas Distribuídos, no âmbito da Licenciatura em Engenharia Informática. O objetivo principal deste trabalho consistiu na implementação de uma aplicação distribuída para troca de ficheiros entre utilizadores, utilizando as tecnologias Java RMI (*Remote Method Invocation*), Sockets e REST *web services*. A aplicação foi concebida para permitir a partilha e transferência de ficheiros de forma direta e eficiente ou através de um servidor intermediário.

O projeto seguiu as especificações fornecidas no enunciado, que incluíam a criação de uma interface gráfica para o cliente, autenticação de utilizadores com nomes únicos, atualização automática de listas de utilizadores e ficheiros, transferência de ficheiros com confirmação de substituição em caso de conflito, e registo de log's para monitorizar ações. A escolha pela tecnologia Java RMI deveu-se à sua transparência no acesso a objetos remotos, simplificando a comunicação entre clientes e servidor.

4 Trabalho desenvolvido

Para a realização da aplicação distribuída de troca de ficheiros recorri à tecnologia Java RMI para efetuar a comunicação entre clientes e servidor.

Java RMI é uma infraestrutura de comunicação baseada em RMI (*Remote Method Invocation*) capaz de executar métodos em objetos de uma outra JVM (*Java Virtual Machine*) (Afonso, 2024).

O motivo pelo qual optei por esta tecnologia foi pela da transparência de acesso a objetos remotos, utilizando a mesma sintaxe e semântica de programas Java (Afonso, 2024). Com isto tornaria a realização do projeto bastante mais simples.

Tabela 1 - Tabela de requisitos

Interface gráfica	Implementado
Selecionar pasta para partilha	Implementado
Introduzir dados do servidor (endereço IP e porto)	Implementado
Autenticação ao servidor (apenas com um nome único)	Implementado
Listagem de todos os utilizadores ligados ao servidor	Implementado
Atualização automática (Listagem de utilizadores, ficheiros e log's)	Implementado
Listagem de todos os ficheiros presentes na pasta partilhada	Implementado
Transferência de ficheiros	Implementado
Logout do servidor	Implementado
Mensagens de erro ou de sucesso	Implementado
Substituição do ficheiro	Implementado
Log's	Implementado

4.1 Interface gráfica do cliente

Para a criação da interface da interface gráfica do cliente, criei uma JFrame Form e recorri à ferramenta de edição visual do NetBeans que gera automaticamente o código JAVA necessário para desenvolver o que projetei no editor.

Relativamente à arquitetura do GUI (*Graphical User Interface*) utilizei os seguintes componentes Swing:

- JButton → Classe representante dos botões, estes permitem a chamada de métodos. Os métodos são acionados através do método "actionPerfomed" que pertence à interface JAVA ActionListener (Oracle America Inc., n.d.-a);
- JLabel → Classe utilizada para a escrita de texto informativo na GUI
- JList → Classe que permite a listagem dos "Vector's" de utilizadores, ficheiros (tanto do próprio utilizador como de outros) e log's (Oracle America Inc., n.d.-b);
- JOptionPane → Classe utilizada para a implementação das mensagens informativas ou de ação. Estas são implementadas através dos métodos "showMessageDialog()" e "showConfirmDialog()" da classe referida (Oracle America Inc., n.d.-c);
- JTextField → Classe utilizada para a introdução de texto por parte do cliente;
- JPanel → Classe que serve de *container* para agrupar os vários componentes respetivos a várias funcionalidades (log's, utilizadores, ficheiros, etc...);
- JScrollPane → Classe que permite que a listagem nas JList's tenham uma scrollbar. Para isso utilizamos o método "setViewPortView()" que define qual é o componente que será mostrado na viewport. Este é mais direto pois não precisamos de instanciar um objeto da classe JViewport (Oracle America Inc., n.d.-d).

Através desta pequena listagem de componentes, consegui desenvolver uma GUI simples, funcional, interativa e autoexplicativa (Figura 1).

