1. Protocolo

O presente trabalho tem como objetivo evoluir a arquitetura de comunicação entre os dispositivos **WAVY**, os **Agregadores** e o **Servidor**, anteriormente baseada em comunicação direta via TCP, para uma **arquitetura distribuída mais robusta e escalável**, utilizando novas tecnologias e paradigmas de comunicação.

Nesta nova versão do sistema, a comunicação entre os diferentes componentes passa a ser realizada por meio de dois modelos principais:

- **RPC (Remote Procedure Call)** para ligação entre agregadores e serviços de pré-processamento, bem como entre o servidor e serviços de análise de dados;
- **Publicação/Subscrição (Pub/Sub)**, baseado em **RabbitMQ**, para a comunicação entre as WAVYs e os Agregadores.

Arquitetura Geral

A arquitetura proposta segue o seguinte fluxo de comunicação:

1. As **WAVYs** enviam dados de sensores (em formatos como texto, CSV, XML ou JSON) e **publicam** os mesmos em tópicos específicos no broker RabbitMQ.
2. Os **Agregadores** estão **subscritos** aos tópicos de interesse, recebem os dados publicados pelas WAVYs e realizam um pré-processamento dos mesmos através de chamadas **RPC**.
3. Os dados pré-processados são depois enviados ao **Servidor**, que pode realizar chamadas **RPC** a servidores externos para análise estatística ou deteção de padrões.
4. O **Servidor** guarda os dados recebidos e os resultados das análises numa base de dados, possibilitando a sua visualização e análise posterior através de uma interface web.

Interface e Persistência

- Todos os dados e resultados de análises são armazenados numa **base de dados** (Microsoft SQL Server).
- A interface permite aos utilizadores:
    - Consultar dados e resultados;
    - Iniciar novas análises, parametrizando os critérios (por ex. temperatura e profundidade).
    - O utilizador poderá também adicionar uma nova WAVY manualmente.

2. Implementação

```
//mandar WavyID

await channel.QueueDeclareAsync(queue: "ID", durable: false, exclusive: false, autoDelete: false,
    arguments: null);

Console.WriteLine("Que tipo de Dados queres enviar: (1 - Manipulação de Strings) / (2 - Ficheiro TXT) / (3 - Ficheiro CSV) / (4 - Ficheiro JSON)");
string resposta = Console.ReadLine();
if (resposta == "1")
{
    string message3 = "WAVY_ID[1]:Manipulação_Strings:200MB:127.0.0.1:" + numeroAleatorioTemp + ":" + numeroAleatorioOndas + ":" + numeroAleatorioAlturaOndas + ":" + numeroAleatorioProfundidade + ":" + DATA + ":" + EstadoFinal;
    Console.WriteLine(message3);
    var body3 = Encoding.UTF8.GetBytes(message3);

    await channel.BasicPublishAsync(exchange: string.Empty, routingKey: "ID", body: body3);
    Console.WriteLine($" [x] Sent {message3}");
```

Na seguinte imagem, foi denominada um queue em RabbitMQ chamada por ID onde cada wavy vai enviar os seus dados para o agregador.

Henrique Dias 78804
Rafael Silva 78532
Pedro Afonso 77181

```
//receber ID

await channel.QueueDeclareAsync(queue: "ID", durable: false, exclusive: false, autoDelete: false,
    arguments: null);

var consumer3 = new AsyncEventingBasicConsumer(channel);
consumer3.ReceivedAsync += (model, ea) =>
{
    var body3 = ea.Body.ToArray();
    var message3 = Encoding.UTF8.GetString(body3);
    byte[] resposta = Encoding.UTF8.GetBytes(message3);
    string[] IDcompleto = message3.Split(":");
    var temp = Convert.ToInt64(IDcompleto[4]);
    var VelOndas = Convert.ToInt64(IDcompleto[5]);
    var alturaOndas = Convert.ToInt64(IDcompleto[6]);
    var profundidade = Convert.ToInt64(IDcompleto[7]);
    var date = Convert.ToString(IDcompleto[8]);
    string estado = IDcompleto[9];

    if (IDcompleto[1] == "Manipulação_Strings")
    {
        Console.WriteLine($" [x] Received ID:{message3}");
        Console.WriteLine("Resposta RPC: [ID] -> " + replyIDStrings +
            " [Pre_Processamento] -> " + replyIDProcessamento +
            " [Volume_Dados] -> " + replyIDDados +
            " [Servidor] -> " + replyIDServidor +
            " [Temperatura] -> " + temp +
            " [Velocidade de Ondas] -> " + VelOndas +
            " [Altura de Ondas] -> " + alturaOndas +
            " [Profundidade] -> " + profundidade +
            " [Data] -> " + date +
            " [Estado] -> " + estado);
```

