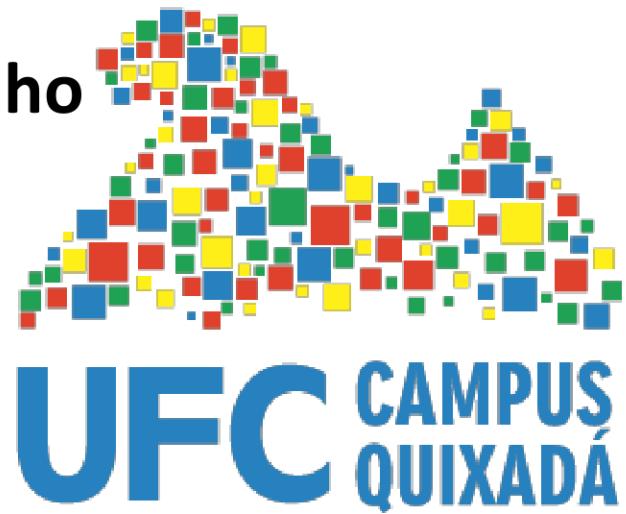


Desenvolvimento de software para Dispositivos Móveis

REVISÃO



Prof. Sidartha Carvalho



Agenda

- 📍 MarketShare Android

- 📍 Adapters
 - 📍 Simple Adapter
 - 📍 RecyclerView

- 📍 CardView

