Relatório Tópicos

Chat com protocolo MQTT

Autores: Júlio Gonçalves, Thiago Spanevello
Instituição: Universidade Federal da Fronteira Sul(UFFS)

Resumo

Este trabalho apresenta a implementação de uma aplicação de bate-papo (*chat*) desenvolvida em JavaScript, responsável tanto pela lógica de comunicação quanto pelo gerenciamento dos elementos visuais da interface. A estilização foi realizada utilizando CSS padrão, garantindo uma aparência simples e funcional.

O principal objetivo do projeto é compreender o funcionamento do protocolo MQTT, que adota o modelo de publicador/assinante (também conhecido como produtor/consumidor). Nesse modelo, as mensagens são enviadas para tópicos criados no broker, aos quais diferentes clientes podem se inscrever. O Mosquitto, atuando como broker MQTT, é o responsável por intermediar essa comunicação, recebendo as publicações dos produtores e distribuindo-as aos assinantes correspondentes.

Introdução

Com a popularização de aplicativos de mensagens, entender como funcionam as comunicações em tempo real se tornou cada vez mais importante. Entre os protocolos usados para isso, o MQTT se destaca por ser leve e eficiente, funcionando bem mesmo em sistemas distribuídos ou dispositivos com recursos limitados.

O MQTT usa o modelo publicador/assinante: os clientes publicam mensagens em tópicos e podem se inscrever nos tópicos que querem acompanhar. Isso permite que a comunicação seja mais flexível e escalável, sem que os participantes precisem se conhecer diretamente.

Neste trabalho, desenvolvemos um chat em JavaScript que usa MQTT (via paho-mqtt). A aplicação cuida tanto da lógica de envio e recebimento de mensagens quanto da interface visual, que foi estilizada com CSS simples. O Mosquitto foi usado como broker para gerenciar a entrega das mensagens entre os usuários.

Arquitetura do projeto

Tópicos utilizados

- userTopic: Mensagens de controle do usuário: "clientId_" + name.
- Grupos criados: “GROUPS”.
- Status dos usuários: “usersStatus”.
- Chat privado: “chat/requisitante_requisitado_timestamp”.
- Grupo específico: “group/código”.

userTopic é usado na criação do cliente, sendo apenas assinado pelo próprio cliente que é referente àquele tópico, usado para controle de operações. Ex: cliente A quer iniciar um chat com o cliente B, cliente A manda para o tópico userTopic de cliente B uma mensagem de solicitação de chat, tendo o fato que apenas cliente B consegue visualizar as mensagens enviadas até seu userTopic.
Mensagem de solicitação de chat:

```
const invite = new Paho.MQTT.Message(
    JSON.stringify({
      type: "invite",
      from: id,
      to: targetId,
      timestamp: new Date().getTime(),
    })
  );
  invite.destinationName = "clientId_" + targetId;
  invite.qos = 2;
  client.send(invite);
```

GROUPS é o tópico de grupos criados, sendo assim feito o controle de qual código de grupo já existe e também a solicitação para entrar em um grupo via código.
Mensagem de criação de grupo:

```
  const status = new Paho.MQTT.Message(
    JSON.stringify({
      type: "group-taken",
      code,
      admin: id,
      members: [id],
    })
  );
  status.destinationName = "GROUPS";
  status.qos = 2;
  status.retained = true;
  client.send(status);
```

usersStatus é o tópico em que os clientes ficam mandando periodicamente se estão online ou não, para ser feito o controle de entrega de mensagens para um usuário desconectado.
Mensagem de status de usuário:

```
statusInterval = setInterval(() => {
        const status = new Paho.MQTT.Message(
          JSON.stringify({
            type: "status",
            from: id,
            timestamp: new Date().getTime(),
          })
        );
        status.destinationName = "usersStatus";
        status.qos = 0;
        client.send(status);
      }, 2500);
```