Aqui o agregador recebe por RabbitMQ todos os dados da wavy onde estes vão ser pré-processados num servidor externo e enviados novamente para o agregador por RPC.

```
7       // The greeting service definition.
8     service Greeter {
9       // Sends a greeting
10      rpc SayHello (HelloRequest) returns (HelloReply);
11      rpc MandarID (wavyID) returns (wavyIDReply);
12    }
13
14    // The request message containing the user's name.
15    message HelloRequest {
16      string name = 1;
17    }
18
19    // The response message containing the greetings.
20    message HelloReply {
21      string message = 1;
22    }
23
24    message wavyID {
25        string ID = 1;
26        string Pre_Processamento = 2;
27        string Volume_Dados_Enviar = 3;
28        string Servidor_Associado = 4;
29    }
30
31    message wavyIDReply {
32        string IDrecebida = 1;
33        string Pre_ProcessamentoRecebido = 2;
34        string Volume_Dados_Recebido = 3;
35        string Servidor_Associado = 4;
36    }
```

Em RPC vão ser pré-processados os dados mais importantes que provêm da wavy, ou seja, o seu o ID, o tipo de dados, o volume de dados que o agregador recebeu e para que servidor vão ser enviados.

Henrique Dias 78804
Rafael Silva 78532
Pedro Afonso 77181

```
//recebe wavy id com pre processamento
byte[] wavyProcessamento = new byte[1024];
int wavyProcessamentoBytes = clientSocket.Receive(wavyProcessamento);
string wavyProcessamentoDados = Encoding.UTF8.GetString(wavyProcessamento, 0, wavyProcessamentoBytes);
string[] wavyProcessamentoFinal = wavyProcessamentoDados.Split(":");
string wavyProcessamentoSQL = wavyProcessamentoFinal[1];
var temperatura = Convert.ToInt64(wavyProcessamentoFinal[4]);
var velocidadeOndas = Convert.ToInt64(wavyProcessamentoFinal[5]);
var alturaOndas = Convert.ToInt64(wavyProcessamentoFinal[6]);
var profundidade = Convert.ToInt64(wavyProcessamentoFinal[7]);
var date = wavyProcessamentoFinal[8];
var estado = wavyProcessamentoFinal[9];
Console.WriteLine($"{wavyProcessamentoDados}");
```

Depois dos dados serem pré-processados por um serviço externo, o servidor vai recebê-los por protocolo TCP do agregador.

```
string connectionString = "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=SD;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Server Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False";


using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();

    string query = "INSERT INTO DadosAgregador (IPWavy, WavyProcessamento, Temperatura, Velocidade_Ondas, Altura_Ondas, Profundidade, Data_Dados, Estado) " +
                   "VALUES (@IPWavy, @WavyProc, @Temp, @VelOndas, @AlturaOndas, @Profundidade, @Data, @Estado)";

    using (SqlCommand command = new SqlCommand(query, connection))
    {
        command.Parameters.AddWithValue("@IPWavy", ipwavydados.Trim());
        command.Parameters.AddWithValue("@WavyProc", wavyProcessamentoSQL.Trim());
        command.Parameters.AddWithValue("@Temp", temperatura);
        command.Parameters.AddWithValue("@VelOndas", velocidadeOndas);
        command.Parameters.AddWithValue("@AlturaOndas", alturaOndas);
        command.Parameters.AddWithValue("@Profundidade", profundidade);
        command.Parameters.AddWithValue("@Data", date);
        command.Parameters.AddWithValue("@Estado", estado);

        Console.WriteLine("Dados inseridos na base de dados.");
```

Depois dos dados serem recebidos o servidor vai enviar para uma base de dados em Microsoft SQL Server onde estes vão ser armazenados.