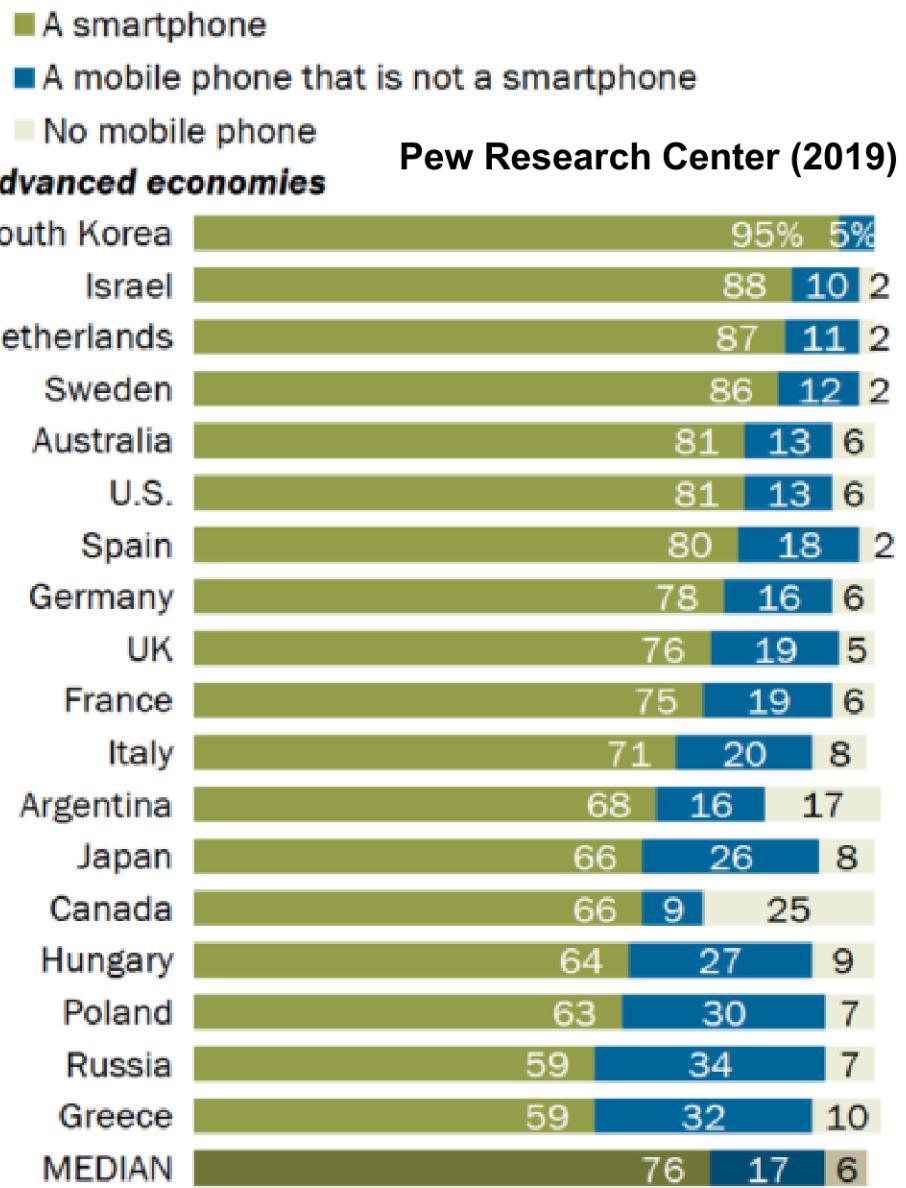
- 📍 Serviços

- 📍 Eventos de Broadcast

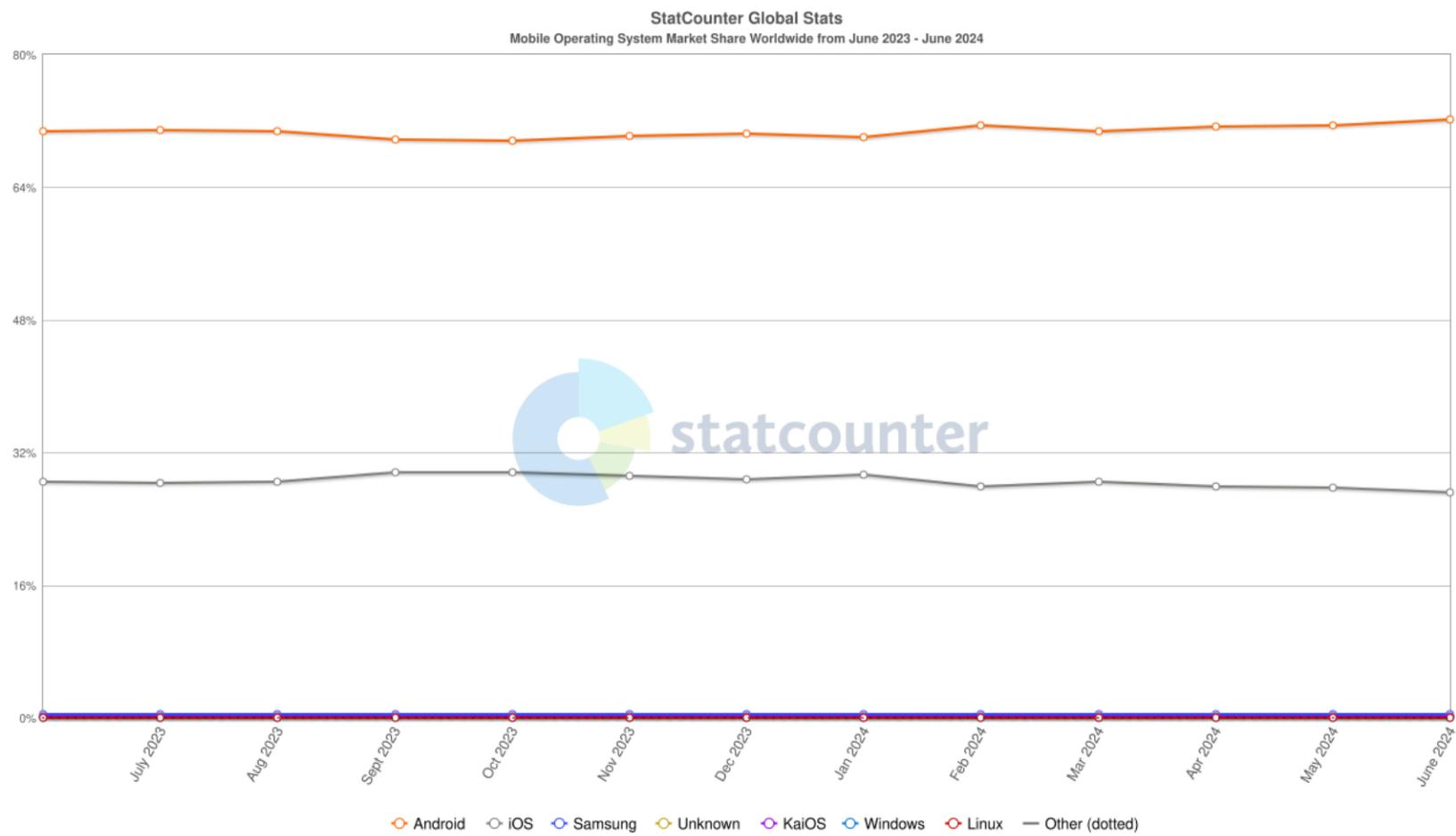
Crescimento dos smartphones

Nos países desenvolvidos,
93% da população possui
celulares, sendo **76%**
smartphones.

