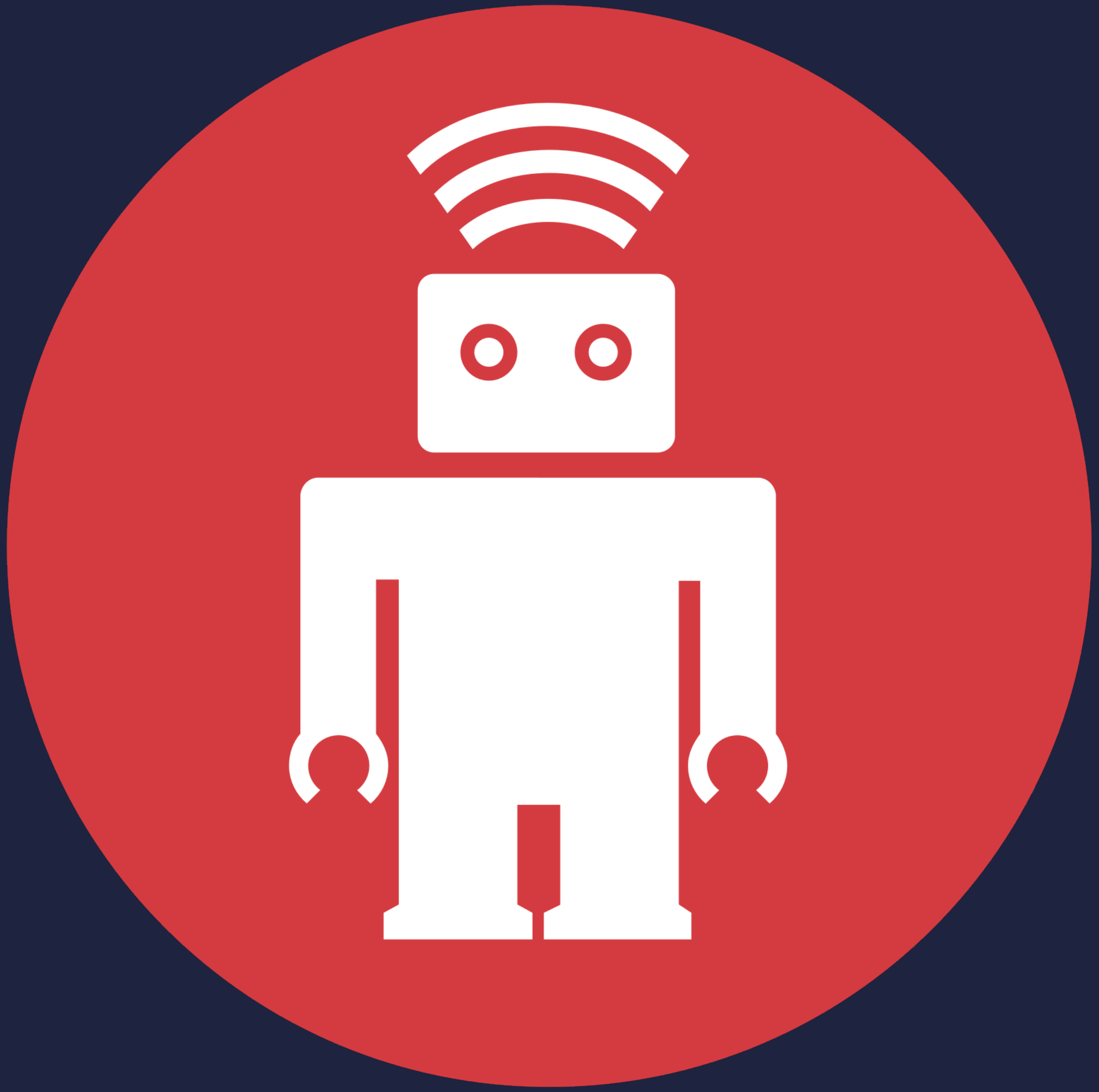
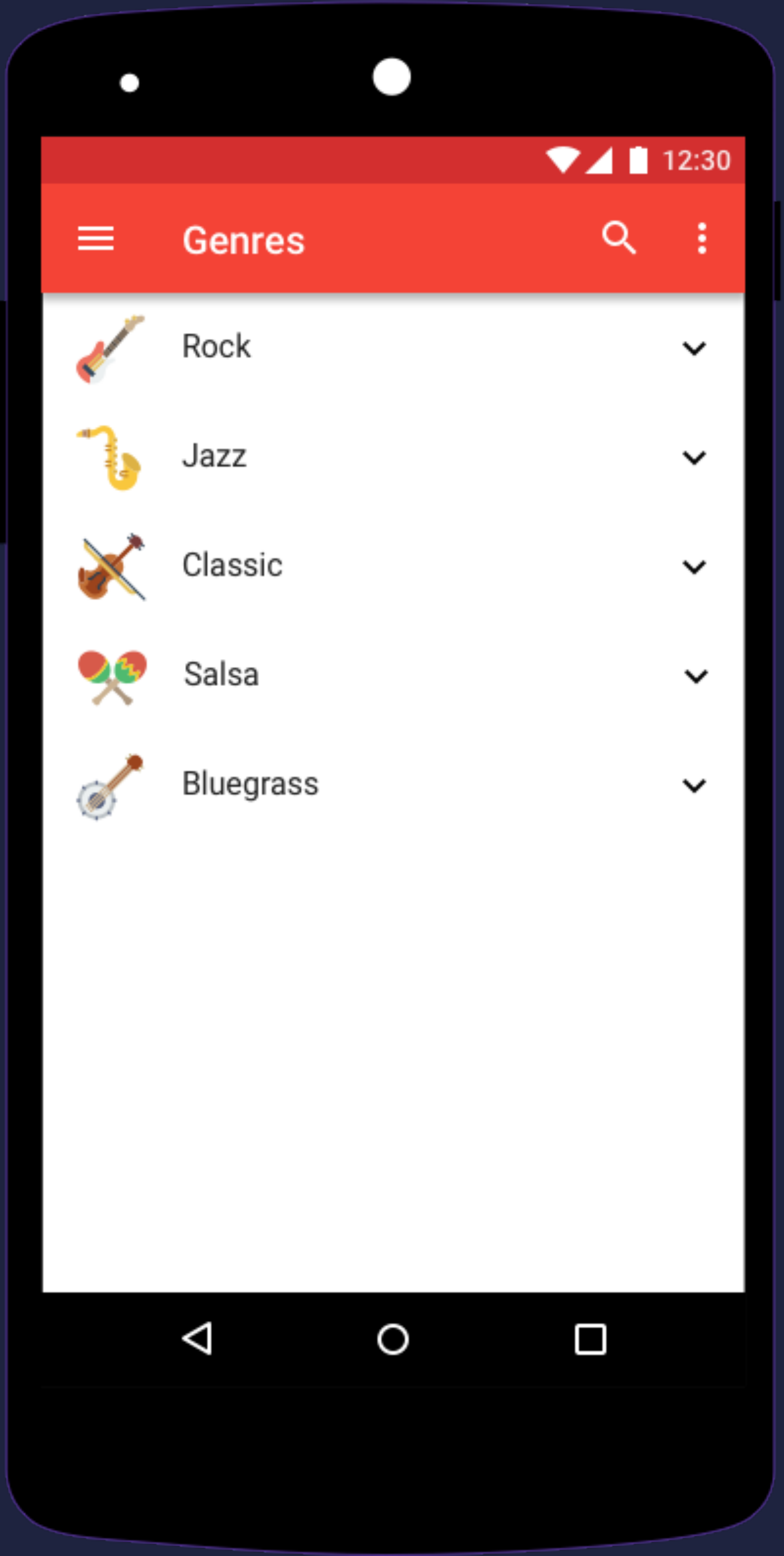
EXPANDABLE RECYCLERVIEW AND YOU

