



Universidade do Minho  
Escola de Engenharia

# Universidade do Minho

## Licenciatura em Engenharia Informática

### Unidade Curricular de

# Sistemas Distribuídos

2024/2025

Trabalho realizado por:

Alexandre Miranda a104445

André Carvalho a100818

Miguel Barbosa a104451

Oleksii Tantsura a102131

<b>1. Introdução.....</b>	<b>3</b>
<b>2. Arquitetura do sistema.....</b>	<b>3</b>
<b>3. Comunicação cliente-servidor.....</b>	<b>3</b>
<b>4. Implementação do servidor.....</b>	<b>4</b>
<b>5. Implementação do cliente.....</b>	<b>5</b>
<b>6. Testes e avaliação de desempenho.....</b>	<b>5</b>
<b>7. Conclusões e trabalho futuro.....</b>	<b>5</b>

# 1. Introdução

Este trabalho prático foca-se na implementação de um serviço de armazenamento de dados partilhado em memória, acessível remotamente, que utiliza uma arquitetura cliente-servidor. Este sistema tem como objetivo demonstrar conceitos fundamentais de sistemas distribuídos, como concorrência, comunicação por sockets e operações atômicas.

O projeto visa implementar um serviço de armazenamento no formato chave-valor, no qual a informação é mantida em memória e disponibilizada através de uma interface remota. A interação entre clientes e servidor ocorre via sockets TCP, garantindo a comunicação eficiente e segura. O servidor suporta múltiplos clientes simultâneos, atendendo-os de forma concorrente, e integra funcionalidades básicas e avançadas, como operações compostas e controle de concorrência, para maximizar desempenho e escalabilidade.

Este relatório vai seguir a seguinte estrutura:

- Arquitetura do sistema -> Breve descrição da arquitetura que foi desenvolvida tendo em conta a comunicação cliente-servidor e o armazenamento em memória
- Comunicação cliente-servidor -> Como foi estabelecida a comunicação entre cliente e servidor e como são transmitidos os dados e as mensagens
- Implementação do servidor -> Funcionalidades apresentadas pelo servidor
- Implementação do cliente -> Funcionalidades do cliente
- Testes e avaliação de desempenho -> apresentação dos testes realizados e análise dos resultados obtidos
- Conclusões e trabalho futuro -> Aspectos a conter do trabalho realizado e possíveis implementações futuras

## 2. Arquitetura do sistema

O sistema desenvolvido está dividido em 5 classes, sendo estas a classe Servidor que estabelece a conexão com o cliente, criando depois uma nova thread para cada cliente que é adicionado e gere as conexões existentes, a classe Cliente que verifica se o cliente é válido e depois recebe as operações que o cliente pretende efetuar e os respectivos dados, a classe ClienteHandler que verifica se o user existe na base de dados e processa os comandos enviados pelos clientes, a classe ServidorArmazenamento que armazena as informações que os clientes enviarem e permite uma leitura dos mesmos, a classe ServidorUsuario que regista novos users e permite fazer o login dos já existentes.

## 3. Comunicação cliente-servidor

A comunicação começa no cliente com a tentativa de comunicação com o servidor enviando a operação a efetuar (login ou registo) e os seus username e password. O

servidor cria um thread para cada cliente recebido e quando houver disponibilidade, isto é, quando o número de clientes a comunicar com o servidor não estiver no máximo, conecta-se com o cliente e recebe estes parâmetros para validar a autenticação com base no que recebe. Após estar com o login feito, o cliente pode selecionar que operação pretende realizar (leitura ou escrita) e de que tipo (simples ou múltipla ou condicional no caso da leitura). Esta seleção é feita localmente para reduzir o número de mensagens transmitidas entre o cliente e o servidor, sendo enviada apenas a escolha final do cliente.

Após esta seleção, o cliente envia os dados necessários para a operação selecionada, por exemplo, a chave para leitura ou a chave e o valor para escrita. O servidor vai receber essa informação e armazenar em memória se for uma operação de escrita ou consultar a informação usando a chave e transmitindo ao cliente.

