

## Relatório de Arquitetura Orientada a Eventos – 4C

Eduardo Cardoso <sup>1</sup>

<sup>1</sup> Acadêmico Bacharel em Ciência da Computação

[edu.cardoso516@gmail.com](mailto:edu.cardoso516@gmail.com)

**Abstract.** *This article describes implementing a RabbitMQ server using Docker, programming a RabbitMQ client in Java, and the difficulties encountered during the process..*

**Key-words:** *RabbitMQ ; Docker; Java.*

**Resumo.** *Este artigo descreve a implementação de um servidor RabbitMQ usando Docker, a programação de um cliente RabbitMQ em Java, e as dificuldades encontradas durante o processo.*

**Palavras-chave:** *RabbitMQ ; Docker; Java.*

### 1. Introdução

Ao desenrolar da atividade foi realizado a Instalação do RabbitMQ, como servidor, foi utilizado o Docker para mais fácil instalação e utilização em vista que ao utilizar Docker o ambiente já vem com as variáveis instaladas e devidamente configuradas pronto para uso.

Foi utilizada a Linguagem Java para desenvolvimento da aplicação de teste que na verdade só enviava uma string simples como “Olá mundo” e deixava disponível para a aplicação de consulta.

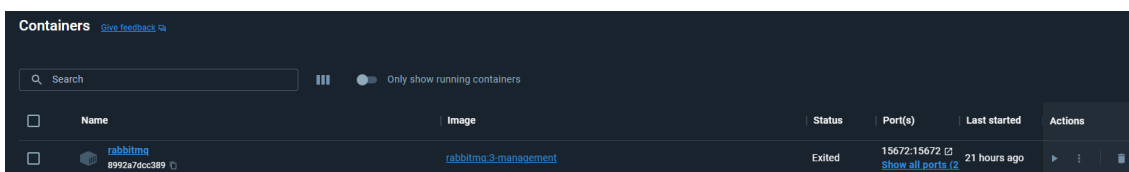
Não foi encontrado um clint disponível ou compatível de maneira direta(Play Store) para o Android.



### 2. Instalação do RabbitMQ no docker

A atividade foi realizada sobre o Sistema Windows 10 que ja continha o docker instalado. Foi utilizado o comando no powerShell(Terminal do Windows) :

```
docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management
```

Onde ao utilizar o comando o mesmo ja configura as portas 5672 para comunicação e 15672 para o painel de controle web.



| Containers <a href="#">Give feedback</a>                                                       |                                                                                                             |                       |        |                                                                                                                                  |              |
|------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-----------------------|--------|----------------------------------------------------------------------------------------------------------------------------------|--------------|
| <input type="text" value="Search"/> <span>☰</span> <span>●</span> Only show running containers |                                                                                                             |                       |        |                                                                                                                                  |              |
| <input type="checkbox"/>                                                                       | Name                                                                                                        | Image                 | Status | Port(s)                                                                                                                          | Last started |
| <input type="checkbox"/>                                                                       |  rabbitmq<br>8992a7dc389 | rabbitmq:3-management | Exited | 15672:15672  <a href="#">Show all ports</a> | 21 hours ago |

**Imagem 1 - Captura da Instância do Rabbit no docker - Fonte : Autor**

Na imagem 1 podemos ver já instalada e instanciada.

**3. IDE de trabalho e Requisitos**

Foi utilizada a Linguagem Java 21 para o experimento e utilizado o Netbeans 17 como estação de desenvolvimento. Foi Criado um projeto do com utilização do maven para realizar a busca automatizada das bibliotecas no repositório java.

Para ser possível baixar corretamente as bibliotecas, é necessário adicionar ao pom do projeto as seguintes linhas na parte de dependências do arquivo pom:

```
<dependencies>
  <dependency>
    <groupId>com.rabbitmq</groupId>
    <artifactId>amqp-client</artifactId>
    <version>5.14.3</version>
  </dependency>
</dependencies>
```

**Imagem 2 - Adicionando as dependência no arquivo pom- Fonte : Autor**

Como podemos ver é necessário essa adição no arquivo pom para ser possível incorporar de forma automatizada ao projeto as bibliotecas amqp-client-5.14.3.jar e slf4j-api-1.7.32 .

