Mudando o layout do aplicativo

Voltando a nossa Activity, continuamos com o objetivo de colocar a foto do aluno, ao lado dos contatos e o nome mais destacado.

No método carregaLista do arquivo ListaAlunoActivity, temos o adapter.

ArrayAdapter<Aluno> adapter = new ArrayAdapter <~>(this, android.R.layout.simple\_list\_item\_1,alunos);  
listaAlunos.setAdapter(adapter);

O adapter era responsável por converter os objetos do Java em um view do Android. Entre os parâmetros que passamos do Adapter, estava o layout. Vamos substitui-lo pelo list\_item que acabamos de criar.

ArrayAdapter<Aluno> adapter = new ArrayAdapter <~>(this, android.R.layout.list\_item,alunos);  
listaAlunos.setAdapter(adapter);

Passamos o layout para o adapter, no entanto, se tentarmos rodar nosso aplicativo, receberemos uma mensagem de erro. Isto indica que temos uma exception no código:

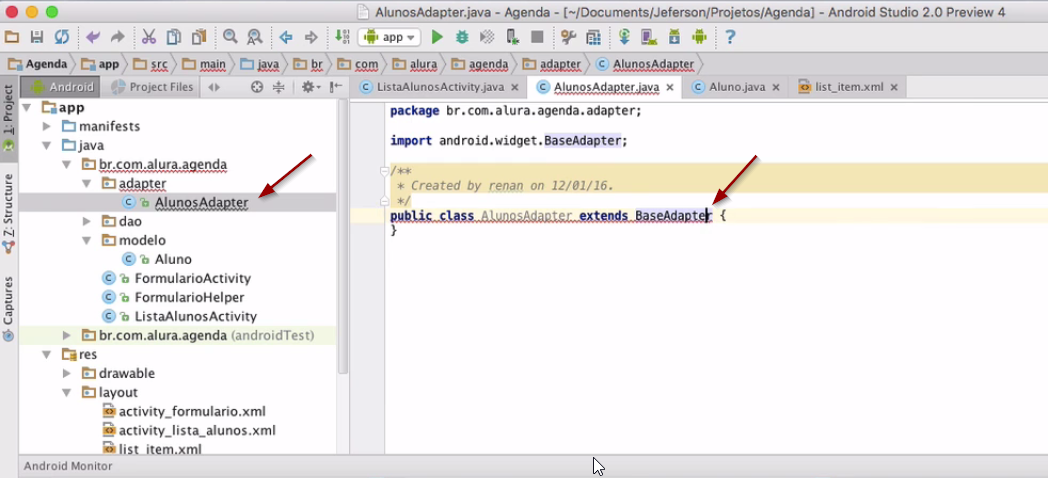
java.lang.IllegalStateException: ArrayAdapter requires the resource ID to be a TextView  
//...

Para recordar, antes no nosso aplicativo, conseguíamos ver o ID e o nome do aluno. O ArrayAdapter pega o objeto java, transforma em um *String* e coloca em um TextView - o layout em que inserimos simple\_list\_item\_1.

Por isso, no arquivo modelo, nós usamos o método toString para definir o que deveria aparecer na lista.

@Override  
public String toString() { return getId() + " - " + getNome(); }   
  
}

Porém, só isso não será mais suficiente, porque temos uma grande quantidade de dados - temos foto, nome, endereço e telefone. E queremos colocá-los no item da nossa lista, que já está mais complexo. Se o ArrayAdapter não está mais funcionando, uma alternativa é criar o nosso próprio adapter especializado, que irá transformar os alunos na View que desejamos.

Iremos até o pacote padrão br.com.alura.agenda, e vamos criar o pacote adapter e dentro dele, vamos adicionar uma nova classe AlunosAdapter. Iremos informar que ele já está herdando o comportamento de uma classe abstrata do Android, chamada extends BaseAdapter. 

Após importá-la, pediremos para o Android Studio implementar todos os métodos que precisamos implementar.

public class AlunosAdapter extends BaseAdapter {  
 @Override  
 public int getCount() {  
 return 0;  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return null;  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return 0;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 return null;  
 }  
}

No ListaAlunosActivity, vamos adicionar novas linhas, logo abaixo do carregaLista:

AlunosAdapter adapter = new AlunosAdapter();  
listaAlunos.setAdapter(adapter);

Se rodarmos agora, teremos um adapterque trabalha com os nossos alunos. Mas a agenda mostrará uma tela em branco.

Vamos entender como funciona o adapter.

O método mais importante que temos no AlunosAdapter é o getView, que é invocado pela lista quando ela precisa mostrar algum item.

Como a lista irá precisar do adapter para devolver uma View e inserir os alunos.

@Override  
public View getView(int position, View convertView, ViewGroup parent) {  
 return null;  
}

Em algum momento, a lista precisará parar de devolver estas Views. Precisaremos definir quantos itens têm na lista de alunos, ela fará isto perguntando novamente para o adapter, usando o métodogetCount.

public class AlunosAdapter extendes BaseAdapter {  
 @Override  
 public int getCount() {  
 return 0;  
 }

Quando instanciamos o adapter no ListaAlunosActivity, não informamos a quantidade de itens que ele precisa gerenciar.

private void carregaLista() {  
 AlunoDAO dao = new AlunoDAO(this);  
 List<Aluno> alunos = dao.buscaAlunos();  
 dao.close();  
  
 AlunosAdapter adapter = new AlunosAdapter(alunos);  
 listaAlunos.setAdapter(adapter);  
}

Precisaremos criar um construtor alunos, na classe AlunosAdapter.

Faremos isto com o comando ALT + ENTER, o adapter vai entender que precisa receber a lista de alunos. Guardaremos a lista no atributo alunos

public class AlunosAdapter extends Baseadapter {  
 private final List<Aluno> alunos;  
  
 public AlunosAdapter(List<Aluno> alunos) {  
 this.alunos = alunos;  
  
 }  
\...

Agora, o getCount pode retornar outro resultado além de 0. Usaremos o método size, que irá informar quantos itens têm na lista.

@Override  
public int getCount() {  
 return alunos.size();  
}

Nossa *listView* já pode pedir cada um dos itens para o adapter.

@Override  
public View getView(int position, View convertView, ViewGroup parent) {  
 return null;  
}

Iremos devolver uma *View* para cada item, em determinada posição. Faremos isso, adicionando new TextView.

@Override  
public View getView(int position, View convertView, ViewGroup parent) {  
 new TextView;  
 return null;  
}

Como parâmetro, traremos o contexto de fora, mais especificamente, do ListaAlunosActivity. Iremos usar o this do AlunosAdapter e depois, iremos adicioná-lo ao construtor.

AlunosAdapter adapter = new AlunosAdapter(this, alunos);  
listaAlunos.setAdapter(adapter);

Voltaremos para o AlunosAdapter.java, e veremos que ele já adicionou o ListaAlunosActivity.

public class AlunosAdapter extends BaseAdapter {  
 private final List<Aluno> alunos;  
  
 public AlunosAdapter(ListaAlunosActivity ListAlunosActivity, list<aluno> alunos) {  
 this.alunos = alunos;  
 }  
}

Mas iremos alterar com uma activity específica e usaremos a mais genérica, Context. Faremos a mesma alteração com o tipo.

public AlunosAdapter(Context context, List<Aluno> alunos) {  
 this.alunos = alunos;  
}

O trecho do nosso código ficará assim:

public class AlunosAdapter extends BaseAdapter {  
 private final Lista<Aluno> alunos;  
 private final Context context;  
  
 public AlunosAdapter(Context context, List<Aluno> alunos) {  
 this.context = context;  
 this.alunos = alunos;  
 }  
  
 @Override  
 public int getCount() {  
 return alunos.size();  
 }  
  
//...

Como já temos o contexto, podemos passar para o nosso TextView.

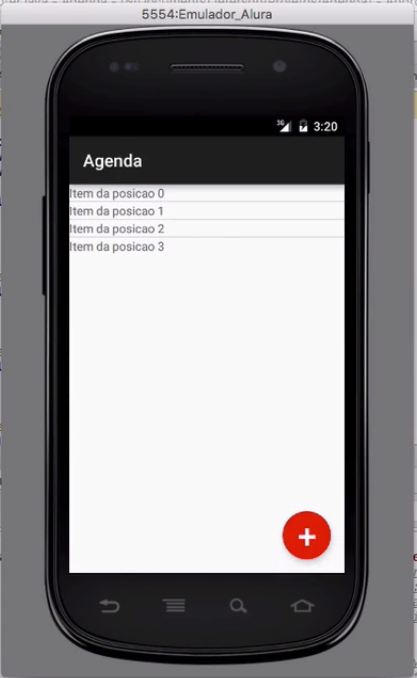
@Override  
public View getView(int position, View convertView, ViewGroup parent) {  
 TextView View = new TextView(context);  
 view.setText("Item da posicao " + position);  
 return null;  
}

Adicionamos o Item da posicao concatenada com a position que o Android nos informou.

Para terminar, iremos finalizar com o getView devolvendo a nossa view.

return view;

Se rodarmos nossa aplicação, veremos a seguinte tela:

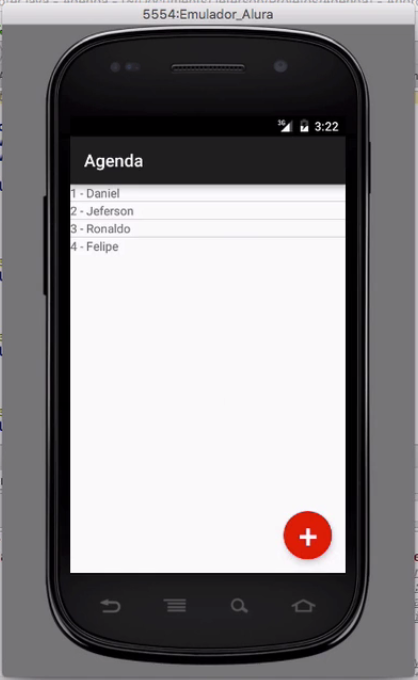


Enxergamos quatro itens, o adapter já começa a funcionar. Precisamos fazer o mesmo que o ArrayAdapter fez antes: mostrar o nome e o telefone.

Para isto, faremos alterações no getView e inserir aluno.toString.

@Override  
public View getView(int position, View convertView, ViewGroup parent) {  
 TextView view = new TextView(context);  
 Aluno aluno = alunos.get(position);  
 view.setText(aluno.toString());  
 return view;   
}

Vamos ver o resultado, rodando o aplicativo:



Mas se clicarmos nos nomes cadastrados, veremos que os formulários estão vazios.

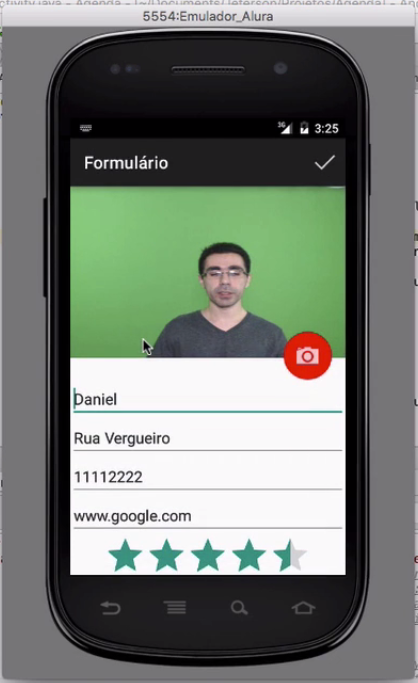
A nossa lista não sabe com quais objetos está trabalhando, ela precisa do adapter para informá-la sobre quais itens ocupam determinadas posições.

listaAluno.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> lista, View item, int position, long id) {  
 Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(position);  
  
 Intent intentVaiProFormulario = new Intent(ListaAlunosActivity.this, FormularioActivity.this, FormularioActivity.class);  
 intentVaiProFormulario.putExtra("aluno", aluno);  
 startActivity(intentVaiProFormulario);  
 }  
  
}

Em seguida, iremos implementar os métods getItem e getItemId, no AlunosAdapter:

@Override  
public Object getItem(int position) {  
 return alunos.get(position);  
}  
  
@Override  
public long getItemId(int position) {  
 return alunos.get(position).getId();  
}

O long id será referente ao item clicado e que será devolvido. Com as alterações, os dados do formulário já irão aparecer na app.



Até aqui, nós substituímos o ArrayAdapter que não era capaz de pegar o layout - com a foto, nome e telefone - e construímos o nosso array, chamado AlunosAdapter.