PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Prof. Luiz Carlos Querino Filho
luiz.querino@fatec.sp.gov.br

Fatec Garça – 2019

pdm-03

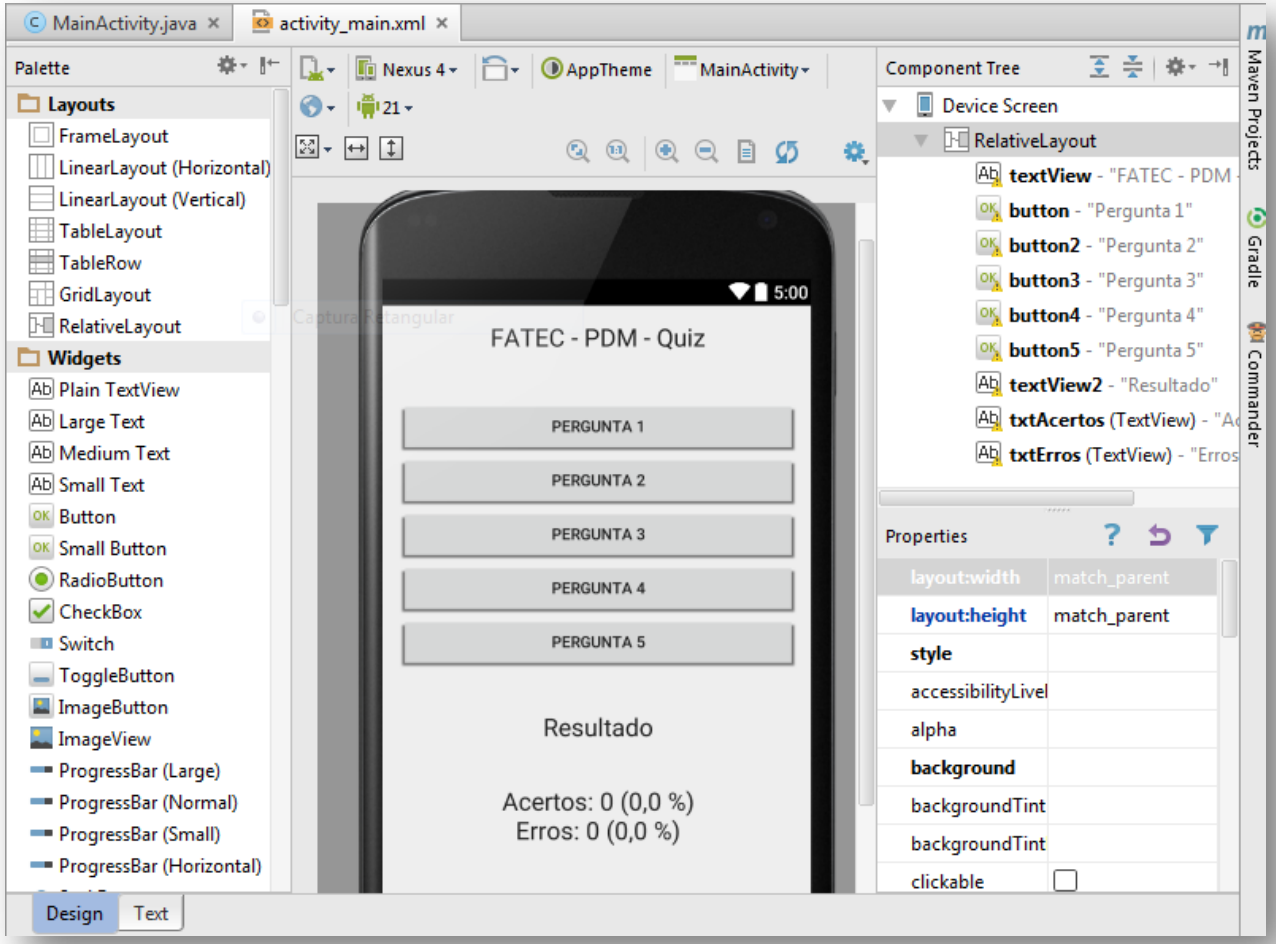
ADICIONANDO NOVAS TELAS AO PROJETO

Uma nova tela no seu aplicativo

- Ter uma nova tela no projeto envolve as seguintes ações:
  - Criar uma nova subclasse de **ActionBarActivity** (ou outra Activity base existente no Android) para atuar como controladora da nova tela.
  - Criar um novo **arquivo XML** de layout para esta tela.
  - Vincular estes dois elementos usando o método **setContentView** no evento **onCreate** da **Activity** em questão.
  - Declarar a nova **Activity** no arquivo `AndroidManifest.xml`
- Tudo isso pode ser feito manualmente ou utilizando o ***assistente de criação*** de uma nova **Activity** do Android Studio.
- O assistente realiza todas as etapas listadas acima automaticamente.

