Injetando componentes com Butterknife

*Começando deste ponto? Você pode fazer o* [*DOWNLOAD*](https://s3.amazonaws.com/caelum-online-public/android-chat/stages/capitulo-5.zip) *do projeto completo do capítulo anterior e continuar seus estudos a partir deste capítulo.*

Conseguimos diminuir bastante o acoplamento com o uso de Injeção de dependências, mas a forma com que construimos a nossa tela pode ser um problema à medida que o projeto for crescendo. O problema é que ainda precisamos buscar os componentes através de **findViewById(..)**, além de trabalhar com várias classes anônimas para atribuir comportamento aos componentes (*listeners*), que traz problemas de legibilidade e manutenção.

Como esse problema é bem comum e afetou diversos desenvolvedores, várias bibliotecas foram desenvolvidas para resolver iss. Nesse projeto usaremos a **ButterKnife**.

Então, vamos adicionar as suas dependências ao projeto, no **build.gradle** (Módulo):

dependencies {  
 ...  
 compile 'com.jakewharton:butterknife:8.1.0'  
 apt 'com.jakewharton:butterknife-compiler:8.0.1'  
 ...  
}

Agora, vamos resolver o primeiro problema que citamos, que foi a quantidade de **findViewById(.)**que estamos fazendo em nosso código. Para isso faremos uso de uma das facilidades do ButterKnife, que é o uso de *Annotations*.

Como queremos injetar uma **View** usaremos a annotation **@BindView\*, que fará a busca no XML atráves de um** id\*\*:

public class MainActivity extends Activity {  
  
 @BindView(R.id.lv\_mensagens)  
 ListView listaDeMensagens;  
  
 @BindView(R.id.et\_texto)  
 EditText editText;  
  
 @BindView(R.id.btn\_enviar)  
 Button button;  
  
 // restante do código comentado  
  
}

Agora podemos remover as linhas abaixo do método onCreate:

// MainActivity.java  
  
listaDeMensagens = (ListView) findViewById(R.id.lv\_mensagens);  
editText = (EditText) findViewById(R.id.et\_texto);  
button = (Button) findViewById(R.id.btn\_enviar);

Podemos ver uma melhora significativa no nosso código, mas somente com essas configurações vamos receber um NullPointerException quando usarmos algum componente pela primeira vez. No nosso caso na linha onde estamos *setando* o *listener*:

button.setOnClickListener(...);

Isso acontece porque ainda não especificamos para o ButterKnife onde ele deve fazer a busca daquele componente. Para fazer isso, basta invocarmos o método bind() da classe ButterKnife, passando a referência da Activity como parâmetro:

// MainActivity.java   
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 ButterKnife.bind(this);  
  
 ChatApplication app = (ChatApplication) getApplication();  
 component = app.getComponent();  
 component.inject(this);  
  
 // restante do código omitido  
  
}

## **Eliminando Listeners**

Com ButterKnife, também podemos eliminar o *listener* do nosso código, usando a anotação @OnClick. Queremos que, ao clicarem no botão R.id.btn\_enviar, a mensagem seja enviada, portanto vamos criar o método **enviarMensagem** anotado:

@OnClick(R.id.btn\_enviar)  
public void enviarMensagem() {  
 chatService.enviar(new Mensagem(idDoCliente, editText.getText().toString())).enqueue(new EnviarMensagemCallback());   
}

E com isso, podemos remover também as linhas abaixo:

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

# **O que aprendemos nesse capítulo?**

* Injeção de dependências ajuda a diminuir o acoplamento entre classes;
* Como aplicar Injeção de dependências no Andorid usando Dagger;
* Melhorar a visibilidade e a manutenibilidade do nosso código usando ButterKnife.