Recebendo mensagens

Já sabemos enviar mensagens, agora temos que recebê-las! Para implementar o recebimento das mensagens, vamos implementar o mesmo GET que fizemos anteriormente com curl. Vamos agora para nossa classe ChatService e crie um método que se chamará ouvirMensagens, até então sem implementação:

// ChatService.java  
  
public void ouvirMensagens() {   
  
}

Novamente vamos usar a API do java.io e java.net. Lembrando que o código deve que ficar dentro de uma outra Thread, ok? :)

// ChatService.java  
  
public void ouvirMensagens() {  
  
 new Thread(new Runnable() {  
  
 @Override  
 public void run() {  
 try {  
 // Altere para o seu IP  
 URL url = new URL("http://192.168.0.208:8080/polling");  
  
 HttpURLConnection httpConnection = (HttpURLConnection) url.openConnection();  
  
 httpConnection.setRequestMethod("GET");  
  
 httpConnection.setRequestProperty("Accept", "application/json");  
  
 httpConnection.connect();  
  
 Scanner scanner = new Scanner(httpConnection.getInputStream());  
 StringBuilder builder = new StringBuilder();  
  
 while(scanner.hasNextLine()) {  
 builder.append(scanner.nextLine());  
 }  
  
 String json = builder.toString();  
  
 JSONObject jsonObject = new JSONObject(json);  
 Mensagem mensagem = new Mensagem(jsonObject.getInt("id"), jsonObject.getString("text"));  
  
 // o que fazer com essa mensagem?  
  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }).start();  
}

De alguma forma é preciso jogar essa mensagem na tela, só que essa não é a responsabilidade do ChatService e sim da MainActivity já que é ela a classe responsável pela tela. Por isso vamos precisar de uma referência da MainActivity dentro do ChatService.

Crie um construtor para a classe ChatService que receberá uma MainActivity e a associará a um atributo:

// ChatService.java  
  
private MainActivity activity;  
  
public ChatService(MainActivity activity) {   
 this.activity = activity;  
}

Agora, no **ouvirMensagens**, vamos colocar a mensagem na lista:

// ChatService.java  
  
public void ouvirMensagens() {  
  
 new Thread(new Runnable() {  
  
 @Override  
 public void run() {  
 try {  
 // Altere para o seu IP  
 URL url = new URL("http://192.168.0.208:8080/polling");  
  
 HttpURLConnection httpConnection = (HttpURLConnection) url.openConnection();  
  
 httpConnection.setRequestMethod("GET");  
  
 httpConnection.setRequestProperty("Accept", "application/json");  
  
 httpConnection.connect();  
  
 Scanner scanner = new Scanner(httpConnection.getInputStream());  
 StringBuilder builder = new StringBuilder();  
  
 while(scanner.hasNextLine()) {  
 builder.append(scanner.nextLine());  
 }  
  
 String json = builder.toString();  
  
 JSONObject jsonObject = new JSONObject(json);  
 Mensagem mensagem = new Mensagem(jsonObject.getInt("id"), jsonObject.getString("text"));  
  
 // Colocando a mensagem na lista, mas esse método não existe ainda, certo?  
 activity.colocaNaLista(mensagem);  
  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }).start();  
}

Na **MainActivity**, vamos criar o método **colocaNaLista(Mensagem mensagem)**, que adiciona a mensagem na lista e cria um novo MensagemAdapter:

// MainActivity.java  
  
public void colocaNaLista(Mensagem mensagem) {  
 mensagens.add(mensagem);  
 MensagemAdapter adapter = new MensagemAdapter(idDoCliente, mensagens, this);  
 listaDeMensagens.setAdapter(adapter);  
}

Mas para o método ter acesso às **mensagens** e **listaDeMensagens**, precisamos torná-las atributos da classe:

public class MainActivity extends AppCompatActivity {  
  
 // outros atributos  
  
 private ListView listaDeMensagens;  
 private List<Mensagem> mensagens;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 listaDeMensagens = (ListView) findViewById(R.id.lv\_mensagem);  
  
 mensagens = Arrays.asList(new Mensagem(1, "olá alunos de android"), new Mensagem(2, "oi"));  
  
 // restante do método comentado  
 }  
  
 public void colocaNaLista(Mensagem mensagem) {  
 mensagens.add(mensagem);  
 MensagemAdapter adapter = new MensagemAdapter(idDoCliente, mensagens, this);  
 listaDeMensagens.setAdapter(adapter);  
 }  
}

Agora, para ouvir as mensagens, vamos criar um atributo do **ChatService** e utilizá-lo no método **onCreate** para ouvir as mensagens. Vamos aproveitar para limpar o array de mensagens, deixando somente as mensagens que enviarmos:

public class MainActivity extends AppCompatActivity {  
  
 // outros atributos  
 private ChatService chatService;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 listaDeMensagens = (ListView) findViewById(R.id.lv\_mensagem);  
  
 // limpando as mensagens  
 mensagens = new ArrayList<>();  
  
 MensagemAdapter adapter = new MensagemAdapter(idDoCliente, mensagens, this);  
  
 listaDeMensagens.setAdapter(adapter);  
  
 button = (Button) findViewById(R.id.btn\_enviar);  
  
 editText = (EditText) findViewById(R.id.et\_texto);  
  
 // instanciando um chatService e ouvindo as mensagens  
 chatService = new ChatService(this);  
 chatService.ouvirMensagens();  
  
 button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // utilizando o atributo chatService aqui  
 chatService.enviar(new Mensagem(idDoCliente, editText.getText().toString());  
 }  
 });  
 }  
}

Mas temos um problema, esse código está sendo executado em uma nova Thread e teremos problemas ao tentar atualizar a tela. Portanto, precisamos executar esse código de dentro da **UIThread**. Para isso, podemos utilizar o método **runOnUiThread**:

// ChatService.java  
// dentro do método ouvirMensagens()  
  
activity.runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 activity.colocaNaLista(mensagem);  
 }  
});

Agora podemos testar e enviar uma mensagem. Podemos reparar no navegador que a API recebeu a mensagem e o nosso aplicativo também! Mas e se enviarmos uma segunda mensagem? A API recebe a mensagem, mas o nosso aplicativo não!

Isso acontece porque ao chamarmos o método ouvirMensagens(), ele irá disparar um GET na API e ficará aguardando por mensagens, mas e quando elas chegarem? Vamos precisar colocá-las na lista e voltar de novo à API para esperar mais mensagens!

Portanto, logo no final do método **colocaNaLista**, vamos chamar o método **ouvirMensagens()**novamente:

// MainActivity.java  
  
public void colocaNaLista(Mensagem mensagem) {  
 mensagens.add(mensagem);  
 MensagemAdapter adapter = new MensagemAdapter(idDoCliente, mensagens, this);  
 listaDeMensagens.setAdapter(adapter);  
  
 chatService.ouvirMensagens();  
}

Podemos rodar novamente a aplicação, e ver que agora conseguimos ir ao servidor quando a aplicação é iniciada para poder receber a mensagem, e quando a mensagem chega, vamos novamente ao servidor para poder receber novas mensagens.