Quiz Fatec

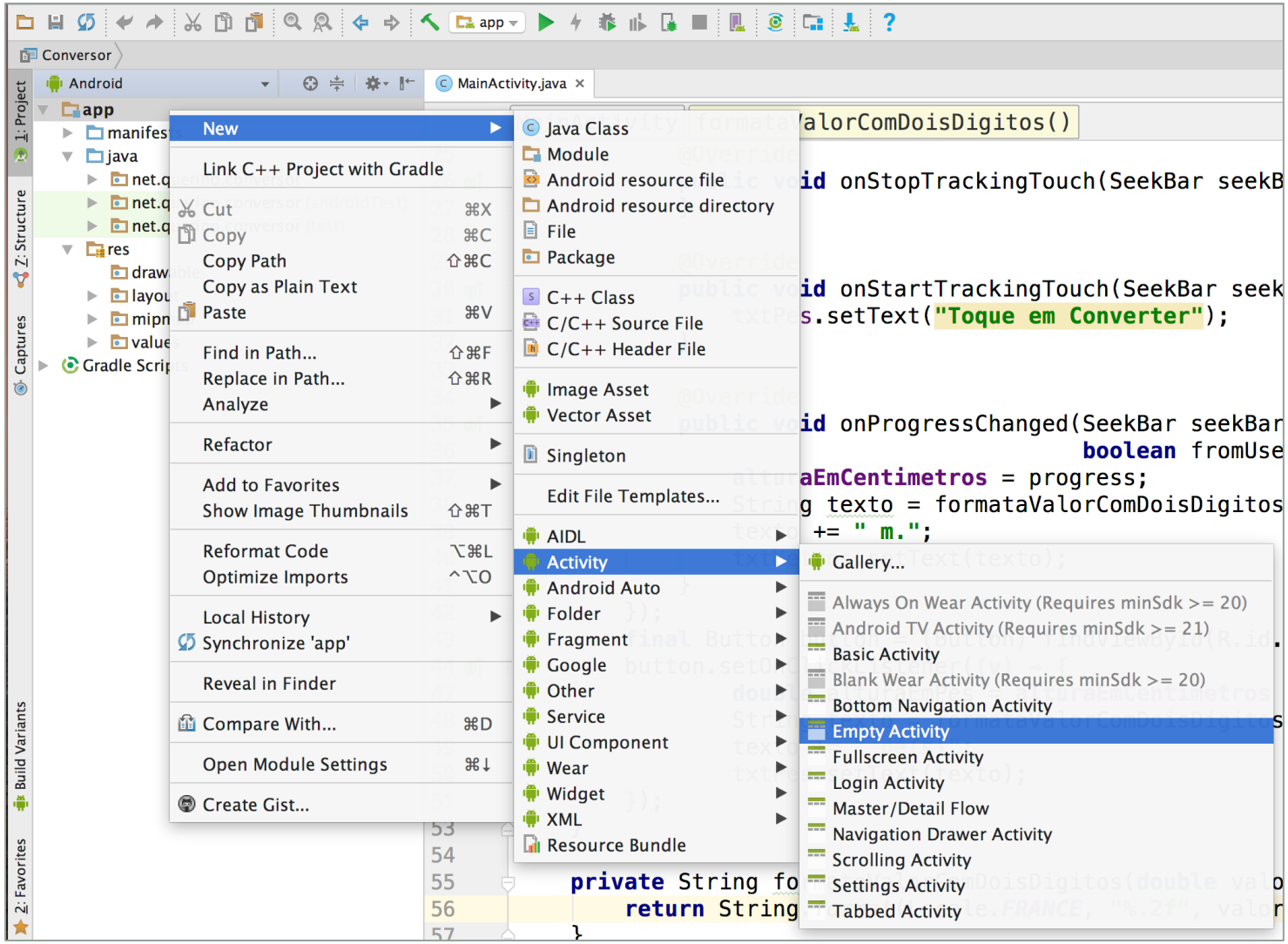
- Como prática da abertura de novas telas, vamos criar um app para um jogo de perguntas e respostas.
- Crie um novo projeto e monte sua tela principal de acordo com a figura:



Uma nova tela no seu aplicativo

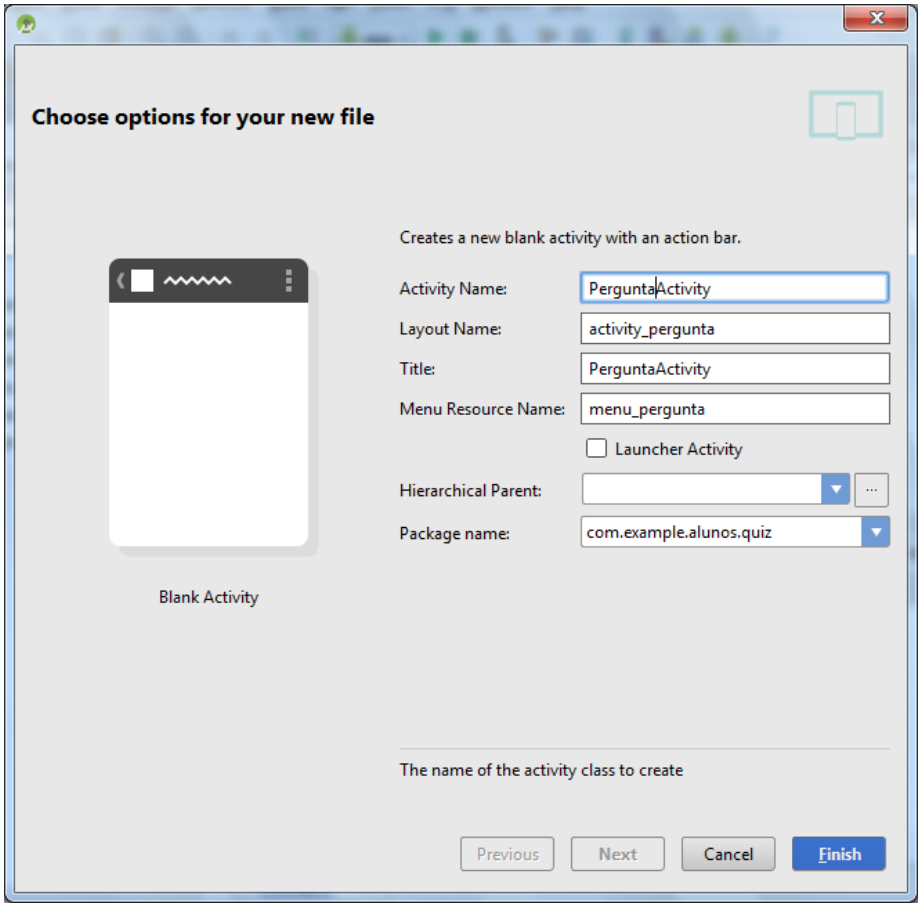
- Vamos montar agora uma tela para as perguntas.
- Para adicionar uma nova tela usando o assistente, clique com o botão direito do mouse sobre a estrutura do seu projeto e selecione **New > Activity > Empty Activity**.
- Veja no próximo slide como chegar até este menu.