**Imagem 3 - Classes da atividade - Fonte : Autor**

Por se tratar de um trabalho de teste em laboratório foi criado as duas classes Consulta e Pública, na qual a Pública envia uma mensagem para o servidor Rabbit e a Consulta “Recebe” a mensagem do servidor.

**4. Servidor Rabbit**

Por usarmos o docker como instalação podemos acessar o servidor através do link : <http://localhost:15672/>, tendo como usuário e senha os padrões usuario: “guest” senha: “guest”



Username:  \*

Password:  \*

**Imagem 4 - Página de acesso ao servidor - Fonte : Autor**

The screenshot shows the RabbitMQ Overview page. At the top, it displays the RabbitMQ logo, version 3.13.0, and Erlang 26.2.2. The page is refreshed every 5 seconds. The main navigation bar includes Overview, Connections, Channels, Exchanges, Queues and Streams, and Admin. The Overview section shows global counts: Connections: 0, Channels: 0, Exchanges: 7, Queues: 0, and Consumers: 0. Below this, there is a table of nodes. The first node is 'rabbit@8992a7dcc389' with 40 file descriptors, 0 socket descriptors, 429 Erlang processes, 157 MiB memory, and 953 GiB disk space. The page also includes links for documentation, tutorials, and other resources.

| Name                | File descriptors | Socket descriptors | Erlang processes | Memory  | Disk space | Uptime  | Info             | Reset stats         |
|---------------------|------------------|--------------------|------------------|---------|------------|---------|------------------|---------------------|
| rabbit@8992a7dcc389 | 40               | 0                  | 429              | 157 MiB | 953 GiB    | 20m 53s | basic disc 2 rss | This node All nodes |

**Imagem 5 - Ao acessar o servidor - Fonte : Autor**

Como podem ver na imagem 5, aqui temos a disponibilidade das ferramentas estatísticas do servidor e acessos.

## 5. Código

Foi realizado a implementação de duas classes como descrito na seção 3 deste relatório, ao decorrer desta seção veremos como foi a implementação do código das classes.

```
package publica;

import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;

import java.io.FileWriter;
import java.io.IOException;

public class Main {
    private final static String QUEUE_NAME = "fila";
    private final static String LOG_FILE = "mensagens_enviadas.log";

    public static void main(String[] argv) throws Exception {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("localhost");

        try (Connection connection = factory.newConnection(); Channel channel = connection.createChannel()) {
            channel.queueDeclare(QUEUE_NAME, false, false, false, null);
            String mensagem = "lenda";

            // Armazenar a mensagem enviada no arquivo de log
            logMessage(mensagem);

            channel.basicPublish("", QUEUE_NAME, null, mensagem.getBytes());
            System.out.println(" [x] Enviado '" + mensagem + "'");
        }

        private static void logMessage(String mensagem) {
            try (FileWriter writer = new FileWriter(LOG_FILE, true)) {
                writer.write(mensagem + System.lineSeparator());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

**Imagem 6 - Classe com responsabilidade de Publicar as informações - Fonte : Autor**

Na imagem 6 está sendo desenvolvida a funcionalidade de publicação de mensagens em um servidor RabbitMQ. Inicialmente, é estabelecida a conexão com o servidor RabbitMQ, configurando-se o endereço do host como "localhost" por meio de um objeto ConnectionFactory. Em seguida, uma conexão e um canal são criados, onde é declarada uma fila chamada "fila" por meio do método queueDeclare(). A mensagem a ser enviada é definida como "lenda" e é publicada no canal utilizando o método basicPublish(). Cada mensagem enviada é registrada em um arquivo de log denominado "mensagens\_enviadas.log", garantindo o rastreamento e o armazenamento das mensagens enviadas para fins de auditoria ou registro. Este código exemplifica um processo básico de publicação de mensagens em um servidor RabbitMQ, fornecendo uma base para o desenvolvimento de sistemas distribuídos e assíncronos.