< Sistemas Distribuídos >

Tabela de Dados WAVY

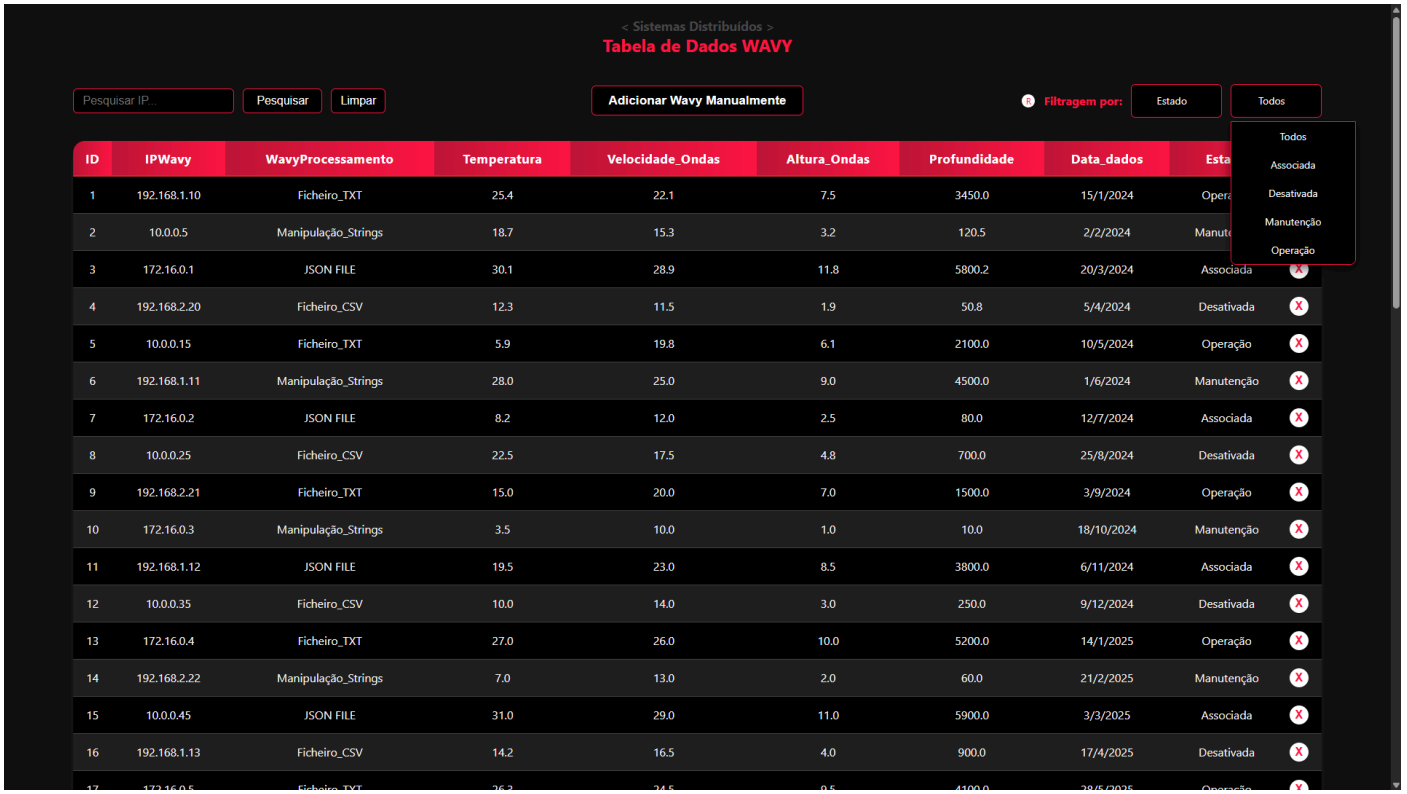
Pesquisar IP... | Pesquisar | Limpar | Adicionar Wavy Manualmente | Filtragem por: Estado | Todos