Uma nova tela no seu aplicativo



Uma nova tela no seu aplicativo

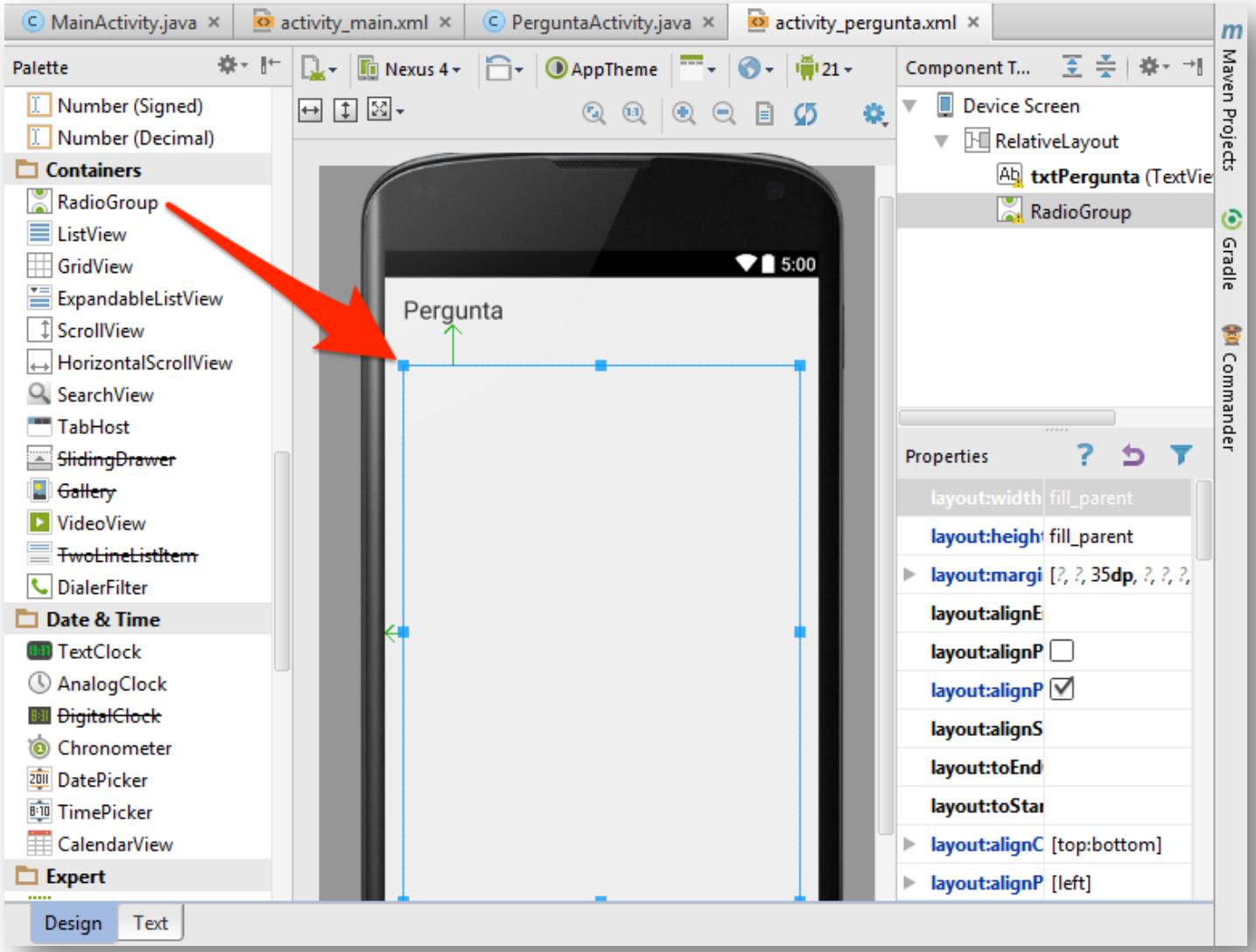
- Se quiser, especifique um novo nome para a Activity e seu arquivo de layout.
- É UMA BOA PRÁTICA entre programadores Android nomear suas Activities sempre terminando com …Activity
- **Após criar a tela, adicione um TextView Grande no topo, onde exibiremos a pergunta.**



android.widget.RadioButton e RadioGroup

- Para as alternativas de cada pergunta (A, B, C e D), usaremos quatro widgets **RadioButton**.
- Queremos que o usuário escolha apenas uma destas quatro alternativas.
- Por isso, devemos agrupar estes **RadioButtons**, de modo que fiquem *mutuamente exclusivos*.
- A forma mais simples de agrupá-los é colocá-los dentro de um **RadioGroup**.
- Arraste um **RadioGroup**, existente na categoria **Containers** da palheta, para a tela, colocando-o logo abaixo do **TextView** da pergunta, como indicado no slide seguinte.