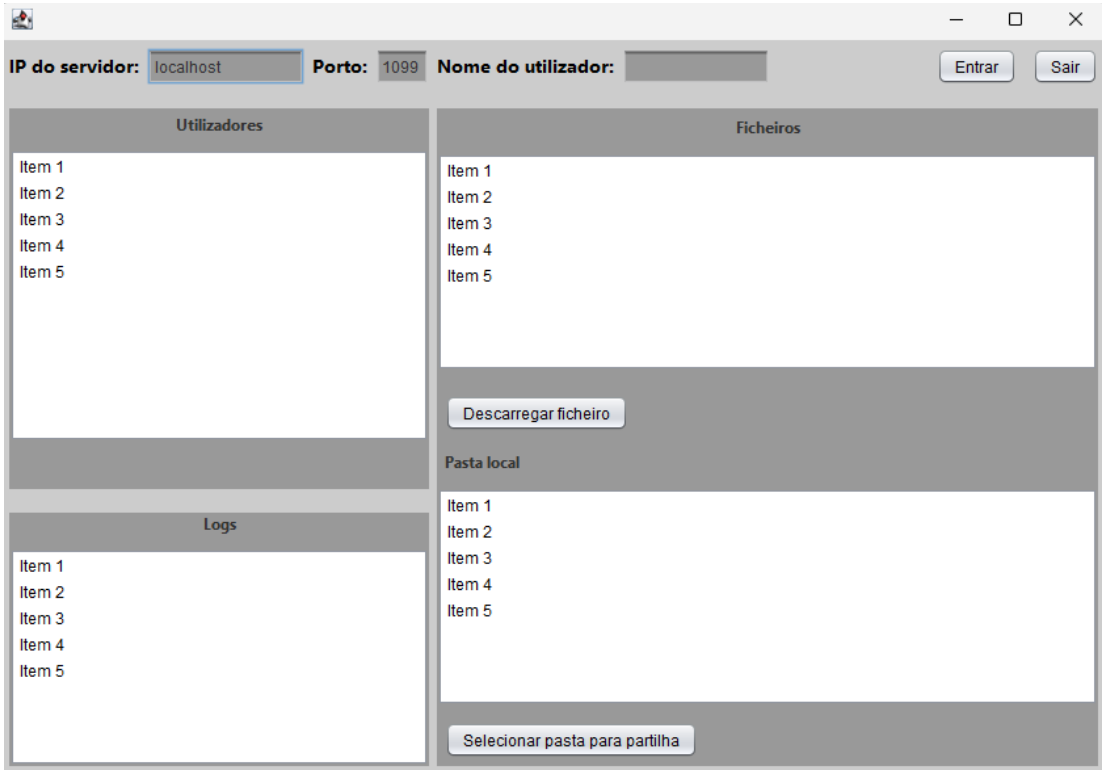

Figura 1 – Arquitetura da GUI

4.2 Autenticação

Para um utilizador entrar no servidor deverá passar pelas seguintes validações:

- IP do servidor: endereço IP, ou se estiver a rodar na mesma máquina o cliente e servidor, poderá ser apenas "localhost";
- Porto: deverá ser sempre o porto 1099, definido no lado do servidor "LocateRegistry.createRegistry(1099)";
- Nome do utilizador: este poderá ser qualquer nome, exceto, quando igual a outro cliente (*case sensitive*);

Na Figura 2 podemos visualizar um teste de autenticação bem-sucedido com o utilizador "Rafael".

