Dagger, me entregue um EventBus!

Agora já conseguimos utilizar o **EventBus** para comunicar o resultado da nossa aplicação, só que nós ainda temos que "correr atrás" do EventBus para fazer o registro, o post, etc:

EventBus.getDefault().register(this);

Seria interessante que nós conseguíssemos **injetar** o **EventBus** com o Dagger:

public class MainActivity extends AppCompatActivity {  
  
 // outros atributos  
  
 @Inject  
 EventBus eventBus;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
  
 // código omitido  
 eventBus.register(this);  
  
 }  
  
 @Override  
 protected void onStop() {  
 super.onStop();  
  
 eventBus.unregister(this);  
 }  
  
 // código omitido  
}

## **Ensinando o Dagger a criar o EventBus**

Só que agora precisamos ensinar o Dagger a criar esse EventBus para nós. E faremos isso no **ChatModule**, criando o método **getEventBus()** e utilizando o *builder* do EventBus:

// ChatModule.java  
  
@Provides  
public EventBus getEventBus() {  
 return EventBus.builder().build();  
}

Por último, temos que modificar a classe **OuvirMensagensCallback** para que use o EventBus criado pelo Dagger, então vamos recebê-lo no construtor e depois utilizá-lo:

public class OuvirMensagensCallback implements Callback<Mensagem> {  
  
 private MainActivity activity;  
 private EventBus eventBus;  
  
 public OuvirMensagensCallback(EventBus eventBus, MainActivity activity) {  
 this.activity = activity;  
 this.eventBus = eventBus;  
 }  
  
 @Override  
 public void onResponse(Call<Mensagem> call, Response<Mensagem> response) {  
 if(response.isSuccessful()) {  
 Mensagem mensagem = response.body();  
  
 eventBus.post(new MensagemEvent(mensagem));   
 }  
  
 }  
  
 // onFailure omitido  
  
}

Essa alteração quebra o método **ouvirMensagem** da classe **MainActivity**, então vamos alterá-lo para voltar a funcionar corretamente, passando o **eventBus** por parâmetro para o construtor da classe **OuvirMensagemCallback**:

// MainActivity.java  
  
@Subscribe  
public void ouvirMensagem(MensagemEvent mensagemEvent) {  
 Call<Mensagem> call = chatService.ouvirMensagens()  
 call.enqueue(new OuvirMensagemCallback(eventBus, this));  
}

Podemos rodar a aplicação e ver que tudo continua funcionando normalmente, mas agora com o **Dagger** injetando o **EventBus** :)

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

* A importância de usar eventos para eliminar o acoplamento entre a Activity e o Callback;
* Disparar eventos com LocalBroadcast;
* Disparar eventos de forma mais simples com o EventBus.