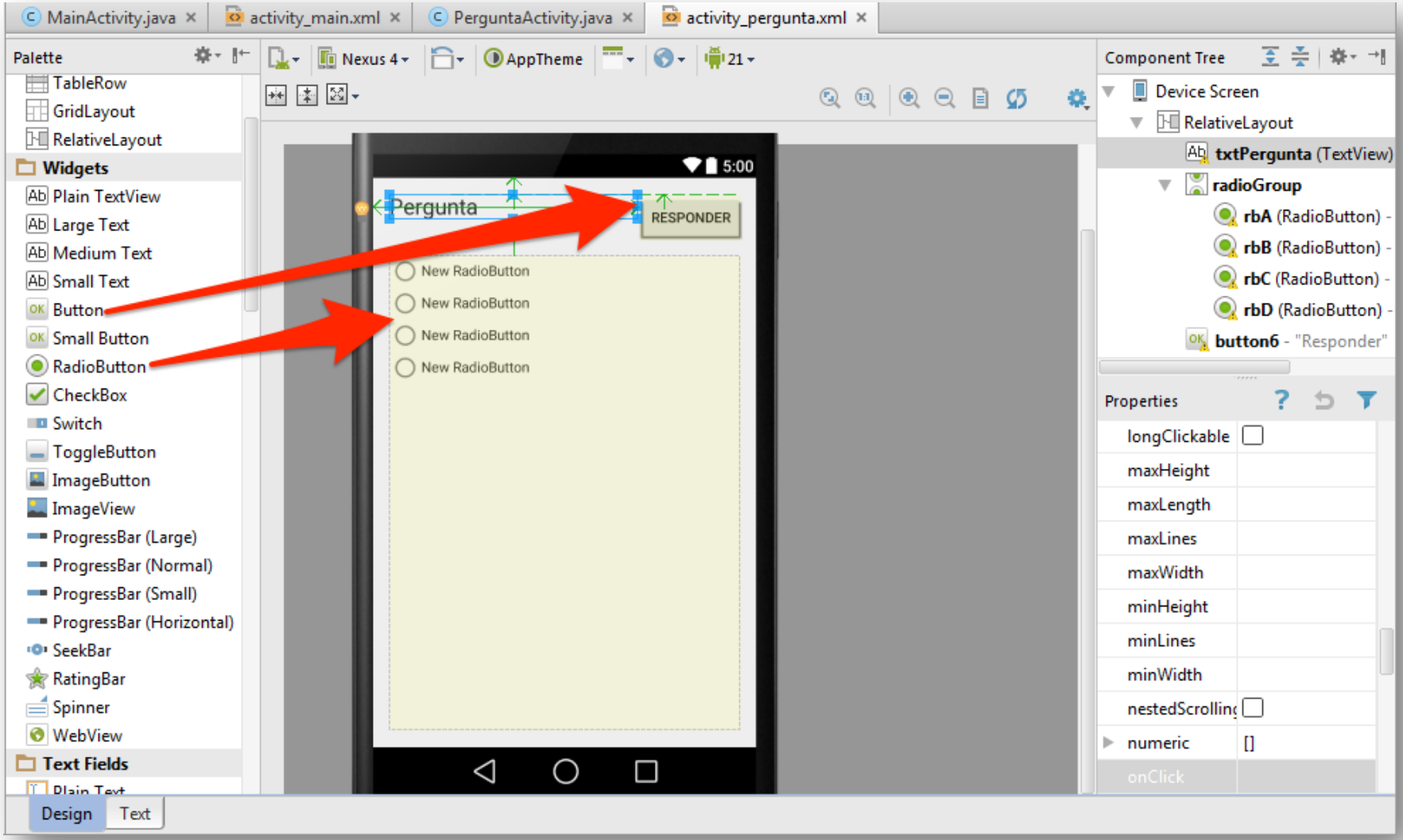
android.widget.RadioButton e RadioGroup



Montando a tela da pergunta

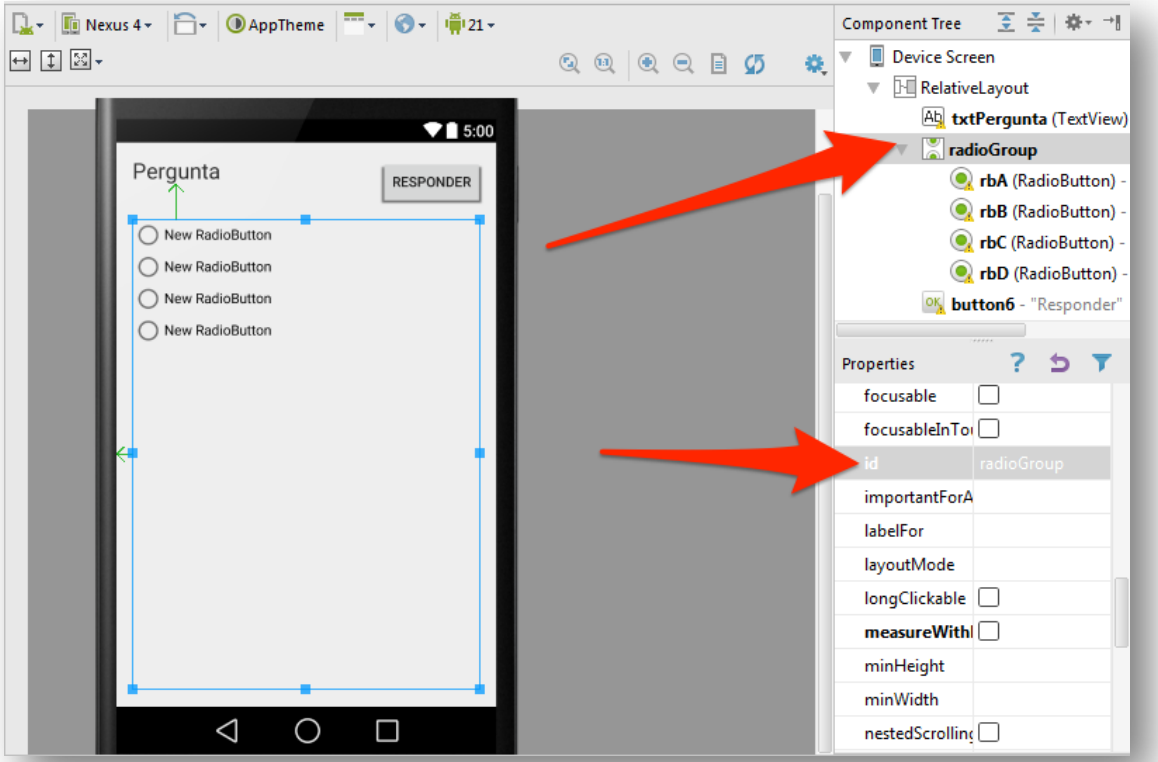
- Agora, adicione dentro do **RadioGroup** quatro **RadioButton** (que estão dentro da categoria **Widgets** da palheta).
- Coloque para os quatro **RadioButton** os seguintes IDs: **`rbA`**, **`rbB`**, **`rbC`** e **`rbD`**
- Aproveite e acrescente também um **Button** contendo o texto Responder, no canto superior direito.
- Se tiver dúvida quanto ao posicionamento dos widgets, siga o próximo slide.