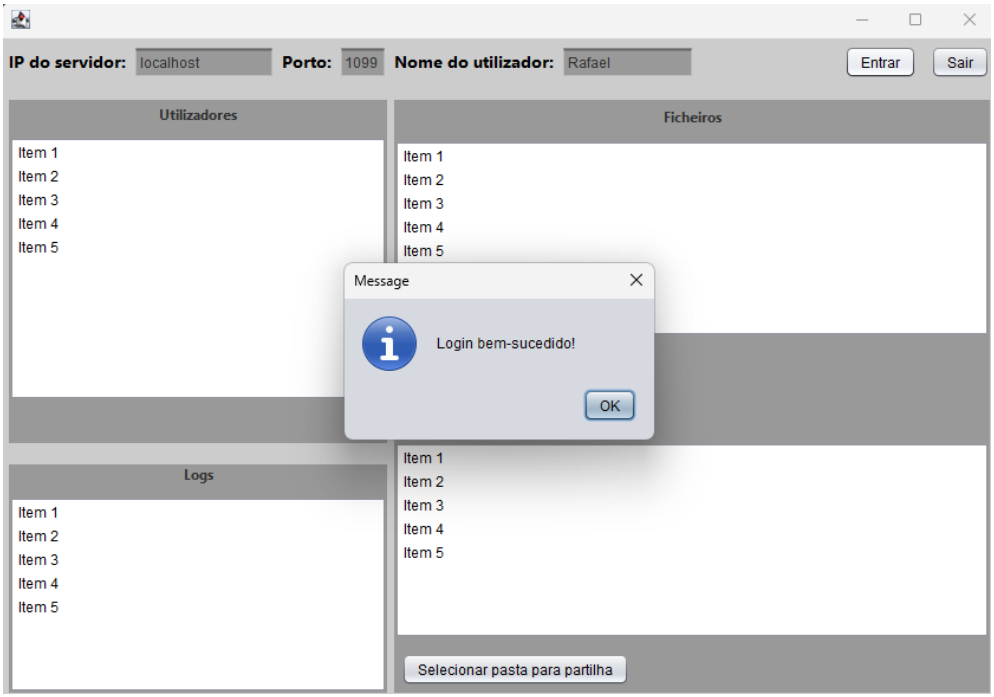

Figura 2 – Autenticação válida

Na Figura 3 foi provocado um erro na autenticação, neste caso a tentativa de autenticação de outro utilizador "rafael" e podemos ver que este não é autorizado a entrar no servidor, uma vez que o cliente "Rafael" encontra-se ligado ao mesmo.

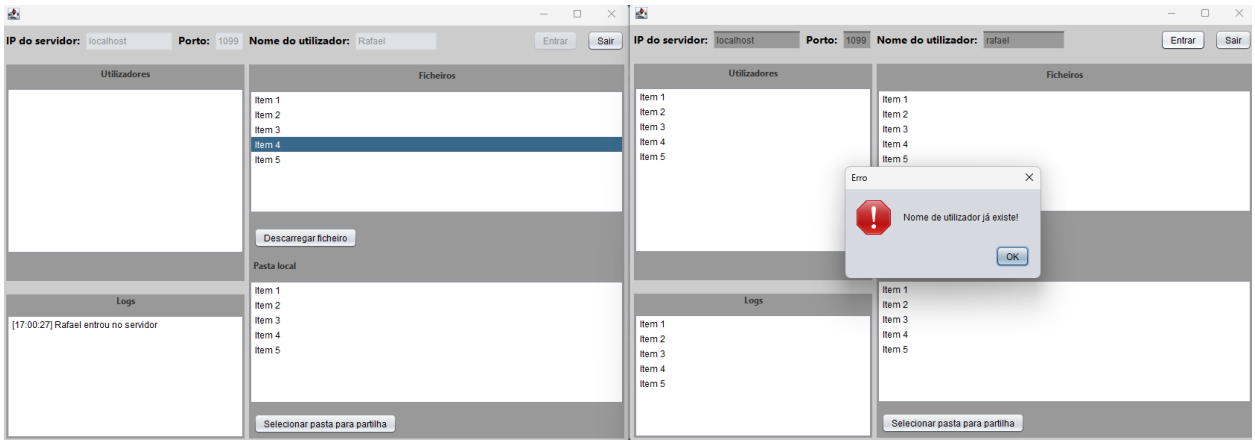

Figura 3 – Autenticação com nome já existente

A última validação efetuada na autenticação é apenas um tratamento de exceções lançadas quando ocorre um erro na conexão ao servidor. Situações em que estas acontecem, são em casos como um porto inválido, endereço IP diferente do servidor ou nome do utilizador "em branco" (Figura 4).

