Programação para dispositivos móveis

Aula 6: Mensagens e Notificações

INTRODUÇÃO

O conceito de mensagem é amplamente implementado em aplicativos Android. Esta aula visa apresentar os conceitos de mensagem, bem como suas principais classes em Android.

OBJETIVOS

Definir o conceito de mensagens em Android.

Demonstrar as classes Toast, Notification, AlertDialog, ProgressDialog, DatePickerDialog e a TimePickerDialog.

TOAST

Como já discutido em aulas anteriores, a classe Toast(android.widget.Toast) é muito utilizada quando desejamos exibir mensagens de alertas para o usuário.

Esta exibe uma pequena tela, que pode ser personalizada, sem caráter permanente, ou seja, desaparecendo sem qualquer tipo de intervenção do usuário.

Para implementarmos este tipo de mensagem, é necessário obter uma instância de objeto Toast através do método estático, desta mesma classe, makeText(), que possui três parâmetros, conforme demonstrado abaixo:

Toast.makeText (contexto, texto, duração)

• No parâmetro **contexto**, definimos o contexto do aplicativo a ser exibido na mensagem;

• No parâmetro **texto**, definimos o texto a ser exibido na mensagem;

• No parâmetro **duração**, podemos definir o tempo de duração que a mensagem será exibida. A própria classe Toast nos oferece constantes para definirmos este tempo de duração, sendo elas Toast.LENGTH_SHORT e Toast.LENGTH_LONG.

Para exibir essa mensagem, precisamos apenas chamar o método show() do objeto.

Veja os exemplos abaixo:

NOTIFICATIONS

Uma notificação Android nada mais é do que uma mensagem que aparece na barra de status Android. Seu objetivo é notificar ao usuário sobre algum evento, seja através de um ícone na barra de status, através de vibração, som ou acender de luzes do dispositivo.

É muito comum seu uso quando temos uma aplicação que roda em segundo plano, como, por exemplo, um aplicativo de mensagem instantânea, e não podemos ser interrompidos em uma aplicação que roda em primeiro plano, como, por exemplo, uma ligação telefônica.

Opções de notificação
Além de exibir o ícone na barra de status, a classe Notification nos oferece mais três opções para configurar nossa notificação:

VIBRAÇÃO

O dispositivo vibra rapidamente quando a notificação é recebida.

SOM

Soa um alarme, seja um tom de chamada ou um tom gravado, quando a notificação é recebida.

LUZ

Led pisca quando a notificação é recebida.
A partir do Android 4.1 Jelly Bean, as notificações podem possuir os recursos:

VISUALIZAÇÃO EXPANSÍVEL

Podemos expandir uma notificação usando o gesto “apertar e ampliar”.

BOTÕES DE AÇÃO

Não estamos limitados a somente um botão de ação. Podemos adicionar até três botões.

ESTILOS VARIADOS

BigTextStyle: Exibe um TextView com diversas linhas.
BigInboxStyle: Exibe uma lista de informações.

TAMANHO MAIOR

Podem ser tão altas quanto 256 dp.
Para maiores informações você poderá acessar o link:
https://developer.android.com/guide/topics/ui/notifiers/notifications.html (glossário)

Para implementar uma notificação Android, precisamos obter a classe NotificationManager, visto que é um serviço do sistema não só executa, como gerencia as notificações.

As linhas abaixo demonstram os passos necessários para obter-se a NotificationManager:

```
NotificationManager notificationManager =(NotificationManager) getSystemService(Context.NOT
```

Após obtermos um NotificationManager, usamos a classe Notification.Builder para configurar nossa notificação.

Através dela, podemos configurar título, ícone e outros.