Chat privado ou Chat em grupo é feito procurando qual o chatTopic atual do chat que o usuário tem selecionado na tela, então quando ele envia uma mensagem no chat é chamada a função handleSendMessage() de handler.js, que possui essa parte para enviar a mensagem ao tópico.
Mensagem de chat e grupo:

```
  if (active_chat.split("/")[0] === "group")
    chat = groups.find((c) => c.chatTopic === active_chat);
  if (!chat) return;

  const newMessage = {
    type: "message",
    from: id,
    timestamp: new Date().getTime(),
    message: messageText,
  };

  const msg = new Paho.MQTT.Message(JSON.stringify(newMessage));
  msg.destinationName = active_chat;
  msg.qos = 2;
  msg.retained = true;
  client.send(msg);
```

Mecanismos Aplicados

Conexão ao broker

A conexão é iniciada quando o usuário informa seu nome e clica no botão de conectar. Nesse momento, o sistema cria um cliente MQTT identificado pelo nome do usuário e tenta se conectar ao broker Mosquitto. São definidos callbacks para lidar com eventos de conexão, desconexão e perda de comunicação. Após a conexão bem-sucedida, o status visual da interface é atualizado, mostrando que o cliente está online e apto a interagir com outros usuários.

Envio e recebimento de mensagens

As mensagens de texto e imagem são enviadas através da função handleSendMessage() e do evento de envio de arquivos. Cada mensagem é encapsulada em um objeto JSON contendo informações como tipo ("message"), remetente, conteúdo e timestamp. Esse objeto é então convertido em uma instância de Paho.MQTT.Message e publicado no tópico correspondente ao chat ativo. Para garantir confiabilidade, é utilizado o **nível de QoS 2**, assegurando que a mensagem será entregue uma única vez. O recebimento é tratado por meio da assinatura nos mesmos tópicos, atualizando o histórico e exibindo as mensagens na tela. A entrega de mensagens para usuários offline é realizada pelo próprio Mosquitto em **nível QoS 2,** além de mostrar todas as mensagens da conversa pelo fato de existir um histórico do chat armazenado no localStorage.

Criação e gerenciamento de chats e grupos

O sistema permite a criação de conversas privadas e grupos. Nos chats diretos, quando um usuário convida outro, é enviada uma mensagem MQTT do tipo "invite" para o cliente de destino. Esse convite pode ser aceito ou recusado, sendo a resposta enviada com o tipo "inviteResponse". Em caso de aceitação, é criado um novo tópico exclusivo no formato chat/id1_id2_timestamp, no qual ambos os usuários passam a publicar e assinar. Já os grupos utilizam tópicos group/<código> e são gerenciados por mensagens do tipo "reqResponse" e "group-taken", que controlam os membros e administradores.

Atualização de status e presença

O status de conexão dos usuários é monitorado a partir de atualizações periódicas enviadas e armazenadas no objeto usersStatus. Cada cliente mantém um registro com o último momento em que foi detectado online. O sistema compara esses timestamps para determinar se um usuário está ativo (menos de seis segundos desde a última atualização). Essa informação é exibida em tempo real ao lado de cada contato na lista de chats, com indicadores visuais de "Online" e "Offline".

Interface e interação do usuário

Toda a parte visual da aplicação é controlada com JavaScript e CSS padrão. A interface conta com botões, abas, modais e toasts para interação dinâmica. As abas alternam entre contatos e grupos, enquanto os modais são utilizados para convites e criação de novos chats. O feedback visual das ações é dado através de notificações rápidas (showToast) e alterações no DOM. Além disso, a interface é responsiva, adaptando-se à presença de uma sidebar que pode ser aberta ou fechada conforme a necessidade do usuário.

Implementação

A implementação do projeto teve início com a escolha da linguagem de programação a ser utilizada. Ambos os integrantes concordaram em empregar JavaScript, por já possuírem experiência prévia com a linguagem e se sentirem à vontade em seu uso. O único controle adicional necessário foi garantir que um cliente não recebesse a própria mensagem enviada ao tópico, mantendo a comunicação correta entre os usuários.