Toda a comunicação que ocorre entre o cliente e o servidor é feita em formato binário, usando as funções de write e de read de Data[Input|Output]Stream, que transformam em binário antes de serem enviadas e transformam no formato que era antes de ser passado para binário ao chegar ao servidor, fazendo com que a comunicação entre cliente e servidor seja feita em formato binário. Para a escrita, foram utilizadas as funções writeUTF para enviar as Strings, writeInt para enviar o comprimento dos arrays de bytes e write para enviar os arrays de bytes. Para a leitura foram utilizadas as funções readUTF para ler Strings, readInt para ler o tamanho dos arrays de bytes e readFully para ler os arrays de bytes.

Quando o cliente deseja terminar a comunicação, uma das operações que este pode selecionar é sair que termina a conexão entre o cliente e o servidor. Para evitar que o servidor fique bloqueado com clientes que não efetuam operações, foi definido um timeout para que, se o cliente não fizer nenhuma operação durante um certo intervalo de tempo, este seja desconectado automaticamente, permitindo assim que outros clientes se conectem e efetuem as operações desejadas.

## 4. Implementação do servidor

O servidor está subdividido em 4 classes, que são Servidor, responsável pela gestão das conexões, ClienteHandler, responsável pela comunicação com o cliente, ServidorArmazenamento, responsável pelo armazenamento da informação enviada pelos clientes, ServidorUsuarios, responsável pela validação dos users.

A classe Servidor é composta pelas funções gerenciarConexao que verifica a possibilidade de efetuar a conexão, main que aceita a conexão do cliente e cria uma thread para que ele possa efetuar operações.

A classe ServidorUsuarios é composta pelas funções loadUsers que adiciona um novo user, saveUsers, registerUser, authenticate que verifica se a password corresponde ao username.

A classe `ServidorArmazenamento` é composta pelas funções `getArmazenamento` que devolve os dados armazenados, `getLock` que devolve o lock para a memória partilhada, `getDataChangedCondition` que devolve a condition para a leitura condicional.

A classe `ClienteHandler` é composta pelas funções `processarComandos` que recebe os argumentos do cliente e efetua as operações escolhidas por este, `autenticarCliente` que valida se o user está registado ou não.

## 5. Implementação do cliente

O cliente está subdividido em 2 classes, que são `Cliente`, responsável pela gestão da conexão, `OperationHandler`, responsável pela comunicação com o servidor.

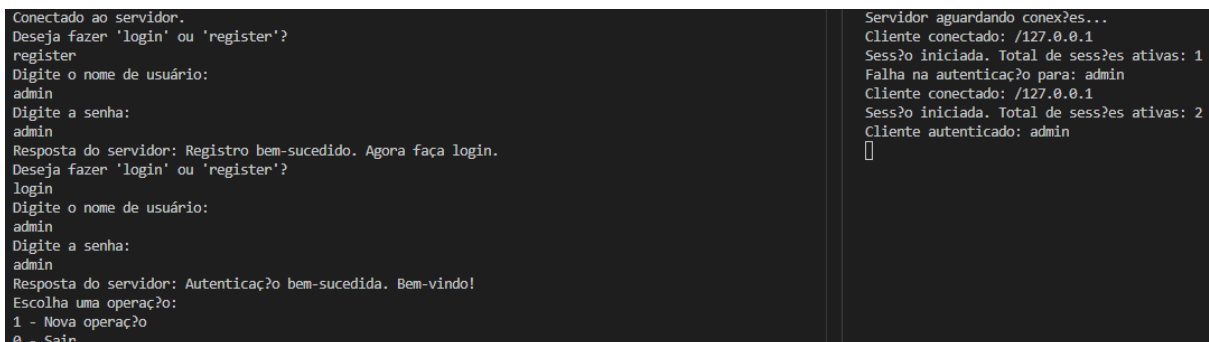
A classe `Cliente` tem a função `main` que recebe os dados do cliente e o regista ou faz o seu login se esses dados forem válidos.

A classe `OperationHandler` tem a função `run` que, com base na escolha da operação por parte do cliente, trata do envio e/ou receção de dados com o servidor.

A classe `OperationHandler` contém funções como `put`, `multiPut`, `get`, `multiGet` e `getWhen`, além da função `run`, que, com base na escolha da operação por parte do cliente, trata do envio e/ou receção de dados com o servidor:

1. `Put`: Insere um par chave-valor único no sistema.
2. `MultiPut`: Realiza a inserção de múltiplos pares chave-valor de uma vez, otimizando operações em lote.
3. `Get`: Recupera o valor associado a uma única chave específica.
4. `MultiGet`: Permite consultar múltiplas chaves em uma única operação, retornando todos os pares chave-valor encontrados.
5. `GetWhen`: Efetua uma leitura condicional, onde o valor de uma chave é recuperado somente se uma condição sobre outra chave (`keyCond`) for atendida.

