06

# Adicionando log nas requisições do Retrotif

Caso você não tenha o projeto com as alterações realizadas na aula passada, você pode baixá-lo por meio deste link (https://github.com/alura-cursos/android-sync/archive/15bfb8b77a5a2d32ad4d998e84556681b09988d5.zip).

Nesta aula vimos que quando tentamos adicionar um novo registro na nossa App a partir de um segundo celular nada acontece no servidor. Por que será? Do jeito que está atualmente não sabemos, afinal não temos nenhum feedback que nos indique o que aconteceu! Em outras palavras, precisamos configurar a nossa App, ou melhor, o responsável em realizar as requisições, que no caso é o Retrofit, para que ele nos diga o que está acontecendo.

## Entendendo as possibilidades para log das requisições

O Retrofit por si só não tem uma configuração para isso! Porém, ele tem uma alternativa que seria pegar as informações que vem no `Callback`, ou seja, fazer uso tanto do parâmetro de `call` como `response` do método `onResponse()`, por exemplo. É uma opção válida, porém, trabalhosa, pois precisaremos nos atentar a isso a cada requisição que criarmos a partir do Retrofit. Em outras palavras, o ideal seria adicionar algum tipo de filtro que ficasse sempre observando quando acontecesse uma requisição, e então, ele nos detalhasse todo o ciclo de vida da mesma.

Felizmente existe uma biblioteca que resolve esse problema para nós! Essa lib é conhecida como ***logging interceptor*** (https://github.com/square/okhttp/wiki/Interceptors) e faz parte de um módulo da biblioteca **OkHttp** (http://square.github.io/okhttp/) que também é uma biblioteca da Square Up. Vimos em aula que o Retrofit tem como core o OkHttp, ou seja, ele faz uso do OkHttp internamente para realizar de fato as requisições HTTP, portanto, faremos uso de um client do OkHttp com um logging interceptor configurado e atribuiremos ao Retrofit!

## Adicionando o logging interceptor no projeto

Para adicionarmos o logging interceptor faremos da mesma forma como fizemos com o Retrofit e o plugin do Jackson, ou seja, vá até **File** > **Project Structure...** > **app** > Aba Dependencies > clique no maisinho verde > **Library Dependency** > busque por **logging-interceptor**. Aparecerá a biblioteca `com.squareup.okhttp3:logging-interceptor:3.5.0`. Selecione essa dependência no projeto e aguarde o gradle terminar o procedimento de sincronização com o projeto.

## Configurando o logging interceptor

Agora que temos o logging interceptor no projeto basta apenas configurarmos o client do OkHttp, para isso vá até a classe `RetrofitInicializador`. Em seguida, dentro do construtor, faça instância da classe `HttpLoggingInterceptor` e atribua para o objeto chamado `interceptor` do tipo `HttpLoggingInterceptor` (lembrando que pode fazer isso por meio do **Alt + Enter** como vimos em aula).

## Indicando nível de log

A única configuração que precisamos realizar é justamente indicar o nível de log que desejamos. Para indicar o nível basta apenas chamar o método `setLevel()` do objeto `interceptor`. Portanto, chame o método `setLevel()`.

Em seguida precisaremos enviar o nível a partir do enum `HttpLoggingInterceptor.Level`. O valor padrão do interceptor é `Level.NONE` que indica que não teremos log algum. Neste caso queremos informações mais detalhes da requisição, como por exemplo, url, status code, body e qualquer outra informação que nos ajude a entender o que está acontecendo, certo? Portanto, adicione o nível com o valor `HttpLoggingInterceptor.Level.BODY`. Caso precise saber mais sobre os níveis basta consultar o Javadoc da API (https://square.github.io/okhttp/3.x/logging-interceptor/).

## Adicionando o logging interceptor no OkHttp

Com o interceptor já configurado em mãos precisamos apenas criar uma instância de um client do OkHttp, portanto, faça instância da classe `OkHttpClient.Builder()` e atribua para uma variável chamada `client`. Então, adicione o interceptor para o objeto `client` por meio do método

`addInterceptor()` e envie o interceptor por parâmetro.

### Adicionando client com logging interceptor no Retrofit

Por fim, basta apenas adicionar o client do OkHttp no Retrofit. Para isso chame o método `client()` e envie o parâmetro `client.build()` para que enviemos esse nosso client do OkHttp. Essa chamada deve ser dentro daquelas chamas de métodos do `Retrofit.Builder` . Lembrando que precisa ser chamado antes do método `build()` pois quando o chamamos finalizamos o processo de construção do Retrofit.

### Testando App

Agora basta apenas executar a App novamente e tentar enviar novamente o aluno com o segundo celular para o servidor.