**5,8 bilhões de pessoas
com smartphones !**



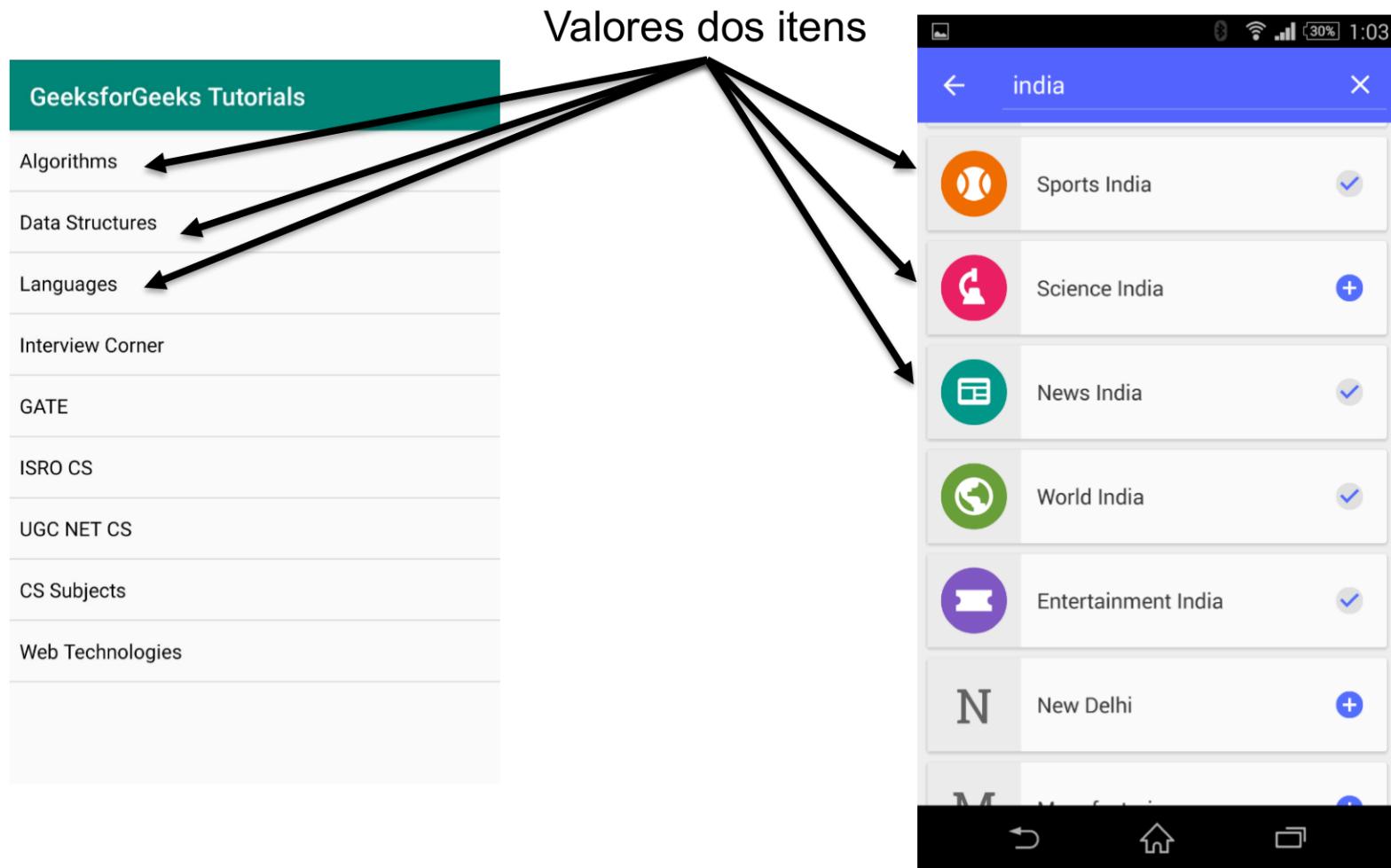
Crescimento do sistema operacional



Adapters no Android

Adapters

- Servem para popular alguns Widgets



Adapters

- Adapter
 - Ponte entre uma fonte de dados e um Adapter View.
 - Provê acesso aos dados dos itens
 - Na prática, o Adapter é necessário para fornecer uma interface conhecida de como acessar os dados de um item, permitindo que o Android crie os itens dentro de listas para o usuário
 - Fontes de dados
 - ArrayList, HashMap, SQLite, etc
 - Adapter View – Componente de UI
 - ListView, RecyclerView, GridView, etc

Adapters

- Simple Adapter
- Recycler View

Simple Adapter

- Usado para popular o widget ListView
1. Criar a fonte dos dados
 2. Criar um objeto adapter
 - `ArrayAdapter(Context, LayoutItem, fonteDados);`
 3. Linkar o widget ListView com o Adapter criado

Simple Adapter

```
// 1- AdapterView: ListView  
lv_cidades = findViewById(R.id.lv_cidades);  
  
// 2- Data Source: String Array  
String[] cidades = {"Quixada", "Fortaleza", "Quixeramobim", "Banabuiu", "Madalena"};  
  
// 3- Adapter: acts as a bridge between the  
//      'data source' and the 'AdapterView'  
ArrayAdapter<String> adapter = new ArrayAdapter<>(  
    this,  
    android.R.layout.simple_list_item_1,  
    cidades  
);  
  
// 4- Link Listview with the Adapter  
lv_cidades.setAdapter(adapter);
```

GeeksforGeeks Tutorials

Algorithms

Data Structures

Languages

Interview Corner

GATE

ISRO CS

UGC NET CS

CS Subjects

Web Technologies

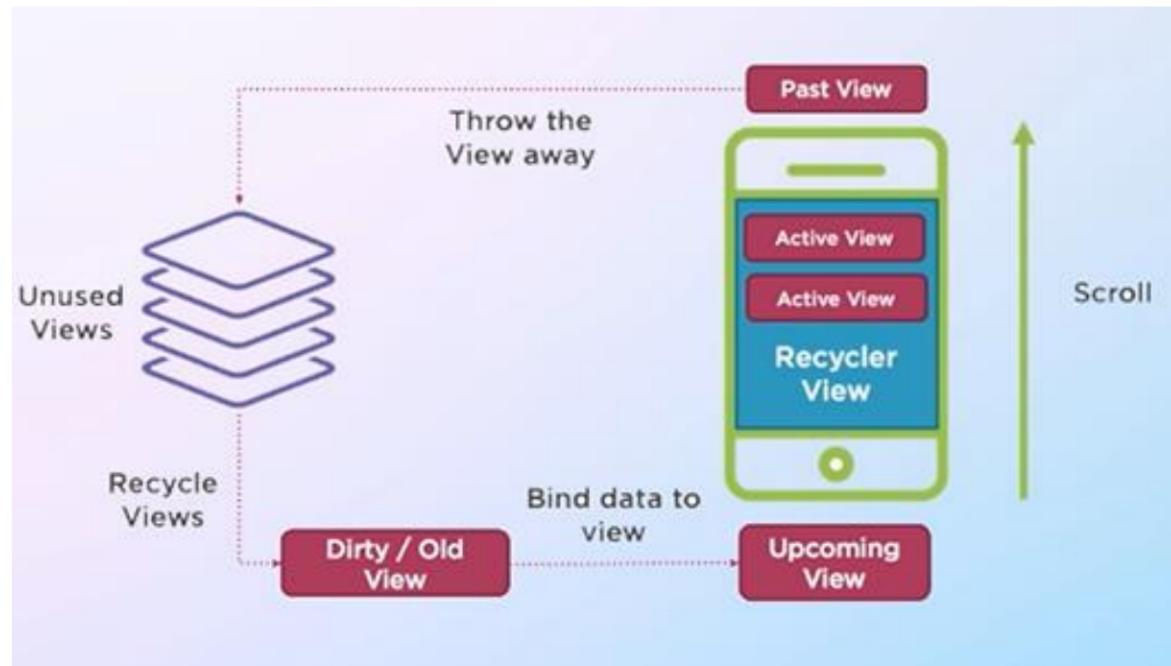


Adapters

- Simple Adapter
- Recycler View

Recycler View

- Permite economizar recursos, mantendo na memória somente os elementos que estão sendo visualizados no momento pelo usuário.
- Sempre que um item for “scrollado” para fora da tela, para cima ou para baixo, ele será removido da memória.
- Os novos itens serão renderizados e carregados na memória conforme a tela for rolando.



Recycler View

- O widget RecyclerView exige que haja um Custom Adapter.
- Vamos ver um exemplo...