Isso é demonstrado no trecho de código abaixo:

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_stat_notification);
builder.setContentIntent(pendingIntent);
builder.setAutoCancel(true);
builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),R.drawable.ic_launcher));
builder.setContentTitle("BasicNotifications Sample");
builder.setContentText("Time to learn about notifications!");
builder.setSubText("Tap to view documentation about notifications.");
```

Devemos implementar, no mínimo, os métodos setSmallIcon(ícone pequeno), setContentTitle(título da notificação) e setContextText(Texto da Notificação).

Finalizando, implementamos um PendingIntent para o nosso builder.

Devemos criar somente um PendingIntent para todas nossas notificações.

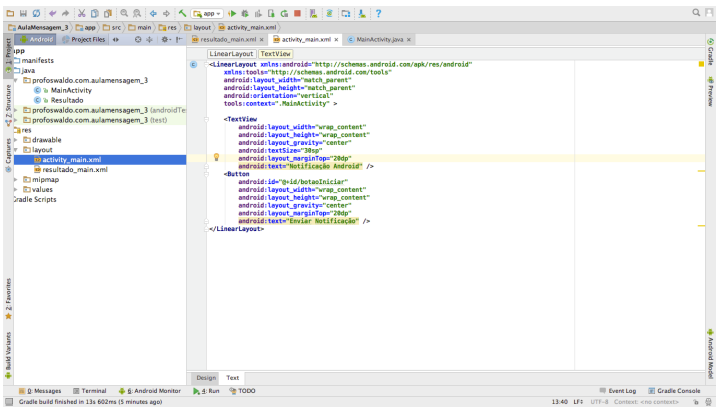
Para facilitar o entendimento, veja o exemplo abaixo:

EXEMPLO - 2.2 Exemplo Notification Android

Em nosso exemplo, quando clicarmos no botão **Enviar Notificação**, será exibido na barra de status do Android um pequeno ícone, indicado pela seta azul.

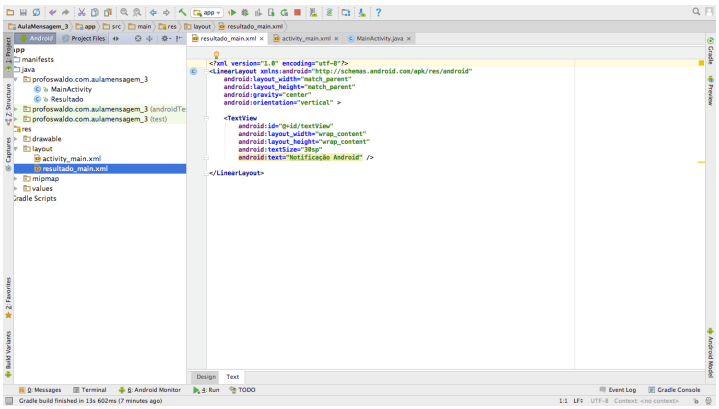
Ao clicarmos neste **ícone**, abrirá uma “gaveta de notificações”. Basta selecionar uma e clicar para ler a notificação por completo.