Todos / Associada / Desativada / Manutenção / Operação

ID	IPWavy	WavyProcessamento	Temperatura	Velocidade_Ondas	Altura_Ondas	Profundidade	Data_dados	Esta
1	192.168.1.10	Ficheiro_TXT	25.4	22.1	7.5	3450.0	15/1/2024	Opera
2	10.0.0.5	Manipulação_Strings	18.7	15.3	3.2	120.5	2/2/2024	Manut
3	172.16.0.1	JSON FILE	30.1	28.9	11.8	5800.2	20/3/2024	Associada
4	192.168.2.20	Ficheiro_CSV	12.3	11.5	1.9	50.8	5/4/2024	Desativada
5	10.0.0.15	Ficheiro_TXT	5.9	19.8	6.1	2100.0	10/5/2024	Operação
6	192.168.1.11	Manipulação_Strings	28.0	25.0	9.0	4500.0	1/6/2024	Manutenção
7	172.16.0.2	JSON FILE	8.2	12.0	2.5	80.0	12/7/2024	Associada
8	10.0.0.25	Ficheiro_CSV	22.5	17.5	4.8	700.0	25/8/2024	Desativada
9	192.168.2.21	Ficheiro_TXT	15.0	20.0	7.0	1500.0	3/9/2024	Operação
10	172.16.0.3	Manipulação_Strings	3.5	10.0	1.0	10.0	18/10/2024	Manutenção
11	192.168.1.12	JSON FILE	19.5	23.0	8.5	3800.0	6/11/2024	Associada
12	10.0.0.35	Ficheiro_CSV	10.0	14.0	3.0	250.0	9/12/2024	Desativada
13	172.16.0.4	Ficheiro_TXT	27.0	26.0	10.0	5200.0	14/1/2025	Operação
14	192.168.2.22	Manipulação_Strings	7.0	13.0	2.0	60.0	21/2/2025	Manutenção
15	10.0.0.45	JSON FILE	31.0	29.0	11.0	5900.0	3/3/2025	Associada
16	192.168.1.13	Ficheiro_CSV	14.2	16.5	4.0	900.0	17/4/2025	Desativada

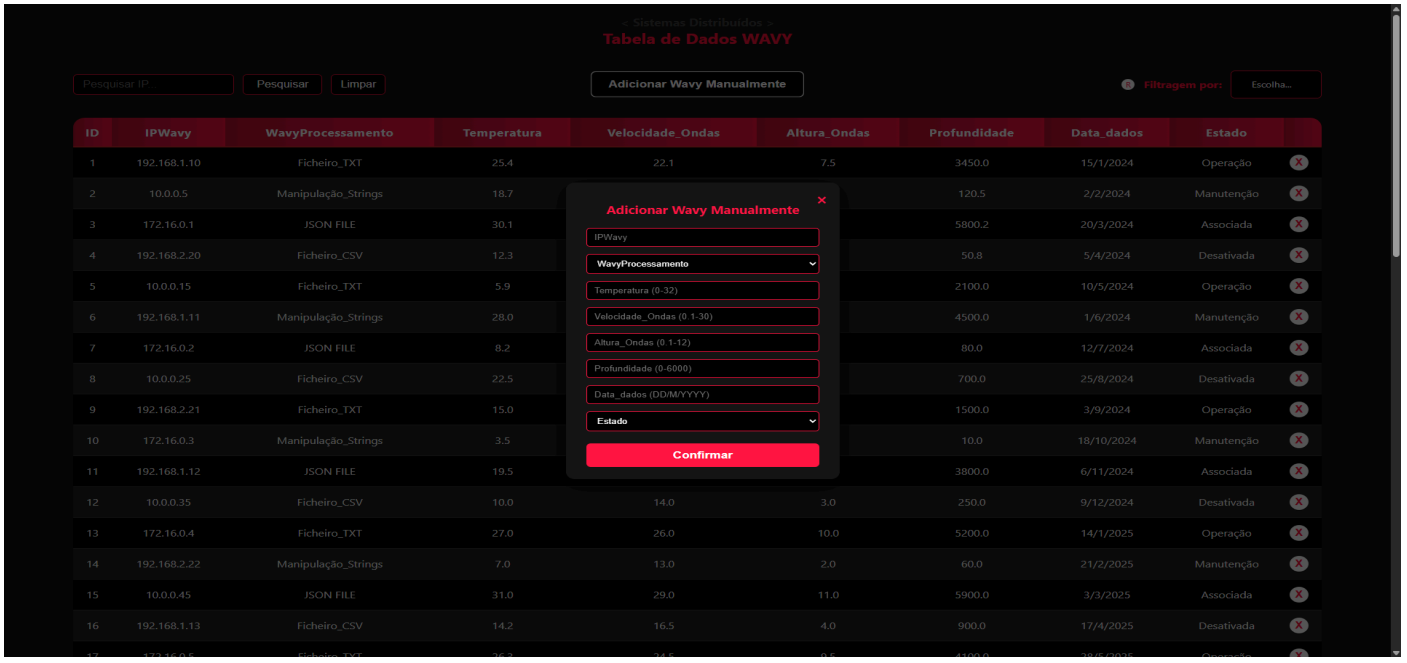
Henrique Dias 78804
Rafael Silva 78532
Pedro Afonso 77181