BY: AMANDA HILL

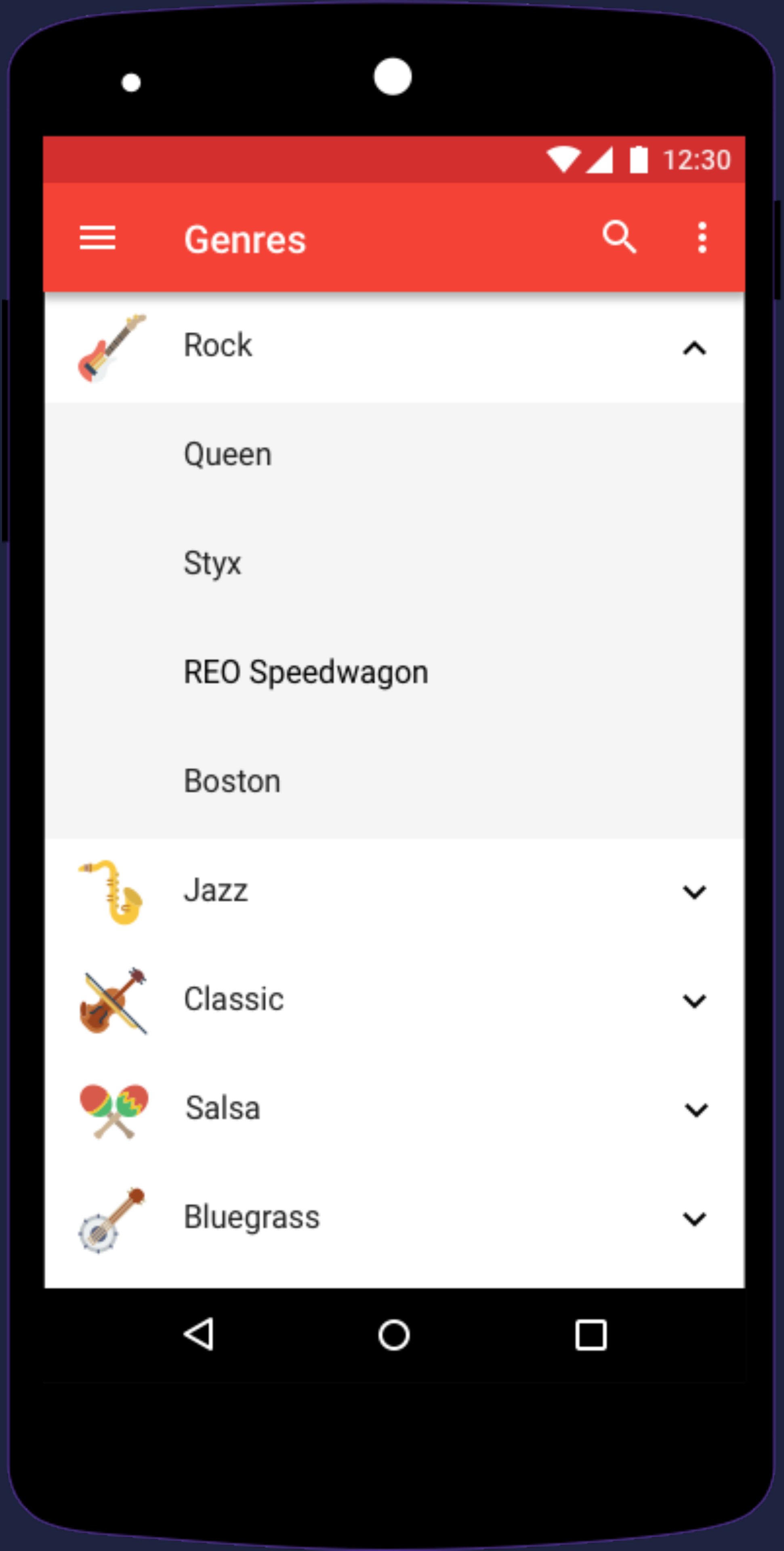


THOUGHTBOT

ExpandableRecyclerView is an open source library for expanding and collapsing groups using `RecyclerView.Adapter`