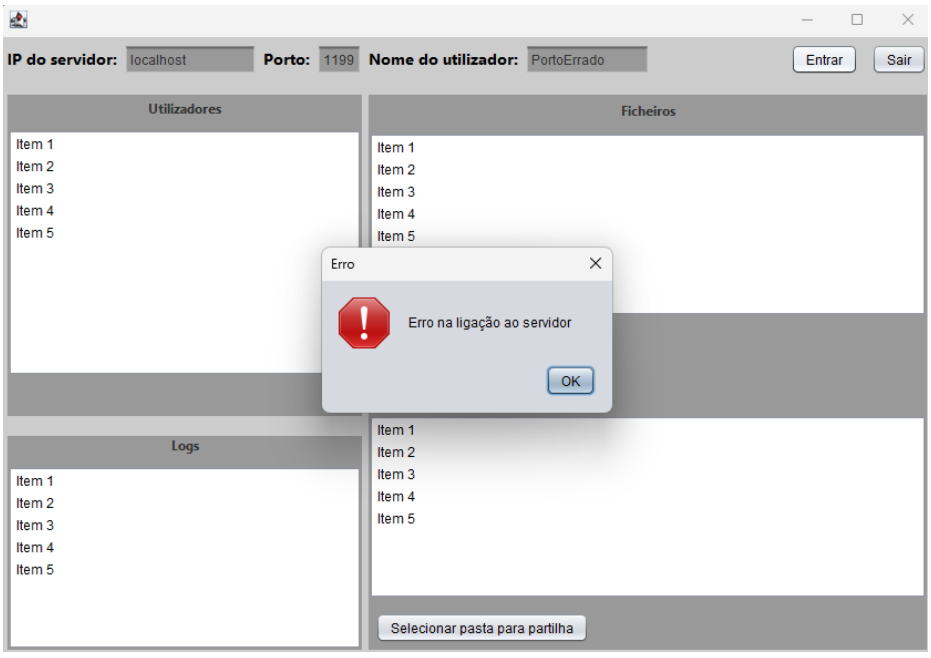

Figura 4 – Autenticação com conexão errada

4.3 Selecionar pasta local

Para selecionar uma pasta local, o cliente deverá clicar no botão “Selecionar pasta para partilha” e será chamado o componente JFileChooser onde ele poderá selecionar qual a pasta deseja partilhar (Figura 5).

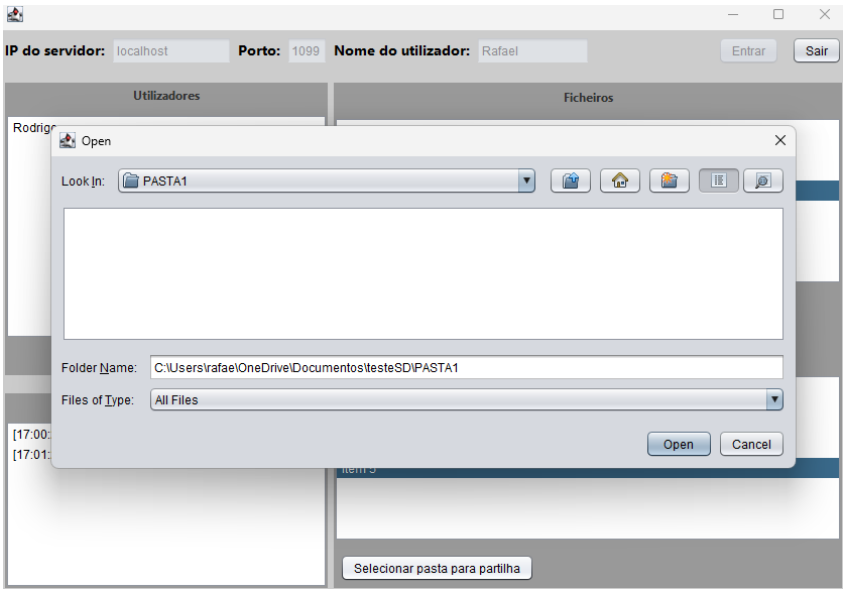

Figura 5 – Seleção de uma pasta para partilha

Após a seleção das pastas partilhadas dos dois clientes “Rafael” e “Rodrigo”, podemos listar quais os ficheiros que ambos estão a partilhar, clicando no respetivo utilizador, presente na listagem de utilizadores (Figura 6).