Em seguida, foram estruturados os tópicos necessários para a comunicação, sendo o desenvolvimento realizado gradualmente até que se compreendesse, na prática, o funcionamento do protocolo MQTT, bem como o envio e recebimento de mensagens por parte do broker. Inicialmente, o foco esteve nas funcionalidades básicas, especialmente na criação de chats privados e em grupo, para depois prosseguir com o envio de mensagens, exibição de status dos usuários e outros detalhes complementares.

Com a parte lógica consolidada, o foco passou para o desenvolvimento da interface visual, buscando oferecer um ambiente agradável e funcional. Foram priorizados aspectos essenciais, como um layout organizado e intuitivo, além de elementos visuais adicionais, como notificações de status (toasts) de conexão e desconexão durante o login e logout.

Por fim, com a base do projeto praticamente concluída, optou-se por incluir funcionalidades extras que tornaram a aplicação mais completa e interessante, como o envio de imagens via chat e o histórico de mensagens, permitindo que o usuário visualize imagens enviadas anteriormente, mesmo após sair e retornar ao sistema.

Instruções para compilação

Pode ser encontrado no arquivo README.md do projeto, porém vamos colocar o passo a passo aqui também:

Passo 1: iniciar o broker Mosquitto com as configurações do projeto.

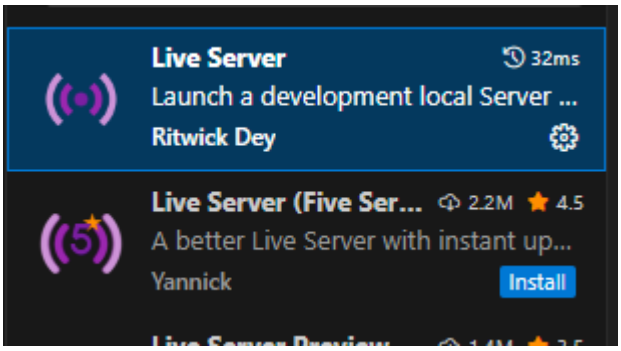
cd “/pasta que se encontra o projeto/MQTT”

mosquitto -c c.conf

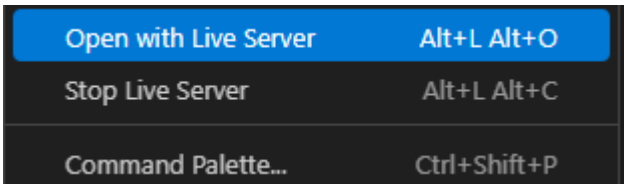
abrir o documento html

Maneira que recomendamos:

Recomendamos o uso da extensão live server ou fire server do vscode.



Após instalar algumas das opções, basta abrir o documento html, clicar com o botão direito do mouse e selecionar a opção abaixo:



Ou utilize o próprio comando citado acima para abrir: Alt + L + Alt + O.

Conclusão

O desenvolvimento deste projeto permitiu compreender de forma prática o funcionamento do protocolo MQTT e sua aplicação em sistemas de comunicação baseados no modelo publicador/assinante. A implementação da aplicação de bate-papo demonstrou como o protocolo pode ser utilizado de maneira eficiente para transmitir mensagens em tempo real entre diferentes clientes conectados a um broker.

Além do aprendizado técnico sobre a estrutura e o fluxo das mensagens MQTT, o projeto proporcionou uma experiência enriquecedora na organização e integração entre lógica de programação e interface visual, utilizando JavaScript e CSS de forma complementar.

Como resultado final, obteve-se uma aplicação funcional, de fácil utilização e com recursos adicionais, como envio de imagens, histórico de mensagens e notificações de status, que tornam o uso mais completo e intuitivo. O trabalho, portanto, atingiu seu objetivo principal de consolidar o entendimento do protocolo MQTT e demonstrar sua aplicabilidade em um cenário real de comunicação entre usuários.