COLLAPSED



12:30

Genres

Rock

Queen

Styx

REO Speedwagon

Boston

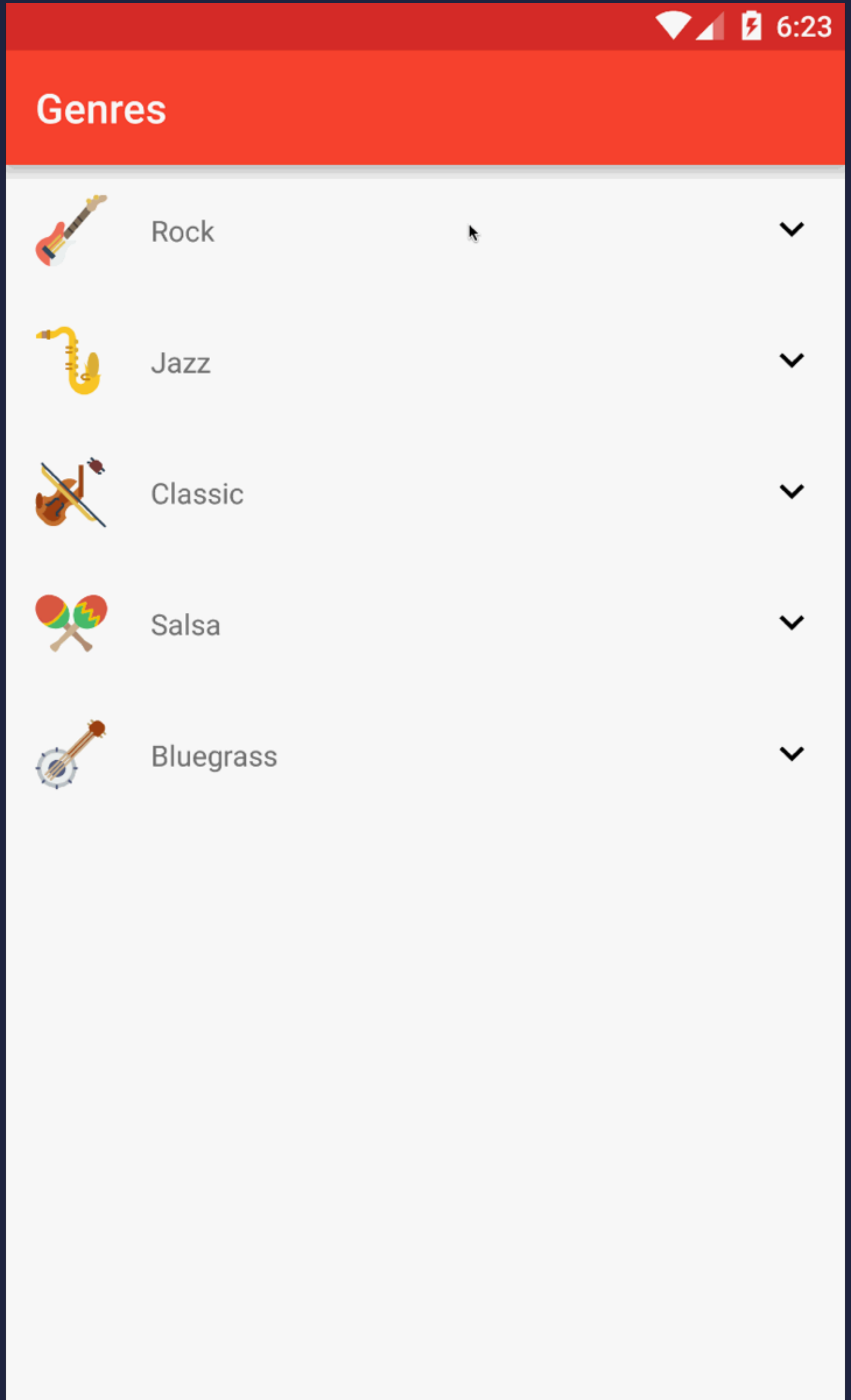
Jazz

Classic

Salsa

Bluegrass

EXPANDED


6:23
Genres
Rock
Jazz
Classic
Salsa
Bluegrass

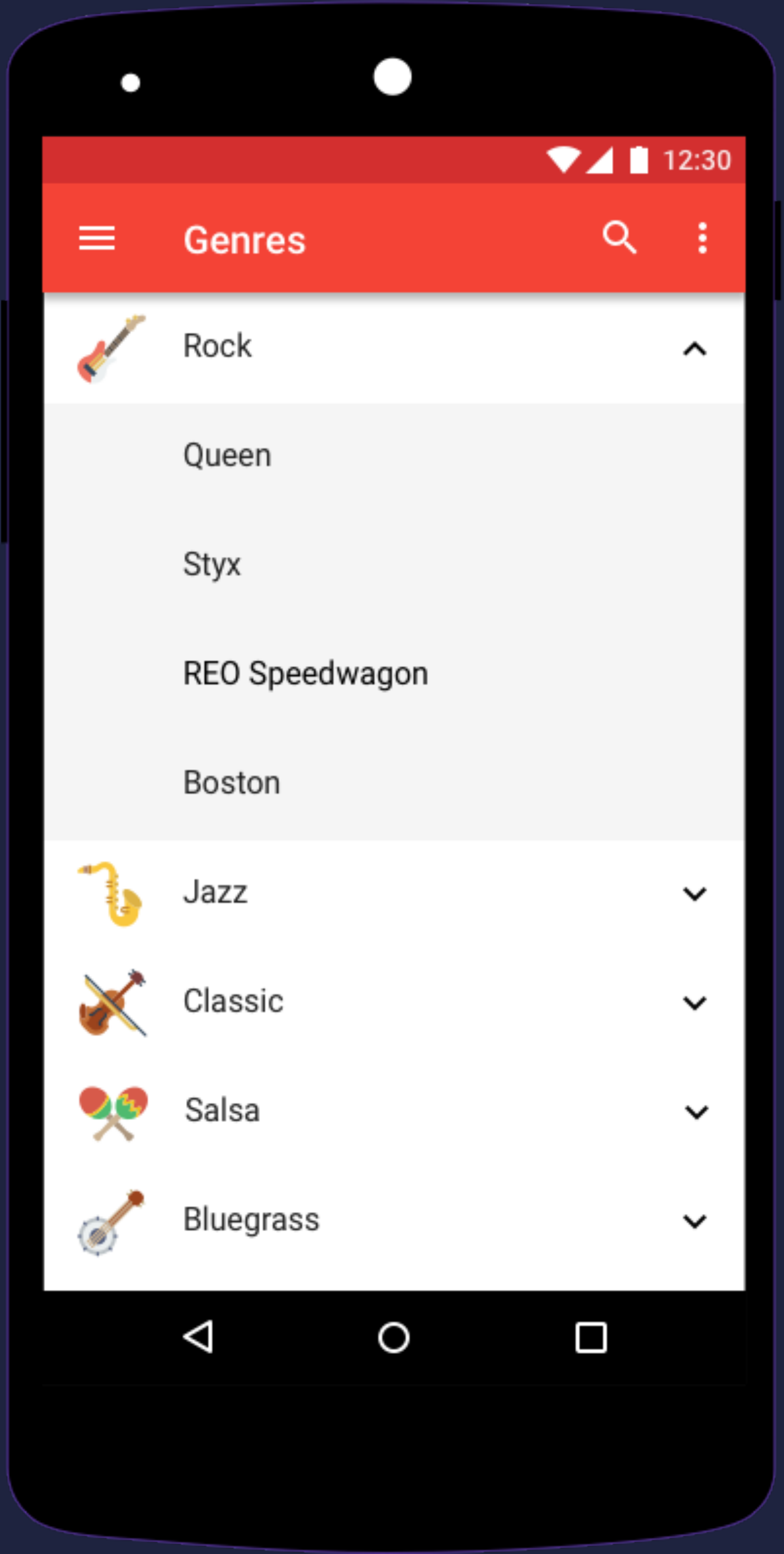
THE WHY

Because the Android SDK doesn’t provide an out of the box solution and my client *really really* wanted it

BUT AREN’T THERE ALREADY A ZILLION OF LIBRARIES OUT THERE?

YUP.

THE HOW (DO I USE IT)


12:30
Genres
Rock
Queen
Styx
REO Speedwagon
Boston
Jazz
Classic
Salsa
Bluegrass

1. MODELING

THE CHILD 👶

```
public class Artist {

  private String title;

  public Artist(String title) {
    this.title = title;
  }
}
```

THE GROUP 👪

```
public class ExpandableGroup<T> {

  private String title;
  private List<T> items;

  public ExpandableGroup(String title, List<T> items) {
    this.title = title;
    this.items = items;
  }
}
```

THE GROUP 👪

```
public class Genre extends ExpandableGroup<Artist> {

  public Genre(String title, List<Artist> items) {
    super(title, items);
  }
}
```

2. THE VIEW (HOLDERS)

THE CHILD 👶

```
public class ArtistViewHolder extends RecyclerView.ViewHolder {


  private TextView childTextView;

  public ArtistViewHolder(View itemView) {
    super(itemView);
    childTextView = (TextView) itemView.findViewById(R.id.list_item_artist_name);
  }
}
```

THE CHILD 👶

```
- public class ArtistViewHolder extends RecyclerView.ViewHolder {
+ public class ArtistViewHolder extends ChildViewHolder {

  private TextView childTextView;

  public ArtistViewHolder(View itemView) {
    super(itemView);
    childTextView = (TextView) itemView.findViewById(R.id.list_item_artist_name);
  }
}
```

THE GROUP 👪

```
public class GenreViewHolder extends RecyclerView.ViewHolder {


  private TextView bandName;
  private ImageView arrow;
  private ImageView icon;

  public GenreViewHolder(View itemView) {
    super(itemView);
    bandName = (TextView) itemView.findViewById(R.id.list_item_genre_name);
    arrow = (ImageView) itemView.findViewById(R.id.list_item_genre_arrow);
    icon = (ImageView) itemView.findViewById(R.id.list_item_genre_icon);
  }
```

THE GROUP 👪

```
- public class GenreViewHolder extends RecyclerView.ViewHolder {
+ public class GenreViewHolder extends GroupViewHolder {

  private TextView bandName;
  private ImageView arrow;
  private ImageView icon;

  public GenreViewHolder(View itemView) {
    super(itemView);
    bandName = (TextView) itemView.findViewById(R.id.list_item_genre_name);
    arrow = (ImageView) itemView.findViewById(R.id.list_item_genre_arrow);
    icon = (ImageView) itemView.findViewById(R.id.list_item_genre_icon);
  }
```

3. THE ADAPTER

`ExpandableRecyclerViewAdapter`

```
ExpandableRecyclerViewAdapter extends RecyclerView.Adapter
```

```
ExpandableRecyclerViewAdapter<GVH extends GroupViewHolder, CVH extends ChildViewHolder> extends RecyclerView.Adapter
```

THE CONSTRUCTOR

```java
public ExpandableRecyclerViewAdapter(List<? extends ExpandableGroup> groups) {
  super(groups);
}
```

FOUR ABSTRACT METHODS

FOUR METHODS:

```
public GVH onCreateGroupViewHolder(ViewGroup parent, int viewType);
```

FOUR METHODS:

```
public GVH onCreateGroupViewHolder(ViewGroup parent, int viewType);

public CVH onCreateChildViewHolder(ViewGroup parent, int viewType);
```