Como funcionalidades adicionais criámos uma interface em python, html, css e javascript para a visualização de dados das wavy's



ID	IPWavy	WavyProcessamento	Temperatura	Velocidade_Ondas	Altura_Ondas	Profundidade	Data_dados	Estado
1	192.168.1.10	Ficheiro_TXT	25.4	22.1	7.5	3450.0	15/1/2024	Opera
2	10.0.0.5	Manipulação_Strings	18.7	15.3	3.2	120.5	2/2/2024	Manut
3	172.16.0.1	JSON FILE	30.1	28.9	11.8	5800.2	20/3/2024	Associada
4	192.168.2.20	Ficheiro_CSV	12.3	11.5	1.9	50.8	5/4/2024	Desativada
5	10.0.0.15	Ficheiro_TXT	5.9	19.8	6.1	2100.0	10/5/2024	Operação
6	192.168.1.11	Manipulação_Strings	28.0	25.0	9.0	4500.0	1/6/2024	Manutenção
7	172.16.0.2	JSON FILE	8.2	12.0	2.5	80.0	12/7/2024	Associada
8	10.0.0.25	Ficheiro_CSV	22.5	17.5	4.8	700.0	25/8/2024	Desativada
9	192.168.2.21	Ficheiro_TXT	15.0	20.0	7.0	1500.0	3/9/2024	Operação
10	172.16.0.3	Manipulação_Strings	3.5	10.0	1.0	10.0	18/10/2024	Manutenção
11	192.168.1.12	JSON FILE	19.5	23.0	8.5	3800.0	6/11/2024	Associada
12	10.0.0.35	Ficheiro_CSV	10.0	14.0	3.0	250.0	9/12/2024	Desativada
13	172.16.0.4	Ficheiro_TXT	27.0	26.0	10.0	5200.0	14/1/2025	Operação
14	192.168.2.22	Manipulação_Strings	7.0	13.0	2.0	60.0	21/2/2025	Manutenção
15	10.0.0.45	JSON FILE	31.0	29.0	11.0	5900.0	3/3/2025	Associada
16	192.168.1.13	Ficheiro_CSV	14.2	16.5	4.0	900.0	17/4/2025	Desativada

Aqui tal como na imagem anterior, criámos um sistema de filtragem e ordenação de dados(canto superior direito da imagem)



Por fim, para comunicar com a nossa arquitetura, o utilizador pode adicionar uma nova wavy manualmente através da interface

REPOSITÓRIO GITHUB: https://github.com/hd9s1lk/Sistemas-Distribuidos---Grupo-34

Henrique Dias 78804
Rafael Silva 78532
Pedro Afonso 77181