Relatório final – Etapa 2 – MQTT

Nomes: Paulo Henrique Hollenbach Muller e David Fambre Mezadri

Serviço de troca de comunicações usando Paho-MQTT

1 Descrição do projeto:

O projeto consiste no desenvolvimento de um sistema de comunicação distribuída baseada no protocolo MQTT, no qual os usuários podem podem criar grupos, anunciar sua existência, descobrir grupos ativos, solicitar entrada, aprovar ou rejeitar pedidos e enviar mensagens em grupo.

1.1 Fluxo:

O fluxo é organizado por meio de dois services principais:

- **MqttService** – responsável pela comunicação com o broker MQTT, incluindo conexão, publicação, inscrição em tópicos e repasse de mensagens.
- **GroupService** - responsável pela lógica da aplicação, controle de grupos, tratamento de eventos e gerenciamento de mensagens.

O sistema foi projetado para funcionar em qualquer ambiente capaz de executar JavaScript/TypeScript com suporte ao cliente **Paho MQTT**, permitindo comunicação em tempo real entre múltiplos usuários.

2 Arquitetura do sistema/aplicativo:

A arquitetura é organizada em **camadas independentes**, seguindo princípios de modularidade, com divisão clara de responsabilidades:

2.1 Camada de comunicação (MqttService):

Responsável pela interface com o broker MQTT, suas funções incluem:

- Conexão através do Paho.Client
- Publica mensagens nos tópicos
- Transferências em JSON
- Qos (Quality of Service) com entrega garantida (Status 1)
- Handlers para interceptação de eventos

2.2 GroupService (Gerenciamento de grupos):
Camada para cuidar das funcionalidades dos grupos:

Nessa camada, temos os recursos:

2.2.1 Criação de grupos:

- Gera um groupId para cada grupo criado
- Pública no método GroupAdvertisement em group/list/<groupId>
- Faz a inscrição automática no tópico de chat do grupo

2.2.2 Descoberta de grupos:

- Ao usuário verificar os grupos ativos, dispara em group/list/#
- Cada advertisement vira um evento group_discovered

2.2.3 Solitação de entrada e aprovação/rejeição:

- Envia group_join_request direto para o admin via group/control/<adminId>

Para aceitar/rejeitar a entrada, o admin pode responder com:

- group_join_approval ou group_join_rejection

Se aprovado, o usuário é inscrito no tópico group/chat/<groupId> gerando o evento group_join_approved

2.2.4 Troca de mensagens do grupo:

Envio de mensagens pelo método sendGroupMessage() no qual todos os participantes do grupo possuem visualização instantânea com entrega de garantia.

3 NewChatService (Chats individuais, presença, convites e eventos unificados)

O NewChatService complementa o sistema, ofereçendo:

- Convites diretos entre usuários
- Aceitação e rejeição de convite
- Criação automática de tópicos de chat individual
- Carregamento de conversas
- Eventos de presença online/offline
- Recebimento de mensagens individuais

- Integração entre grupos

3.1 Troca de mensagens do grupo:

Tópicos usados pelo NewChatService

- Controle individual: control/<userId>
- Presença: presence/<userId>
- Presença geral: presence/#
- Carregar conversas: control/loadconversation/<userId>
- Chat individual: chat/<topic>
- Listagem de grupos: group/list

3.2 Eventos tratatos pelo NewChatService:

Ele gera eventos tanto de chat quanto de grupo (os nomes são autoexplicativos). São eles:

- invite_received
- invite_accepted
- invite_reject
- message_received
- chat_subscribed
- presence_update (atualiza a presença do usuário)
- load_conversation

3.3 Funcionamento interno do NewChatService:

3.3.1 Inicialização:

O método initialize() ativa o carregamento de conversas antigas, prepara o canal de controle, marca o usuário como online e começa a escutar anuncios de grupos. Ao entrar, publica status: online, se inscreve em presence/#. Ao sair, publica status: offline.

3.3.2 Convites:

Envia um requestId e publica em control/<destino>. Ao ser recebido, vira um evento invite_received

Se aceito:

- Cria o tópico ex: chat/123
- assina esse novo tópico
- Gera o evento invite_accepted

3.3.3 Eventos de grupo:

Como o NewChatService usa o global message handler, ele também captura outros eventos como:

- group_join_request

- group_join_approved
- group_join_rejected
- group_message_received
- group_discovered
- group_removed

4 Para utilizar o aplicativo:

Tenha o Paho-MQTT instalado na máquina e conecte-o na porta 9001, ao pegar o código, certifique-se de que tenha pelo menos a versão 18 do Node.js e v11.0.0 do npm. Execute npm i e em seguida npm run dev.