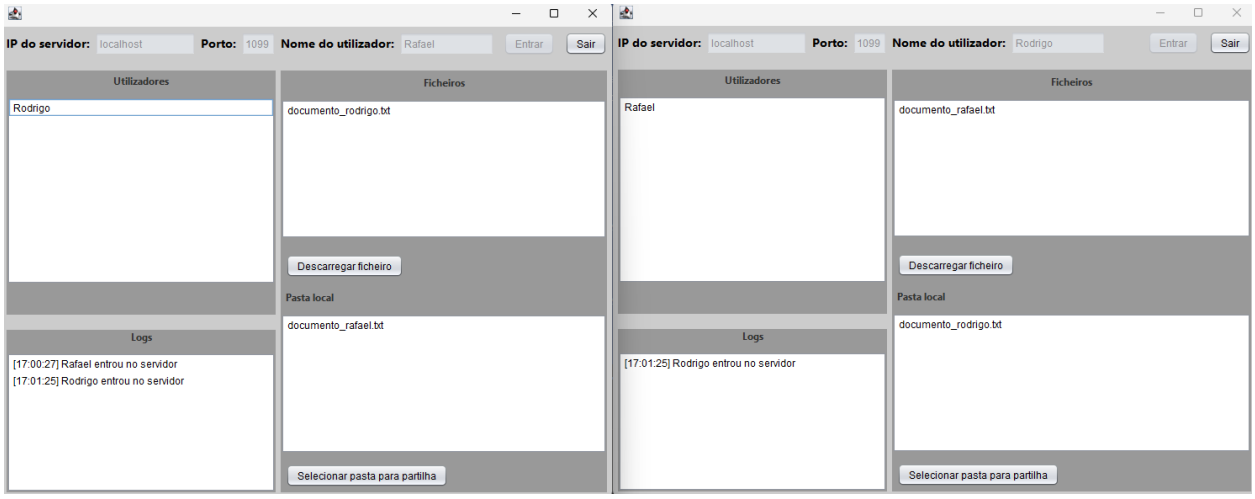

Figura 6 – Verificação da pasta partilhada

4.4 Transferência de ficheiros

A funcionalidade de transferência de ficheiros entre utilizadores é efetuada diretamente entre os servidores, um utilizador ao selecionar outro cliente, de seguida selecionar um ficheiro e clicar em "Descarregar ficheiro" o método "downloadFile()" será chamado. Este irá passar como parâmetro o utilizador selecionado e o nome do ficheiro escolhido, depois será verificada a existência desse ficheiro, caso este exista será escrito na pasta partilhada pelo cliente que efetuo o download (Figura 7).