FOUR METHODS:

```
public GVH onCreateGroupViewHolder(ViewGroup parent, int viewType);

public CVH onCreateChildViewHolder(ViewGroup parent, int viewType);

public void onBindGroupViewHolder(GVH holder, int flatPosition, ExpandableGroup group);
```

FOUR METHODS:

```java
public GVH onCreateGroupViewHolder(ViewGroup parent, int viewType);

public CVH onCreateChildViewHolder(ViewGroup parent, int viewType);

public void onBindGroupViewHolder(GVH holder, int flatPosition, ExpandableGroup group);

public void onBindChildViewHolder(CVH holder, int flatPosition, ExpandableGroup group, int childIndex);
```

GENRE ADAPTER

GENRE ADAPTER

```
public class GenreAdapter extends ExpandableRecyclerViewAdapter<GenreViewHolder, ArtistViewHolder> {

  public GenreAdapter(List<Genre> groups) {
    super(groups);
  }

  {...}
  {...}
  {...}
  {...}

}
```

GENRE ADAPTER

```
public class GenreAdapter extends ExpandableRecyclerViewAdapter<GenreViewHolder, ArtistViewHolder> {

  {...}

  @Override
  public GenreViewHolder onCreateGroupViewHolder(ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.list_item_genre, parent, false);
    return new GenreViewHolder(view);
  }

  {...}
  {...}
  {...}

}
```

GENRE ADAPTER

```
public class GenreAdapter extends ExpandableRecyclerViewAdapter<GenreViewHolder, ArtistViewHolder> {

  {...}
  {...}

  @Override
  public ArtistViewHolder onCreateChildViewHolder(ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.list_item_artist, parent, false);
    return new ArtistViewHolder(view);
  }

  {...}
  {...}

}
```

GENRE ADAPTER

```
public class GenreAdapter extends ExpandableRecyclerViewAdapter<GenreViewHolder, ArtistViewHolder> {

  {...}
  {...}
  {...}

  @Override
  public void onBindChildViewHolder(ArtistViewHolder holder, int flatPosition,
      ExpandableGroup group, int childIndex) {

    final Artist artist = ((Genre) group).getItems().get(childIndex);
    holder.bindArtist(artist);
  }

  {...}

}
```

GENRE ADAPTER

```
public class GenreAdapter extends ExpandableRecyclerViewAdapter<GenreViewHolder, ArtistViewHolder> {

  {...}
  {...}
  {...}
  {...}

  @Override
  public void onBindGroupViewHolder(GenreViewHolder holder, int flatPosition,
      ExpandableGroup group) {

    holder.bindGenre((Genre) group);
  }
}
```

GENRE ADAPTER

```java
public class GenreAdapter extends ExpandableRecyclerViewAdapter<GenreViewHolder, ArtistViewHolder> {

  public GenreAdapter(ListGenre> groups) {
    super(groups);
  }

  @Override
  public GenreViewHolder onCreateGroupViewHolder(ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.list_item_genre, parent, false);
    return new GenreViewHolder(view);
  }

  @Override
  public ArtistViewHolder onCreateChildViewHolder(ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.list_item_artist, parent, false);
    return new ArtistViewHolder(view);
  }

  @Override
  public void onBindChildViewHolder(ArtistViewHolder holder, int flatPosition,
      ExpandableGroup group, int childIndex) {

    final Artist artist = ((Genre) group).getItems().get(childIndex);
    holder.bindArtist(artist);
  }

  @Override
  public void onBindGroupViewHolder(GenreViewHolder holder, int flatPosition,
      ExpandableGroup group) {

    holder.bindGenre((Genre) group);
  }
}
```

4. THE ACTIVITY

```
public class GenreActivity extends AppCompatActivity {



  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_genre);
    getSupportActionBar().setTitle("Genres");







  }
}
```

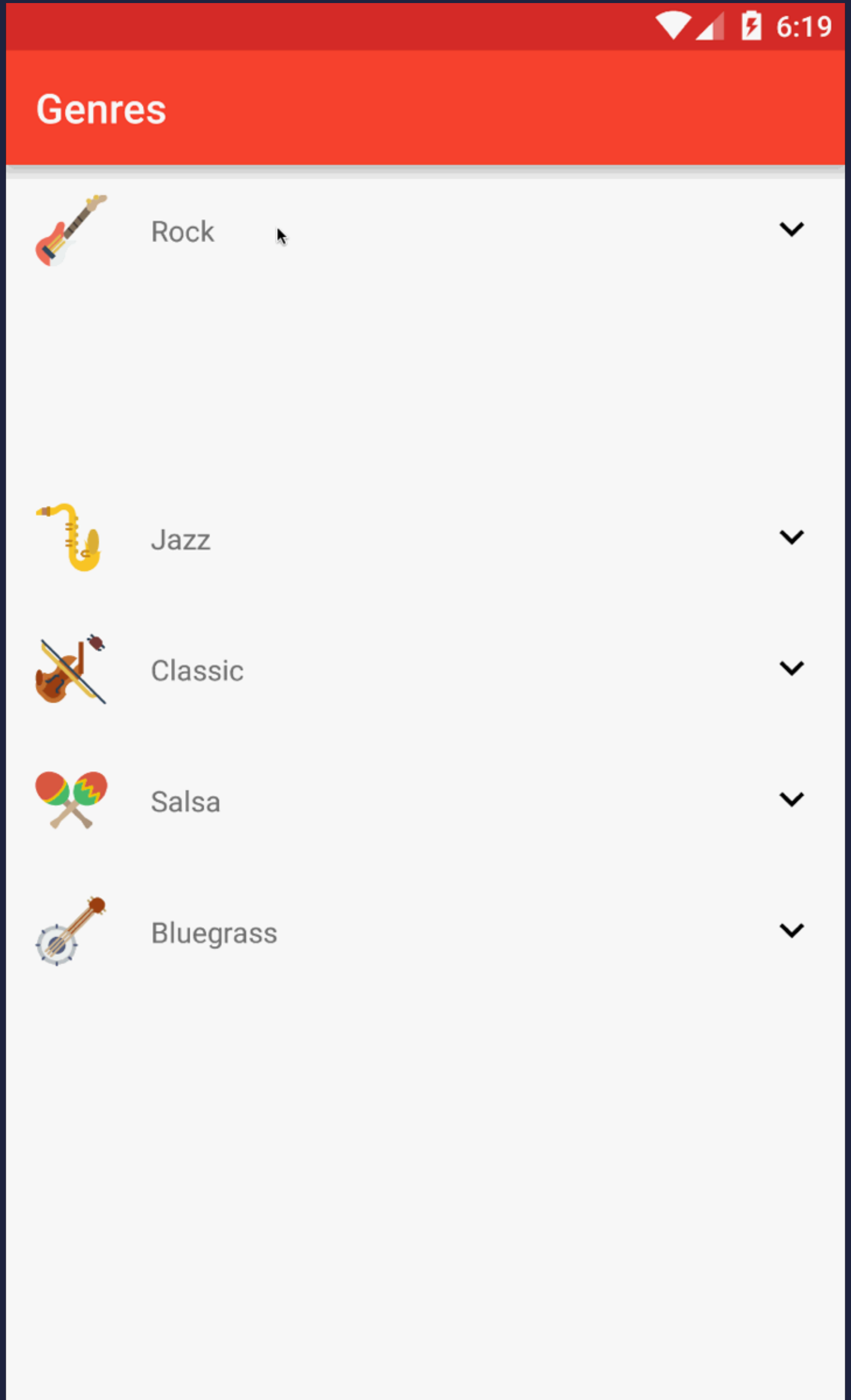
```
public class GenreActivity extends AppCompatActivity {

+ public GenreAdapter adapter;

  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_genre);
    getSupportActionBar().setTitle("Genres");

+   RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
+   LinearLayoutManager layoutManager = new LinearLayoutManager(this);

+   adapter = new GenreAdapter(makeGenres());
+   recyclerView.setLayoutManager(layoutManager);
+   recyclerView.setAdapter(adapter);
  }
}
```