```

package consulta;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.DeliverCallback;

public class Main {
    private final static String QUEUE_NAME = "fila";

    public static void main(String[] argv) throws Exception {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("localhost");

        try (Connection connection = factory.newConnection(); Channel channel = connection.createChannel()) {
            channel.queueDeclare(QUEUE_NAME, false, false, false, null);
            System.out.println(" [*] Aguardando mensagens. Para sair, pressione CTRL+C");

            DeliverCallback deliverCallback = (consumerTag, delivery) -> {
                String mensagem = new String(delivery.getBody(), "UTF-8");
                System.out.println(" [x] Recebido '" + mensagem + "'");
            };

            channel.basicConsume(QUEUE_NAME, true, deliverCallback, consumerTag -> {
            });
        }
    }
}

```

**Imagem 7 - Classe com responsabilidade de Consultar as informações - Fonte : Autor**

Enquanto na classe responsavel utiliza basicPublish() para subir as mensagens, na consulta as mensagens são consumidas da fila utilizando o método basicConsume() a qual ja recebe os valores do servidor e após printa em tela.

Agora podemos ver a execução nas imagens a baixo:

```

cd C:\Users\Usuario\Documents\NetBeansProjects\AMQP; "JAVA_HOME=C:\Program Files\Java\jdk-21" cmd /c "
Scanning for projects...

-----< com.mycompany:AMQP >-----
[ ] Building AMQP 1.0-SNAPSHOT
    from pom.xml
-----[ jar ]-----
[ ] --- resources:3.3.1:resources (default-resources) @ AMQP ---
- skip non existing resourceDirectory C:\Users\Usuario\Documents\NetBeansProjects\AMQP\src\main\resources
[ ] --- compiler:3.11.0:compile (default-compile) @ AMQP ---
- Nothing to compile - all classes are up to date
[ ] --- exec:3.1.0:exec (default-cli) @ AMQP ---
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[x] Enviado 'lenda'

BUILD SUCCESS

Total time: 1.279 s
Finished at: 2024-03-20T19:22:58-03:00
|

```

**Imagem 8 - Subindo os dados para o servidor - Fonte : Autor**

```

cd C:\Users\Usuario\Documents\NetBeansProjects\AMQP; "JAVA_HOME=C:\\Program Files\\Java\\jdk-21" cmd /c "
Scanning for projects...

-----< com.mycompany:AMQP >-----
[ ] Building AMQP 1.0-SNAPSHOT
    from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ AMQP ---
- skip non existing resourceDirectory C:\Users\Usuario\Documents\NetBeansProjects\AMQP\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ AMQP ---
- Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ AMQP ---
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[*] Aguardando mensagens. Para sair, pressione CTRL+C
[x] Recebido 'lenda'

BUILD SUCCESS

Total time: 0.928 s
Finished at: 2024-03-20T19:25:40-03:00

```

**Imagem 8 - Recebendo os dados do servidor - Fonte : Autor**

## 6. Dificuldades

Ao tentar instalar o Docker no Pop!\_OS, esbarrei em pacotes quebrados, o que atrapalhou a instalação. Isso me fez perder tempo tentando resolver esses problemas antes de poder continuar.

Também tive dificuldade em encontrar um aplicativo na Play Store que funcionasse como cliente para acessar o servidor RabbitMQ no meu celular Android. Passei um tempo procurando, mas não consegui encontrar nada que atendesse às minhas necessidades. Isso dificultou testar a integração do RabbitMQ com dispositivos móveis.

## 4. Referências

Stack Overflow. (2017). How to open RabbitMQ in browser using Docker container? <https://stackoverflow.com/questions/47290108/how-to-open-rabbitmq-in-browser-using-docker-container>

Apps Developer Blog. (s.d.). Run RabbitMQ Docker Container Command. Recuperado de <https://www.appsdeveloperblog.com/run-rabbitmq-docker-container-command/>

RabbitMQ. (s.d.). Consumers. <https://www.rabbitmq.com/docs/consumers>