Recycler View

1. MainActivity
 - Referência a UI RecyclerView
 - Definir o Layout e o Adapter
2. Criação do XML contendo o RecyclerView Widget
3. list_item.xml
 - Criação do modelo XML que define cada item da lista
4. Item.java
 - Classe Java para permitir o acesso aos atributos do Item
5. ItemArrayAdapter.java
 - extends RecyclerView.Adapter<ItemArrayAdapter.ViewHolder>
 - Implementar vários métodos obrigatórios

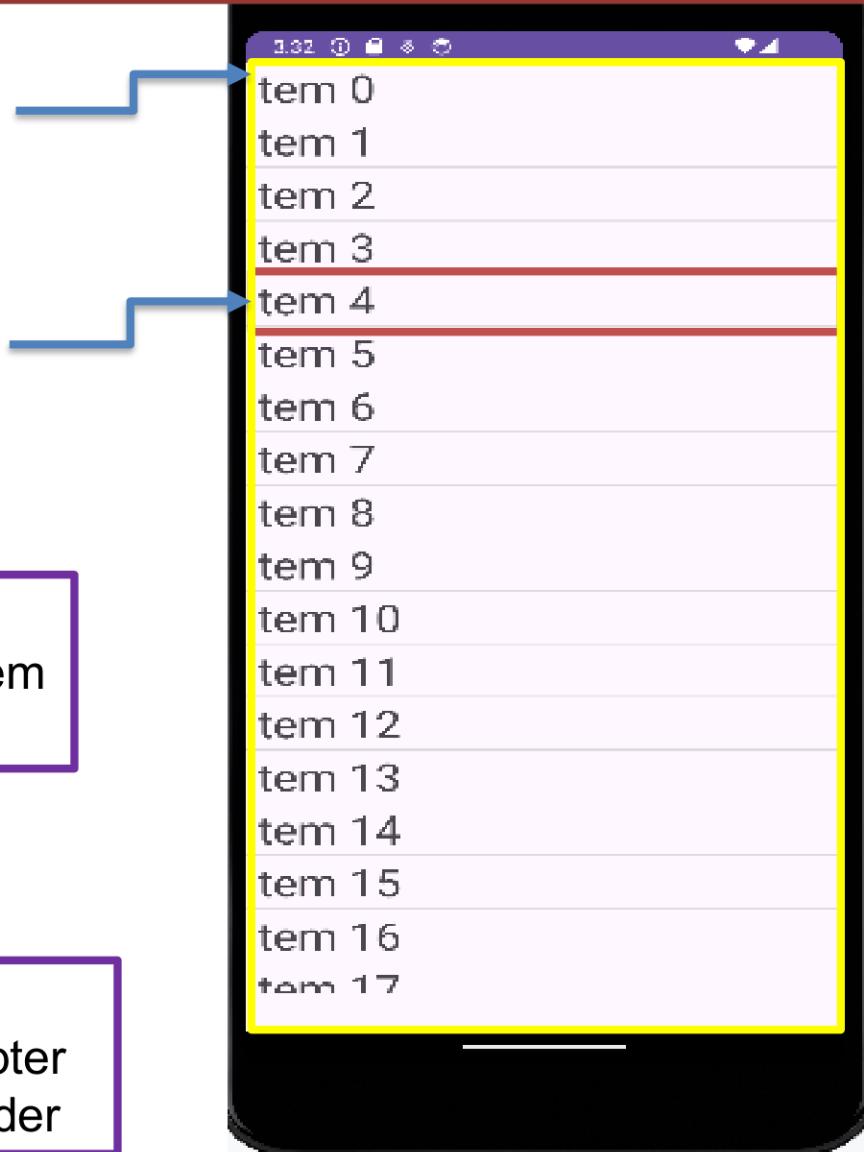
Recycler View: 4 Elementos Essenciais

XML Lista:
Possui o Widget
RecyclerView

XML Item

Item.java
Manipular os atributos do item
(strings, imagens, etc)

ItemArrayAdapter.java
- Extends RecyclerView.Adapter
- Implementa classe ViewHolder



Recycler View: Step 1 - MainActivity

```
// 1- AdapterView: ListView
rv_cidades = findViewById(R.id.rv_cidades);

// Initializing list view with the custom adapter
ArrayList<Item> itemList = new ArrayList<Item>();

ItemArrayAdapter itemArrayAdapter = new ItemArrayAdapter(R.layout.list_item, itemList);

rv_cidades = (RecyclerView) findViewById(R.id.rv_cidades);

rv_cidades.setLayoutManager(new LinearLayoutManager(this));

rv_cidades.setAdapter(itemArrayAdapter);

// Populating list items
for(int i=0; i<100; i++) {
    itemList.add(new Item("Item " + i));
}
```

Recycler View: Step 2 - activity_recycler_view_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".RecyclerViewExample">

    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="409dp"
        android:layout_height="729dp"
        android:id="@+id/rv_cidades"
        tools:layout_editor_absoluteX="1dp"
        tools:layout_editor_absoluteY="1dp"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9

Recycler View: Step 3 - list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/row_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30dp"
        android:gravity="center"/>
</LinearLayout>
```

Recycler View: Step 4 - Item.java

```
public class Item {  
    private String name;  
  
    public Item(String n) {  
        name = n;  
    }  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

Recycler View: Step 5 - ItemArrayAdapter.java

```
public class ItemArrayAdapter extends RecyclerView.Adapter<ItemArrayAdapter.ViewHolder> {

    private int listItemLayout;
    private ArrayList<Item> itemList;

    // Constructor of the class
    public ItemArrayAdapter(int layoutId, ArrayList<Item> itemList) {
        listItemLayout = layoutId;
        this.itemList = itemList;
    }

    // get the size of the list
    @Override
    public int getItemCount() {
        return itemList == null ? 0 : itemList.size();
    }

    // specify the row layout file and click for each row
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(listItemLayout, parent, false);
        ViewHolder myViewHolder = new ViewHolder(view);
        return myViewHolder;
    }
}
```

CONTINUA NO PROXIMO SLIDE...

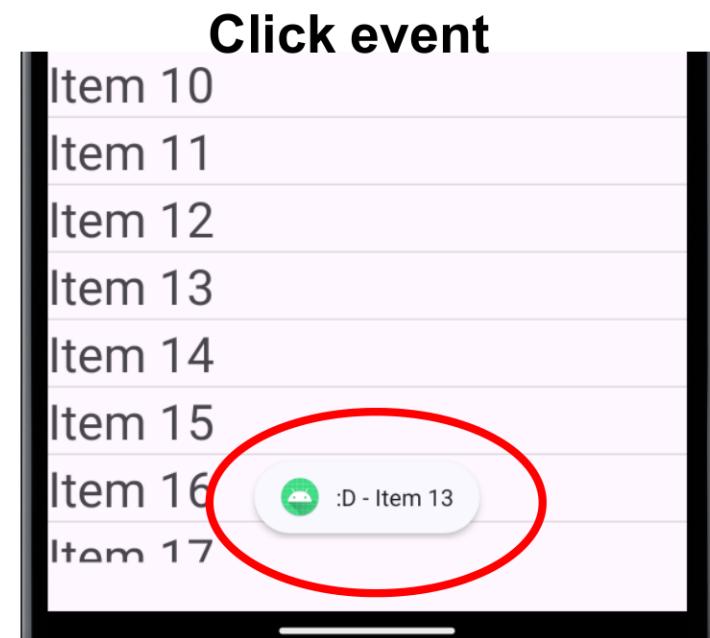
Recycler View: Step 5 - ItemArrayAdapter.java

```
// load data in each row element
@Override
public void onBindViewHolder(final ViewHolder holder, final int listPosition) {
    TextView item = holder.item;
    item.setText(itemList.get(listPosition).getName());
}

// Static inner class to initialize the views of rows
static class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    public TextView item;
    public ViewHolder(View itemView) {
        super(itemView);
        itemView.setOnClickListener(this);
        item = (TextView) itemView.findViewById(R.id.row_item);
    }

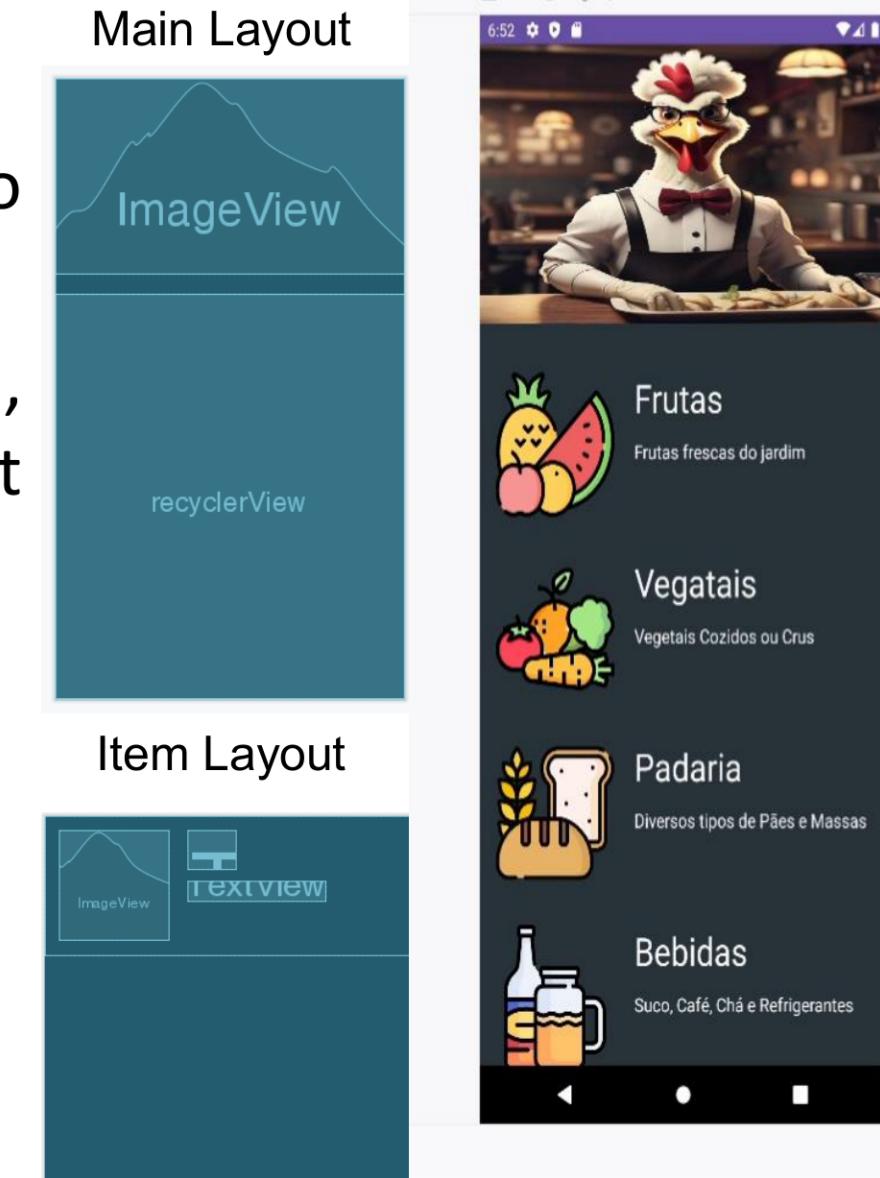
    @Override
    public void onClick(View view) {
        Toast.makeText(view.getContext(), ":D - " + item.getText(), Toast.LENGTH_SHORT).show();
        Log.d("onclick", "onClick " + getLayoutPosition() + " " + item.getText());
    }
}
```

Recycler View: Resultado



Exemplo RecyclerView

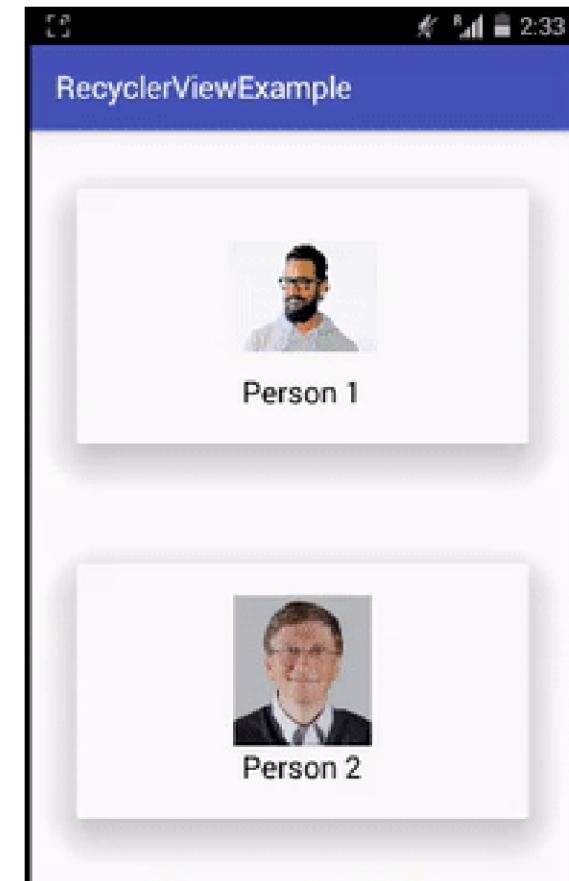
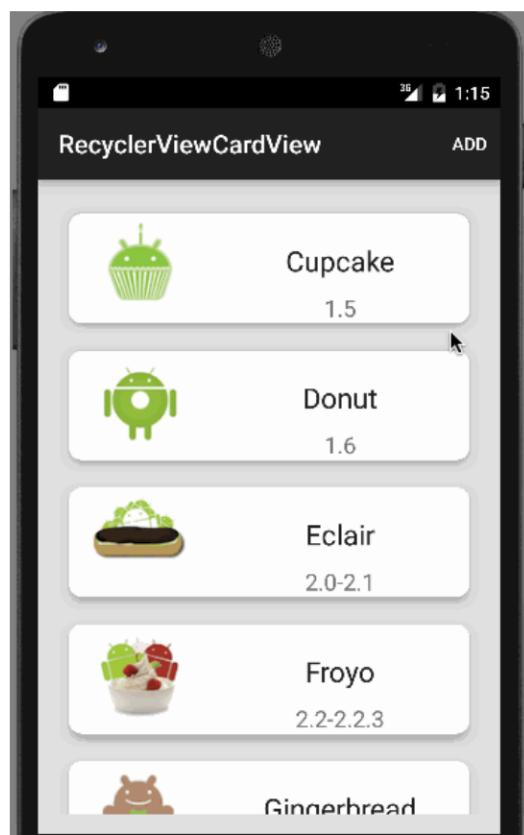
- MarketApp
 - RecyclerView rolável com o menu do restaurante
 - Ao clicar em cada item, mostrar um Toast indicando onde foi clicado



CardView no Android

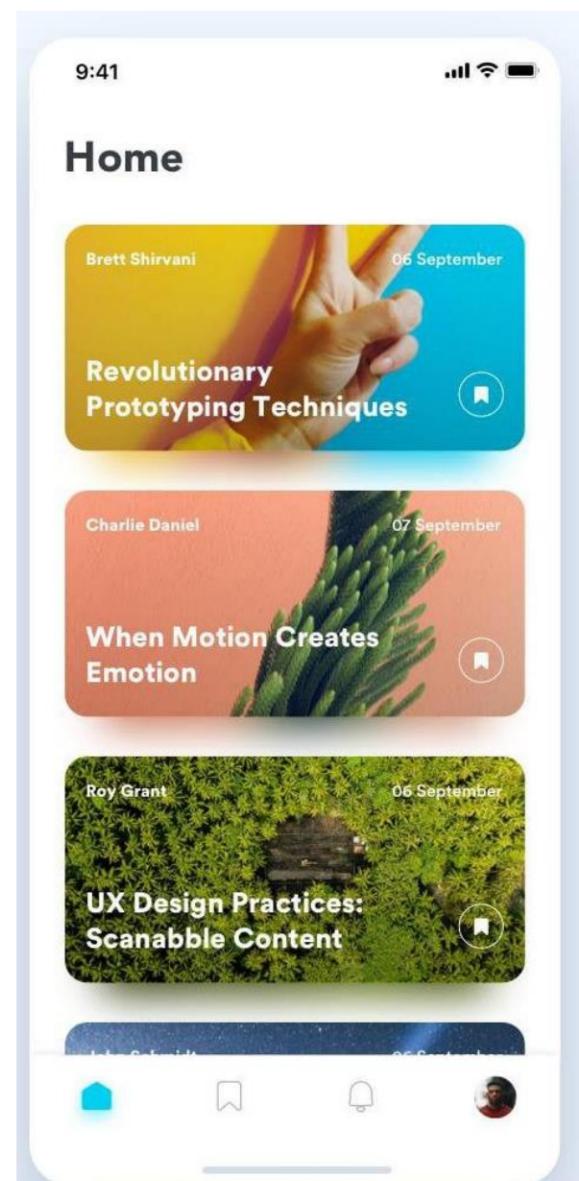
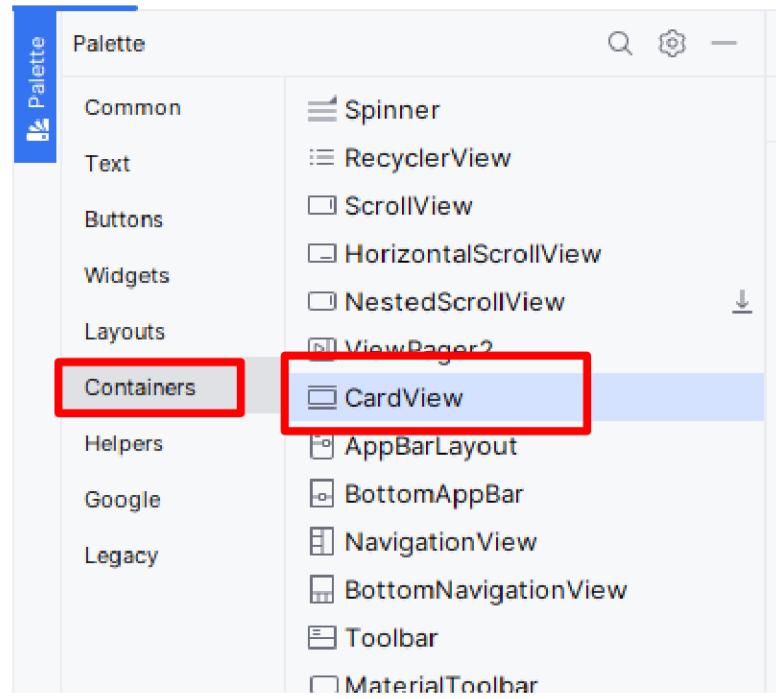
CardView: Widget Android

- Permite separar itens por “cards”, tornando-os mais bonitos e com mais opções de customização.



CardView: Widget Android

- O CardView pode ser usado dentro de um RecyclerView, como um item de layout.



Proposta de App: Quixadá Night Events



Passos para a construção do App

1. Criar o layout do item usando CardView
2. Criar o layout da ActivityMain usando RecyclerView
3. Criar uma classe de modelo para armazenar os dados de cada CardView (Titulo, Descrição e Imagem de fundo)
4. Criar o Adapter e o ViewHolder
5. Criar as notícias e associar ao Adapter na MainActivity

Passo 1

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardCornerRadius="50dp"
    app:cardElevation="100dp"
    android:layout_margin="10dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/card_layout"
        android:background="@drawable/backgroundgradient1">

        <TextView
            android:id="@+id/tv_card_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="28dp"
            android:layout_marginBottom="47dp"
            android:text="Titulo do evento"
            android:textColor="@color/black"
            android:textSize="40dp"
            android:textAlignment="center"
            android:textStyle="bold"
            app:layout_constraintBottom_toTopOf="@+id/tv_card_subject"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/tv_card_subject"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_marginStart="34dp"
            android:layout_marginEnd="34dp"
            android:layout_marginBottom="46dp"
            android:text="Descrição do meu evento."
            android:textSize="20dp"
            android:textAlignment="center"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/tv_card_title" />
    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```



Passo 2



Passo 2

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/iv_logo"
        android:layout_width="409dp"
        android:layout_height="185dp"
        android:layout_marginTop="4dp"
        android:layout_marginBottom="1dp"
        android:scaleType="fitXY"
        app:layout_constraintBottom_toTopOf="@+id/rv_news"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_news"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/iv_logo" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9



rv_news

Passo 3

```
package br.ufc.quixada.cardviewwithrecyclerview;

public class NewsModel {

    String title;
    String description;
    int backgroundImage;

    NewsModel(String title, String description, int backgroundImage) {
        this.title = title;
        this.description = description;
        this.backgroundImage = backgroundImage;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public int getBackgroundImage() {
        return backgroundImage;
    }

    public void setBackgroundImage(int backgroundImage) {
        this.backgroundImage = backgroundImage;
    }
}
```

Criar uma classe de modelo para armazenar os dados de cada CardView (Titulo, Descrição e Imagem de fundo)

Passo 4

```
public class NewsAdapter extends RecyclerView.Adapter<NewsAdapter.NewsViewHolder>{  
  
    private List<NewsModel> newsList;  
  
    public NewsAdapter(List<NewsModel> newsList){  
        this.newsList = newsList;  
    }  
  
    @NonNull  
    @Override  
    public NewsViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
        View itemView = LayoutInflater.from(parent.getContext()).inflate(R.layout.card_item_layout, parent, false);  
        return new NewsViewHolder(itemView);  
    }  
  
    @Override  
    public void onBindViewHolder(@NonNull NewsViewHolder holder, int position) {  
        NewsModel news = newsList.get(position);  
        holder.tv_title.setText(news.getTitle());  
        holder.tv_subject.setText(news.getDescription());  
        holder.card_layout.setBackgroundResource(news.getBackgroundImage());  
    }  
  
    @Override  
    public int getItemCount() {  
        return newsList.size();  
    }  
}
```

Adapter

Passo 4

```
public static class NewsViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
  
    TextView tv_title, tv_subject;  
    ConstraintLayout card_layout;  
  
    public NewsViewHolder(@NonNull View itemView) {  
        super(itemView);  
  
        tv_title = itemView.findViewById(R.id.tv_card_title);  
        tv_subject = itemView.findViewById(R.id.tv_card_subject);  
        card_layout = itemView.findViewById(R.id.card_layout);  
  
        tv_title.setOnClickListener(this);  
        tv_subject.setOnClickListener(this);  
        card_layout.setOnClickListener(this);  
    }  
  
    @Override  
    public void onClick(View v) {  
        Toast.makeText(v.getContext(), "Você clicou no item: " + tv_title.getText(), Toast.LENGTH_SHORT).show();  
    }  
}
```

ViewHolder (no mesmo arquivo do Adapter)

Passo 5

```
recyclerView = findViewById(R.id.rv_news);
newsList = new ArrayList<>();

NewsModel news1 = new NewsModel("Matulão",
    "O restaurante estilo barzinho Matulão está com a cantora Stefani esta noite! Venha conferir o melhor do forró romântico.",
    R.drawable.backgroundgradient1);

NewsModel news2 = new NewsModel("Broiler",
    "Restaurante estilo barzinho. Acontece todos os finais de semana.",
    R.drawable.backgroundgradient2);

NewsModel news3 = new NewsModel("Cangaço",
    "Bar com vários paredões de som, acontece todas as sextas e sábados.",
    R.drawable.backgroundgradient3);

NewsModel news4 = new NewsModel("Portal da Lagoa",
    "Bar e restaurante com música brasileira e karaoke, neste final de semana teremos Karaoke.",
    R.drawable.backgroundgradient1);

NewsModel news5 = new NewsModel("Abrigo bar e restaurante",
    "Tem que chegar cedo para pegar uma mesa.",
    R.drawable.backgroundgradient2);

newsList.add(news1);
newsList.add(news2);
newsList.add(news3);
newsList.add(news4);
newsList.add(news5);

newsAdapter = new NewsAdapter(newsList);

recyclerView.setLayoutManager(new LinearLayoutManager(this));
recyclerView.setAdapter(newsAdapter);
```

MainActivity.java

Quixadá Night Events



Services no Android

Services

- Service é um processamento Android que não exige uma visualização para o usuário (diferente das Activities).
- Não exige um layout.xml.
- Tipos de Service
 - **Foreground**
 - Operação que é perceptível ao usuário.
 - Exige a exibição de uma notificação do App na barra de notificações.
 - Exemplo: tocar uma música
 - **Background**
 - Operação que não é facilmente perceptível ao usuário.
 - Exemplo: enviar os dados do aplicativo para a nuvem ou compactar os dados.
 - **Bounded**
 - Serviço que está vinculado a um aplicativo e permite a comunicação entre o serviço e o app que o chamou.
 - Deve executar somente enquanto a aplicação que o criou estiver sendo executada.