Montando a tela da pergunta



Montando a tela da pergunta

IMPORTANTE: verifique se seu **RadioGroup** possui um **id**. Um bug em algumas versões do Android Studio deixa este widget sem um **id** na interface. Se estiver em branco, coloque `radioGroup`.



ABRINDO UMA NOVA TELA

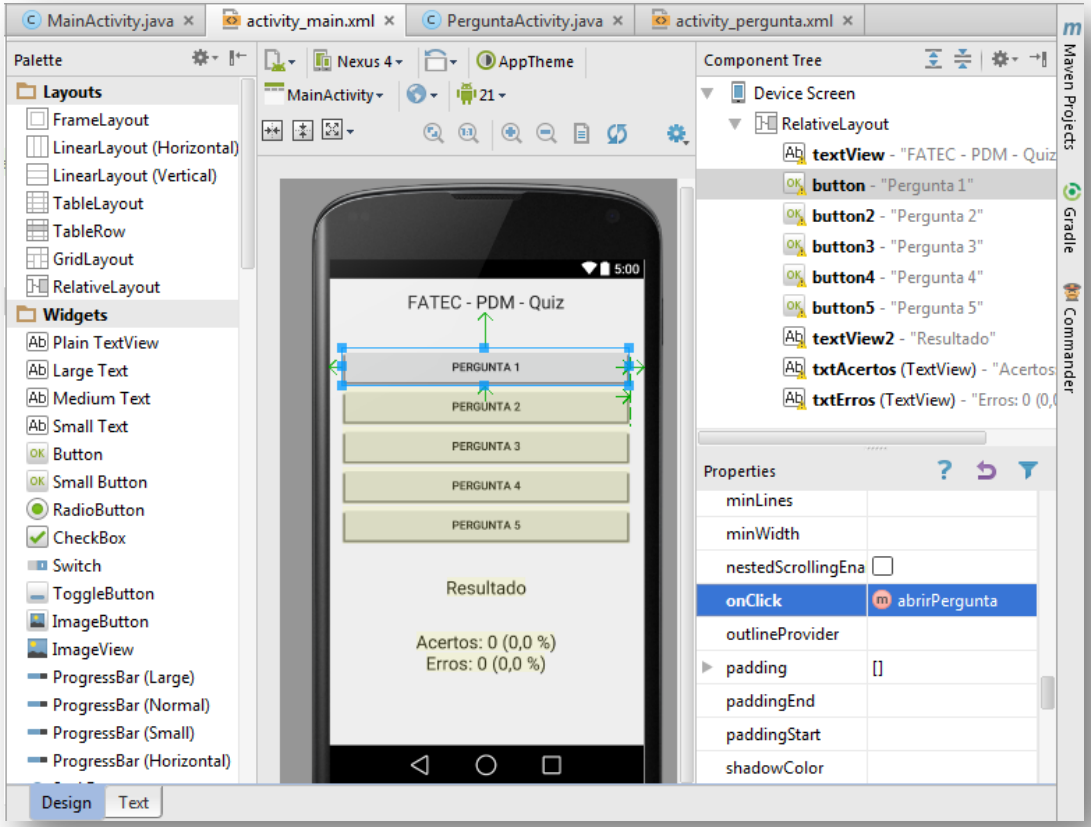
Classe Intent

android.content.Intent

- Usamos a classe **Intent** para passar uma mensagem a um outro componente do aplicativo, sinalizando uma intenção, que pode significar uma ação para…
  - Abrir um nova tela
  - Abrir o navegador padrão em um endereço
  - Iniciar um serviço em background (classe **android.app.Service**)
  - Enviar um broadcast (mensagem para outros apps)
  - Acessar um provedor de conteúdo (**Content Provider**)
- Seu principal uso é, com certeza, a abertura de telas.
- Com ela, além de abrir um nova tela, podemos também *passar valores a esta e receber dados dela*.

Abrindo a tela de perguntas a partir da tela principal

Para testar o uso da classe **Intent** na abertura de novas janelas, volte ao layout da tela principal e configure o nome do método para o clique no primeiro botão (Pergunta 1):



Implemente o método para o clique no botão

- Na **Activity** da tela principal, implemente agora o método **abrirPergunta** de acordo com a tela abaixo:

```
public void abrirPergunta(View view) {
    Intent intent = new Intent(this, PerguntaActivity.class);
    startActivity(intent);
}
```

- Primeiro, declaramos e instanciamos um **objeto** da classe **Intent**. Ao seu **construtor** passamos uma referência ao contexto atual (a **Activity** em que o código está sendo escrito – `this`) e o nome da classe da tela que será aberta (`PerguntaActivity.class`).
- Em seguida, para abrir efetivamente a nova tela, chamamos o método **startActivity**, passando o objeto `intent` a ele.

