Fazendo um Chat usando Socket.IO

Vai ser louco

Antes de qualquer coisa, vamos separar isso em duas partes: Server side e Client side. Calma, você vai entender.

Server Side

① **Mas o que é isso?**

Ótima pergunta! Vamos começar pelo básico: quando fazemos um chat temos que pensar em como será mandada a mensagem. Para isso, precisamos pensar no servidor, que irá receber essas mensagens e mandar elas adiante. No fundo, funciona como um banco de dados como qualquer outro.

② **Tá, e como eu faço isso?**

Ai que está o detalhe: eu também não montei o meu próprio servidor. Não, não me julgue. Eu não tenho paciência para isso, fora que não tem nada de desenvolvimento iOS em si, então acho que não entra no caso. Porém, vou deixar o link aqui de qual eu peguei.

https://github.com/appcoda/SocketIOChat

Além desse link, vou deixar também o tutorial de como baixar e usar esse servidor. Basicamente ele roda um servidor localmente usando node, e esses passos estão bem detalhados no projeto dele.

https://www.appcoda.com/socket-io-chat-app/

③ **Então é isso de servidor? Você não fez nada?**

É. Supere

④ **Mas...**

CLIENT SIDE

Client Side

① **Beleza, client side. Mas naquele link que você mandou da appcoda já ensina como faz essa parte também.**

Verdade! E realmente é muito bom o tutorial que eles fazem. Porém, vou falar aqui de alguns pontos que não são tão óbvios assim, e que ele não ajuda tanto. Além disso, a versão do Swift dele é antiga, então vou deixar meu projeto aqui também, já com algumas coisas corrigidas. MAS eu não fiz como ele fez, o meu projeto está um pouco diferente.

② **OK, e o que não é tão óbvio assim?**

Vamos por pontos:

1. Antes de qualquer coisa, ele usa um singleton para fazer todo o controle do chat, o que faz muito sentido, e que recomendo que também faça. Caso não conheça muito sobre singleton e como usar, ai vai um link pra você

https://cocoacasts.com/what-is-a-singleton-and-how-to-create-one-in-swift

2. Duas funções do Socket.IO que ele usa muito e são essenciais para trabalhar com o chat.

```
func on(_ event: String, callback:
@escaping NormalCallback) -> UUID
```

Quando passamos um **on**, estamos falando que SEMPRE que esse evento ocorrer, essa função será chamada. Uma comparação grosseira mas que me ajudou a entender: Isso é um "get" junto com uma "notification", ou seja, você recebe alguma coisa e sempre que recebe, a função é chamada. No caso dele, ele chama `socket.on("newChatMessage")`, logo, sempre que ele receber alguma mensagem, a função será chamada. Nota: a informação do "get" vem no callback.

```
func emit(_ event: String, _ items:
SocketData..., completion: (() ->
())? = nil)
```

O **emit** é quase o oposto do **on**. Nele, avisamos que algo deve ser passado para o banco e que algo deve acontecer. Seguindo a mesma comparação podemos falar que ele se assemelha bastante com o "post", onde colocamos algo no banco. No caso dele, ele usa o `socket.emit("connectUser", nickname)`, que fala para o servidor conectar o usuário com esse nickname.

③ **Entendi! Mas como funciona esses events? De onde vem esse "newChatMessage" e "connectUser" que ele usa?**

Esse é um ponto extremamente importante! Esses nomes de eventos não é algo padrão, não é algo que você poderá usar em qualquer projeto, pois eles dizem respeito ao banco de dados e aos nomes dados nele. É quase o nome de uma função no banco, logo, o nome dos eventos depende de como foi implementado o servidor.

④ **Mas onde que você viu que eram esses nomes???**

Ai que entra a parte de consultar a parte do servidor de novo! Quando abrimos o arquivo srv-SocketChat baixado no primeiro link, temos um index.js. Quando abrimos isso com algum editor de texto, temos essa parte do código que procuramos. Por exemplo, no nosso socket.on("newChatMessage"), a implementação aparece no index.js assim:

```
io.emit('newChatMessage', clientNickname, message, currentDateTime);
```

⑤ **Nossa mas é muita coisé, né?**

Sim, não é muito simples, mas seguindo os tutoriais fica um pouco mais fácil.

Por isso, nas referências, vou deixar alguns links de consulta, assim como o projeto que eu fiz, se você quiser usar como base!

Referências

https://www.appcoda.com/socket-io-chat-app/

https://github.com/appcoda/SocketIOChat

https://github.com/appcoda/SocketIOChat

https://socket.io/get-started/chat/

https://socket.io/get-started/chat/#Emitting-events

https://github.com/nuclearace/socket.io-client-swift-example

https://github.com/felipekpetersen/ChatSocketGettingStarted

Nota: Usei pod no lugar de colocar os arquivos diretamente

Felipe Kaça Petersen

iOS