*Como implementar um chat?*

*Começando deste ponto? Você pode fazer o* [*DOWNLOAD*](https://s3.amazonaws.com/caelum-online-public/android-chat/stages/capitulo-2.zip) *do projeto completo do capítulo anterior e continuar seus estudos a partir deste capítulo.*

Nesse treinamento, vamos trabalhar em uma aplicativo de Chat onde aprenderemos a trabalhar com alguns frameworks do Android para fazer as mais diversas tarefas como carregar imagens, consumir Endpoints REST sobre HTTP (nossa API de chat) e injetar nossas dependências.

## **Mas calma aí. Como implementar um chat?**

Não, você não leu errado. Eu escrevi **Endpoints REST sobre HTTP** mesmo. Parece óbvio que ao falar de REST estamos falando também de HTTP mas, na verdade, eu quis causar uma pequena dúvida em você.

Se eu consegui, provavelmente você deve estar pensando: o HTTP é um protocolo de requisição e resposta, pelo menos foi isso que me disseram no *curso de HTTP que fiz* [*aqui na Alura*](https://cursos.alura.com.br/course/fundamentos-http/section/4/task/2). Como será que vou conseguir criar um canal de comunicação ***real-time*** com o meu cliente?

O nome é bem bonito, mas as soluções (ou técnicas e existem várias delas!) são bem simples.

Uma das técnicas consiste em fazer diversas requisições em curtos intervalos de tempo. Dessa forma sempre uma nova mensagem chega ao servidor, recebemos como resposta da requisição. O lado ruim dessa técnica é que se as requisições forem feitas em um período que o usuário não está enviando mensagens, elas não terão uma mensagem na resposta. Sendo assim, essa opção se torna inimiga da escalabilidade.

Bom, mas já que trabalhamos com HTTP, requisições não deveriam ser um problema, certo? Por isso, é importante deixar claro que o problema real está no fato de várias requisições serem feitas e retornarem sem mensagem.

Podemos mudar um pouco esse cenário, segurando as requisições no servidor até que elas tenham uma mensagem pra retornar (ou ocorrer um *timeout*). Dessa forma, repare que conseguimos economizar muita das requisições feitas na solução anterior. Essa técnica é o que costumamos chamar de **Long Polling** ou **Comet**.

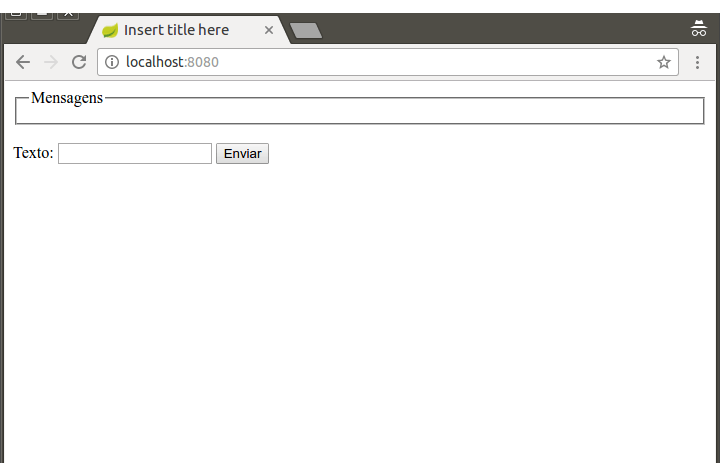
Aqui no curso vamos entregar uma API para que você não precise se preocupar em implementar toda essa bagagem, que não é nosso foco. Mas já adianto que a implementação foi feita em Java e você pode ver o código [aqui](https://github.com/leonardocordeiro/ichat-api).

## **Rodando a API**

Já temos a base do nosso projeto, mas antes começarmos a trabalhar, precisamos rodar uma API, o **ichat-api.jar**, você pode baixá-lo [aqui](https://s3.amazonaws.com/caelum-online-public/android-chat/files/ichat-api.jar). É uma aplicação feita em **Spring Boot** (inclusive temos o curso [**Spring Boot: Agilidade no desenvolvimento java com Spring**](https://cursos.alura.com.br/course/springboot) aqui na Alura, caso você queira saber mais sobre o assunto), e para rodá-la, execute o seguinte comando no seu terminal:

java -jar ichat-api.jar

O Spring Boot irá subir tudo para nós. Após a execução, podemos ir no navegador e digitar a seguinte URL: [http://localhost:8080](http://localhost:8080/).



Se abrirmos duas páginas, por exemplo, para simular dois clientes, digitar um texto no campo, ao clicar em **Enviar** veremos que a mensagem aparecerá no bloco acima, das duas páginas!

Vamos simular uma requisição a partir de uma outra fonte. No nosso caso, vamos usar a ferramenta **curl** do terminal Unix. (Para Windows, faça o download da ferramenta [aqui](https://s3.amazonaws.com/caelum-online-public/android-chat/files/curl_751_0.zip))

Então vamos ao terminal e fazer uma requisição para <http://localhost:8080/polling>, enviando um mensagem no formato JSON:

curl -X POST -H "Content-type: application/json" -d '{"text":"Olá aluno de Android!!!","id":123}' http://localhost:8080/polling

OBS: No Windows, pode ser necessário escapar o ". Dessa forma:

curl -X POST -H "Content-type: application/json" -d '{\"text\":\"Olá aluno de Android!!!\",\"id\":123}' http://localhost:8080/polling

Execute o comando acima e olhe novamente para a página do projeto. Repare que a mensagem **Olá aluno de Android!!!** apareceu no box :)

Da mesma forma, podemos fazer o contrário, podemos fazer uma requisição que irá **receber** uma mensagem:

curl -X GET -H "Accept: application/json" http://localhost:8080/polling

Repare que o terminal fica travado, exatamente como esperávamos, aguardando por mensagens (até um *timeout* de 20 segundos).

Agora faça o teste: Acesse novamente a página [http://localhost:8080](http://localhost:8080/) e envie uma mensagem. Repare que o terminal é liberado e a resposta aparece logo em seguida. Algo como:

