Desenvolvimento Android - Walkthrough

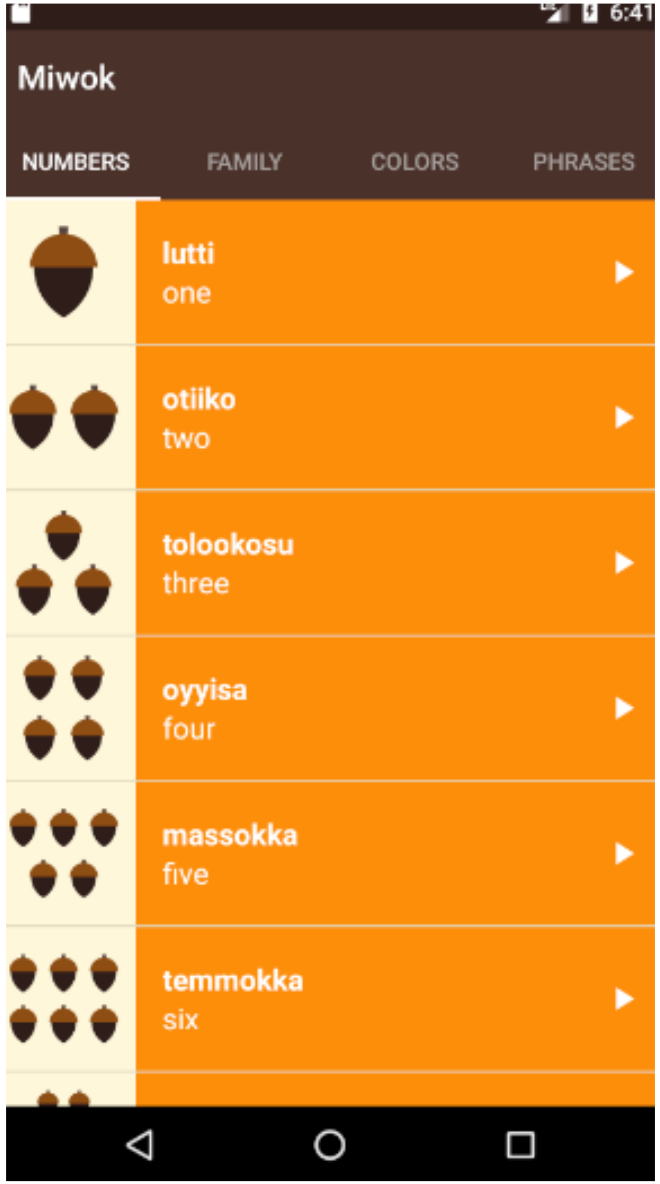
Análise e Desenvolvimento de Sistemas – 6º Período
Thiago Gois Lima
Instituto Federal do Triângulo Mineiro
Ituiutaba-MG

Introdução

- Esse material é uma adaptação de aulas (em inglês) para desenvolvimento de um sistema android da Udacity em parceria com a Google
- Algumas alterações no cronograma e didática adotadas no curso foram feitas a fim de facilitar o entendimento dos alunos em sala de aula
- O objetivo do material, é fornecer total apoio para que ao final do processo, o aluno tenha entendido todo o caminho pelo qual passou até chegar ao app finalizado
- Esse será meu último esforço como professor no IFTM, espero que gostem

Objetivo

- O objetivo final nosso, é ter uma aplicação que mostre uma tradução inteligível para o usuário junto à versão da mesma palavra na língua dos Miwoks
- Como acompanhamento adicional, teremos auxílio visual e de áudio para que seja possível ao usuário entender em várias frentes um pouco mais sobre cada palavra ou sentença apresentada



Configurações iniciais

- Vamos começar criando um novo projeto com o nome de:
  - Miwok
- O domínio será:
  - android.example.com
- Começaremos o projeto com uma:
  - Empty Activity
- Então aceitamos os dois arquivos padrões que nos são recomendados
- Para finalizar, vá em File > Settings > (Digite “import”, sem aspas, na busca) > Marque a opção “Add unambiguous imports on the fly”
  - Isso vai fazer com que pacotes que não contenham nomes duplicados sejam importados automaticamente

Entendendo um pouco do que foi gerado

- Durante o processo de inicialização da aplicação, alguns arquivos foram gerados a medida que a utilização deles for acontecendo, estaremos vendo melhor sua funcionalidade. No momento, nos interessam 4:
- app > java > com.example.android.miwok > MainActivity.java
- app > res > layout > activity_main.xml
- app > res > values > colors.xml
- app > res > values > styles.xml

Entendendo um pouco do que foi gerado

- app > java > com.example.android.miwok > MainActivity.java

- É o nosso arquivo java da principal atividade do projeto
- Quando criada automaticamente pela IDE, esse arquivo sempre é marcado para ser inicializado quando nossa aplicação é executada
- Toda programação (recursos a serem inseridos na tela, interações com usuário, busca em APIs e banco de dados, etc) deve ser feita nesse arquivo
- Vem com um arquivo de layout vinculado à ele o activity_main.xml

Entendendo um pouco do que foi gerado

- APP > RES > LAYOUT > ACTIVITY_MAIN.XML

- Arquivo de layout (apresentação) de telas
- Pode ser estilizado na aba de design ou XML (durante todo essa projeto estaremos utilizando a parte de XML)
- Como é um arquivo XML, ele tem suas próprias regras sintáticas a serem seguidas, sendo a mais importante o fato de que na primeira linha do arquivo, deve sempre aparecer a versão do XML em uso

```
<?xml version="1.0" encoding="utf-8"?>
```

Entendendo um pouco do que foi gerado

- APP > RES > VALUES > COLORS.XML

- Esse arquivo contém informações com relação às cores utilizadas no app
- Torna muito mais fácil gerir atualizações de design no app, já que quando uma cor é trocada, a alteração surte efeito em todo o app e não somente no componente que sofreu alteração
- Cada nova cor é adicionada como um recurso color no XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Entendendo um pouco do que foi gerado

- APP > RES > VALUES > STYLES.XML

- Esse arquivo contém informações com relação aos estilos utilizadas no app
- Torna muito mais fácil gerir atualizações de design no app, pois concentra em um único lugar estilos que são aplicados em diversos componentes de tela
- Cada novo estilo é um recurso do tipo style identificados por nomes e cada componente a ser estilizado é um item

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

Instruções com relação ao material

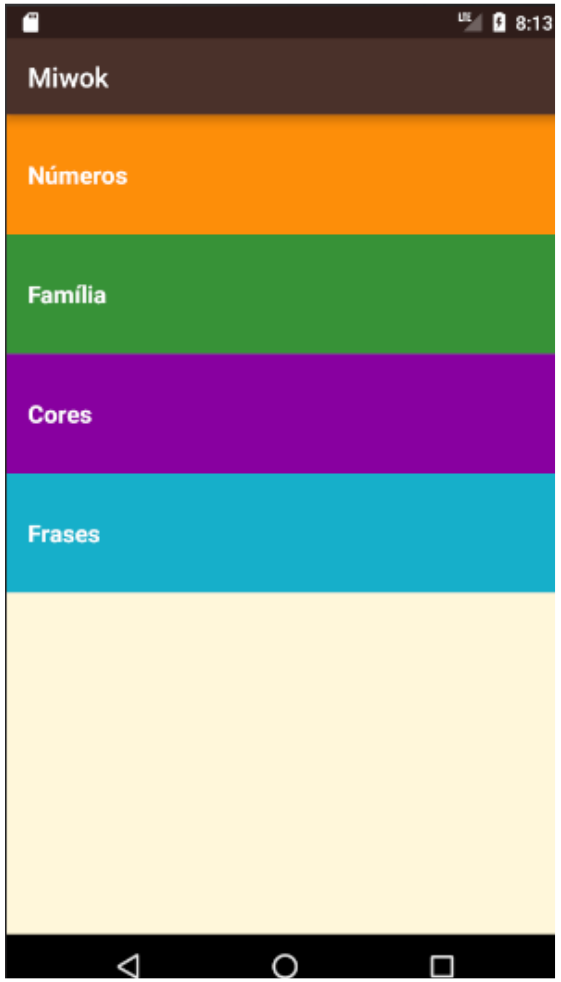
- O material é dividido em 5 partes com objetivos claros em cada parte
- No início do desenvolvimento do projeto, estaremos fazendo alterações de maneira gradual e acompanhada
- Conforme formos avançando no conteúdo, um slide de “pausa” será adicionado, para que você tente bolar uma saída para o problema antes de olhar a resposta
- Alguns desafios julgados fáceis demais nem possuem resposta
- Se você está realmente interessado em aprender a desenvolver Android, eu sugiro fortemente que ao encontrar um slide de pausa, se esforce para resolver o desafio antes de avançar

Parte 1

Criação da aplicação e Estilizações iniciais

Iniciando o desenvolvimento

- Vamos construir de maneira iterativa o seguinte layout



Iniciando o desenvolvimento

- A primeira parte do nosso projeto se concentra em construir uma tela inicial para a nossa aplicação seguindo as seguintes instruções de design (façam alterações de design somente utilizando a parte do XML, por favor):
  1. Serão 4 TextViews a serem inseridas dentro de um LinearLayout.
  2. O Linear Layout deve ser o elemento raiz da aplicação, ou seja, não deve estar inserido dentro de nenhum outro componente. Além disso, ele deve ter:
     1. Altura e largura "match_parent" para cobrir toda a área da tela
     2. Orientação vertical
  3. O texto para cada um dos TextViews deve seguir a seguinte ordem:
     1. Números > Família > Cores > Frases

Pausa

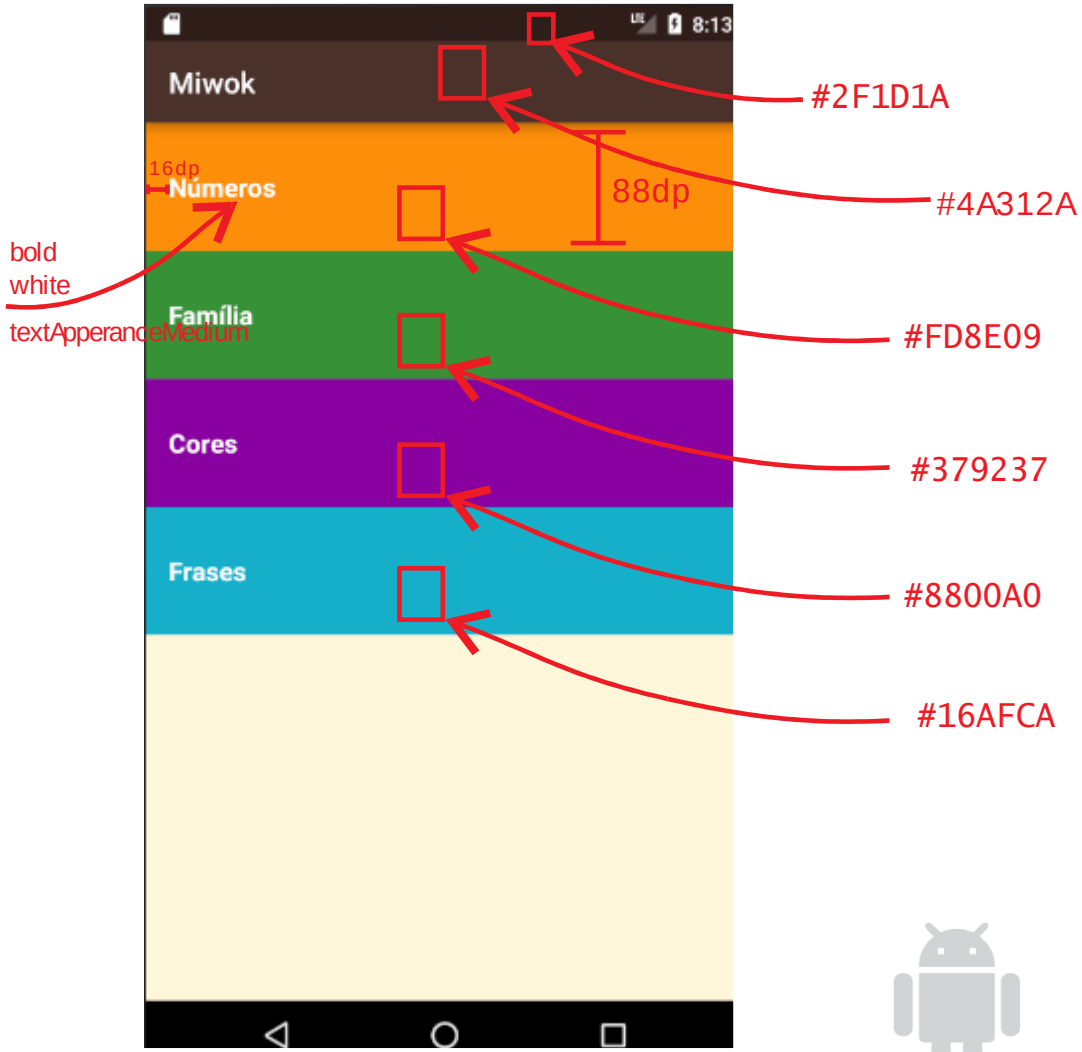
Iniciando o desenvolvimento

- A solução para esse primeiro problema é mostrada ao lado

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.android.miwok.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Números"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Família"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cores"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Frases"/>

</LinearLayout>
```

Adicionando estilo aos TextViews

- O estilo planejado para cada um dos itens da interface é descrito ao lado.
- Com o conhecimento que você já possui, tente chegar o mais próximo o possível desse resultado.



Pausa

Adicionando estilo aos TextViews

- Um resultado possível para o problema é mostrado ao lado
- O atributo gravity pode ser utilizado para posicionar o texto quando estamos trabalhando com layouts lineares
- A solução somente mudaria as cores em cada um dos TextViews

```
<TextView
        android:layout_width="match_parent"
        android:layout_height="88dp"
        android:background="#FD8E09"
        android:padding="16dp"
        android:textAppearance="?android:textAppearanceMedium"
        android:textStyle="bold"
        android:textColor="@android:color/white"
        android:gravity="center_vertical"
        android:text="Números"/>
```

Reorganizando dados

- Essa primeira parte da solução, trouxe-nos resultado para a estilização das TextView, mas não alterou em nada a barra do app
- A alteração dos estilos desse componente somente é feita pelos arquivos de estilização e é neles que vamos focar agora

Reorganizando dados

- Repare no código ao lado, na quantidade de informação repetida que temos nas TextViews
- Essas informações são compartilhadas por todas elas, portanto, podemos simplificar a manutenção dessas informações passando os dados ao arquivo de estilo

```
<TextView
        android:layout_width="match_parent"
        android:layout_height="88dp"
        android:background="#FD8E09"
        android:padding="16dp"
        android:textAppearance="?android:textAppearanceMedium"
        android:textStyle="bold"
        android:textColor="@android:color/white"
        android:gravity="center_vertical"
        android:text="Números"/>

<TextView
        android:layout_width="match_parent"
        android:layout_height="88dp"
        android:background="#379237"
        android:padding="16dp"
        android:textAppearance="?android:textAppearanceMedium"
        android:textStyle="bold"
        android:textColor="@android:color/white"
        android:gravity="center_vertical"
        android:text="Família"/>
```

Reorganizando dados

- Podemos criar um novo recurso de estilos no arquivo styles.xml, como mostrado ao lado

```
...

<style name="Categorias">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">88dp</item>
    <item name="android:padding">16dp</item>
    <item name="android:textAppearance">?android:textAppearanceMedium</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textColor">@android:color/white</item>
    <item name="android:gravity">center_vertical</item>
</style>

...
```

Reorganizando dados

- Com isso, nosso arquivo activity_main.xml fica como ao lado
- Para referenciar um resource de estilo utilizamos o atributo style
- No valor colocamos @style (at style ou em estilo, em português) seguido de /nome_do_estilo

```
...

<TextView
    style="@style/Categorias"
    android:background="#FD8E09"
    android:text="Números"/>

<TextView
    style="@style/Categorias"
    android:background="#379237"
    android:text="Família"/>

<TextView
    style="@style/Categorias"
    android:background="#8800A0"
    android:text="Cores"/>

<TextView
    style="@style/Categorias"
    android:background="#16AFCA"
    android:text="Frases"/>

...
```

Reorganizando dados

- Agora, vamos aproveitar e mover as cores para o arquivo colors.xml
- E é nesse arquivo em que trocamos a cor da barra do nosso app
- Onde colorPrimary representa a cor da barra no app e colorPrimaryDark é a cor da barra de notificações
- OBS: aqui ou deixamos o colorAccent ou removemos essa linha daqui e do arquivo style

```
...

<resources>
    <color name="colorPrimary">#4A312A</color>
    <color name="colorPrimaryDark">#2F1D1A</color>

    <color name="categoria_numeros">#FD8E09</color>
    <color name="categoria_familia">#379237</color>
    <color name="categoria_cores">#8800A0</color>
    <color name="categoria_frases">#16AFCA</color>
</resources>
```

Reorganizando dados

- Com isso, nosso arquivo activity_main.xml fica como ao lado
- Fazemos referência a cor usando @color/nome_da_cor_no_arquivo_colors

```
...

<TextView
    style="@style/Categorias"
    android:background="@color/categoria_numeros"
    android:text="Números"/>

<TextView
    style="@style/Categorias"
    android:background="@color/categoria_familia"
    android:text="Família"/>

<TextView
    style="@style/Categorias"
    android:background="@color/categoria_cores"
    android:text="Cores"/>

<TextView
    style="@style/Categorias"
    android:background="@color/categoria_frases"
    android:text="Frases"/>

...
```

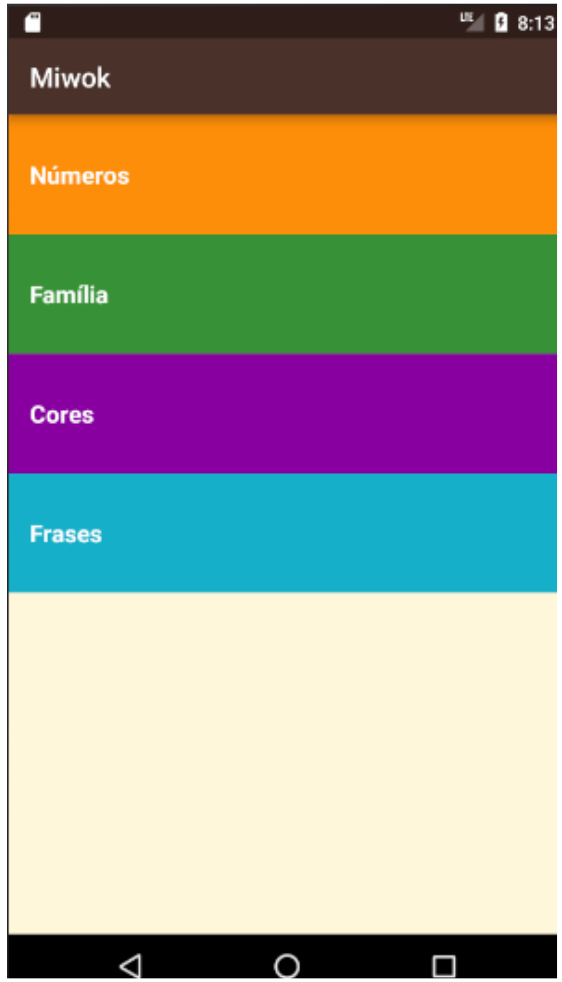
Finalizando a estilização inical

- Para finalizar o estilo inicial
  1. Façam a criação do recurso de string para armazenar o texto que aparece em cada um dos TextViews
  2. Chame o recurso criado no arquivo strings.xml no arquivo activity_main.xml

- Essa atividade é por sua conta e não possui resposta

Finalizando a estilização inical

- Nesse momento seu app deve estar sendo mostrado como na figura ao lado



Activities e Intents

- Toda vez que abandonamos uma tela em um dispositivo android e iniciamos a execução de uma nova tela, seja no nosso app ou em uma outra aplicação, estamos iniciando novas activities
- O processo de iniciar uma nova activity é gerenciado por Intents
- Existem dois tipos de Intents os explícitos e os implícitos

Intents Implícitos e Explícitos

- Intents implícitos são usados quando queremos chamar uma nova activity, mas não queremos forçar que uma determinada aplicação seja executada. Exemplos:
  - Quando queremos enviar um e-mail para reportar um bug na nossa aplicação, mas não queremos especificar qual aplicação de e-mail no dispositivo vai ser responsável pela ação, enviamos um intent implícito com ação de e-mail
  - A mesma coisa caso queiramos abrir um e-mail, tirar uma foto, executar um vídeo, etc

Intents Implícitos e Explícitos

- Intents explícitos são usado quando queremos que determinada aplicação rode a activity
- É esse tipo de intent que usamos para chamar novas activities na nossa aplicação
- Não utilizamos intents explícitos para forçar um link a ser aberto no Chrome, por exemplo, pois pode ser que o usuário tenha desinstalado o navegador, o que faz com que nossa aplicação pare de funcionar

Criando novas Activities

- Na nossa aplicação queremos criar 4 novas activities
- Uma para cada uma das seções existentes no aplicativo. São elas:
  1. NumerosActivity
  2. FamiliaActivity
  3. CoresActivity
  4. FrasesActivity
- Para isso basta ir com o botão direiro em app > New > Activity > Empty Activity e nomeamos como mostrado acima, deixamos marcado a opção para criar o respectivo arquivo XML

Chamando novas activities

- Com as activities criadas, precisamos agora chamar essas activities quando os usuário clica no devido TextView
- Nós sabemos fazer isso utilizando o atributo onClick no componente dentro do arquivo XML
- Agora veremos como fazer isso através de um listener

Listeners

- Para que possamos escutar (listen) a ocorrência de um evento de clique, precisamos identificar os componentes para que o aplicativo saiba onde que o usuário clicou
- Para isso, vamos adicionar ids aos nossos TextViews
- numeros, familia, cores e frases

```
...

<TextView
    android:id="@+id/numeros"
    style="@style/Categorias"
    android:background="@color/categoria_numeros"
    android:text="@string/categoria_numeros"/>

...
```

Listeners

- Agora que podemos identificar nossos TextViews, podemos capturá-los na MainActivity.java

```
...

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

    TextView numerosTextView = (TextView) findViewById(R.id.numeros);
}

...
```

Listeners

- Com o elemento capturado, podemos agora setar um novo listener para onClick com o método setOnClickListener
- Esse método recebe um objeto OnClickListener como parâmetro

```
...

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

    TextView numerosTextView = (TextView) findViewById(R.id.numeros);
    numerosTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this,
                NumerosActivity.class);

            startActivity(intent);
        }
    });
}

...
```

Listeners

- Agora, o Java nos permite criar essa classe anonimamente, ou seja, não precisamos criar uma classe somente para implementar um OnClickListerner
- Tudo o que precisamos fazer é sobrescrever os métodos obrigatórios, no nosso caso o método onClick

```
...

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

    TextView numerosTextView = (TextView) findViewById(R.id.numeros);
    numerosTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this,
                NumerosActivity.class);

            startActivity(intent);
        }
    });
}

...
```

Listeners

- No método onClick, precisamos criar um intente explícito saindo do contexto da MainActivity e iniciando a classe NumerosActivity
- Feito isso, basta executar o intent

```
...

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

    TextView numerosTextView = (TextView) findViewById(R.id.numeros);
    numerosTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this,
                NumerosActivity.class);

            startActivity(intent);
        }
    });
}

...
```

Criando os demais listeners

- Agora repita o processo de criação dos outros 3 listeners
- Teste se sua aplicação continua rodando

Pausa

Criando os demais listeners

- Você pode acompanhar o resultado final no link:

MainActivity.java

Alterando o título de cada activity

- Para alterar o título de cada activity, precisamos seguir alguns passos simples
  1. Abrir o arquivo AndroidManifest.xml
  2. Procurar pelo componente que descreve cada uma das nossas activities
  3. Adicionar em cada activity o parâmetro android:label com o título que queremos dar à activity

Alterando título das activities

- Um exemplo de solução para o problema é mostrado ao lado
- Complete por sua conta a alteração nas outras 3 activies

```
...

<activity android:name=".NumerosActivity"
    android:label="@string/categoria_numeros" />

...
```

Parte 2

ArrayLists, Gerenciamento de Memória, Classes e Reciclagens de Views

Objetivo

- O objetivo nesta fase do desenvolvimento, é conseguir mostrar nossos dados em listas
- Apresentando suas versões em português e Miwok para cada uma das nossas palavras

Arrays e ArrayLists

- Uma maneira muito convencional de armazenar dados que de alguma forma possuem alguma relação em qualquer linguagem de programação são os arrays
- Quando queremos armazenar um conjunto de valores como, por exemplo, diversas marcas de carro, automaticamente a estrutura de array vem a nossa mente
- String[] marcas = new String[]{“Fiat”, “Ford”, “Volkswagen”};
- O problemas dos arrays é que eles possuem tamanho fixo

Arrays e ArrayLists

- Se, posteriormente, quisermos aumentar a quantidade de marcas de carros temos duas saídas:
  1. Atualizamos a quantidade de marcas do nosso vetor: o que gera uma carga de trabalho adicional
  2. Ou criamos um vetor com espaço de sobra: o que gera desperdício de recursos computacionais

Arrays e ArrayLists

- ArrayLists são um tipo de collection de dados existente em Java que nos permite trabalhar com arrays que variam de tamanho ao longo do tempo de maneira bem mais simples
- A diferença básica de um array para um ArrayList é que nos arrays nós podemos ter tipos de dados primitivos (como double, int, boolean, etc), ao passo que o ArrayList somente pode ser construído para objetos

ArrayList

- Então vamos criar nosso primeiro ArrayList para armazenar as nossas palavras (apenas em português, à princípio) dentro do método onCreate
- Em NumerosActivity, fazemos o seguinte
  - Criamos um ArrayList com o nome de palavras
  - Adicionamos cada item do ArrayList com o método add

```
...

ArrayList<String> palavras = new ArrayList<>();

        palavras.add("um");
        palavras.add("dois");
        palavras.add("três");
        palavras.add("quatro");
        palavras.add("cinco");
        palavras.add("seis");
        palavras.add("sete");
        palavras.add("oito");
        palavras.add("nove");
        palavras.add("dez");

...
```

ArrayList

- Adicionalmente, podemos printar no log do android studio, o conteúdo do array palavras para saber se tudo está correto

```
...

palavras.add("nove");
palavras.add("dez");

// Percorre o array completamente e printa no log
// de que classe que o log vem e a mensagem que
// deve ser exibida
for(int i = 0; i < palavras.size(); i++) {
    Log.v("NumerosActivity", "Posição " + i + ": "
     + palavras.get(i));
}

...
```

Printar as palavras na activity

- Primeiramente precisamos alterar nossa números_activity de um constraint layout para um linear layout vertical
- E, então, identificá-lo para que possamos referenciá-lo corretamente

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/raiz"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.android.miwok.NumerosActivity">

</LinearLayout>
```

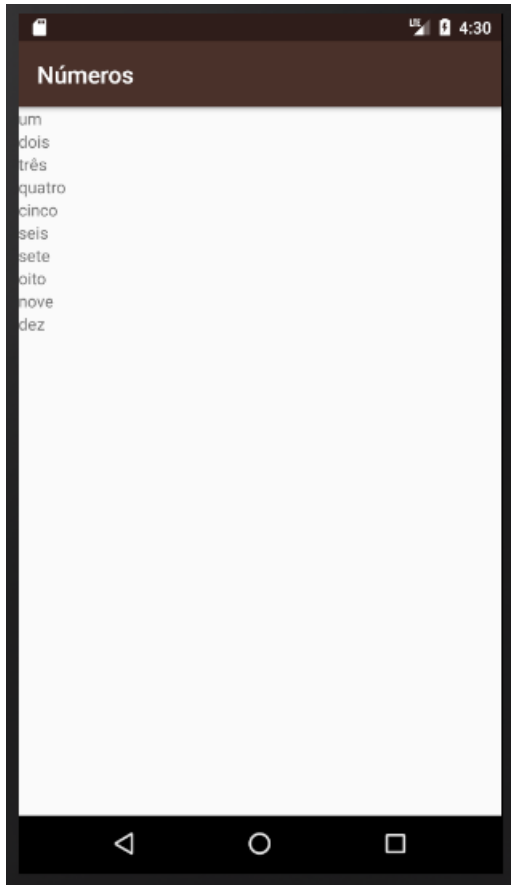
Printar as palavras na activity

- Feito o passo anterior, voltamos em NumerosActivity e ao invés de mostrar os números em log
  - Resgatamos a view onde queremos posicionar nosso texto
  - Criamos uma TextView e setamos seu texto corretamente no java
  - E adicionamos essa view no elemento raiz

```
...

palavras.add("nove");
palavras.add("dez");

LinearLayout raiz = (LinearLayout) findViewById(R.id.raiz);

for(int i = 0; i < palavras.size(); i++) {
    TextView palavraTextView = new TextView(this);
    palavraTextView.setText(palavras.get(i));

    raiz.addView(palavraTextView);
}

...
```

Printar as palavras na activity

- O RESULTADO DESSAS ALTERAÇÕES É MOSTRADO AO LADO



Memória é um recurso limitado

- Quando desenvolvíamos aplicações para outras plataformas, muitas vezes deixávamos de lado nossa preocupação com o gerenciamento de memória
- Mas isso é um tópico que não pode ficar de lado quando tratamos de desenvolvimento mobile
- A memória nesses dispositivos é um recurso muito escasso
- Então devemos sempre fazer o melhor uso dela possível
- Se a nossa aplicação crescer consideravelmente e deixarmos de mostrar a tradução de 1 a 10 e passarmos a mostrar a tradução de 1 a 1000, o custo de memória da nossa aplicação cresce espantosamente a ponto de ela se tornar uma péssima opção para se aprender línguas

Memória é um recurso limitado

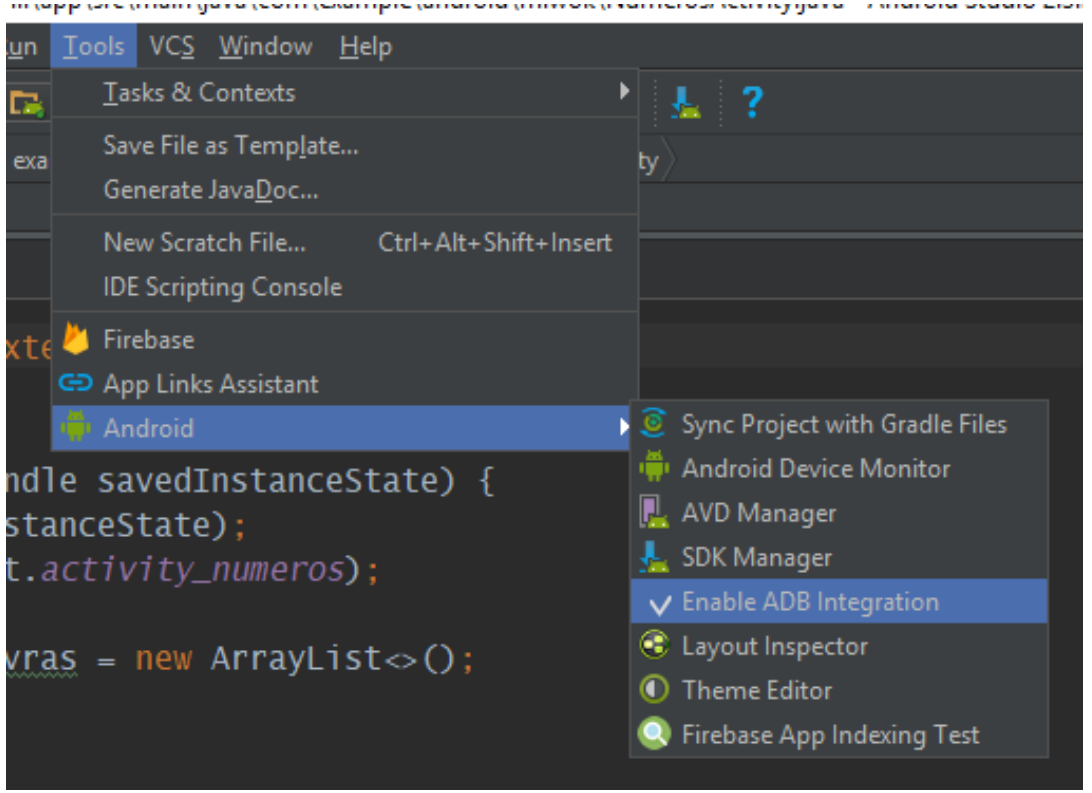
- Pensando um pouco mais na situação de mostrarmos 1000 palavras...
  - Não há necessidade de carregarmos todas elas assim que o usuário inicia a activity, pois apenas uma porção delas serão apresentadas na tela por vez
  - Então ao invés de carregarmos todas as palavras, carregamos o que precisamos para mostrar os dados na tela conforme o usuário utiliza a aplicação
  - Um exemplo disse é o seguinte:
    - Pense na sua lista de contatos. Se você tem 500 nomes nessa lista mas procurou por um nome como Bernardo, todos os nomes que foram carregados após o Bernardo, foram basicamente desperdício de memória

Memória é um recurso limitado

- Para resolvermos esse problema, o Android tem o que chamamos de reciclagem de views
- Que funciona basicamente criando-se 2 componentes
  - Um ListView
  - E um adapter
- Esse processo cria uma quantidade de views para mostrar os dados na tela, e quando rolamos a tela para ver o conteúdo da sequência, os dados de uma view que não está mais na tela, são trocados pelos dados da view que entrou na tela e nenhum outro componente é criado

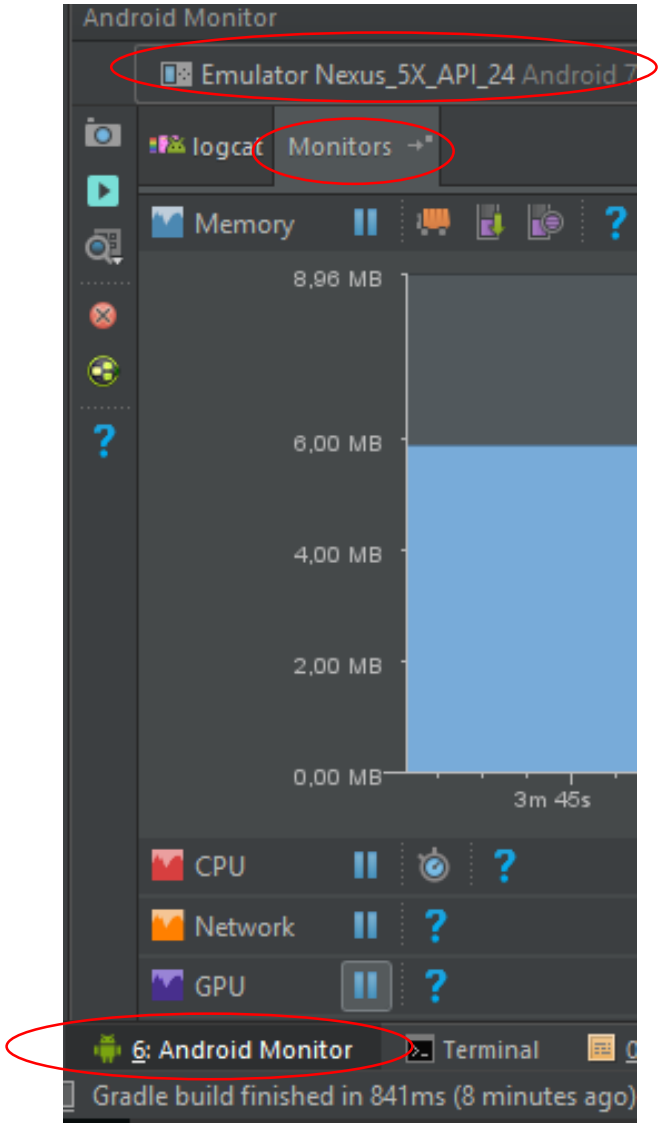
Visualizando a quantidade de memória gasta pelo app

- Para que possamos ver o quanto de memória nossa aplicação gasta vamos em:
  - Tools
  - Android
  - Enable ADB Integration



Visualizando a quantidade de memória gasta pelo app

- Para acompanharmos o custo de memória, rodamos nossa aplicação novamente e:
  - Abrimos o android monitor
  - Clicamos em monitors
  - Pode ser que o dispositivo tenha que ser trocado para o monitor mostrar os dados corretamente



Inserindo ArrayAdapter

- Primeiramente, devemos substituir o Layout de activity_numeros por um ListView

```
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/list"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context="com.example.android.miwok.NumerosActivity"/>
```

Inserindo ArrayAdapter

- Depois:
  - Criamos o ArrayAdapter
  - Resgatamos o ListView
  - E inserimos o adapter no ListView
- Esse código irá substituir o laço for de inserção de dados

```
...

palavras.add("nove");
palavras.add("dez");

// O construtor do adapter usa 3 parâmetros
// O contexto onde o adapter será inserido
// O id do layout que será usado para construir a lista
// O array com os dados a serem inseridos
ArrayAdapter<String> itensAdapter =
        new ArrayAdapter<String>(
                this,
                android.R.layout.simple_list_item_1,
                palavras);

ListView listView = (ListView) findViewById(R.id.list);

listView.setAdapter(itensAdapter);

...
```

Inserindo ArrayAdapter

- O resultado dessas alterações é mostrado ao lado
- Essa linha separando os itens é nativa do ListView

Layouts Padrões

- Você deve ter notado esse `R.layout.simple_list_item_1`
- Esse é um layout que já vem pronto do framework Android e ele desenha na tela um TextView simples
- Você pode ver mais informações AQUI

```
...

palavras.add("nove");
palavras.add("dez");

// O construtor do adapter usa 3 parâmetros
// O contexto onde o adapter será inserido
// O id do layout que será usado para construir a lista
// O array com os dados a serem inseridos
ArrayAdapter<String> itensAdapter =
        new ArrayAdapter<String>(
                this,
                android.R.layout.simple_list_item_1,
                palavras);

ListView listView = (ListView) findViewById(R.id.list);

listView.setAdapter(itensAdapter);

...
```

Layout customizado para ArrayAdapter

- Para o que temos na nossa aplicação até o momento esse `R.layout.simple_list_item_1` é suficiente
- Mas e se decidirmos adicionar agora, a versão em Miwok de cada uma das palavras?
  - Como esse layout provê apenas um TextView e nada mais, isso seria impossível (a menos que fôssemos mostrar as duas palavras em um mesmo TextView, o que não é o caso)
  - Para isso devemos ter a condição de armazenar cada palavra em sua versão padrão e Miwok em uma estrutura única
    - E é nesse momento que o conceito de classe entra

Criando nossa classe Palavra

- Vamos agora criar a nossa classe palavra que deve:
  1. Armazenar uma tradução padrão
  2. Armazenar uma tradução Miwok
  3. Construir um objeto passando as duas palavras no momento da inicialização
  4. Permitir resgatar os valores dessas variáveis posteriormente
- Antes de olhar a resposta, tente fazer isso você mesmo

Pausa

Criando a classe Palavra

- Criamos nova classe java clicando com o botão direito em app > java > com.example.android.miwok e escolhendo New > Java Class
- As variáveis para a tradução privadas (esse m minúsculo no início da palavra é um padrão que se utiliza para facilitar a identificação de variáveis globais e locais e exclui a necessidade do uso do this)
- Junto ao construtor e aos getters

```
public class Palavra {

    private String mTraducaoPadrao;
    private String mTraducaoMiwok;

    public Palavra(String traducaoPadrao, String traducaoMiwok) {
        mTraducaoPadrao = traducaoPadrao;
        mTraducaoMiwok = traducaoMiwok;
    }

    public String getTraducaoPadrao() {
        return mTraducaoPadrao;
    }

    public String getTraducaoMiwok() {
        return mTraducaoMiwok;
    }
}
```

Definindo o arquivo de recurso para os itens da lista

- Precisamos criar um arquivo de recurso de layout para os itens que queremos mostrar no ListView. Para isso vamos em:
  - app
  - res
  - layout com o botão direito
  - New > Layout Resource File
    - O nome do arquivo será item_lista.xml
    - E o root elemento será um linear layout

Arquivo item_lista.xml

- Agora devemos definir um layout que atenda as diretrizes mostradas ao lado
- Devemos identificar os dois TextViews para que possamos fazer as referência no Java de maneira adequada

Pausa

Arquivo item_lista.xml

- Código para resolução do problema
- Um conceito novo aqui é o de namespace
  - Em arquivos xml do android o namespace é a parte que aparece antes dos :
  - Aqui utilizamos o namespace tools para inserir texto, pois o android não vai gerar alerta de string
  - Ele reconhece o text do namespace tools como texto de exemplo e que não deve aparecer no app
  - Esse namespace é importado no elemento raiz do arquivo xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView
        android:id="@+id/miwok"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="lutti"/>

    <TextView
        android:id="@+id/padrao"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="one"/>

</LinearLayout>
```

Populando o ArrayList com Palavras no lugar de Strings

- Vá ao arquivo NumerosActivity
- Faça a alteração para que, agora, ele receba a palavra com as suas traduções padrão e Miwok
- Link para a lista de palavras e traduções (inglês)

Pausa

Adicionando palavras e traduções

- Resposta do desafio
- Essa alteração pode (e provavelmente vai) gerar um erro no ArrayAdapter logo abaixo
  - Resolveremos isso depois

```
...

ArrayList<Palavra> palavras = new ArrayList<>();

palavras.add(new Palavra("um", "lutti"));
palavras.add(new Palavra("dois", "otiiko"));
palavras.add(new Palavra("três", "tolookosu"));
palavras.add(new Palavra("quatro", "oyyisa"));
palavras.add(new Palavra("cindo", "massokka"));
palavras.add(new Palavra("seis", "temmokka"));
palavras.add(new Palavra("sete", "kenekaku"));
palavras.add(new Palavra("oito", "kawinta"));
palavras.add(new Palavra("nove", "wo’e"));
palavras.add(new Palavra("dez", "na’aacha"));

...
```

Criando nosso ArrayAdapter Personalizado

- Para que possamos mostrar algo diferente de um TextView em um ArrayAdapter, precisamos criar nosso próprio ArrayAdapter
- Fazemos isso dando um extends em ArrayAdapter<objeto_customizado_de_dados>
- A seguir tem um link para um arquivo de exemplo de um ArrayAdapter, quero que você dê seu melhor para tentar fazer o seu ArrayAdapter funcionar por conta própria

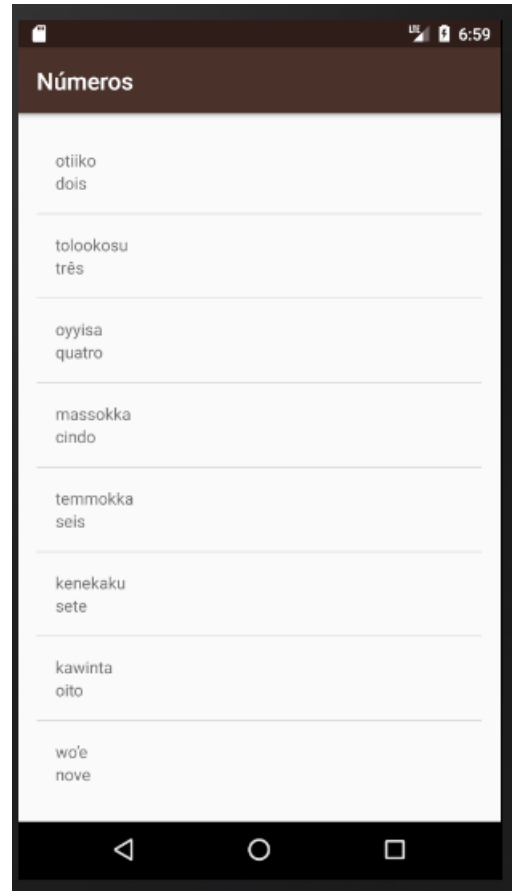
Criando nosso ArrayAdapter Personalizado

- Clique AQUI para ver o arquivo de exemplo de um ArrayAdapter
- As alterações que devem ser feitas em NumerosActivity são mostradas ao lado

```
...

PalavraAdapter itensAdapter = new PalavraAdapter(this, palavras);

ListView listView = (ListView) findViewById(R.id.list);

listView.setAdapter(itensAdapter);

...
```

Criando nosso ArrayAdapter Personalizado

- Depois de implementar as alterações, seu aplicativo deve se parecer com o exemplo ao lado



Pausa

Criando nosso ArrayAdapter Personalizado

- Primeiramente devemos definir o construtor
  - Ele recebe a activity onde deve ser criado o adpater e o array de palavras
  - E passa a criação para a classe ArrayAdapter com o super
  - O parâmetro 0, indica que estaremos criando uma visualização personalizada

```
...

public class PalavraAdapter extends ArrayAdapter<Palavra> {

    public PalavraAdapter(Activity context, ArrayList<Palavra> palavras) {
        super(context, 0, palavras);
    }

...
```

Criando nosso ArrayAdapter Personalizado

- Sobrescrevemos o método getView
  - Colocamos o convertView em uma variável itemListView para trabalharmos ela localmente
  - O if confere se eu vou usar uma view reciclada ou se vou inflar (criar) uma nova view
  - Resgato o item que está sendo construído com o método getItem
  - Coloco as informações nos TextViews
  - Retorno o componente criado

```
...

@NonNull
@Override
public View getView(int position, @Nullable View convertView,
                    @NonNull ViewGroup parent) {
    View itemListaView = convertView;

    if(itemListaView == null) {
        itemListaView = LayoutInflater.from(getContext())
            .inflate(R.layout.item_lista, parent, false);
    }

    Palavra palavraAtual = getItem(position);

    TextView miwok = (TextView) itemListaView.findViewById(R.id.miwok);
    miwok.setText(palavraAtual.getTraducaoMiwok());

    TextView padrao = (TextView) itemListaView.findViewById(R.id.padrao);
    padrao.setText(palavraAtual.getTraducaoPadrao());

    return itemListaView;
}

...
```

Ajustes finais do ArrayAdapter

- Agora queremos repassar essas alterações para todos as outras activities
- Para que não tenhamos que reescrever o ListView em todos os componentes, vamos renomear o activity_numeros.xml para lista_palavras.xml (botão direito no arquivo refactor > rename)

Ajustes finais do ArrayAdapter

- Automaticamente, o layout chamada em NumerosActivity passou de activity_numeros para lista_palavras
- Para que você possa usar esse mesmo ListView em outras activities, bas chamar lista_palavras no setContentView de cada activity

```
...

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

    ArrayList<Palavra> palavras = new ArrayList<>();
...
```

Ajustes finais do ArrayAdapter

- Faça as alterações necessárias para que todas as activities funcionem corretamente com seus ArrayAdpters
- Essa atividade serve apenas para reforçar os conceitos e não tem resposta, portanto, preste bastante atenção e leia os logs de erros caso alguma das suas alterações não funcione como o esperado

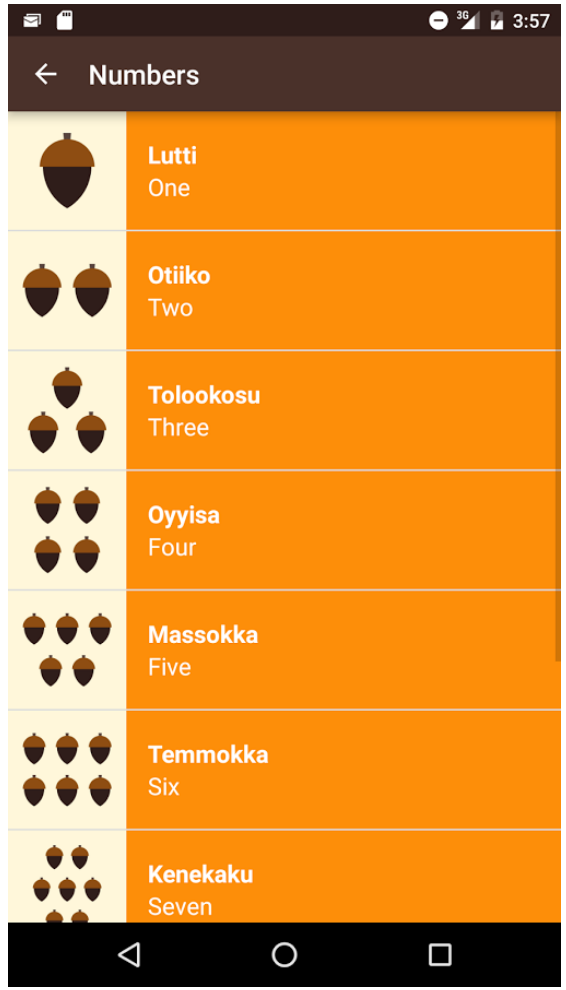
Pausa

Parte 3

Ajustes visuais,
Adição de imagens e
Tratando itens que fogem ao padrão

Planejando ajustes visuais

- O objetivo dessa parte do material é chegar nesse layout para a nossa aplicação
- Qual seria o fluxo total de alterações a serem realizadas?
- Tire uns segundos, analise tudo o que você já fez e tente imaginar os lugares que deveriam sofrer alterações para que conseguíssemos atingir esse layout



Pausa

Planejando ajustes visuais

- Basicamente, o que temos que fazer é:

1. Ajustar a nossa listagem para que seja capaz de receber uma imagem e para que se pareça um pouco mais com o layout que é apresentado na imagem
2. Ter as imagens para colocar em cada item da lista
3. Ajustar a classe Palavra para que ela seja capaz de armazenar a imagem que está vinculada a cada Palavra
4. Corrigir como o item é listado no adapter

- Agora que sabemos quais itens temos que alterar, qual seria uma ordem para a execução dessas atividades?
- Pense em um cenário onde teríamos as imagens assim que começamos essa fase e em um outro cenário onde essa imagem pode demorar um pouco para chegar

Pausa

Planejando ajustes visuais

- Dentre as atividades que tínhamos, uma sequência válida é:
  - 1, 3, 4, 2
  - Ajustamos a listagem e para execução das demais atividades, pegamos uma imagem qualquer e utilizamos como dado provisório da nossa listagem, assim desvencilhamos o nosso desenvolvimento da necessidade de um item que pode demorar a chegar

1. Ajustar a nossa listagem para que seja capaz de receber uma imagem e para que se pareça um pouco mais com o layout que é apresentado na imagem
2. Ter as imagens para colocar em cada item da lista
3. Ajustar a classe Palavra para que ela seja capaz de armazenar a imagem que está vinculada a cada Palavra
4. Corrigir como o item é listado no adapter

Executando as modificações

- Então, vamos começar pela atividade:

1. Ajustar a nossa listagem para que seja capaz de receber uma imagem e para que se pareça um pouco mais com o layout que é apresentado na imagem

- Vamos começar adicionando uma imagem temporária aqui
- Coloque o componente de imagem para referenciar a ícone do nosso app
  - O local é @mipmap/ic_launcher

Pausa

Executando as modificações

- Quando aplicamos essa alteração, temos uma imagem sobre as palavras
- O que queremos é que a imagem fique horizontalmente alinhada às palavras
- E que as palavras fiquem verticalmente alinhadas entre si
- Para isso podemos utilizar o conceito de ViewGroups aninhados, tente chegar no nosso objetivo de layout

```
<ImageView
    android:id="@+id/container_imagem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/ic_launcher"/>
```

Pausa

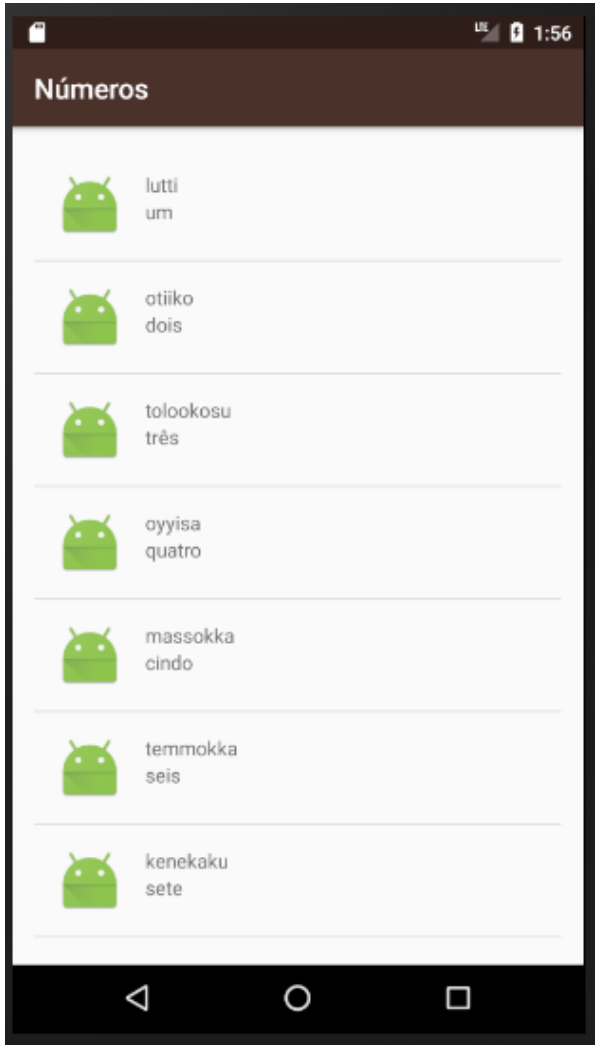
Executando modificações

- Uma solução possível para o nosso problema é mostrada ao lado
- Um linear layout externo agrupando todos os componentes horizontalmente
- E um linear layout vertical agrupando apenas as palavras

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <ImageView
        android:id="@+id/container_imagem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"/>

    <LinearLayout
        android:id="@+id/container_palavras"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="16dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/miwok"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            tools:text="lutti"/>

        <TextView
            android:id="@+id/padrao"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            tools:text="one"/>

    </LinearLayout>
</LinearLayout>
```

Resultado

- Quando rodamos nossa aplicação, temos o resultado ao lado



Executando ajustes de layout

- Agora que nosso container para cara item da lista já consegue receber uma imagem, podemos executar a atividade 3:

3. Ajustar a classe Palavra para que ela seja capaz de armazenar a imagem que está vinculada a cada Palavra

- Para isso, guardaremos uma referência para a imagem que queremos vincular a cada palavra
- Primeiramente vamos fazer um teste, peça ao Android Studio para mostrar em log na main activity ao final do método onCreate o que recebemos quando fazemos referência à nossa imagem mipmap
- `Log.v("MainActivity", "O QUE TEMOS QUANDO CHAMAMOS UMA REFERÊNCA R DE UM OBJETO MIPMAP NO JAVA É: " + R.mipmap.ic_launcher);`

Executando ajustes de layout

- Você vai ver que a resposta vai ser um inteiro, ou seja, para que consigamos fazer referência ao objeto de imagem, precisamos armazenar um inteiro na nossa classe Palavra,
- Sabendo disso, execute as alterações devidas na classe palavra

Pausa

Alterando a classe Palavra

- O resultado da classe palavra deve ser o mostrado ao lado
- Essa alteração gera um erro em cascata
- Volte e corrija todas as activities para adicionarem a imagem mipmap no momento em que constroem um objeto Palavra

```
public class Palavra {

    private String mTraducaoPadrao;
    private String mTraducaoMiwok;
    private int mReferenciaImagem;

    public Palavra(String traducaoPadrao, String traducaoMiwok,
                   int referenciaImagem) {
        mTraducaoPadrao = traducaoPadrao;
        mTraducaoMiwok = traducaoMiwok;
        mReferenciaImagem = referenciaImagem;
    }

    public String getTraducaoPadrao() {
        return mTraducaoPadrao;
    }

    public String getTraducaoMiwok() {
        return mTraducaoMiwok;
    }

    public int getReferenciaImagem() { return mReferenciaImagem; }
}
```

Corrigindo os erros nos construtores

- Essa seria a correção feita para adicionar a imagem à activity Numeros
- Faça o mesmo para as demais activities

```
palavras.add(new Palavra("um", "lutti", R.mipmap.ic_launcher));
palavras.add(new Palavra("dois", "otiiko", R.mipmap.ic_launcher));
palavras.add(new Palavra("três", "tolookosu", R.mipmap.ic_launcher));
palavras.add(new Palavra("quatro", "oyyisa", R.mipmap.ic_launcher));
palavras.add(new Palavra("cindo", "massokka", R.mipmap.ic_launcher));
palavras.add(new Palavra("seis", "temmokka", R.mipmap.ic_launcher));
palavras.add(new Palavra("sete", "kenekaku", R.mipmap.ic_launcher));
palavras.add(new Palavra("oito", "kawinta", R.mipmap.ic_launcher));
palavras.add(new Palavra("nove", "wo’e", R.mipmap.ic_launcher));
palavras.add(new Palavra("dez", "na’aacha", R.mipmap.ic_launcher));
```

Executando alterações

- Passemos agora para a seguinte tarefa:

4. Corrigir como o item é listado no adapter

- Vamos corrigir o adpater para que ele adicione em tempo de execução, as imagens para cada uma das palavras do nosso dicionário Miwok
- Pense e tente executar essa alteração
- Nesse momento o app não deve mudar nada, pois as imagens que referenciamos e que usamos no nosso arquivo de layout, será a mesma

Pausa

Alterando o adapter

- As alterações no adapter são simples
- Para adicionar um recurso de imagem precisamos informar seu ID, e esse é justamente o valor que temos, por isso utilizamos o método setImageResource
- Temos várias maneiras de adicionar recursos de imagem, essa é apenas uma delas e a única que se encaixa no modo como desenvolvemos nosso app até agora

```
ImageView imagem = (ImageView) itemListaView
                    .findViewById(R.id.container_imagem);
imagem.setImageResource(palavraAtual
            .getReferenciaImagem());
```

Executando alterações

- Passemos agora para a seguinte tarefa:

1. Ter as imagens para colocar em cada item da lista

- Nesse momento obtemos acesso às imagens que serão utilizadas no app
- Elas se encontrar nesse LINK
- Faça o download delas como zip
- Descompacte o arquivo e navegue na pasta até chegar no conteúdo da pasta assets
- Copie todas 5 pastas aqui
- Volte no android studio, clique com o botão direito na pasta res e selecione a opção colar
- Agora corrija as referências para as imagens
- NOTA: não temos imagens para a categoria de frases e vamos dedicar um tempo a corrigir essa questão específica

Pausa

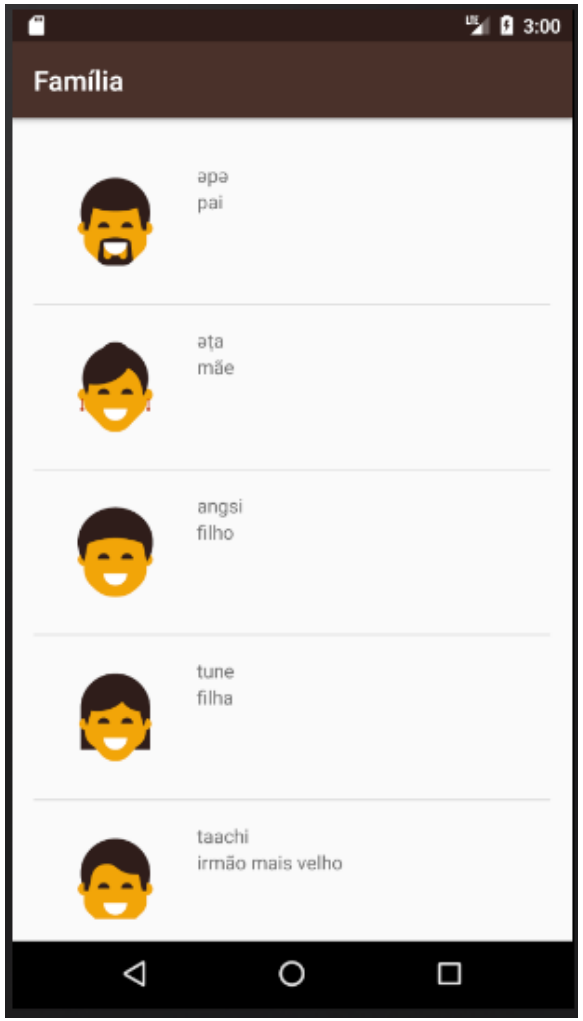
Alterando os resources

- Essa fica sendo a alteração para a activity Familia
- Faça o mesmo processo de atualização para as activities Numeros e Cores
- Depois rode seu programa e se certifique de que as imagens para essas 3 categorias são mostradas corretamente

```
palavras.add(new Palavra("pai", "əpə", R.drawable.family_father));
palavras.add(new Palavra("mãe", "əṭa", R.drawable.family_mother));
palavras.add(new Palavra("filho", "angsi", R.drawable.family_son));
palavras.add(new Palavra("filha", "tune", R.drawable.family_daughter));
palavras.add(new Palavra("irmão mais velho", "taachi", R.drawable.family_older_brother));
palavras.add(new Palavra("irmão mais novo", "chalitti", R.drawable.family_younger_brother));
palavras.add(new Palavra("irmã mais velha", "tete", R.drawable.family_older_sister));
palavras.add(new Palavra("irmã mais nova", "kolliti", R.drawable.family_younger_sister));
palavras.add(new Palavra("avó", "ama", R.drawable.family_grandmother));
palavras.add(new Palavra("avô", "paapa", R.drawable.family_grandfather));
```

Alterando imagens

- O RESULTADO PARA A ALTERAÇÕES DESSAS IMAGENS É MOSTRADA AO LADO



Corrigindo Frases

- Para que a activity de frases funcione corretamente, ela não deve apresentar nenhuma imagem
- Para resolver esse problema vamos abordá-lo de maneira iterativa
- Primeiro vamos começar criando um construtor na classe Palavra que aceite um objeto ser criado sem uma imagem

Pausa

Criando um construtor sem imagem

- Esse processo é bem simples e basta que tenhamos um novo construtor igual ao que tínhamos antes de adicionar uma variável para reter a imagem

```
public Palavra(String traducaoPadrao, String traducaoMiwok) {
    mTraducaoPadrao = traducaoPadrao;
    mTraducaoMiwok = traducaoMiwok;
}
```

Agora atualizamos a criação das frases

- Retornamos o construtor de frases para o modelo que somente aceita a palavra e sua tradução

```
palavras.add(new Palavra("onde você está indo?", "minto wuksus"));
palavras.add(new Palavra("qual o seu nome?", "tinnə oyaase'nə"));
palavras.add(new Palavra("meu nome é...", "oyaaset..."));
palavras.add(new Palavra("como está se sentindo?", "michəksəs?"));
palavras.add(new Palavra("estou me sentindo bem", "kuchi achit"));
palavras.add(new Palavra("você está vindo?", "əənəs'aa?"));
palavras.add(new Palavra("sim, estou indo", "həə’ əənəm"));
palavras.add(new Palavra("estou indo", "əənəm"));
palavras.add(new Palavra("vamos", "yoowutis"));
palavras.add(new Palavra("venha aqui", "ənni'nem"));
```

Polimento da classe Palavra

- Agora precisamos de uma maneira de dizer à classe PalavraAdapter se uma palavra tem ou não imagem
- Fazemos isso com pequenas alterações na classe Palavra
- Como a variável mReferenciaImagem é um int com o id da imagem, vamos fornecer um id inválido (no caso -1)
- Poderia ter colocado esse valor direto na inicialização da variável?

```
...

private int mReferenciaImagem = SEM_IMAGEM_FORNECIDA;
private static final int SEM_IMAGEM_FORNECIDA = -1;

....
....

public boolean hasImagem() {
    return mReferenciaImagem != SEM_IMAGEM_FORNECIDA;
}


...
```

Polimento da classe Palavra (cont.)

- Sim, poderia.
- Mas é uma boa prática de programação inserir constantes para esse tipo de cenário, pois elas facilitam o entendimento do código
- Adicionalmente, criamos um método hasImagem que retorna true se tem imagem e false, caso contrário

```
...

private int mReferenciaImagem = SEM_IMAGEM_FORNECIDA;
private static final int SEM_IMAGEM_FORNECIDA = -1;

....
....

public boolean hasImagem() {
    return mReferenciaImagem != SEM_IMAGEM_FORNECIDA;
}


...
```

Visibilidade

- Quando queremos que determinado componente apareça ou suma da tela, devemos trabalhar seu atributo de visibilidade
- Para isso utilizamos o método setVisibility()
  - Esse método recebe um inteiro que informa a visibilidade do componente
    - Mas como saberemos qual inteiro representa qual tipo de visibilidade?
      - Não precisamos saber!! Para isso, temos as constantes estáticas da classe View: VISIBLE, INVISIBLE e GONE
      - View.VISIBLE: faz o componente visível
      - View.INVISIBLE: faz o componente invisível, mas ele continua a ocupar espaço na tela
      - View.GONE: o objeto não aparece mais na tela e não ocupa mais espaço
- Sabendo disso, qual seria a alteração na classe PalavaAdapter a ser feita para que o componente funcione corretamente?

Pausa

Alterando PalavraAdapter

- Precisamos conferir se a palavra em questão possui imagem
- Se sim, encontramos ela e mostramos
- Se não, sumimos com o componente de imagem da tela

```
...

ImageView imagem = (ImageView) itemListaView.findViewById(R.id.container_imagem);
if(palavraAtual.hasImagem()) {
    imagem.setImageResource(palavraAtual.getReferenciaImagem());
    imagem.setVisibility(View.VISIBLE);
} else {
    imagem.setVisibility(View.GONE);
}

...
```

Resultado

- O RESULTADO NA TELA DE FRASES DEVE SER O SEGUINTE

Finalizando o polimento visual

- Para que possamos finalizar o polimento visual nós temos o link a seguir mostrando as referências que devemos utilizar

  Clique aqui para o PDF com as referências

- Preocupe-se primeiramente em acertar as dimensões passadas no arquivo e pode deixar uma mesma cor em todas as categorias

Pausa

lista_palavras.xml

- Vamos começar removendo o padding do componente do ListView

```xml
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/list"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.miwok.NumerosActivity"/>
```

colors.xml

- Agora vamos adicionar o tom pálido que deve ser usado atrás das imagens ao arquivo de cores

```
...

<color name="palida">#FFF7DA</color>

...
```

item_lista.xml

- Agora vamos estilizar o texto que deve aparecer em cada item da lista

```
...

<LinearLayout
    android:id="@+id/container_palavras"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="88dp"
    android:paddingLeft="16dp"
    android:background="@color/categoria_numeros"
    android:orientation="vertical">

    <TextView
        android:id="@+id/miwok"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textColor="#FFFFFF"
        android:textStyle="bold"
        android:textSize="18sp"
        android:gravity="bottom"
        tools:text="lutti"/>

    <TextView
        android:id="@+id/padrao"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textColor="#FFFFFF"
        android:textSize="18sp"
        tools:text="one"/>

</LinearLayout>

...
```

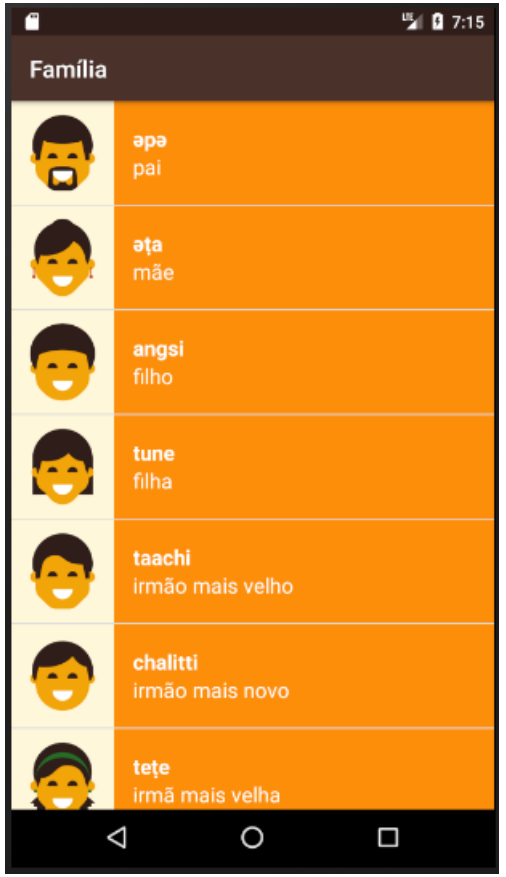
item_lista.xml

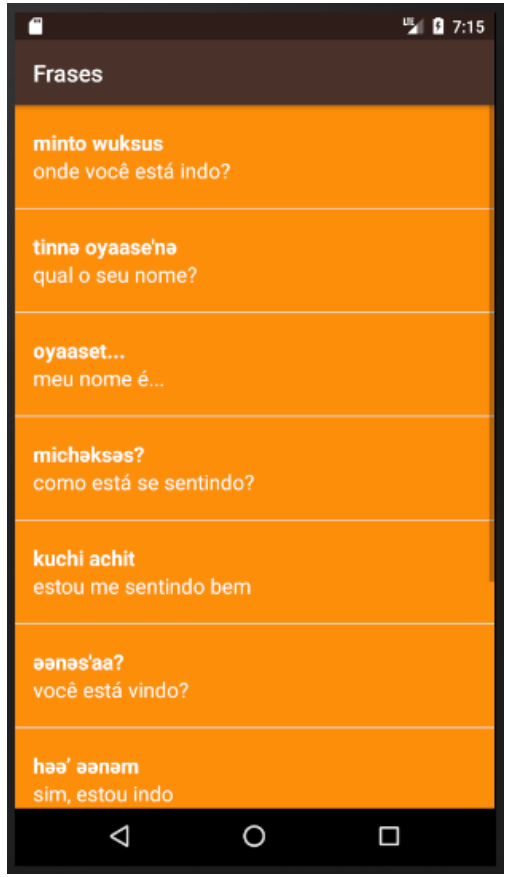
- Agora vamos estilizar o componente de imagem para que tenha as mesmas características definidas no layout
- Adicionalmente, se seu componente mais externo tiver algum padding aqui, remova-o

```
...

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/container_global">

    <ImageView
        android:id="@+id/container_imagem"
        android:layout_width="88dp"
        android:layout_height="88dp"
        android:background="@color/palida"/>

...
```

Finalizando primeira etapa

- Após fazer as alterações passadas a interface da sua aplicação deve se parecer com o seguinte





Planejando background

- Agora pense o seguinte?
- Onde seria o melhor lugar para eu adicionar a informação das cores a serem utilizadas na aplicação? Reformulando, onde eu poderia inserir essa informação de modo a ser o menos redundante possível?
  - Classe Palavra?
  - Classe PalavraAdapter?
  - Nas Activities?

Pausa

Planejando background

- A resposta mais adequada à pergunta é na classe
  - PalavraAdapter
- A classe Palavra também seria uma alternativa, o problema é que seria muito redundante ter essa informação em cada uma das minhas palavras
- Sabendo disso, tente fazer com que cada uma das categorias tenha sua cor corretamente aplicadas no PalavraAdapter

Pausa

Alterando as Activities

- No momento em que cada uma das activities cria o seu PalavraAdapter, ele passa o contexto e o vetor de palavras, agora ele também irá passar um recurso de cor
- Ao lado o exemplo aplicado na activity Familia

```
...

PalavraAdapter itensAdapter =
    new PalavraAdapter(this, palavras, R.color.categoria_familia);


...
```

Alterando PalavraAdapter

- Agora precisamos receber essa cor no construtor de PalavraAdapter bem como adicionar uma variável de classe para armazenar essa informação
- Lembrando que o super deve sempre ser o primeiro comando a ser executado em um servidor, caso ele exista

```
...

private int mCorFundo;

public PalavraAdapter(Activity context, ArrayList<Palavra> palavras,
                      int corFundo) {
    super(context, 0, palavras);
    mCorFundo = corFundo;
}


...
```

Alterando PalavraAdapter

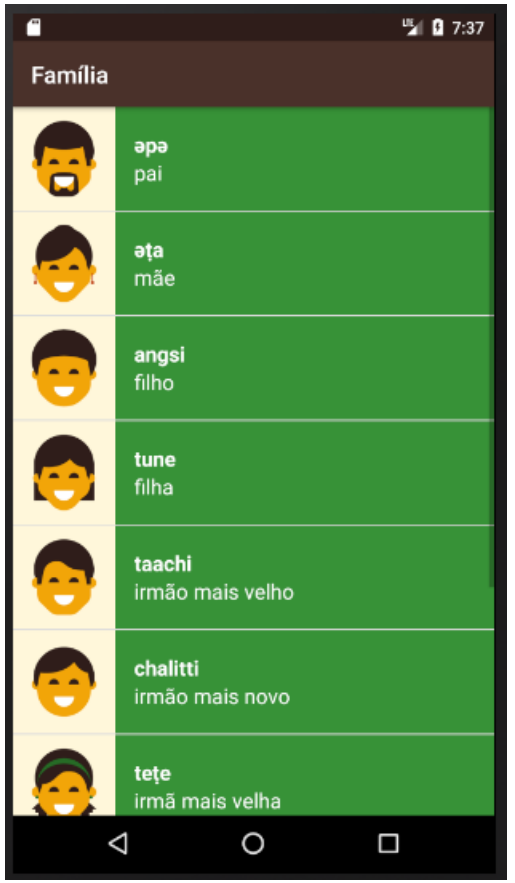
- Ainda na mesma classe, precisamos agora resgatar o componente de cada item da lista e estilizá-lo com a cor correta
- Se depois de testar sua aplicação as cores não forem aplicadas corretamente, tente tirar o background definido provisoriamente no arquivo item_lista.xml

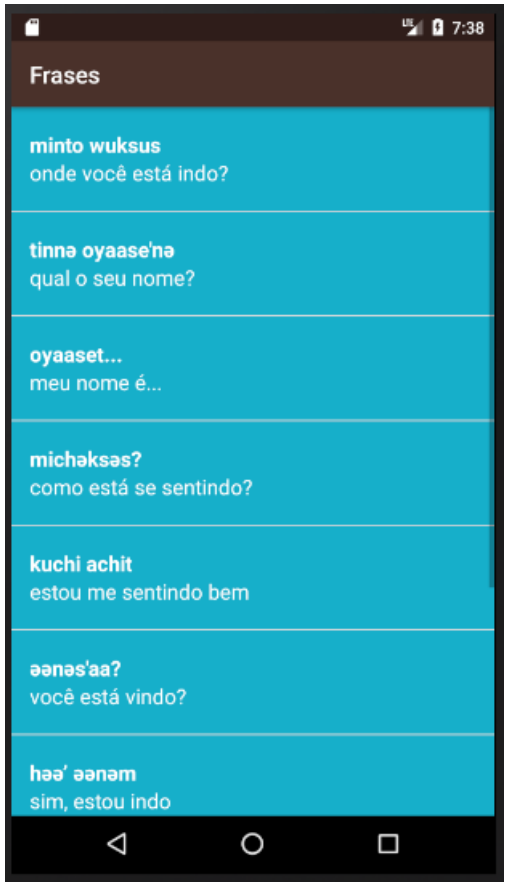
```
...

LinearLayout layout = (LinearLayout) itemListaView
        .findViewById(R.id.container_global);
layout.setBackgroundResource(mCorFundo);


...
```

Resultado de todas as alterações

- Nesse momento você finalizou a Parte 3 do projeto e seu app deve se parecer como o da imagem a seguir
- O estado atual do aplicativo pode ser encontrado no link abaixo

  Estado atual do projeto





Parte 4

Trabalhando com Áudio

Planejando a inserção do áudio

- Tire alguns minutinhos para imaginar como seria a inserção de áudio na nossa aplicação
- De que recursos poderíamos precisar?
- De que funcionalidades a aplicação irá necessitar para que tenhamos o comportamento esperado?
- Quais alterações no estilo poderíamos fazer para facilitar ao usuário identificar que há áudios que podem ser tocados para cada uma das palavras?

Pausa

Media Player

- Uma das coisas que nos vem a mente de cara é a necessidade de ter uma maneira de reproduzir arquivos de áudio no android
- Para isso, o framework já traz uma classe implementada pronta para trabalhar com arquivos de áudio a classe media player
- Nesse link você vai encontrar um pouco sobre o funcionamento da classe media player, bem como os arquivos de áudio suportados no android
- NOTA: A classe media player não é a única existente no android para lidar com esse problema

Media Player

- Documentação da classe MediaPlayer
- Esse vídeo em inglês (com legendas automáticas do YouTube) tem uma explicação muito legal sobre os estados de um áudio no Android
- Eu vou explicar em texto um pouco sobre os estados:
  - Idle: O seu objeto media player está pronto para reproduzir áudio mas ainda não decidiu o que tocar (pode se mover para prepared)
  - Prepared: Seu objeto está pronto e já sabe o que vai tocar (pode se mover para started)
  - Started: Seu áudio começa a tocar no dispositivo (pode se mover para paused ou stopped)
  - Paused: Seu áudio é pausado e fica aguardando para ser retomado (pode se mover para started)
  - Stopped: Seu áudio terminou de ser tocado ou foi finalizado pelo usuário (pode se mover para prepared)

Pequeno desafio

- Nesse momento, gostaria que você tirasse um tempo para criar um novo app de áudio
- Bem simples com 2 botões, um para tocar o áudio e outro para parar, coloque para tocar uma música que você goste e veja o resultado
- Para facilitar vou dar a dica de que os arquivos de áudio devem ser armazenados em uma pasta chamada raw dentro de res
- Você pode referenciar o áudio usando R.raw.nome_da_sua_musica
- Fica mais fácil para você se não salvar o nome do áudio com espaços

Pausa

Voltando ao nosso app

- Espero que tenha tentado fazer algo por conta própria e visto o resultado que um pouco de pesquisa sobre um conteúdo ainda não aprendido pode nos trazer de conhecimento
- Para começar as alterações do nosso app, faça os seguintes passos:
  - Baixe os arquivos de áudio desse LINK (clique em clone or download e baixe como zip)
  - Descompacte todos os arquivos e cole-os em uma pasta raw dentro de res no nosso app (exatamente como foi feito no app de exemplo)

Adicionando eventos de toque

- Agora nós temos os nossos arquivos de áudio e queremos fazer com que eles sejam reproduzidos corretamente de acordo com o item do ListView em que eles forem clicados
- Sabemos que quando queremos capturar um evento de clique em algum componente desenhado na tela nós usamos o OnClickListener
- Mas os itens de um ListView são criados dinamicamente
- Tire um tempo para pesquisar sobre qual eventos deveríamos capturar para que possamos tocar o nosso áudio corretamente para cada uma das nossas palavras

Pausa

Eventos de clique em itens no ListView

- Para que possamos capturar um evento de clique em um item de um ListView, devemos utilizar o método
  - setOnItemClickListener que recebe um objeto do tipo OnItemClickListener (de maneira muito similar ao que acontece em cliques em objetos comuns)
- Agora que sabemos o que devemos fazer, temos que definir onde vamos fazer
  - Para resolver esse problema, devemos tratar o clique em um item do ListView em um local onde temos controle sobre os elementos que estão sendo criados no ListView
  - No nosso caso, cada ListView é criado dentro das activities para cada uma das categorias

Adicionando evento de clique no ListView

- O CÓDIGO AO LADO MOSTRA COMO O NOSSO OBJETO PODE SER CRIADO PARA TRATAR OS EVENTOS DE CLIQUE NOS ITENS DA LISTA

```
...

ListView listView = (ListView) findViewById(R.id.list);

listView.setAdapter(itensAdapter);

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        MediaPlayer mediaPlayer =
                MediaPlayer.create(NumerosActivity.this, R.raw.number_one);
        mediaPlayer.start();
    }
});


...
```

Fazendo com que áudios sejam reproduzidos

- Vamos começar nossa implementação fazendo com que o mesmo áudio seja tocado em todos os itens da lista
- Adicione no evento de clique a ação necessária para que um objeto media player toque o som da palavra number_one
- Adicionalmente, crie uma variável de classe para armazenar o objeto MediaPlayer e utilize-a no evento

Pausa

Reproduzindo áudio

- Esse código cria um objeto media player dizendo que quer tocar um áudio na activity numbers e informa qual áudio deve ser tocado, ou seja, prepara o áudio
- Logo depois disso, ele começa a tocar o áudio e o faz até que ele seja finalizado

```
...

private MediaPlayer mMediaPlayer;

...
...
...

ListView listView = (ListView) findViewById(R.id.list);

listView.setAdapter(itensAdapter);

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        mMediaPlayer = MediaPlayer
                .create(NumerosActivity.this, R.raw.number_one);
        mMediaPlayer.start();
    }
});


...
```

Tocando áudios corretos

- No momento, nós tocamos o mesmo áudio para qualquer que seja a palavra na qual clicamos
- Onde eu devo armazenar a informação do áudio para cada palavra?
- Que tipo de informação eu devo armazenar para conseguir identificar corretamente o áudio posteriormente?

Pausa

Alterando Palavra

- Para que possamos armazenar o áudio para cada palavra, precisamos alterar a classe palavra para que ela receba uma informação de áudio
- Quando referenciamos um arquivo dentro da pasta raw, nós o fazemos utilizando inteiros, logo este deve ser o tipo da variável

```
private String mTraducaoPadrao;
private String mTraducaoMiwok;
private int mReferenciaImagem = SEM_IMAGEM_FORNECIDA;
private int mReferenciaAudio;
private static final int SEM_IMAGEM_FORNECIDA = -1;
```

Alterando Palavra

- Agora precisamos receber nos construtores o arquivo de áudio referente a cada uma das palavras
- Logo, também devemos alterar a criação dos objetos para que informem os áudios para cada palavra no momento de sua instanciação

```
...

//Activities
palavras.add(new Palavra("um", "lutti",
                R.drawable.number_one, R.raw.number_one));
palavras.add(new Palavra("dois", "otiiko",
        R.drawable.number_two, R.raw.number_two));
palavras.add(new Palavra("três", "tolookosu",
        R.drawable.number_three, R.raw.number_three));

...
...
...
//Classe Palavra
public Palavra(String traducaoPadrao, String traducaoMiwok,
               int referenciaAudio) {
    mTraducaoPadrao = traducaoPadrao;
    mTraducaoMiwok = traducaoMiwok;
    mReferenciaAudio = referenciaAudio;
}

public Palavra(String traducaoPadrao, String traducaoMiwok,
               int referenciaImagem, int referenciaAudio) {
    mTraducaoPadrao = traducaoPadrao;
    mTraducaoMiwok = traducaoMiwok;
    mReferenciaImagem = referenciaImagem;
    mReferenciaAudio = referenciaAudio;
}


...
```

Alterando Palavra

- O estado atual da aplicação pode ser encontrado no link abaixo

Estado atual da aplicação

Tocando o áudio correto

- No passo anterior, vimos como armazenar as informações de áudio corretamente, agora vamos avançar para o ponto onde pegamos essa informação e fazemos com que o áudio correto seja reproduzido baseado no item em que clicamos
- O método que implementamos dentro do OnItemClickListener recebe esses parâmetros:
  - `AdapterView<?> adapterView, View view, int i, long l`
- Acesse esse LINK para a documentação e tente descobrir com qual desses parâmetros conseguiremos acessar o item que foi clicado

Pausa

Tocando o áudio correto

- A resposta correta é a variável position, pois ela faz referência à view dentro do view adapter
- O problema é que não há veriável position no nosso código, isso ocorre pois o auto implemente do método quando criamos o nosso OnItemClickListener o gerou assim
- Para ficar mais fácil a identificação, vamos trocar o nome do terceiro parâmetro de I para position

```
...

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view,
                            int position, long l) {
        mMediaPlayer = MediaPlayer
                .create(NumerosActivity.this, R.raw.number_one);
        mMediaPlayer.start();
    }
});

...
```

Tocando o áudio correto

- Agora que sabemos que temos que pegar o item pela posição, recuperamos a palavra clicada através do método get (passando a posição do item) no array list de palavras
- Automaticamente, o Android Studio irá setar o array list palavras como final, se ele não fizer isso e você estiver recebendo um erro em palavras basta adicionar final antes da sua declaração
- Isso tem que ser feito para que a variável exista tanto no método quanto dentro do objeto OnItemClickListener que criamos
- Adicionalmente, crie o get de áudio na classe Palavra

```
...

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view,
                            int position, long l) {
        Palavra palavraClicada = palavras.get(position);
        mMediaPlayer = MediaPlayer
                .create(NumerosActivity.this, palavraClicada.getReferenciaAudio());
        mMediaPlayer.start();
    }
});

...
```

Tocando o áudio correto

- Replique as alterações nas demais activities de família, cores e frases para que funcionem corretamente

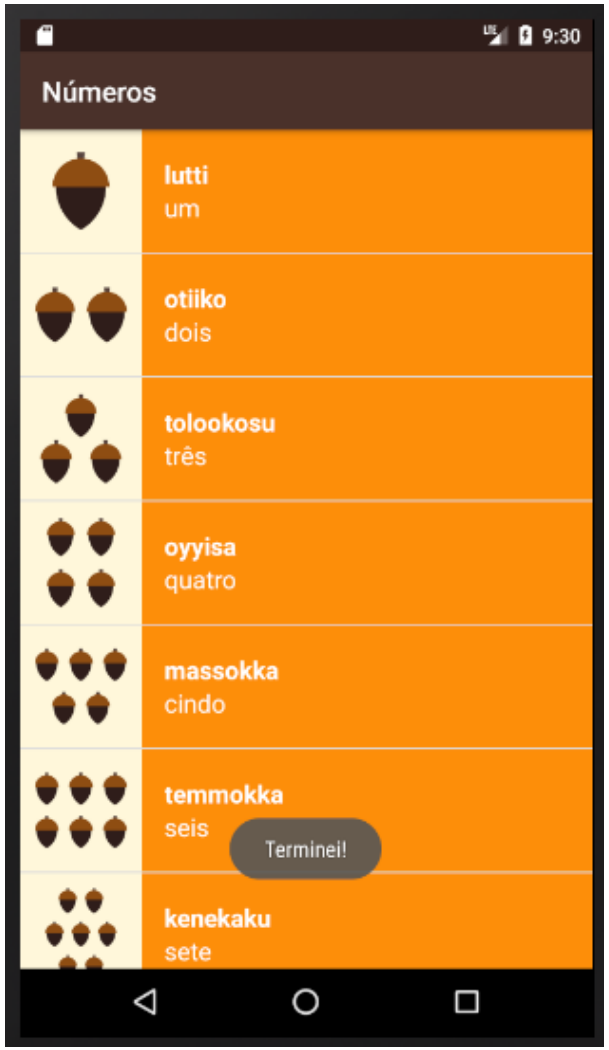
Pausa

Asynchronous Callbacks

- É basicamente fazer outra coisa enquanto você espera que um evento ocorra
- Isso é muito útil para a nossa situação, pois se você reparar, verá que o próximo áudio em que clicarmos, só será iniciado caso o anterior termine
- Para resolver esse problema, vamos recorrer aos async callbacks
- Faça uma pesquisa sobre asynchonous callbacks do media player e descubra qual deles notifica quando um áudio terminou de tocar
- Feito isso, implemente um callback para mostrar um Toast com o texto “Terminei!” quando o áudio tiver terminado de tocar
- Para que esse callback funcione, ele deve ser implementado depois que o áudio for colocado para tocar

Asynchronous Callbacks

- O RESULTADO NA TELA DEVE SER COMO O MOSTRADO AO LADO



Pause

Asynchronous Callbacks

- O RESULTADO EM CÓDIGO QUE IMPLEMENTA UMA SOLUÇÃO PARA O PROBLEMA ANTERIOR É MOSTRADO AO LADO

```
...

public void onItemClick(AdapterView<?> adapterView, View view,
    int position, long l) {

    Palavra palavraClicada = palavras.get(position);
    mMediaPlayer = MediaPlayer
            .create(NumerosActivity.this, palavraClicada.getReferenciaAudio());
    mMediaPlayer.start();

    mMediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mediaPlayer) {
            Toast.makeText(NumerosActivity.this,
                "Terminei!", Toast.LENGTH_SHORT).show();
        }
    });
}

...
```

Liberando recursos de memória

- Mais uma vez o assunto é como garantir que nossa aplicação consuma o mínimo de recursos possível
- Nesse caso, quando não estivermos mais tocando o áudio, queremos liberar o recurso de memória utilizado, pois não necessitamos mais dele

Liberando recursos de memória

- Para facilitar a codificação, adicione esse trecho de código logo abaixo e fora do método onCreate em cada uma das nossas activities
- Agora libere os recursos de memória antes de tocar um áudio e, também, quando um áudio finaliza sua execução

```
/**
 * Libera o recurso de memória utilizado pelo MediaPlayer.
 * Se o media player não for igual a null, significa que
 * ele pode estar tocando um áudio, mas isso não importa,
 * pois queremos finalizá-lo antes, liberar seu recurso
 * de memória e reiniciar o nosso objeto MediaPlayer.
 */
private void releaseMediaPlayer() {
    if (mMediaPlayer != null) {
        mMediaPlayer.release();
        mMediaPlayer = null;
    }
}

...
```

Pausa

Liberando recursos de memória

- O resultado para o desafio é mostrado ao lado
- Agora, para limparmos um pouco a nossa tela, vamos fazer com que o nosso objeto MediaPalyer.OnCompletionListener seja armazenado em uma variável de classe

```
...

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView,
        View view, int position, long l) {

        Palavra palavraClicada = palavras.get(position);

        releaseMediaPlayer();

        mMediaPlayer = MediaPlayer
                .create(NumerosActivity.this,
                    palavraClicada.getReferenciaAudio());
        mMediaPlayer.start();

        mMediaPlayer.setOnCompletionListener(
            new MediaPlayer.OnCompletionListener() {

            @Override
            public void onCompletion(MediaPlayer mediaPlayer) {
                releaseMediaPlayer();
            }
        });
    }
});

...
```

Liberando recursos de memória

- Na primeira parte do código, coloco meu objeto OnCompletionListener na variável mOnCompletionListener
- E na segunda parte, simplesmente seto o método para fazer referência à variável que eu acabei de criar
- Faça essas alterações em todas as activities

```
...

private MediaPlayer.OnCompletionListener mOnCompletionListener
        = new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
        releaseMediaPlayer();
    }
};

...
...
...

mMediaPlayer.setOnCompletionListener(mOnCompletionListener);


...
```

Clico de vida das activities

- Pode ser que em um determinado momento minha aplicação esteja consumindo um recurso e o usuário deseja fechar minha aplicação enquanto esse recurso é consumido
- É uma boa prática, liberar esse recurso assim que minha aplicação saia da tela do usuário, a menos que por algum motivo de projeto você queira o contrário
- Para tratar esse cenário precisamos trabalhar com ciclos de vida da aplicação
- Você pode achar mais informações sobre o ciclo de vida de uma activity nesse link da documentação e também um exemplo prático nesse vídeo

Ciclo de vida das activities

- Baseado nas informações que você obteve acessando o link da documentação e/ou assistindo o vídeo de exemplo, responda o seguinte:
  - Em qual momento do ciclo da vida da minha aplicação eu deveria liberar o recurso de áudio caso o usuário saia da minha aplicação?

Pausa

Ciclo de vida das activities

- A resposta correta é no momento em que o callback onStop() é chamado, pois é ele que é acionado quando o usuário abandona a nossa aplicação
- Sabendo disso, implemente as alterações para que a aplicação interrompa corretamente um áudio caso o usuário saia da aplicação

Pausa

Ciclo de vida das activities

- O resultado dessa operação é mostrado ao lado
- Replique a alteração para todas as activities

```
...

@Override
protected void onCreate(Bundle savedInstanceState) {...
}

@Override
protected void onStop() {
    super.onStop();
    releaseMediaPlayer();
}

/**...
private void releaseMediaPlayer() {...
}


...
```

Audio Focus

- Para que nossa aplicação interaja melhor com outras aplicações, devemos aprender sobre Audio Focus
- Imagine que o usuário se encontra com nossa aplicação aberta e estamos reproduzindo um áudio
- De repente uma chamada começa e não queremos que nossa aplicação impeça o usuário de ouvir o toque do celular, muito menos que continue tocando enquanto ele atende a chamada
- Para impedir inconvenientes como esses temos o conceito de áudio focus

Audio Focus

- Para essa simples atividade, quero que você leia a este artigo em inglês e/ou este item da documentação e dê seu melhor para encontrar a resposta para as seguintes perguntas:
  1. Qual método devemos chamar quando queremos requisitar o áudio focus?
  2. Qual método devemos chamar para liberar o áudio focus?
  3. Qual método de callback devemos implementar para que sejamos notificados caso haja mudança no áudio focus?

Pausa

Audio Focus

1. Qual método devemos chamar quando queremos requisitar o áudio focus?
   - requestAudioFocus()

2. Qual método devemos chamar para liberar o áudio focus?
   - abandonAudioFocus()

3. Qual método de callback devemos implementar para que sejamos notificados caso haja mudança no áudio focus?
   1. onAudioFocusChange()

- Todos as requisições de Audio Focus são tratadas pela classe Audio Manager

Audio Manager

- Sabemos que para trabalharmos com áudio focus precisamos utilizar a classe áudio manager
- Também sabemos que para que tenhamos direito ao áudio focus devemos chamar o mátodo requestAudioFocus()
- Tendo essas informações em mãos faça o seguinte:
  - Acesse esse LINK para ver os parâmetros do método requestAudioFocus()
  - Responda:
    - Qual o valor que devemos passar como segundo parâmetro (streamType)?
    - Qual o valor que devemos passar como terceiro parâmetro (durationHint)?
      - Dica: ambos são constantes da classe Audio Manager mais informações sobre essas constantes AQUI

Pausa

Audio Manager

- Qual o valor que devemos passar como segundo parâmetro (streamType)?
  - AudioManager.STREAM_MUSIC

- Qual o valor que devemos passar como terceiro parâmetro (durationHint)?
  - AudioManager.AUDIOFOCUS_GAIN_TRANSIENT

Audio Focus change

- Para que possamos tratar de maneira correta os diversos estados de áudio focus vamos analisar tipos existentes e o comportamento do nosso app quando cada um deles ocorrer:
  1. AUDIOFOCUS_GAIN: recebe o focus novamente depois de ter perdido previamente => na nossa aplicação, retomamos a reprodução do áudio
  2. AUDIOFOCUS_LOSS: perda permanente do áudio focus => na nossa aplicação, paramos de executar o media player e liberamos os recursos
  3. AUDIOFOCUS_LOSS_TRANSIENT: perda de áudio focus temporária => na nossa aplicação, devemos pausar o áudio
  4. AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK: perda temporária do áudio focus, mas pode continuar a reprodução diminuindo o volume => na nossa aplicação, deveremos pausar o áudio, pois é importante que o usuário compreenda a pronúncia com clareza

Implementando o que foi aprendido

- Vamos começar implementando o código para requisitar o áudio focus

Pausa

Requisitando Audio Focus

- Para requisitar o Audio Focus, vamos começar instanciando um objeto Audio Manager

```
...

private MediaPlayer mMediaPlayer;

private AudioManager mAudioManager;

private MediaPlayer.OnCompletionListener mOnCompletionListener
        = new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
        releaseMediaPlayer();
    }
};

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

    mAudioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

    ...
}


...
```

Requisitando Audio Focus

- Agora devemos requisitar o áudio focus e devemos fazer isso logo antes de reproduzir nosso áudio
- A variável resultado vai receber se obtivemos sucesso no nosso pedido de áudio focus ou não
- Para isso, a requisição precisa enviar um mOnAudioFocusChangeListener (que implementaremos daqui a pouco) junto com o tipo de áudio e o tempo que precisamos desse áudio
- Feito isso, se tivermos conseguido o foco, procedemos com a reprodução do áudio

```
...

@Override
public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {
    Palavra palavraClicada = palavras.get(position);

    releaseMediaPlayer();

    int resultado = mAudioManager.requestAudioFocus(mOnAudioFocusChangeListener,
            AudioManager.STREAM_MUSIC,
            AudioManager.AUDIOFOCUS_GAIN_TRANSIENT);

    if (resultado == AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
        mMediaPlayer = MediaPlayer
                .create(NumerosActivity.this, palavraClicada.getReferenciaAudio());
        mMediaPlayer.start();

        mMediaPlayer.setOnCompletionListener(mOnCompletionListener);
    }
}

...
```

Implementando o OnAudioFocusChangeListener

- Agora vamos implementar o callback que fica esperando por uma alteração no áudio focus

Pausa

Criando uma instância de OnAudioFocusChangeListener

- Adicione o mOnAudioFocusChangeListener às suas variáveis de classe
- Nela adicione um objeto OnAudioFocusChangeListener e implemente o seu método obrigatório
- Nesse método, você recebe o resultado da troca de foco no parâmetro (aqui mudei o nome para focusChange para ficar mais fácil de entender)
- E dentro desse método, você vai implementar as consições particulares de comportamento de áudio da nossa aplicação

```
...

private AudioManager.OnAudioFocusChangeListener mOnAudioFocusChangeListener =
        new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT ||
                focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK) {
            //Queremos pausar o audio
        } else if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            //Queremos reproduzir o audio
        } else if (focusChange == AudioManager.AUDIOFOCUS_LOSS) {
            //queremos parar o audio
        }
    }
};

...
```

Criando uma instância de OnAudioFocusChangeListener

- Refinando nossa condição if/else temos o que é mostrado ao lado
- Onde queremos pausar, nós pausamos e retornamos ao início
- Onde queremos reproduzir chamamos o método start
- E quando perdermos completamente o áudio focus, liberamos o recurso

```
...

if (focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT ||
        focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK) {
    //Queremos pausar o audio
    mMediaPlayer.pause();
    mMediaPlayer.seekTo(0);
} else if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
    //Queremos reproduzir o audio
    mMediaPlayer.start();
} else if (focusChange == AudioManager.AUDIOFOCUS_LOSS) {
    //Queremos parar o audio
    releaseMediaPlayer();
}


...
```

Abandonando o Audio Focus

- Agora implemente o trecho de código que representa a não atual necessidade de áudio focus da nossa aplicação

Pause

Abandonando o Audio Focus

- Toda vez que damos um releaseMediaPlayer, fazemos isso porque não queremos mais utilizar esse recurso
- Então vamos usar esse local para dizer que também não queremos mais o áudio focus

```
...

private void releaseMediaPlayer() {
    if (mMediaPlayer != null) {
        mMediaPlayer.release();
        mMediaPlayer = null;

        mAudioManager.abandonAudioFocus(mOnAudioFocusChangeListener);
    }
}


...
```

Propagando alterações

- Faça essas alterações em todas as demais activities

Pausa

Adicionando ícone de play

- Para facilitar a percepção ao usuário de que cada palavra possui um áudio vinculado, vamos adicionar um ícone de play
- Para isso acesse esse LINK
- Baixe os pngs da versão em branco com resolução de 24dp
- Descompacte os arquivos e copie o conteúdo da pasta android e cole na pasta res
- Feito isso, modifique a listagem para que o ícone de play apareça corretamente

Pausa

Adicionando ícone de play

- Uma das possíveis soluções é mostrada ao lado

```
...

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/container_global">

    <ImageView
        android:id="@+id/container_imagem"
        android:layout_width="88dp"
        android:layout_height="88dp"
        android:background="@color/palida"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="88dp"
        android:orientation="horizontal">

        //componentes do próximo slide entram aqui

    </LinearLayout>
</LinearLayout>

...
```

Adicionando ícone de play

- Uma das possíveis soluções é mostrada ao lado

```
<LinearLayout
    android:id="@+id/container_palavras"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="88dp"
    android:paddingLeft="16dp"
    android:orientation="vertical">

    <TextView
        android:id="@+id/miwok"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textColor="#FFFFFF"
        android:textStyle="bold"
        android:textSize="18sp"
        android:gravity="bottom"
        tools:text="lutti"/>

    <TextView
        android:id="@+id/padrao"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textColor="#FFFFFF"
        android:textSize="18sp"
        tools:text="one"/>

</LinearLayout>

<ImageView
    android:layout_width="24dp"
    android:layout_height="24dp"
    android:layout_marginRight="16dp"
    android:layout_gravity="center_vertical"
    android:src="@drawable/ic_play_arrow_white_24dp"/>

</LinearLayout>
```

Parte 5

Fragments