Em nosso projeto, primeiro configuramos o layout da tela inicial no arquivo activity_main.xml, conforme ilustrado abaixo:



Nosso exemplo é bastante simples. Possui apenas um TexView para identificarmos a tela principal, através da mensagem “Notificação Android”, e um botão que, quando clicado, enviará uma notificação.

Abaixo, veja a ilustração do layout de nossa tela secundária que será chamada quando clicarmos na notificação.

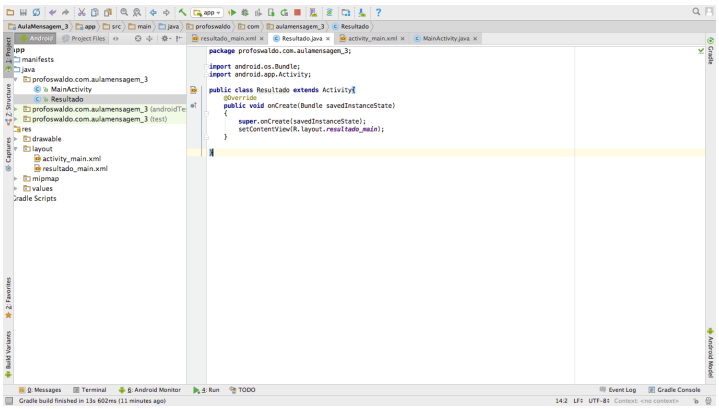


Esta tela possuirá somente um texto.

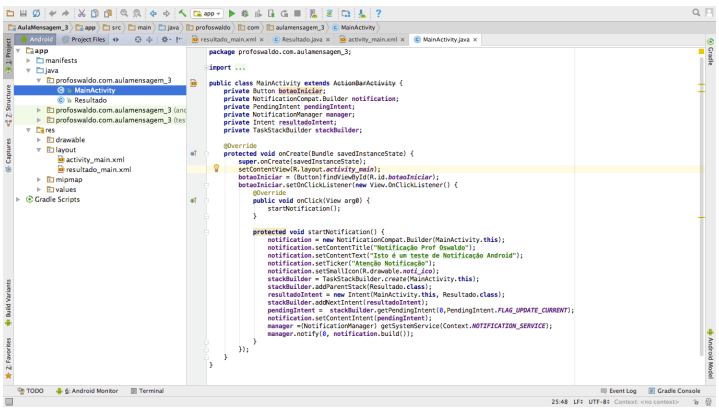
Chegou a vez de programar a nossa activity secundária que será chamada quando selecionarmos a notificação.

Lembrando que também poderia ser um BroadCast Receiver ou um Service. Vai depender do parâmetro configurado na Intent.

Abaixo segue a acitivy Resultado.java.



Por fim, precisamos programar também nossa activity principal. Abaixo segue o código de nosso arquivo MainActivity.java:

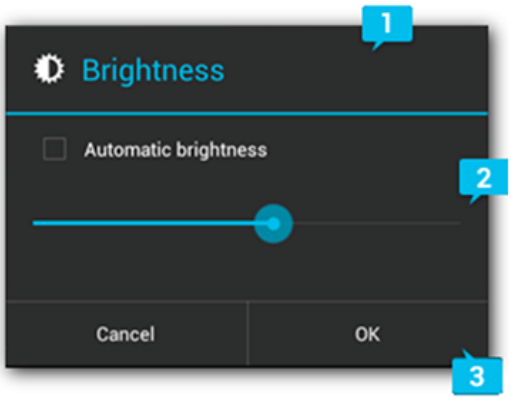


ALERTDIALOG

Esta classe pode exibir uma caixa de diálogo com um título de até três botões.

Diferente da classe Toast, podemos interagir com o usuário pressionando botões.

Existem três regiões na AlertDialog que devemos configurar, conforme ilustrado na imagem abaixo:



1. Título:
Embora seja opcional, é onde definimos o título de nossa janela.

2. Mensagem:
A mensagem de nossa janela.

3. Botões de ação:
Os botões que estarão disponíveis em nossa janela. Não podemos nos esquecer de que não deve haver mais de três.

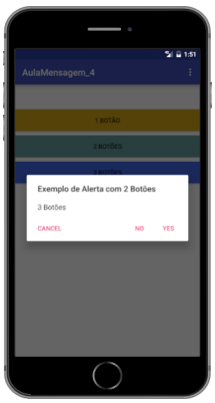
Finalizando, para criar nossa AlertDialog, fazemos uso da classe AlertDialog.Builder.

EXEMPLO - AlertDialog

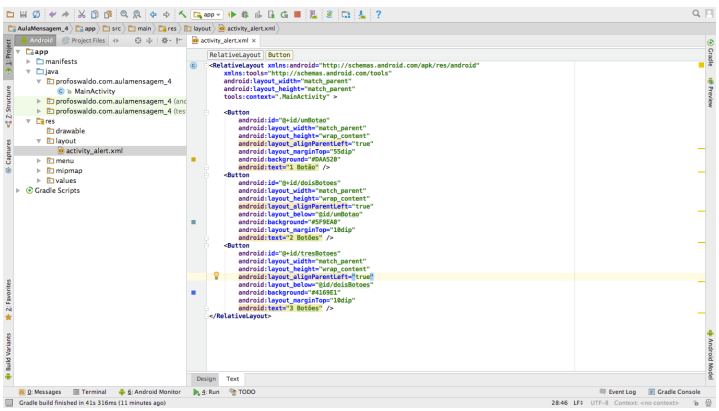
A melhor forma de entendermos o AlertDialog é com um exemplo.

Como demonstrado nas telas abaixo, quando clicarmos no botão 1 BOTÃO, será exibida a segunda tela que possui uma caixa de diálogo com apenas um botão.

O mesmo raciocínio é seguido para os demais botões.

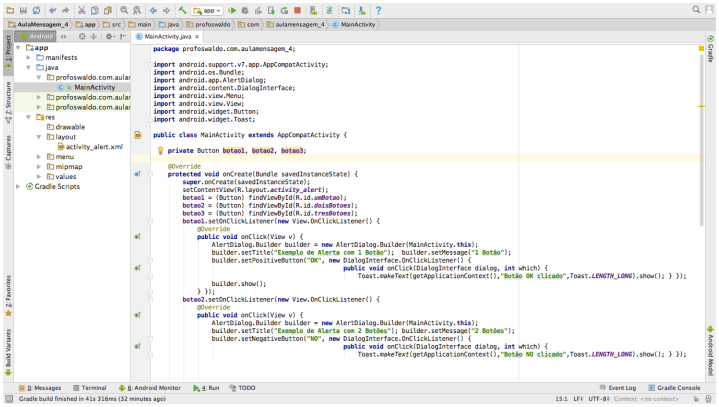


Criamos no arquivo activity_main.xml três botões na tela principal. Cada um deles será responsável por exibir um tipo de AlertDialog.

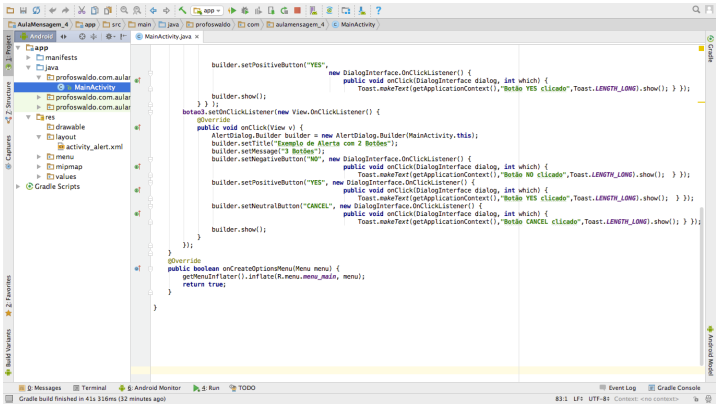


Agora chegou a vez de nossa classe principal.

Veja o arquivo MainActivity.java demonstrado abaixo:



Não foi possível exibir o código todo desse arquivo em somente uma imagem. A tela abaixo complementa este código:



Observe a linha abaixo:

```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
```

Podemos verificar que nossa AlertDialog foi criada no contexto de nossa activity(MainActivity), ou seja, no contexto que será exibida.

Usamos ***MainActivity***.this para parametrizar este contexto.

Nesse mesmo exemplo poderíamos ter usado ***v.getContext()***, pois o contexto de nossa view parametrizada no método onClick() é o mesmo da nossa activity.

Existem várias formas de fazer essa referência, inclusive exploramos isso nos outros tipos de caixa de diálogo discutidos a seguir.

Além disso, a AlertDialog possui vários métodos.

Podemos destacar os seguintes:

setTitle
Define o título de nossa AlertDialog

setMessage
Define a mensagem de nossa AlertDialog

setPositiveButton
Define um botão para aceitar e continuar uma ação (Exemplo: OK)

setNeutralButton
Define um Botão que não cancela e nem confirma uma ação (Exemplo: Notifique-me mais tarde)

setNegativeButton
Define um botão para cancelar a ação (Exemplo: Cancel)

show()
Exibe a AlertDialog

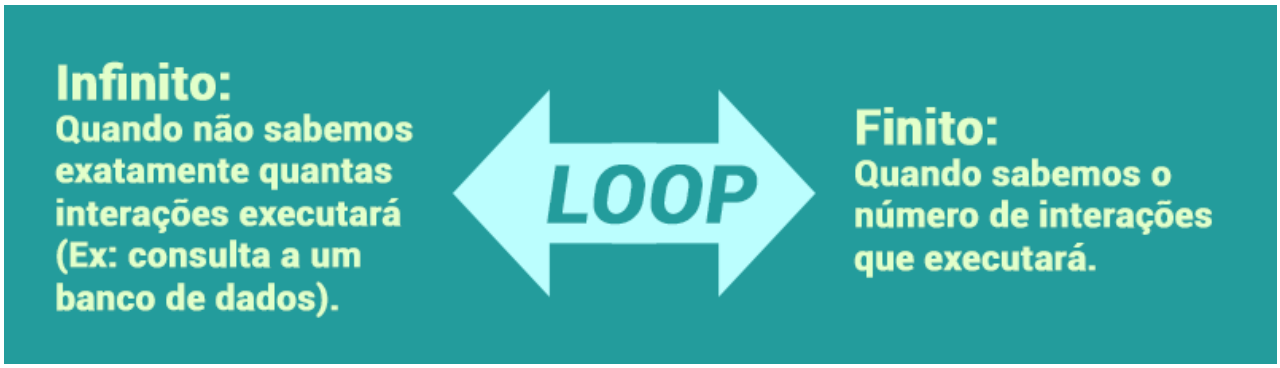
PROGRESSDIALOG

Este tipo de caixa de diálogo exibe um indicador de progresso e uma mensagem de texto opcional.

Seu range de progresso é de 0 a 10000.

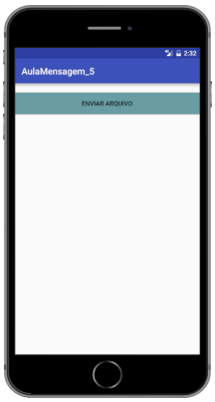
Quando o sistema for efetuar uma operação demorada, podemos implementar a ProgressDialog. Com isso, o usuário não pensará que o aplicativo travou.

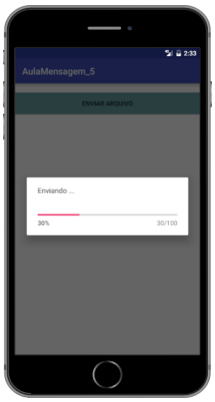
Existem dois tipos de ProgressDialog que podemos implementar:



EXEMPLO - ProgressDialog

Em nosso exemplo, ao clicarmos no botão Enviar arquivo, será exibida uma ProgressDialog com um range de 0 a 100.

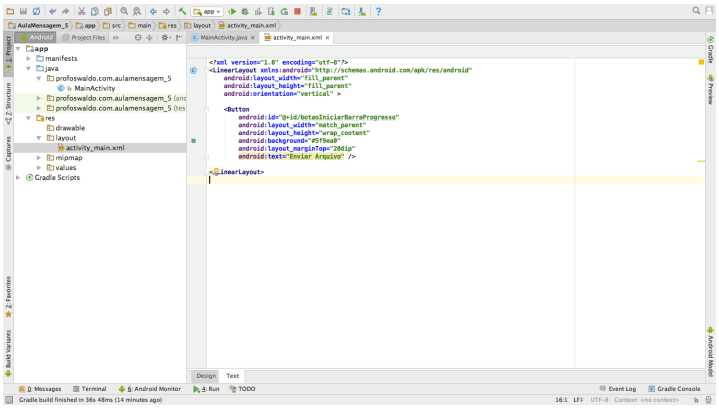




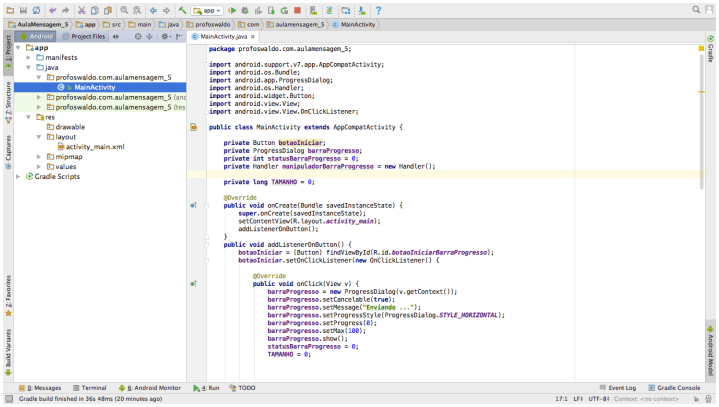
Vamos iniciar pelo layout de nossa tela.

Essa possui somente um botão que chamará nossa ProgressDialog.

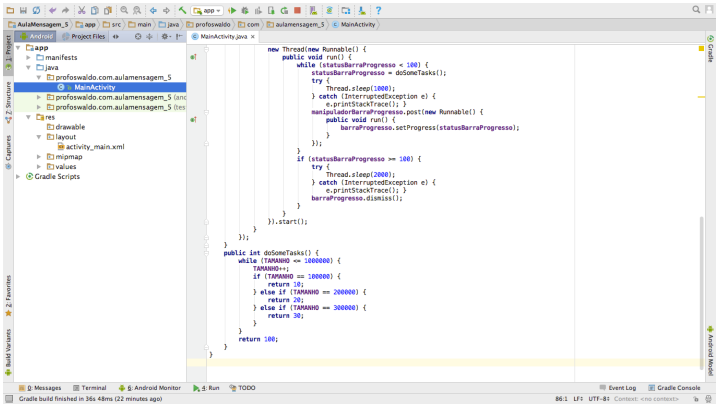
Veja a ilustração abaixo:



Agora estamos demonstrando abaixo o código de nossa MainActivity.java.



Mais uma vez, dividimos o código em duas imagens devido ao seu tamanho.



Observe as linhas abaixo:

```
barraProgresso = new ProgressDialog(v.getContext());
barraProgresso.setCancelable(true);
barraProgresso.setMessage("Enviando ...");
barraProgresso.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
barraProgresso.setMax(100);
barraProgresso.show();
```

A criação de nossa barra de progresso é bastante simples. Bastou apenas passar por parâmetro o contexto de nossa atividade, no exemplo, através do v.getContext().

Assim como a AlertDialog, poderíamos efetuar esta parametrização de forma diferente, lembrando sempre que não é o contexto de nosso aplicativo, como no caso da Toast, mas de nossa activity.

Existem vários outros métodos, mas em nosso exemplo implementamos:

• **setCancelable():** Define se ProgressDialog pode ser cancelada ou não com a back Key. As opções, de parâmetros são True ou False.

• **setMessage:** Define a mensagem de nossa ProgressDialog.

• **setProgressStyle:** Define o estilo de nossa ProgressDialog, podendo ser STYLE_HORIZONTAL(Loop Finito) ou STYLE_SPINNER(Loop Infinito).

• **setMax:** valor máximo do range da ProgressDialog.

• **show:** exibir a ProgressDialog.

DATEPICKERDIALOG

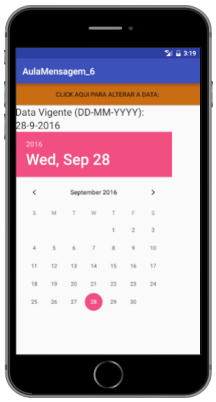
É muito comum precisarmos configurar datas em nossos aplicativos. Para tanto, o Android nos disponibiliza a classe DatePicker, que nos permite selecionar uma data.

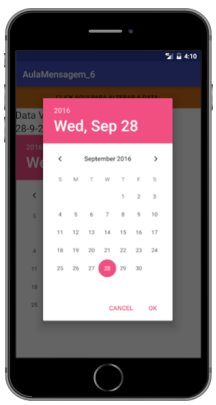
Embora a DatePicker possa ser usada como um widgets independente, ela ocupa muito espaço. É prudente, então, implementá-la dentro de uma caixa de diálogo.

Pensando nisso, o Android nos poupou trabalho fornecendo a classe DatePickerDialog, que é bastante simples de se implementar.

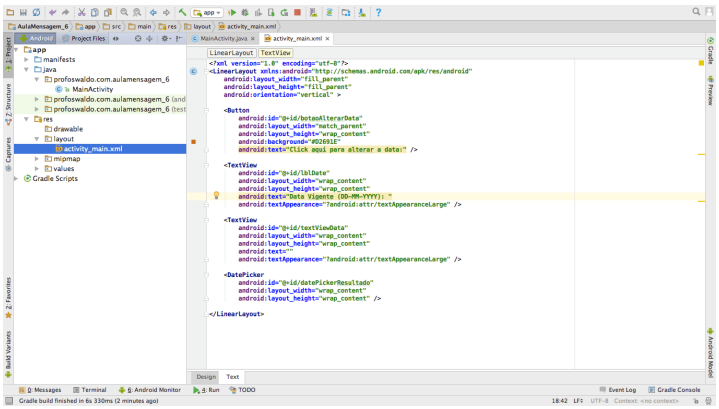
EXEMPLO - DatePickerDialog

Em nosso exemplo, ao clicarmos no botão marrom, será exibida a segunda tela, onde poderemos alterar a data.

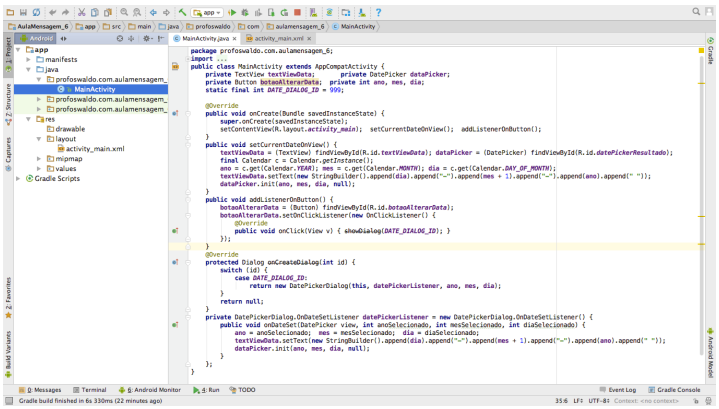




Nosso layout é bastante simples. A única novidade que verificamos é o componente DatePicker.



Agora chegou a vez de nossa MainActivity.java, ilustrada na tela abaixo:



Observe que precisamos da instância da classe Calendar para ter acesso à data de nosso sistema, conforme destacado no trecho de código abaixo:

```
final Calendar c = Calendar.getInstance();
ano = c.get(Calendar.YEAR);
mes = c.get(Calendar.MONTH);
dia = c.get(Calendar.DAY_OF_MONTH);
```

Para exibirmos a data de nosso sistema, usamos o método init de nossa classe DatePicker, conforme demonstrado no trecho de código abaixo:

```
dataPicker.init(ano, mes, dia, null);
```

TIMEPICKERDIALOG

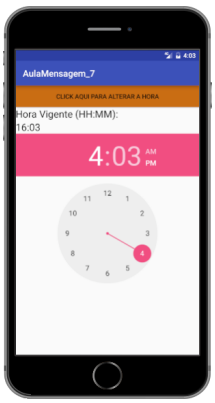
Bastante similar à DatePickerDialog, a classe TimerPickerDialog nos permite manipular o horário de nosso sistema.

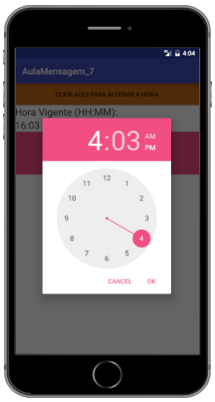
O exemplo abaixo tem por objetivo consolidar nosso entendimento referente a essa classe:

EXEMPLO - TimePickerDialog

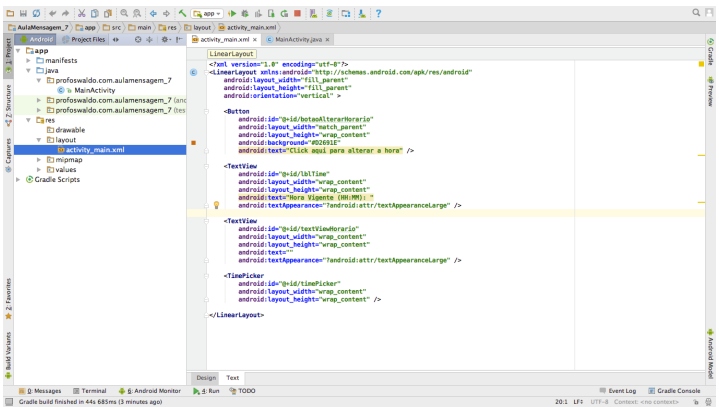
Assim como no exemplo referente à DatePickerDialog, basta clicar no botão marrom e a TimePickerDialog será exibida.

Nela, poderemos alterar a hora e o minuto e escolher entre AM/PM para nosso horário.

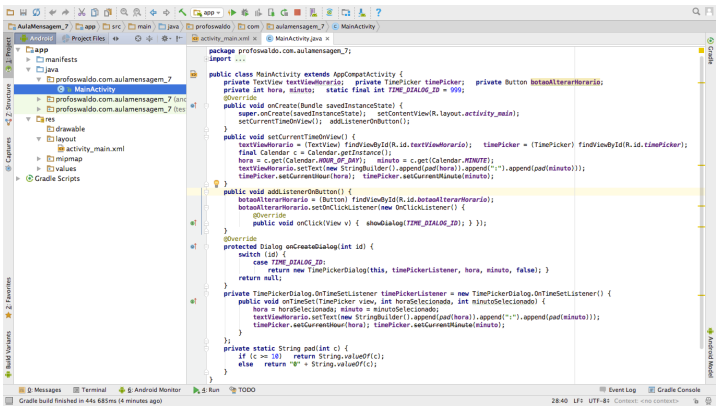




Nosso layout é bastante simples, tendo como novidade o componente TimePicker.



Abaixo estamos implementando a nossa atividade principal.

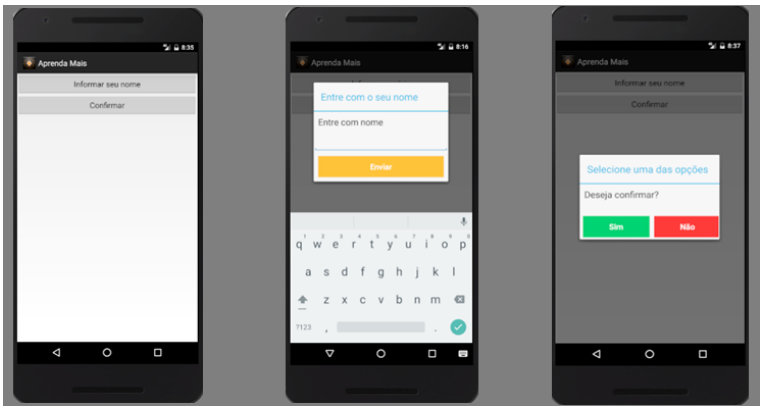


Como comentamos anteriormente, nosso código é bastante parecido. Apenas não estamos mais configurando dia, mês e ano, mas hora e minuto, conforme demonstrado no trecho de código abaixo:

```
timePicker.setCurrentHour(hora);  timePicker.setCurrentMinute(minuto);
```

ATIVIDADE

Observe as imagens:



Desenvolva um pequeno aplicativo que, como demonstrado nas telas acima, ao clicarmos no botão Informar o usuário, deverá exibir a segunda tela.

Nela, você digitará o seu nome e, ao clicar no botão Enviar, uma mensagem(Toast) será exibida informando que o seu nome foi digitado. Porém se, na primeira tela, clicarmos no botão Confirmar, será exibida a terceira tela. Essa também deverá exibir uma mensagem(Toast) confirmando ou não o envio do nome.

Resposta Correta

Glossário