USANDO UM MESMO MÉTODO PARA EVENTOS DE DIFERENTES WIDGETS

Como reutilizar um único método para cliques em diferentes botões

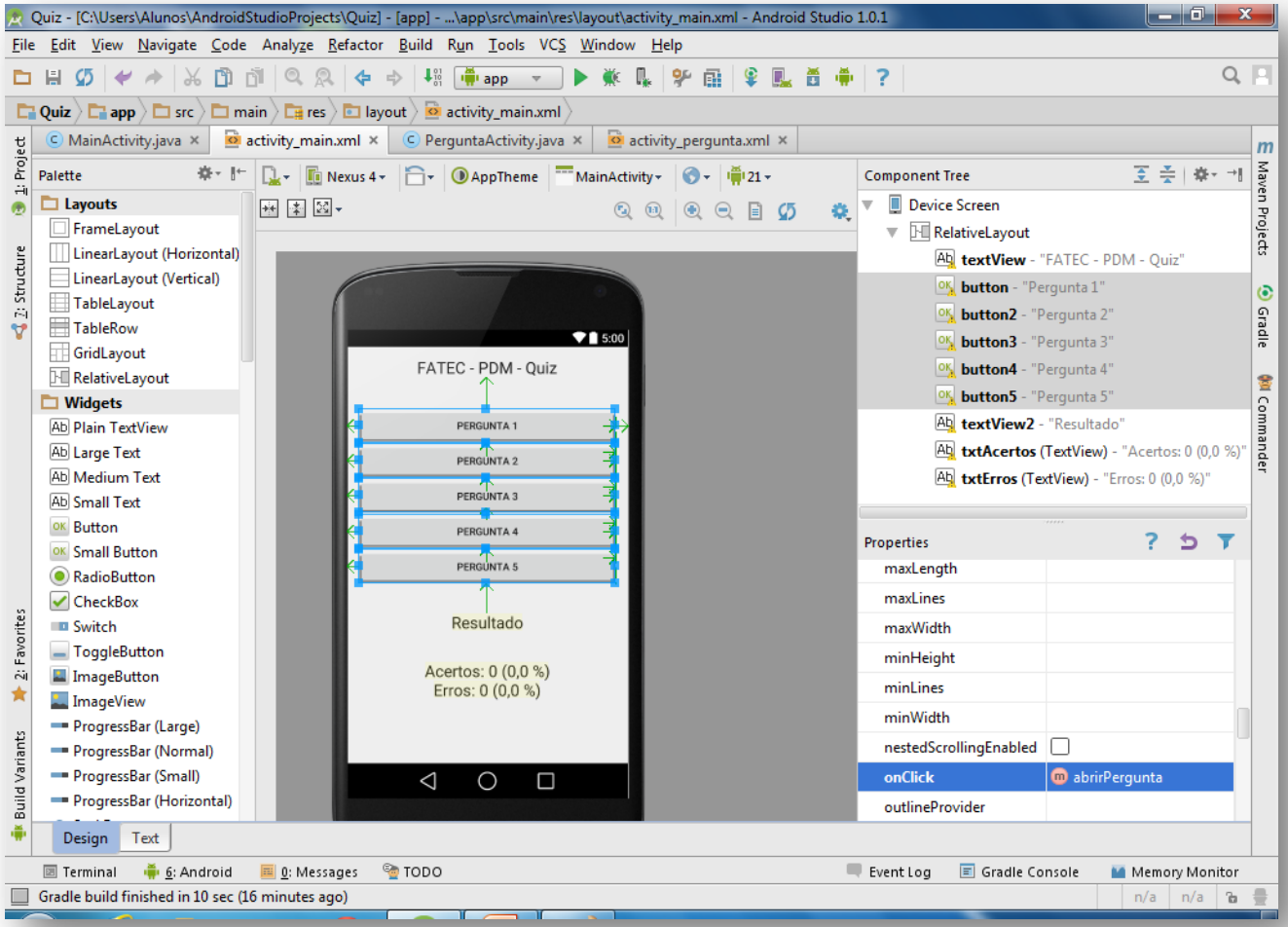
Usando um mesmo método para diferentes objetos

- Os métodos que criamos até agora para tratar o clique em um botão recebem, obrigatoriamente, um objeto do tipo **View** como parâmetro.
- Este parâmetro é uma referência ao objeto que recebeu o evento, ou seja, o botão que foi tocado.
- Dessa forma, podemos facilmente usar um mesmo métodos para tratar cliques em dois ou mais botões, bastando checar, no início do método, qual objeto recebeu o evento.
- Isso pode ser feito usano o método **getId()** da **View** passada por parâmetro. Este pode ser comparado com os IDs dos objetos dentro de **R.id,** identificando assim o "objeto emissário" do evento.

Usando um mesmo método para diferentes objetos

- Antes de modificar o método **abrirPergunta** para conseguir diferenciar o botão clicado, volte à interface.
- Selecione todos os botões do menu principal (mantenha o **Shift** pressionado enquanto clica nos botões – veja no próximo slide)
- Após selecionar todos, localize a propriedade **onClick** e aponte-a para o método **abrirPergunta**.
- Assim, o método **abrirPergunta** será executado no clique de qualquer um dos cinco botões.

Usando um mesmo método para diferentes objetos



Usando um mesmo método para diferentes objetos

- Neste exemplo, conseguimos diferenciar o que acontecerá no método de acordo com o botão clicado, testando o retorno de **view.getId()**

```
public void abrirPergunta(View view) {
    Intent intent = new Intent(this, PerguntaActivity.class);

    switch (view.getId()) {
        case R.id.button:
            Toast.makeText(this, "Botão da Pergunta 1 foi pressionado",
                    Toast.LENGTH_SHORT).show();
            break;
        case R.id.button2:
            Toast.makeText(this, "Botão da Pergunta 2 foi pressionado",
                    Toast.LENGTH_SHORT).show();
            break;
        case R.id.button3:
            Toast.makeText(this, "Botão da Pergunta 3 foi pressionado",
                    Toast.LENGTH_SHORT).show();
    }

    startActivity(intent);
}
```

PASSANDO E RECEBENDO VALORES ENTRE TELAS

Passagem de valores entre telas

- Podemos facilmente passar dados entre telas (**Activities**) no Android.
- Basta, antes de chamar o método **startActivity()**, “colocar” os valores no objeto da classe **Intent** que será usado para abertura.
- Estes parâmetros são os “**Extras**” contidos na **Intent** que abrirá a nova tela.

