Ouvindo mensagens

Isso acontece porque o Retrofit precisa executar a chamada de forma assíncrona, ou seja, não sabemos ao certo quando a chamada irá ocorrer. Por isso, precisamos retornar **Call<T>**, que é um tipo que representa a chamada e nos permite tratar o retorno quando ele chegar:

// ChatService.java  
  
@POST("polling")  
Call<Void> enviar(@Body Mensagem mensagem);  
  
@GET("polling")  
Call<Mensagem> ouvirMensagens();

Nesse tipo de cenário, geralmente passamos um código que será executado pelo Retrofit quando a resposta chegar ou se algo de errado ocorrer. Mas certamente isso não é uma novidade para você, é exatamente a mesma coisa com o clique no botão, afinal não sabemos ao certo quando ele será clicado.

No Retrofit, chamamos esse código de **Callback** (que é também o nome de uma interface). Portanto vamos criar, no pacote br.com.caelum.ichat.callback, uma implementação dessa interface para tratar o resultado:

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

O que queremos fazer se a resposta chegar com sucesso? Queremos colocar a mensagem na tela e realizar uma nova requisição ao servidor para continuar ouvindo as mensagens, certo? Logo:

public class OuvirMensagensCallback implements Callback<Mensagem> {  
  
 @Override  
 public void onResponse(Call<Mensagem> call, Response<Mensagem> response){  
 if(response.isSuccessful()) {  
 Mensagem mensagem = response.body();  
  
 activity.colocaNaLista(mensagem);  
 activity.ouvirMensagens();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<Mensagem> call, Throwable t) {  
  
 }  
}

E para isso funcionar, vamos precisar de um construtor que receba a MainActivity.

public class OuvirMensagensCallback implements Callback<Mensagem> {  
  
 private MainActivity activity;  
  
 public OuvirMensagemCallback(MainActivity activity) {  
 this.activity = activity;  
 }  
  
 // onResponse e onFailure omitidos  
}

De volta à MainActivity, podemos agora passar a implementação dessa interface no método **enqueue**:

// MainActivity.java  
// método onCreate  
  
Retrofit retrofit = new Retrofit.Builder()  
 // Altere para o seu IP  
 .baseUrl("http://192.168.0.208:8080/")  
 .addConverterFactory(GsonConverterFactory.create())  
 .build();  
  
chatService = retrofit.create(ChatService.class);  
  
Call<Mensagem> call = chatService.ouvirMensagens();  
call.enqueue(new OuvirMensagensCallback(this));

Agora precisamos fazer a mesma coisa no método **colocaNaLista**. Se é a mesma coisa, vamos repetir código, certo? Quando temos esses casos, isolamos o código em um método. Então vamos criar o método **ouvirMensagem**:

// MainActivity.java  
  
public void ouvirMensagem() {  
 Call<Mensagem> call = chatService.ouvirMensagens();  
 call.enqueue(new OuvirMensagensCallback(this));  
}

Agora basta chamar o método **ouvirMensagem** dentro dos métodos **onCreate** e **colocaNaLista** :)

Mas e se um erro ocorrer na requisição, devemos nos recuperar rapidamente e voltar a ouvir as mensagens. Então o método **onFailure** da classe **OuvirMensagensCallback** ficará assim:

public class OuvirMensagensCallback implements Callback<Mensagem> {  
  
 // onResponse comentado  
  
 @Override  
 public void onFailure(Call<Mensagem> call, Throwable t) {  
 activity.ouvirMensagem();  
 }  
}

Falta fazermos algo semelhante para **enviar** as mensagens. É isso que faremos no próximo vídeo :)