Criar um Serviço: Passos

1. Criar a classe que irá implementar o serviço, deve “extends Service” e implementar os métodos onStartCommand() e onDestroy().
2. Criar a Intent para iniciar o serviço e chamar o método StartService() passando a Intent.
3. Adicionar o Service no AndroidManifest.

Criar um Serviço: Passo 1

```
public class MusicInBackgroundService extends Service {  
  
    MediaPlayer mediaPlayer;  
  
    @Nullable  
    @Override  
    public IBinder onBind(Intent intent) {  
        return null;  
    }  
  
    @Override  
    public int onStartCommand(Intent intent, int flags, int startId) {  
  
        mediaPlayer = MediaPlayer.create(this, Settings.System.DEFAULT_RINGTONE_URI);  
        mediaPlayer.setLooping(true);  
        mediaPlayer.start();  
  
        return START_STICKY;  
    }  
  
    @Override  
    public void onDestroy() {  
        super.onDestroy();  
  
        mediaPlayer.stop();  
    }  
}
```

Criar um Serviço: Passo 2

```
Intent startMusicIntent = new Intent(v.getContext(), MusicInBackgroundService.class);  
startService(startMusicIntent);
```

```
Intent stopMusicIntent = new Intent(v.getContext(), MusicInBackgroundService.class);  
stopService(stopMusicIntent);
```

Criar um Serviço: Passo 2

```
public void onClick(View v) {  
    if(v == btn_start){  
        Log.d("sidartha", "start music button");  
        Intent startMusicIntent = new Intent(v.getContext(), MusicInBackgroundService.class);  
        startService(startMusicIntent);  
    }else if(v == btn_stop){  
        Log.d("sidartha", "stop music button");  
        Intent stopMusicIntent = new Intent(v.getContext(), MusicInBackgroundService.class);  
        stopService(stopMusicIntent);  
    }  
}
```

Criar um Serviço: Passo 3

```
</application>
```

```
// DECLARAÇÃO DAS ACTIVITIES
```

```
<service android:name=".MusicInBackgroundService"></service>
```

```
</application>
```

Criar um Serviço: Passo 3

- O que criamos até o momento foi um Background Service.
- O bounded é o mesmo, mas com uma comunicação/troca de informações entre a aplicação que chamou e o serviço.
- startService() vs startForegroundService()
 - startForegroundService() deve ser usado para iniciar um serviço em primeiro plano que precisa continuar em execução mesmo em condições de poucos recursos. Isso é necessário para evitar a limitação de serviços em segundo plano em dispositivos com Android Oreo e versões posteriores.
 - Quando se utiliza startForegroundService(), é necessário chamar o método startForeground() dentro do serviço para exibir uma notificação na barra de status. Essa notificação informa ao usuário que o serviço está em execução e permite que ele interaja com o serviço conforme necessário.

Resultado

- Após clicar em Play Music, mesmo se você fechar a aplicação a música continuará a tocar.
- Isso não acontecia no nosso AnimalsApp, quando a aplicação perdia a atenção do usuário, a música era interrompida.



Eventos de Broadcast

Broadcast

- Os apps Android podem enviar ou receber mensagens de Broadcast do sistema Android e de outros aplicativo, similar ao padrão de design Publish–Subscribe.
- Essas transmissões são enviadas quando ocorrem alguns eventos interessantes.
- Exemplos: o dispositivo começa a carregar, o modo avião é ligado ou desligado, o Wifi é desconectado, etc.
- Os apps também podem enviar transmissões personalizadas, por exemplo, para notificar outros apps sobre algo que possam ser do interesse deles.
- Quando uma transmissão é enviada, o sistema a roteia automaticamente para apps que se inscreveram para receber esse tipo específico de transmissão.

Broadcast: recebendo eventos do modo avião

- Passos
 1. Criar uma classe Java que “extends BroadcastReceiver” e implementar o método `onReceive()` necessário.
 2. Escolher qual evento deseja receber dentro do `onReceive()`.
 3. Adicionar a permissão no `AndroidManifest` manualmente ou de forma dinâmica, usando código Java e `IntentFilter` (recomendado).
- Para cada evento há um nome associado
 - Mais de 200 eventos nativos do Android

<https://developer.android.com/about/versions/11/reference/broadcast-intents-30?hl=pt-br>

Broadcast: recebendo eventos do modo avião

```
public class AirplaneModeReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        //Este método é chamado toda vez que um evento de Broadcast  
é lançado  
        if (intent.getAction() != null &&  
            intent.getAction().equals(Intent.ACTION_AIRPLANE_MODE_CH  
ANGED))  
        {  
            boolean isAirplaneModeOn =  
intent.getBooleanExtra("state", false);  
  
            String msg = isAirplaneModeOn ? "Airplane Mode is ON" :  
"Airplane mode is OFF";  
  
            Toast.makeText(context,  
                ""+msg,  
                Toast.LENGTH_LONG).show();  
        }  
    }  
}
```

Broadcast: recebendo eventos do modo avião

```
//MainActivity
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Registrar de forma dinâmica a permissão necessária
    IntentFilter intentFilter = new
IntentFilter("android.intent.action.AIRPLANE_MODE");
    AirplaneModeReceiver br = new AirplaneModeReceiver();
    registerReceiver(br, intentFilter);
}
```

Referências

- <https://developer.android.com/>
- <https://developer.android.com/courses/fundamentals-training/>

Dúvidas?



www.shutterstock.com · 744867163