6:19
Genres
Rock
Jazz
Classic
Salsa
Bluegrass

```java
public class GenreViewHolder extends GroupViewHolder {

  private TextView bandName;
  private ImageView arrow;
  private ImageView icon;

  public GenreViewHolder(View itemView) {
    super(itemView);
    bandName = (TextView) itemView.findViewById(R.id.list_item_genre_name);
    arrow = (ImageView) itemView.findViewById(R.id.list_item_genre_arrow);
    icon = (ImageView) itemView.findViewById(R.id.list_item_genre_icon);
  }








}
```

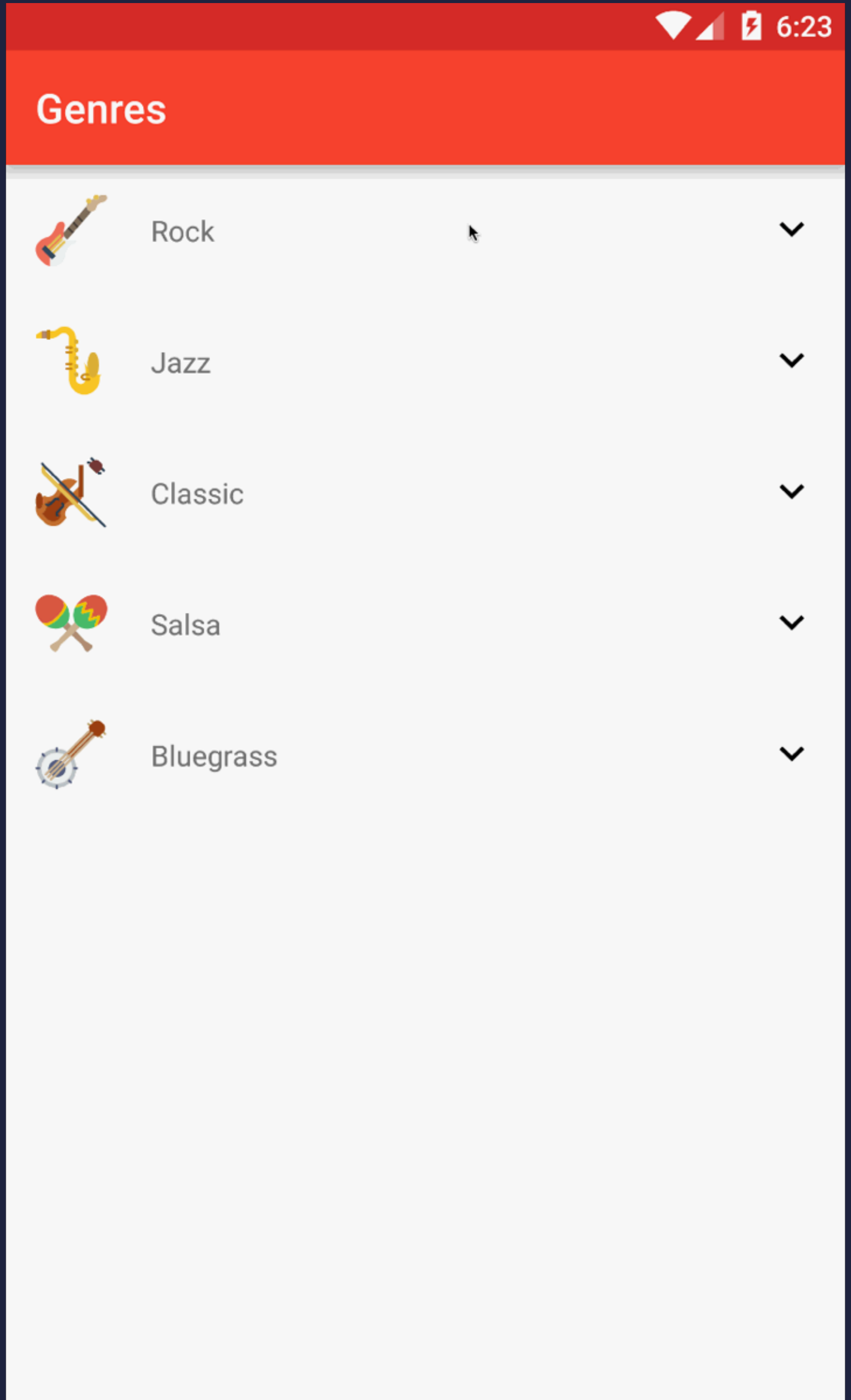
```
public class GenreViewHolder extends GroupViewHolder {

  private TextView bandName;
  private ImageView arrow;
  private ImageView icon;

  public GenreViewHolder(View itemView) {
    super(itemView);
    bandName = (TextView) itemView.findViewById(R.id.list_item_genre_name);
    arrow = (ImageView) itemView.findViewById(R.id.list_item_genre_arrow);
    icon = (ImageView) itemView.findViewById(R.id.list_item_genre_icon);
  }

+  @Override
+  public void expand() {
+    animateExpand();
+  }

+  @Override
+  public void collapse() {
+    animateCollapse();
+  }
}
```


6:23
Genres
Rock
Jazz
Classic
Salsa
Bluegrass

😺

THE HOW (DOES IT WORK)

“[RecyclerView] Adapters provide a binding from an app-specific *data set* to *views* that are displayed within a RecyclerView”

💑

DATA *and* VIEWS

```java
public abstract VH onCreateViewHolder(ViewGroup parent, int viewType);

public abstract void onBindViewHolder(VH holder, int position);

public abstract int getItemCount();
```

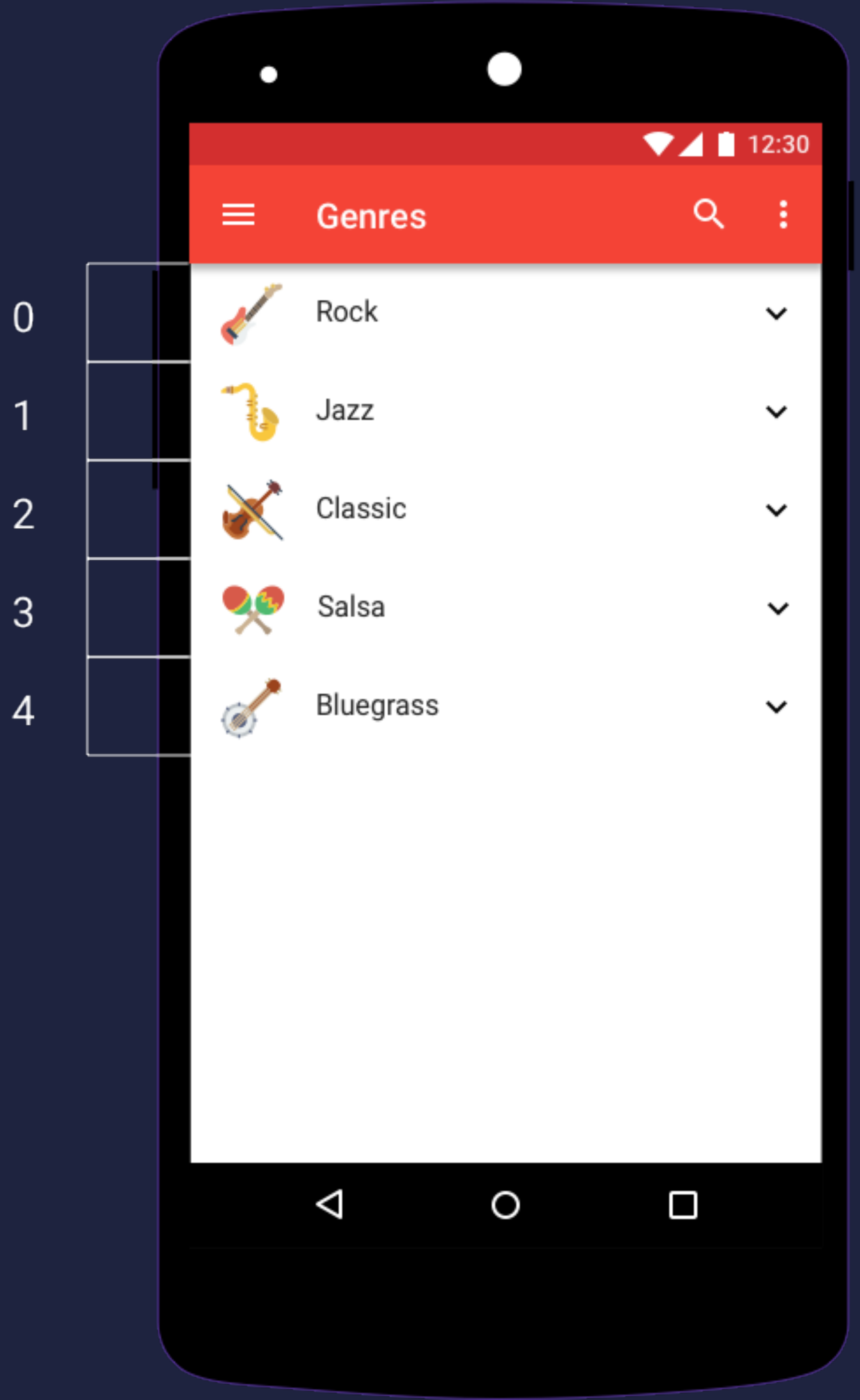
1. `getItemCount()`

1. `getItemCount()`
2. `getItemViewType(int position)`

1. `getItemCount()`
2. `getItemViewType(int position)`
3. `onCreateViewHolder(ViewGroup parent, int viewType)`

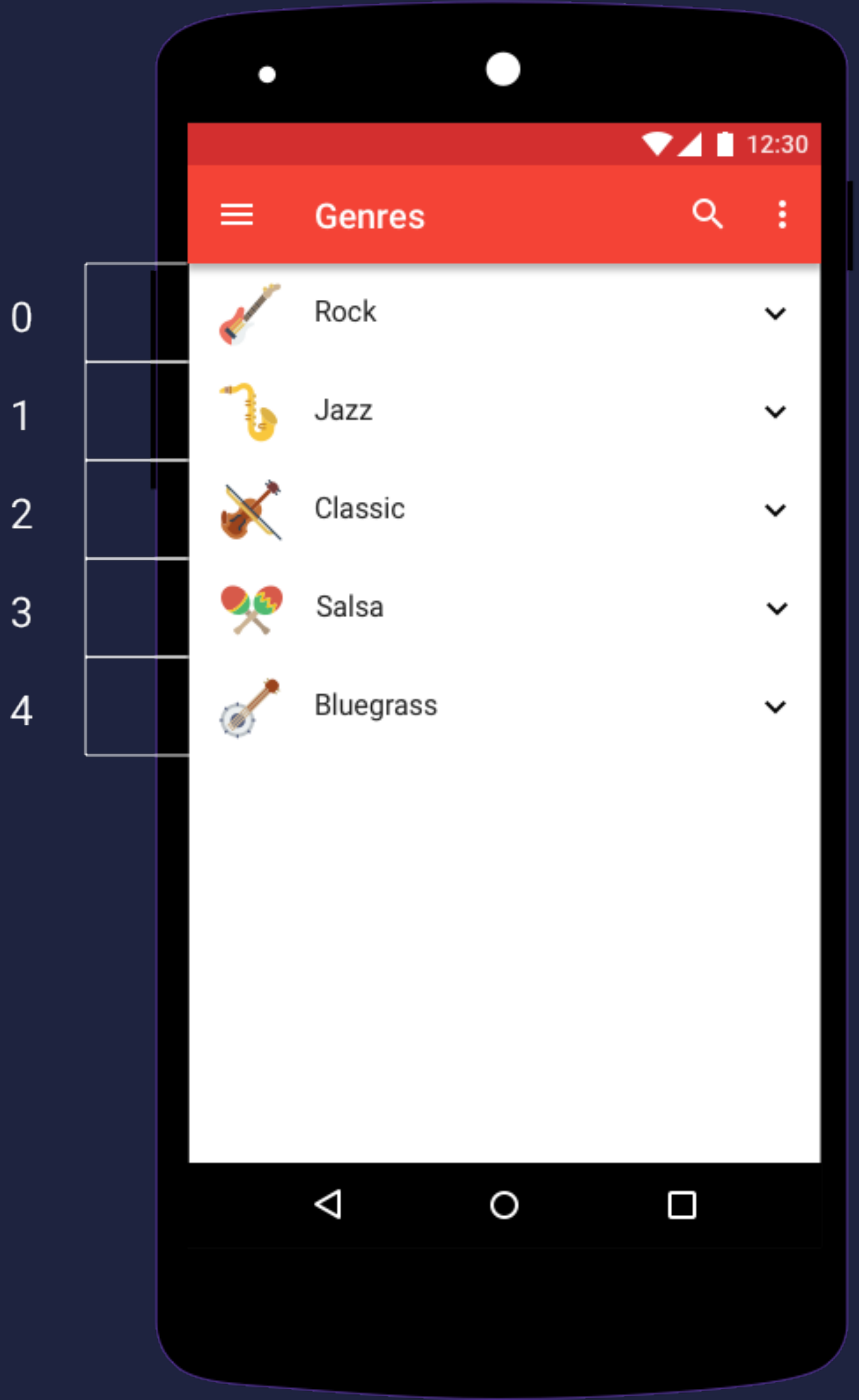
SINGLE DIMENSIONAL DATA

```
0 -[    "rock",
1 -[    "jazz",
2 -[    "classic",
3 -[    "salsa",
4 -[    "bluegrass",
```



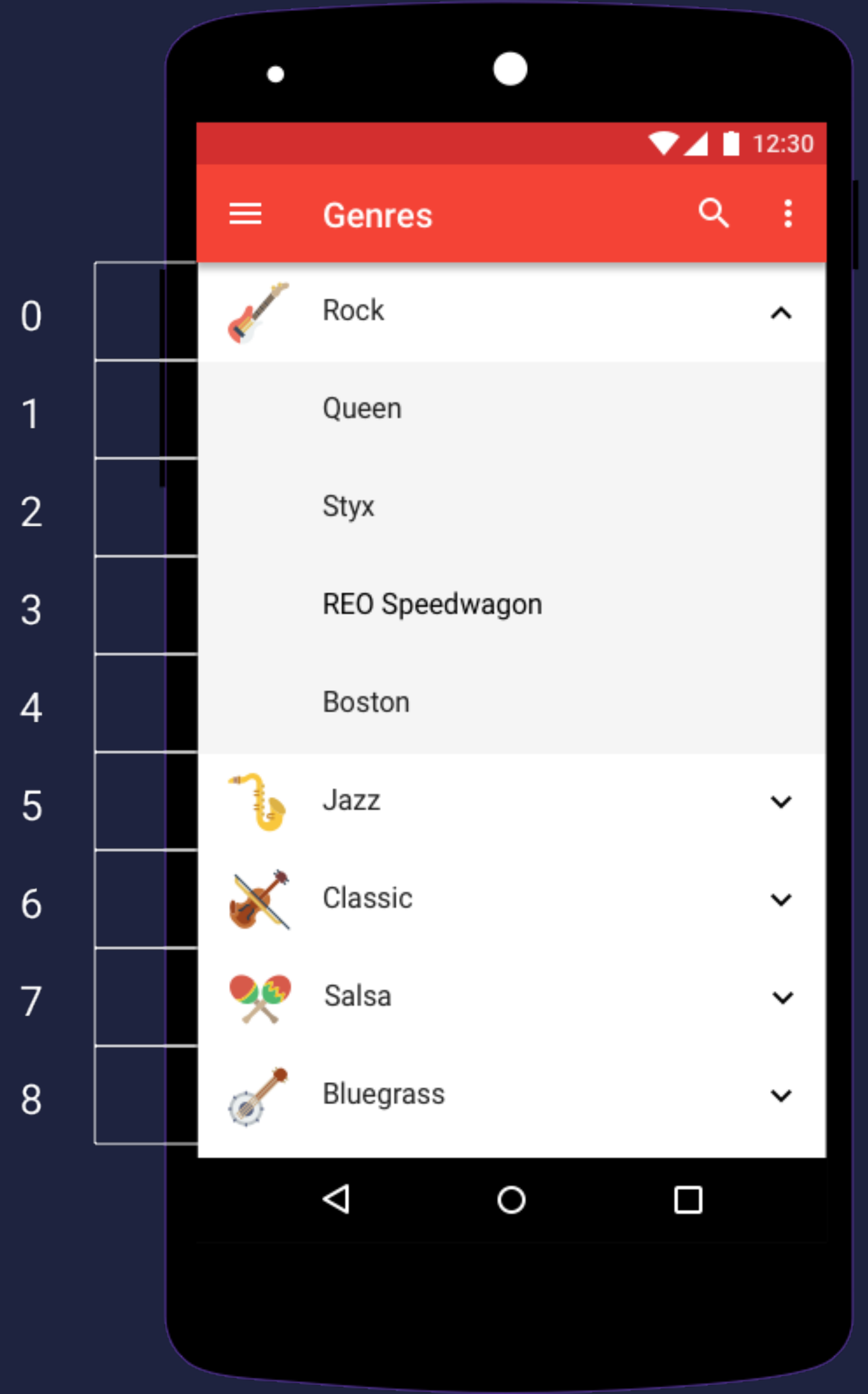
TWO DIMENSIONAL DATA

```
0 -[    "rock",
        "artists":[
1 -[        "queen",
2 -[        "styx",
3 -[        "reoSpeedwagon",
4 -[        "boston",
         ],
5 -[    "jazz",
6 -[    "classical",
7 -[    "salsa",
8 -[    "bluegrass"
```



TWO DIMENSIONAL DATA

```
0 -[    "rock",
        "artists":[
1 -[        "queen",
2 -[        "styx",
3 -[        "reoSpeedwagon",
4 -[        "boston",
         ],
5 -[    "jazz",
6 -[    "classical",
7 -[    "salsa",
8 -[    "bluegrass"
```