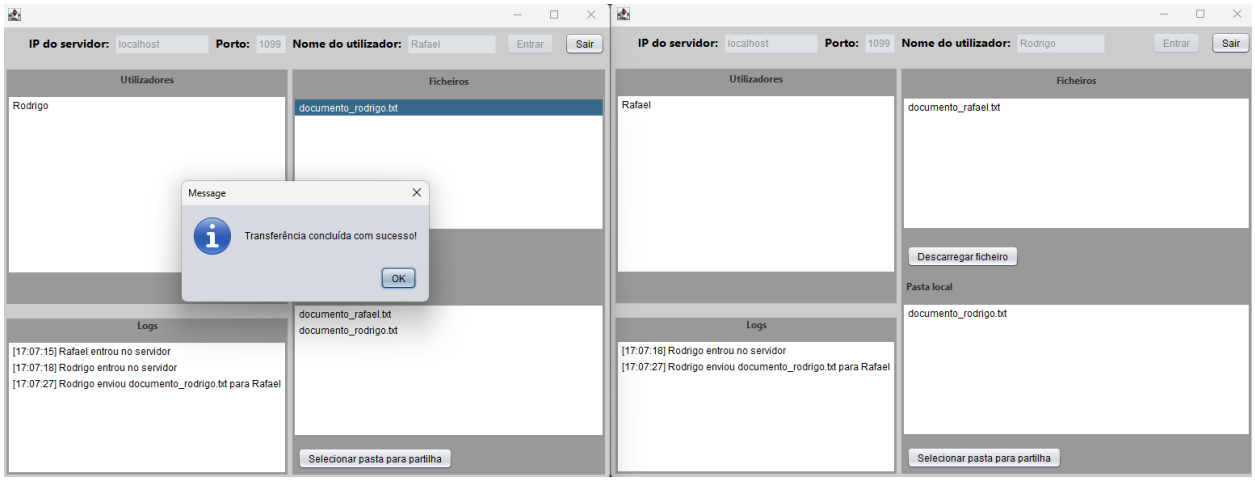

Figura 7 – Transferência de um ficheiro

Na Figura 8 podemos verificar uma validação intermédia na transferência de ficheiros. Esta verifica se o ficheiro já existe na pasta partilhada do utilizador, caso exista será mostrada uma JOptionPane para a confirmação ou não, da substituição do ficheiro.

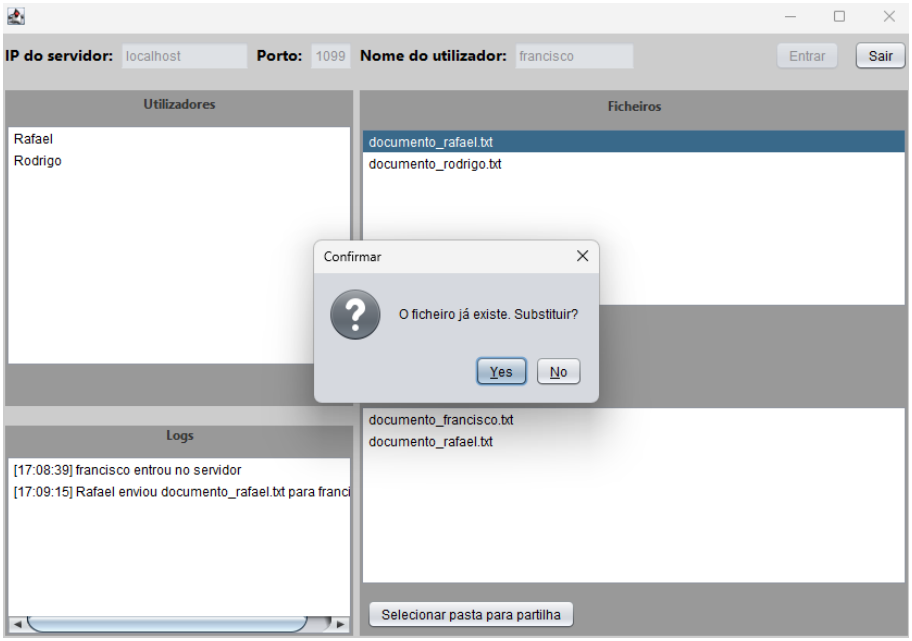

Figura 8 – Substituição de ficheiro existente

Caso o utilizador pretenda prosseguir a transferência, e durante o processo da mesma não surgirem erros, será mostrada uma mensagem informativa com o sucesso da transferência do ficheiro (Figura 9).