Passagem de valores entre telas

- Nos objetos da classe **Intent** existem diversas sobrecargas do método **putExtra()**, que recebem dois parâmetros:
  - O primeiro sempre será uma **String**, que atuará como ***identificador ou "chave" do parâmetro*** (precisamos identificá-los, já que podemos passar mais de um e que eles serão recebidos na próxima tela)
  - O segundo será a ***informação a ser passada***, que pode ser de um tipo primitivo (**int** ou **double**, por exemplo) ou um objeto de uma classe (uma **String**, ou outro objeto que implemente as interfaces **Serializable** ou **Parcelable**)

Passagem de valores entre telas

- No **Quiz**, de acordo com o botão tocado pelo usuário, passaremos valores diferentes para:
  - Pergunta
  - Alternativa A
  - Alternativa B
  - Alternativa C
  - Alternativa D
  - Resposta Certa
- Por isso, no método **abrirPergunta**, declararemos variáveis **String** para estes parâmetros.
- Seus valores deverão ser diferentes para cada botão tocado.
- No final de **abrirPergunta**, passaremos estas variáveis à próxima tela, usando nossa **Intent**.

```
public void abrirPergunta(View view) {

    Intent intent = new Intent(this, PerguntaActivity.class);

    String pergunta = "", resposta = "";
    String respA = "", respB = "", respC = "", respD = "";

    int codigoTela = 0;

    switch (view.getId()) {
        case R.id.button:
            pergunta = "Como se chama a classe usada como " +
                    "controladora de uma tela?";
            respA = "Intent";
            respB = "Screen";
            respC = "Controller";
            respD = "Activity";
            resposta = "d";
            codigoTela = CODIGO_PERGUNTA1;
            break;
        case R.id.button2:
            // Valores para a pergunta 2
            break;
        case R.id.button3:
            // Valores para a pergunta 3
            break;
        case R.id.button4:
            // Valores para a pergunta 4
            break;
        case R.id.button5:
            // Valores para a pergunta 2
    }
```

Passagem de valores entre telas

```
    intent.putExtra("pergunta", pergunta);
    intent.putExtra("a", respA);
    intent.putExtra("b", respB);
    intent.putExtra("c", respC);
    intent.putExtra("d", respD);
    intent.putExtra("resposta", resposta);

    startActivityForResult(intent, codigoTela);
}
```

- Note que, como a tela a ser aberta vai devolver a esta tela principal a indicação se o usuário acertou ou errou a pergunta, mudamos o método de abertura da janela…
- …de **startActivity()** para…
- **startActivityForResult()**
- …que pede também, como segundo parâmetro, um código identificador da tela que será aberta.

Passagem de valores entre telas

- No início da classe da **Activity** no menu principal, declare as seguintes constantes para os códigos de identificação das telas e variáveis contadoras de erros e acertos.
- **Convenção Java**: sempre defina os nomes de constantes em letras maiúsculas!

```
public class MainActivity extends ActionBarActivity {

    private static final int CODIGO_PERGUNTA1 = 0;
    private static final int CODIGO_PERGUNTA2 = 1;
    private static final int CODIGO_PERGUNTA3 = 2;
    private static final int CODIGO_PERGUNTA4 = 3;
    private static final int CODIGO_PERGUNTA5 = 4;

    private int erros = 0;
    private int acertos = 0;
```

startActivity() ou startActivityForResult()?

```
Intent intent = new Intent(this, PerguntaActivity.class);

// Este comando abre uma tela SEM ESPERAR RESULTADO.
startActivity(intent);

// Este comando abre uma tela E ESPERA UM RESULTADO.
// Quando a nova tela for fechada, o método
// onActivityResult nesta Activity será executado
startActivityForResult(intent, codigoDeRetornoDaTela);

// USE APENAS UM DESTES DOIS COMANDOS, DE ACORDO COM O
// QUE VOCÊ PRECISA!!!
```

Recebendo os valores na nova tela

- O melhor “local” na nova tela para o recebimento de valores passados da tela anterior é o evento **onCreate**.
- Dentro dele, o primeiro passo é obter uma referência ao objeto da classe **Intent** que abriu esta nova tela. Isso pode ser feito com o método **getIntent()**.
- Em seguida, chamamos neste mesmo objeto seus métodos **getTipoExtra("chave")** onde **Tipo** é o tipo primitivo (**`Int`**, **`Double`**, etc.) ou classe do objeto/valor passado pela tela anterior, com sua respectiva chave (sempre do tipo **String**) entre parênteses.
- Por exemplo, para obter um valor **String** passado com a chave **nome**, usamos **intent.getStringExtra("nome")**.

Recebendo os valores na nova tela

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_pergunta);

    TextView txtPergunta = (TextView)findViewById(R.id.txtPergunta);
    RadioButton rbA = (RadioButton)findViewById(R.id.rbA);
    RadioButton rbB = (RadioButton)findViewById(R.id.rbB);
    RadioButton rbC = (RadioButton)findViewById(R.id.rbC);
    RadioButton rbD = (RadioButton)findViewById(R.id.rbD);

    Intent intent = getIntent();
    txtPergunta.setText(intent.getExtras().getString("pergunta"));
    rbA.setText(intent.getStringExtra("a"));
    rbB.setText(intent.getStringExtra("b"));
    rbC.setText(intent.getStringExtra("c"));
    rbD.setText(intent.getStringExtra("d"));

    if (intent.getStringExtra("resposta").equals("a"))
        idCorreto = R.id.rbA;
    else if (intent.getStringExtra("resposta").equals("b"))
        idCorreto = R.id.rbB;
    else if (intent.getStringExtra("resposta").equals("c"))
        idCorreto = R.id.rbC;
    else
        idCorreto = R.id.rbD;
}
```

Verificando o acerto ou erro e retornando à tela principal

- Após o usuário selecionar a resposta desejada, irá tocar no botão **Responder**.
- Dentro do método disparado pelo toque, vamos verificar se o usuário acertou ou errou a questão, exibir um **Toast** avisando e retornar ao menu principal.
- Ao retornar, vamos "devolver" um parâmetro chamado `"acertou"`.
- Este parâmetro será um `boolean`, contendo `true` no caso de acerto e `false` no erro.
- Para que o método **responder()** funcione corretamente, vamos acrescentar uma variável na classe **PerguntaActivity** que guardará o identificador do `idCorreto` (já estamos armazenando nele o valor para a resposta certa – **veja o slide anterior**)

```java
public class PerguntaActivity extends ActionBarActivity {

    int idCorreto;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pergunta);

