[Mensagens de plataforma cruzada facilitadas com o TMS MQTT](https://www.tmssoftware.com/site/tmsmqtt.asp)

Na semana passada lançamos um novo produto, o [TMS MQTT](http://www.tmssoftware.com/mqtt) . [O MQTT](http://mqtt.org/) é um protocolo de mensagens de máquina 2 padrões e leve. Isso permite a troca instantânea de mensagens binárias ou de texto entre o código em execução em vários sistemas operacionais. Para aproveitar totalmente isso, desenvolvemos nosso cliente TMS MQTT para trabalhar em aplicativos Windows VCL, FMX Windows, macOS, aplicativos iOS e Android e também em desktop Linux ou variantes do Linux, como Raspbian on Raspberry Pi via FPC / Lazarus.  
  
Neste aniversário de 22 anos do Delphi, eu queria mostrar a você como nosso componente TMS MQTT permite que você desenvolva um aplicativo de mensagens em um modo RAD real. Farei isso implementando um aplicativo de bate-papo que permita que você converse entre clientes em execução em desktops ou dispositivos móveis. Isso efetivamente leva menos de 40 linhas de código usando o componente TMS MQTT do qual 15 linhas já inserem o texto de bate-papo esquerdo e direito em itens de uma caixa de listagem FMX e outras linhas de +/- 10 detectando a origem do texto do bate-papo. recebido para decidir se deseja mostrá-lo à esquerda ou à direita na caixa de listagem. Então, eu diria que apenas cerca de 15 linhas de código dizem respeito ao uso do cliente MQTT para fazer a troca de mensagens.   
  
Para esta demonstração, usamos o corretor de teste Mosquitto MQTT e nos conectamos a ele com duas linhas de código:

1. **begin**
2. TMSMQTTClient1.BrokerHostName := 'test.mosquitto.org';
3. TMSMQTTClient1.Connect();
4. **end**;

Ao se conectar, nós nos inscrevemos no tópico 'tms / chat' que é o tópico que será usado para trocar o texto do chat entre vários clientes. Isso leva 2 linhas de código do evento TMQTTClient.OnConnectedStatusChanged ():

1. **procedure** TForm1.TMSMQTTClient1ConnectedStatusChanged(ASender: TObject;
2. **const** AConnected: Boolean; AStatus: TTMSMQTTConnectionStatus);
3. **begin**
4. **if** AConnected **then**
5. TMSMQTTClient1.Subscribe('tms/chat');
6. **end**;

Para enviar uma mensagem, o texto inserido em um controle de nota é publicado neste tópico com uma linha de código:

TMSMQTTClient1.Publish('tms/chat', TMSMQTTClient1.ClientID+'!'+ memo1.Lines.Text);

Aqui, adicionamos o ID exclusivo do aplicativo enviando o texto do bate-papo e o próprio texto do bate-papo.

Finalmente, o texto do bate-papo recebido no tópico inscrito é recebido pelo evento TMQTTClient.OnPublishReceived (). O ID do cliente exclusivo é recuperado para decidir se deseja colocar o texto do bate-papo à esquerda ou à direita da caixa de listagem da mensagem e isso é feito por meio do código:

1. **procedure** TForm1.TMSMQTTClient1PublishReceived(ASender: TObject;
2. APacketID: Word; ATopic: **string**; APayload: TArray<system.**byte**>);
3. **var**
4. msg,orig: **string**;
5. vp: **integer**;
6. alright: **boolean**;
7. **begin**
8. msg := TEncoding.UTF8.GetString(APayload);
10. vp := pos('!', msg);
12. **if** vp > 0 **then**
13. **begin**
14. orig := copy(msg,1,vp-1);
15. alright := orig <> TMSMQTTClient1.ClientID;
17. msg := copy(msg, vp + 1, Length(msg));
18. AddMessage(msg, alright);
19. **end**;
20. **end**;</system.**byte**>

E então vem o método para nos permitir inserir texto alinhado à esquerda ou à direita como itens em uma caixa de listagem FMX:

1. **procedure** TForm1.AddMessage(AMessage: **string**; AlignRight: **boolean**);
2. **var**
3. li: Tlistboxitem;
4. **begin**
5. li := Tlistboxitem.Create(self);
6. li.StyledSettings := li.StyledSettings - [TStyledSetting.ssOther];
7. li.Text := AMessage;
8. li.Height := 22;
9. li.VertTextAlign := TTextAlign.taTrailing;
11. **if** AlignRight **then**
12. li.TextAlign := TTextAlign.taTrailing
13. **else**
14. li.TextAlign := TTextAlign.taLeading;
16. listbox1.AddObject(li);
17. **end**;

Compile e execute este aplicativo em seu sistema operacional de sua escolha. No meu caso, eu configurei uma pequena conversa entre o cliente compilado para o Windows e o cliente implementado em um iPhone:

Você pode baixar o código-fonte completo deste aplicativo FireMonkey [aqui](http://www.tmssoftware.net/public/TMSQuickAndEasyChat.zip) .   
  
Eu o convido a explorar as numerosas e empolgantes capacidades da troca de mensagens da máquina 2. Note que existem bibliotecas JS que facilitam o envio de mensagens MQTT via websockets de uma web browser. A demonstração incluída no download do TMS MQTT é baseada em mensagens entre um aplicativo cliente de desktop ou móvel e um aplicativo de página da web. Com o cliente TMS MQTT, você também pode usar mensagens de máquina 2 a partir de seu Raspberry Pi e enviar vários dados de sensor capturados pelo Raspberry Pi para aplicativos de desktop ou móveis. Em um blog de acompanhamento, explicaremos e ofereceremos esse projeto de amostra.

https://www.tmssoftware.com/site/img/raspberry-pi-logo.png

Estou ansioso para saber como você usará as mensagens de máquina da máquina 2 do MQTT em seus aplicativos ou projetos de IoT!