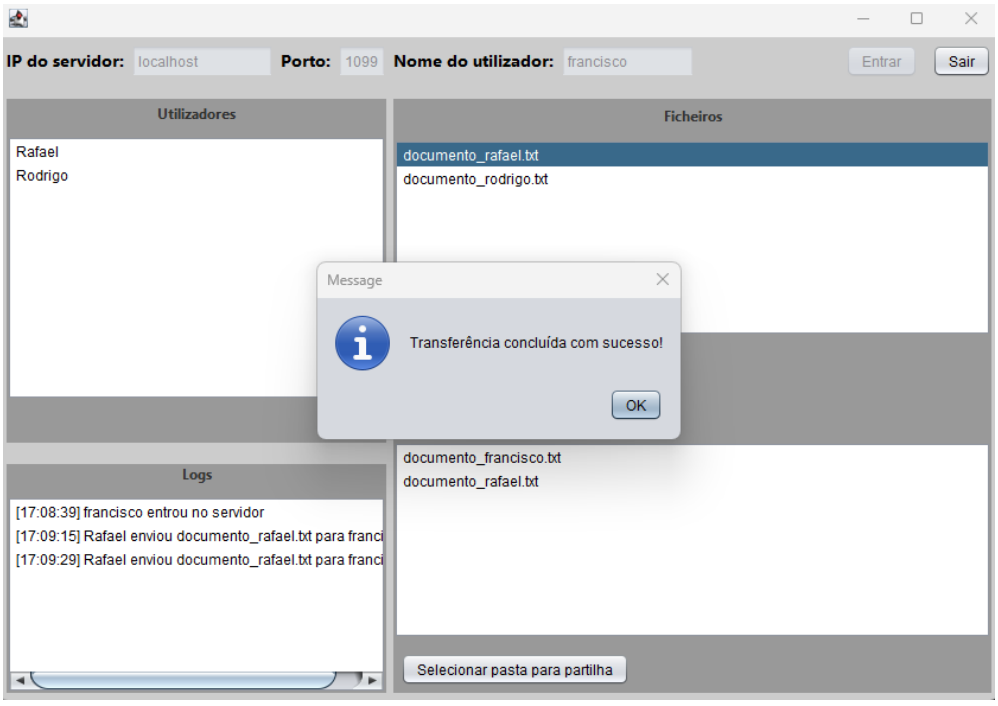

Figura 9 – Comfirmação da substituição do ficheiro substituído

4.5 Log's

A listagem dos log's é diferente para cada utilizador, isto é, quando um utilizador se autêntica no servidor, é guardado o instante de tempo em que isso aconteceu. Este instante de tempo é enviado ao servidor quando o cliente efetua a listagem dos log's e é no servidor que os log's serão filtrados. A filtragem dos mesmos é simples, apenas verifica se o instante de tempo de cada log é superior ou igual ao instante de tempo da autenticação do utilizador, caso seja, adiciona e retorna os log's filtrados ao cliente.

Na Figura 10 podemos ver que o utilizador "Rafael" saiu e voltou a entrar no servidor e todos os log's anteriores não foram novamente mostrados ao mesmo. Enquanto isso o utilizador "francisco" que permaneceu sempre ligado ao servidor, verificou as ações de saída e entrada do "Rafael" no servidor.

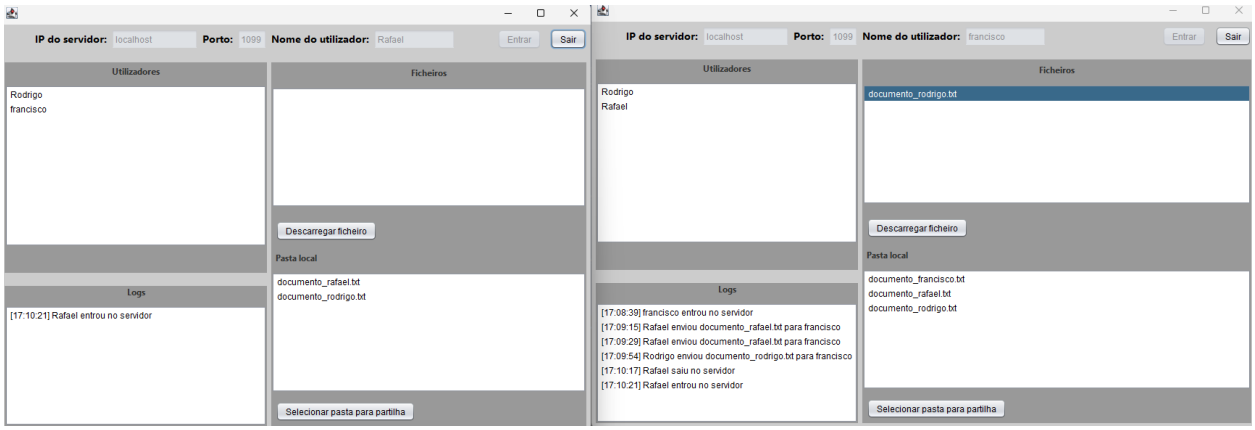

Figura 10 – Listagem dos log's

4.6 Atualização automática

Para a atualização automática da interface (listagem de utilizadores, ficheiros e log's) utilizei a classe Timer, que permite que *thread* possam a calendarizar tarefas para serem executadas em background(Oracle America Inc., n.d.-f).

