Enviando mensagens

Ainda está faltando o método de enviar mensagem, não é mesmo? Bom, também precisamos dispará-lo de forma assíncrona e por isso chamar o método enqueue. Mas o que vamos precisar tratar como resposta? Não precisamos fazer nada e se houver um erro, podemos disparar um Toastavisando o usuário que não foi possível entregar a mensagem.

Vamos criar uma classe chamada de EnviarMensagemCallback dentro do pacote br.com.caelum.ichat.callback:

public class EnviarMensagemCallback implements Callback<Void> {  
  
 @Override  
 public void onResponse(Call<Void> call, Response<Void> response) {  
  
 }  
  
 @Override  
 public void onFailure(Call<Void> call, Throwable t) {  
  
 }  
}

Nela não precisamos fazer nada, já que não queremos tratar as respostas, só queremos **enviar**.

Já na MainActivity, fazemos:

button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 chatService.enviar(new Mensagem(idDoCliente, editText.getText().toString())).enqueue(new EnviarMensagemCallback());  
 }  
});

Se tentarmos enviar uma mensagem, ela chegará vazia. Isso porque na classe **Mensagem** tem um atributo chamado **texto**, e o JSON que trafega na API traz o texto através de uma propriedade chamada **text**, então temos que fazer um mapeamento:

public class Mensagem {  
  
 @SerializedName("text")  
 private String texto;  
  
 // ...  
}

Podemos rodar novamente e ver que tudo está funcionando corretamente, exatamente como antes, mas agora trabalhando com as mensagens de uma forma mais enxuta, utilizando o **Retrofit**.

## **O que aprendemos nesse capítulo?**

* Como implementar um Chat com HTTP;
* Trabalhar com requisições HTTP no Android pode ser um trabalho complexo e muito baixo nível;
* Usando Retrofit para tornar mais fácil e transparente a comunicação com serviços Web.