        TextView txtPergunta = (TextView)findViewById(R.id.txtPergunta);
        RadioButton rbA = (RadioButton)findViewById(R.id.rbA);
        RadioButton rbB = (RadioButton)findViewById(R.id.rbB);
        RadioButton rbC = (RadioButton)findViewById(R.id.rbC);
        RadioButton rbD = (RadioButton)findViewById(R.id.rbD);

        Intent intent = getIntent();
        txtPergunta.setText(intent.getExtras().getString("pergunta"));
        rbA.setText(intent.getStringExtra("a"));
        rbB.setText(intent.getStringExtra("b"));
        rbC.setText(intent.getStringExtra("c"));
        rbD.setText(intent.getStringExtra("d"));

        if (intent.getStringExtra("resposta").equals("a"))
            idCorreto = R.id.rbA;
        else if (intent.getStringExtra("resposta").equals("b"))
            idCorreto = R.id.rbB;
        else if (intent.getStringExtra("resposta").equals("c"))
            idCorreto = R.id.rbC;
        else
            idCorreto = R.id.rbD;
    }
```

O código para o método **responder()**

- Veja a explicação no próximo slide.

```
public void responder(View view) {
    RadioGroup radioGroup = (RadioGroup)findViewById(R.id.radioGroup);
    Intent intent = new Intent();

    if (radioGroup.getCheckedRadioButtonId() == idCorreto) {
        intent.putExtra("acertou", true);
        Toast.makeText(this, "Resposta CERTA! :)", Toast.LENGTH_SHORT).show();
    } else {
        intent.putExtra("acertou", false);
        Toast.makeText(this, "Resposta errada... :(", Toast.LENGTH_SHORT).show();
    }

    setResult(RESULT_OK, intent);
    finish();
}
```

O código para o método **responder()**

- Para verificar se o usuário acertou, chamaremos o método **getCheckedRadioButtonId()** do **RadioGroup**.
- Ele retorna o ID do **RadioButton** escolhido. Comparando ele com o **`idCorreto`**, sabemos se o usuário acertou ou não.
- Criamos também um objeto **`intent`** apenas para retornar o parâmetro de erro ou acerto
- Para fechar uma **Activity**, basta chamar o método **finish()**.
- Como neste caso queremos fechá-la e passar uma **Intent** com dados à tela anterior, antes de finish chamamos **setResult()**.
- Neste caso, o **setResult()** recebe dois parâmetros:
  - Uma constante identificando um fechamento confirmando uma ação do usuário (**`RESULT_OK`**) ou cancelando (**`RESULT_CANCELED`**)
  - Um objeto **`intent`** contendo o parâmetro "acertou".

Recebendo o acerto ou erro na tela principal

- Após clicar no botão Responder, a tela de pergunta será fechada e o usuário retornará ao menu principal.
- Mas em termos de “código Java”, **para onde**, dentro da classe a tela principal, **nós retornamos**?
- Como a tela de pergunta foi aberta com o método **startActivityForResult()**, o retorno ocorrerá em um evento denominado **onActivityResult()**.
- Assim, para programar o recebimento do parâmetro de erro e acerto, atualização do “placar” e desabilitar o botão da pergunta já respondida, devemos escrever este método de acordo com os próximos slides.

Recebendo o acerto ou erro na tela principal

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    int idDoBotao;

    switch (requestCode) {
        case CODIGO_PERGUNTA1:
            idDoBotao = R.id.button;
            break;
        case CODIGO_PERGUNTA2:
            idDoBotao = R.id.button2;
            break;
        case CODIGO_PERGUNTA3:
            idDoBotao = R.id.button3;
            break;
        case CODIGO_PERGUNTA4:
            idDoBotao = R.id.button4;
            break;
        default:
            idDoBotao = R.id.button5;
            break;
    }
```

Recebendo o acerto ou erro na tela principal

```
    if (resultCode == RESULT_OK) {
        Button button = (Button) findViewById(idDoBotao);
        button.setEnabled(false);

        if (data.getBooleanExtra("acertou", false)) {
            acertos++;
        } else {
            erros++;
        }

        TextView txtAcertos = (TextView)findViewById(R.id.txtAcertos);
        TextView txtErros = (TextView)findViewById(R.id.txtErros);

        txtAcertos.setText("Acertos: " + acertos + " " +
                100 * acertos / (acertos + erros) + " %");
        txtErros.setText("Erros: " + erros + " " +
                100 * erros / (acertos + erros) + " %");
    }
}
```

Recebendo o acerto ou erro na tela principal

- O evento/método **onActivityResult()** recebe três parâmetros muito úteis:
  - **`requestCode`**: um **`int`** contendo o valor identificador da tela passado no **startActivityForResult()**;
  - **`resultCode`**: um **`int`** indicando se a tela que acabou de ser fechada resultou em uma confirmação (`RESULT_OK`) ou cancelamento (`RESULT_CANCELED`)
  - **`data`**: o objeto da classe **Intent** contendo os parâmetros retornados pela tela – neste caso, apenas um **`boolean`**: **`“acertou”`**
- Usamos o parâmetro **`requestCode`** para “saber” de qual pergunta estamos voltando.

Recebendo o acerto ou erro na tela principal

- Já um `resultCode == RESULT_OK` confirma que o usuário clicou no botão *Responder* na tela de pergunta; consequentemente:
  - desabilitamos o botão que abre a pergunta correspondente;
  - obtemos o parâmetro **`boolean`** `"acertou"` dentro do objeto **`data`** (classe **Intent**);
  - se `"acertou"` for verdadeiro, incrementamos os acertos;
  - se for falso, incrementamos os erros.
- Por fim, atualizamos os dois **TextView** existentes na tela para exibição dos acertos e erros do usuário

BIBLIOGRAFIA

- QUERINO FILHO, L. C. Desenvolvendo seu Primeiro Aplicativo Android. Novatec Editora. 2013
- DEITEL, H. et al. Android for Programmers: An App-Driven Approach. Pearson Education. 2012.