Com o uso do método "scheduleAtFixedRate" agendei a atualização automática (TimerTask) para ser repetida a cada três segundos(Oracle America Inc., n.d.-g, n.d.-f; Paraschiv, 2024; Tutorials Point India Limited, n.d.).

Um aspeto importante de referir é que sempre que para executar as alterações na interface é necessário (recomendado) colocar essas alterações no método "invokeLater()" da classe SwingUtilities, isto porque o objeto da classe Timer executa as tarefas numa *thread* própria e nas aplicações que usam Swing a *thread* EDT (Event Dispatch Thread) é responsável por efetuar alterar a interface. O método referido anteriormente limita-se a colocar essa tarefa numa fila para ser executada pela *thread* EDT(Oracle America Inc., n.d.-e).

5 Conclusão

O desenvolvimento deste projeto permitiu consolidar os conhecimentos teóricos e práticos sobre sistemas distribuídos, em particular no que diz respeito à comunicação remota entre processos utilizando Java RMI. A aplicação implementada cumpre com 100% os requisitos especificados na Tabela 1, oferecendo uma solução funcional e intuitiva para a troca de ficheiros entre utilizadores. Entre as funcionalidades destacam-se a interface gráfica amigável, a autenticação simples mas eficaz, a transferência de ficheiros com confirmação de substituição, e o registo detalhado de log's para monitorizar as várias operações.

Inicialmente, senti alguma dificuldade na compreensão da tecnologia Java RMI, mas após seguir o exemplo disponível do professor, juntamente com a pesquisa de outras informações, julgo que a escolha por esta tecnologia mostrou-se acertada, dada a sua simplicidade e eficiência na invocação remota de métodos.

Em suma, este projeto não apenas cumpriu os objetivos propostos, mas também proporcionou uma valiosa experiência prática no desenvolvimento de sistemas distribuídos. Como trabalho futuro, poderia ser explorada a integração das outras tecnologias (Sockets e REST *web services*).

6 Referências

Afonso, F. (2024). *Java RMI*.

Oracle America Inc. (n.d.-a). *ActionListener (Java Platform SE 8)*. Retrieved June 27, 2025, from https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionListener.html

Oracle America Inc. (n.d.-b). *JList (Java Platform SE 8)*. Retrieved June 27, 2025, from https://docs.oracle.com/javase/8/docs/api/javax/swing/JList.html

Oracle America Inc. (n.d.-c). *JOptionPane (Java Platform SE 8)*. Retrieved June 27, 2025, from https://docs.oracle.com/javase/8/docs/api/javax/swing/JOptionPane.html

Oracle America Inc. (n.d.-d). *JScrollPane (Java Platform SE 8)*. Retrieved June 27, 2025, from https://docs.oracle.com/javase/8/docs/api/javax/swing/JScrollPane.html

Oracle America Inc. (n.d.-e). *SwingUtilities (Java Platform SE 8)*. Retrieved June 27, 2025, from https://docs.oracle.com/javase/8/docs/api/javax/swing/SwingUtilities.html#invokeLater-java.lang.Runnable-

Oracle America Inc. (n.d.-f). *Timer (Java Platform SE 8)*. Retrieved June 27, 2025, from https://docs.oracle.com/javase/8/docs/api/java/util/Timer.html

Oracle America Inc. (n.d.-g). *TimerTask (Java Platform SE 8)*. Retrieved June 27, 2025, from https://docs.oracle.com/javase/8/docs/api/java/util/TimerTask.html

Paraschiv, E. (2024, January 8). *Java - Timer | Baeldung*. https://www.baeldung.com/java-timer-and-timertask

Tutorials Point India Limited. (n.d.). *Java Timer scheduleAtFixedRate Method*. Retrieved June 27, 2025, from https://www.tutorialspoint.com/java/util/timer_scheduleatfixedrate.htm