## 6. Testes e avaliação de desempenho



```
Conectado ao servidor.
Deseja fazer 'login' ou 'register'?
register
Digite o nome de usuário:
admin
Digite a senha:
admin
Resposta do servidor: Registro bem-sucedido. Agora faça login.
Deseja fazer 'login' ou 'register'?
login
Digite o nome de usuário:
admin
Digite a senha:
admin
Resposta do servidor: Autenticação bem-sucedida. Bem-vindo!
Escolha uma operação:
1 - Nova operação
0 - Sair
```

```
Servidor aguardando conexões...
Cliente conectado: /127.0.0.1
Sessão iniciada. Total de sessões ativas: 1
Falha na autenticação para: admin
Cliente conectado: /127.0.0.1
Sessão iniciada. Total de sessões ativas: 2
Cliente autenticado: admin
[]
```

Figura 1: Processo de Registo e Login

```
Escolha o tipo de operação (1 - Escrita, 2 - Leitura):
1
1 - Escrita simples
2 - Escrita múltipla
2
Digite a quantidade de mensagens:
4
Digite a chave:
Teste1
Digite o valor:
Valor1
Digite a chave:
Teste2
Digite o valor:
valor2
Digite a chave:
teste3
Digite o valor:
valor3
Digite a chave:
teste4
Digite o valor:
valor4
Escolha uma operação:
1 - Nova operação
0 - Sair

Sessão encerrada. Total de sessões ativas:
Cliente conectado: /127.0.0.1
Sessão iniciada. Total de sessões ativas: 1
Cliente autenticado: admin
Escrita: [1] = a
Escrita: [2] = b
Escrita: [3] = c
Escrita: [4] = d
Escrita: [Teste1] = Valor1
Escrita: [Teste2] = valor2
Escrita: [teste3] = valor3
Escrita: [teste4] = valor4
[]
```

Figura 2: Processo de escrita múltipla

```
Escolha o tipo de operação (1 - Escrita, 2 - Leitura):
2
1 - Leitura simples
2 - Leitura múltipla
3 - Leitura condicional
2
Quantas chaves deseja consultar?
3
Digite a chave:
Teste1
Digite a chave:
teste3
Digite a chave:
teste4
Número de pares encontrados: 3
Chave: Teste1, Valor: Valor1
Chave: teste4, Valor: valor4
Chave: teste3, Valor: valor3
Escolha uma operação:
1 - Nova operação
0 - Sair
```

Figura 3: Processo de leitura múltipla

A figura 1 ilustra todo o processo de conexão e autenticação no servidor. À direita temos a visão do utilizador que interage com o menu e à direita temos a consola do servidor durante esse processo. Caso o utilizador tente dar login com uma conta inexistente ele recebe um feedback do servidor.

A figura 2 representa o processo de escrita múltipla em que o utilizador dá o número de chaves que quer escrever e depois introduz as chaves e os valores que vão ser escritos no servidor.

A figura 3 ilustra o processo de leitura múltipla onde o utilizador fornece mais uma vez o número e as chaves que pretende ler e depois obtém os respectivos valores do servidor.

## 7. Conclusões e trabalho futuro

O desenvolvimento deste sistema de armazenamento de dados partilhado em memória demonstrou a importância de conceitos fundamentais em sistemas distribuídos, como concorrência, comunicação eficiente e operações atômicas. Através da arquitetura cliente-servidor implementada, foi possível criar um serviço funcional, capaz de atender múltiplos clientes simultaneamente, enquanto mantinha um desempenho consistente e satisfatório.

No entanto, há espaço para melhorias e extensões futuras:

- **Persistência de dados:** Adicionar um mecanismo de armazenamento persistente, permitindo que os dados sejam recuperados após o reinício do servidor.
- **Segurança:** Implementar criptografia nas comunicações e reforçar a autenticação para proteger os dados contra acessos mal-intencionados.
- **Melhoria na interface do cliente:** Criar uma interface mais amigável para facilitar a interação com o serviço.