12:30

Genres

Rock

Queen

Styx

REO Speedwagon

Boston

Jazz

Classic

Salsa

Bluegrass

THE PROBLEM

WE HAVE TWO DIMENSIONAL DATA 👯, BUT THE RECYCLERVIEW.ADAPTER WORKS WITH SINGLE DIMENSIONAL DATA 💃

THE PROBLEM

WE HAVE TWO DIMENSIONAL DATA 👯, BUT THE RECYCLERVIEW.ADAPTER WORKS WITH ~~SINGLE DIMENSIONAL~~ 💃

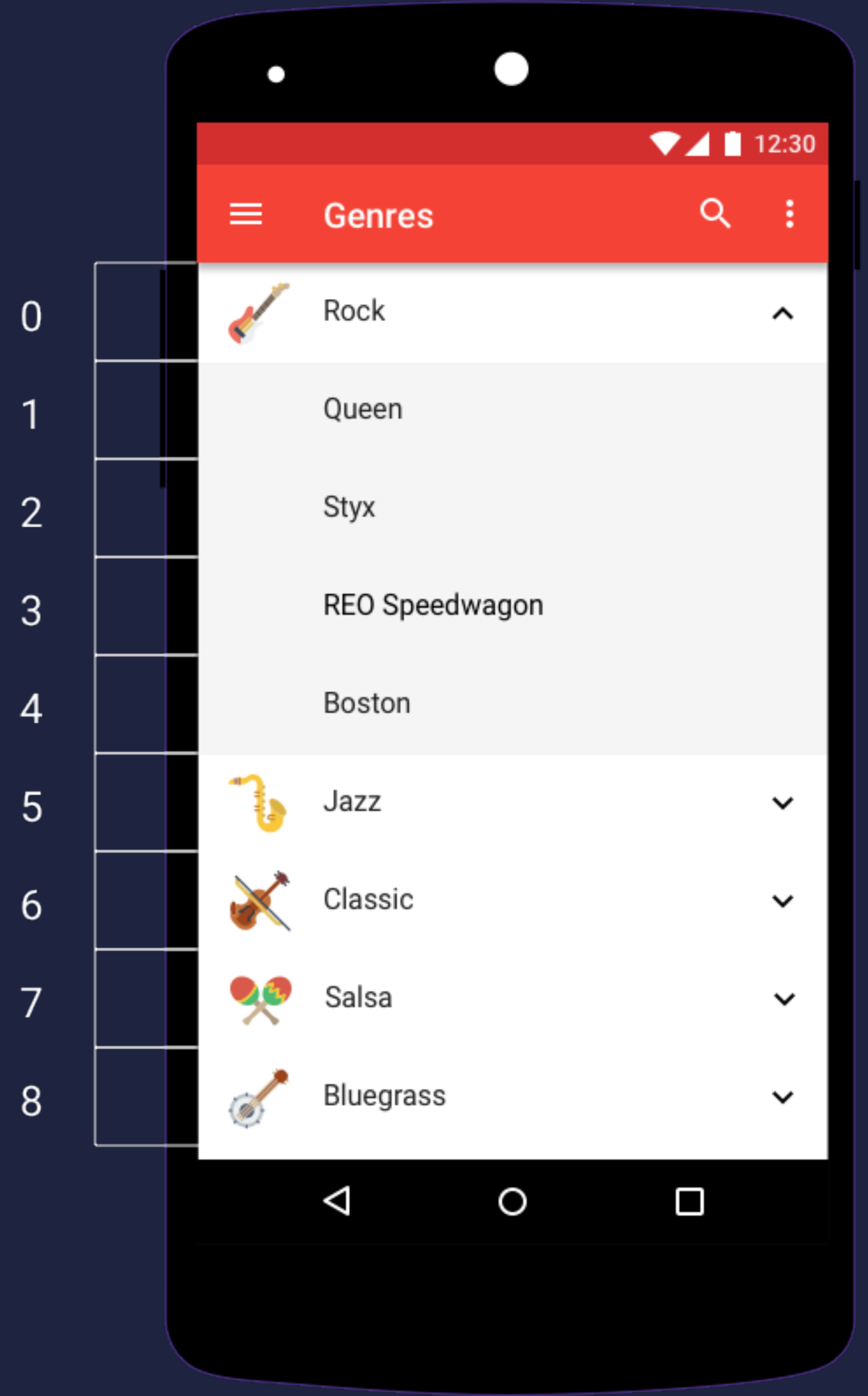
flat list positions

FLAT LIST POSITION

THE POSITION OF AN ITEM RELATIVE TO ALL THE OTHER *visible* ITEMS ON THE SCREEN

ROCK

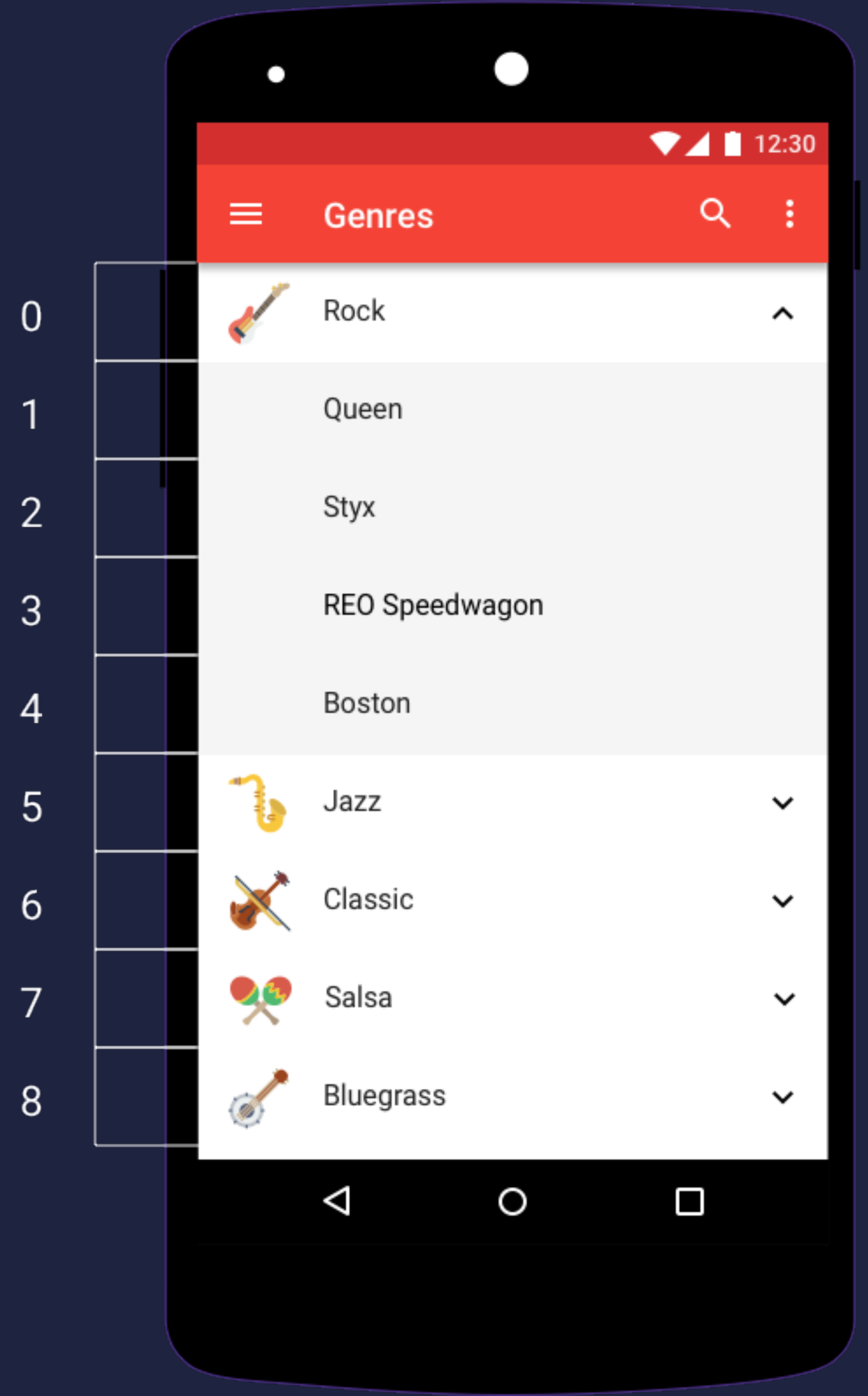
?



0
1
2
3
4
5
6
7
8
12:30
Genres
Rock
Queen
Styx
REO Speedwagon
Boston
Jazz
Classic
Salsa
Bluegrass

ROCK

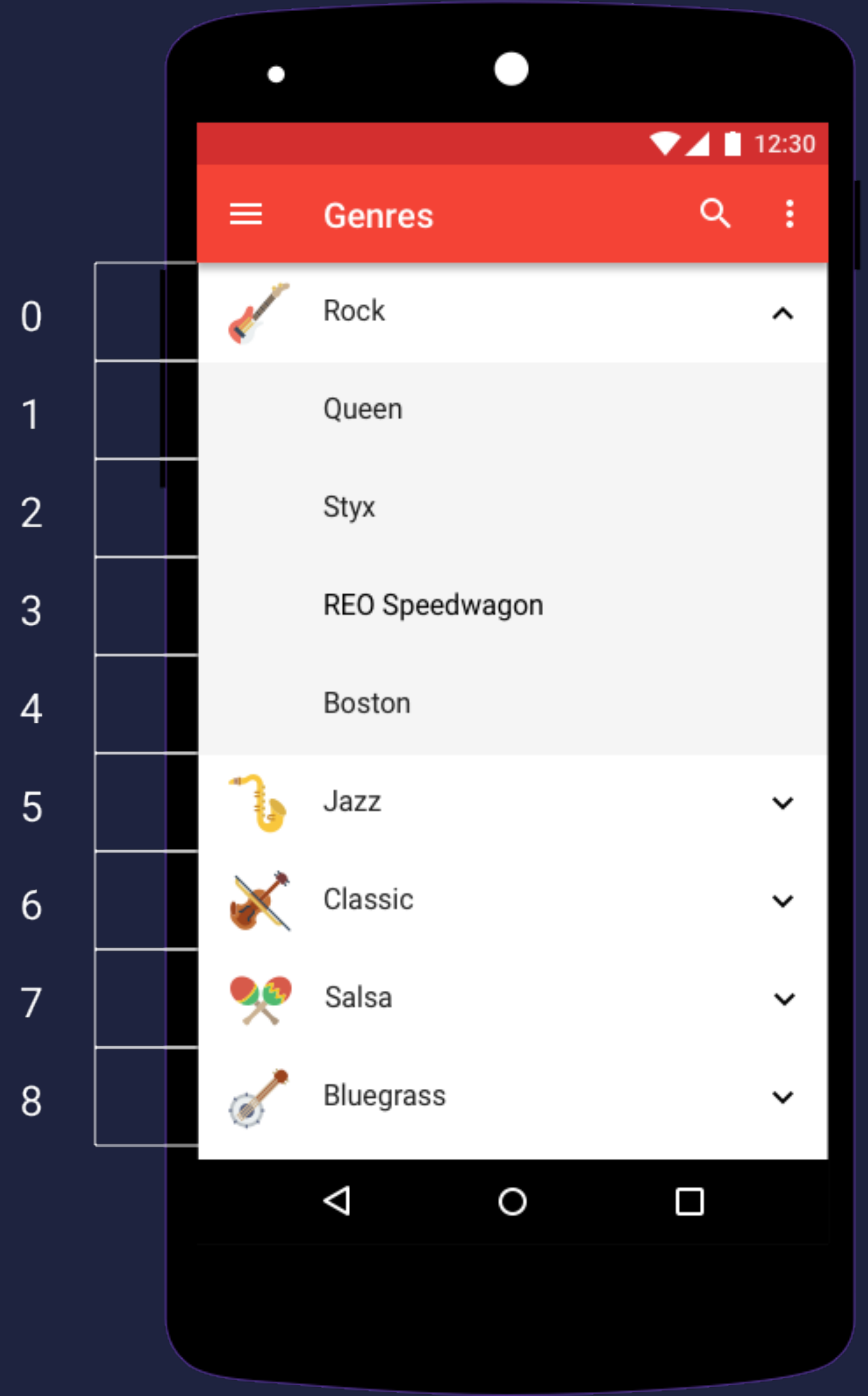
0



0
1
2
3
4
5
6
7
8
12:30
Genres
Rock
Queen
Styx
REO Speedwagon
Boston
Jazz
Classic
Salsa
Bluegrass

CLASSIC

?



Genres

Rock

Queen

Styx

REO Speedwagon

Boston

Jazz

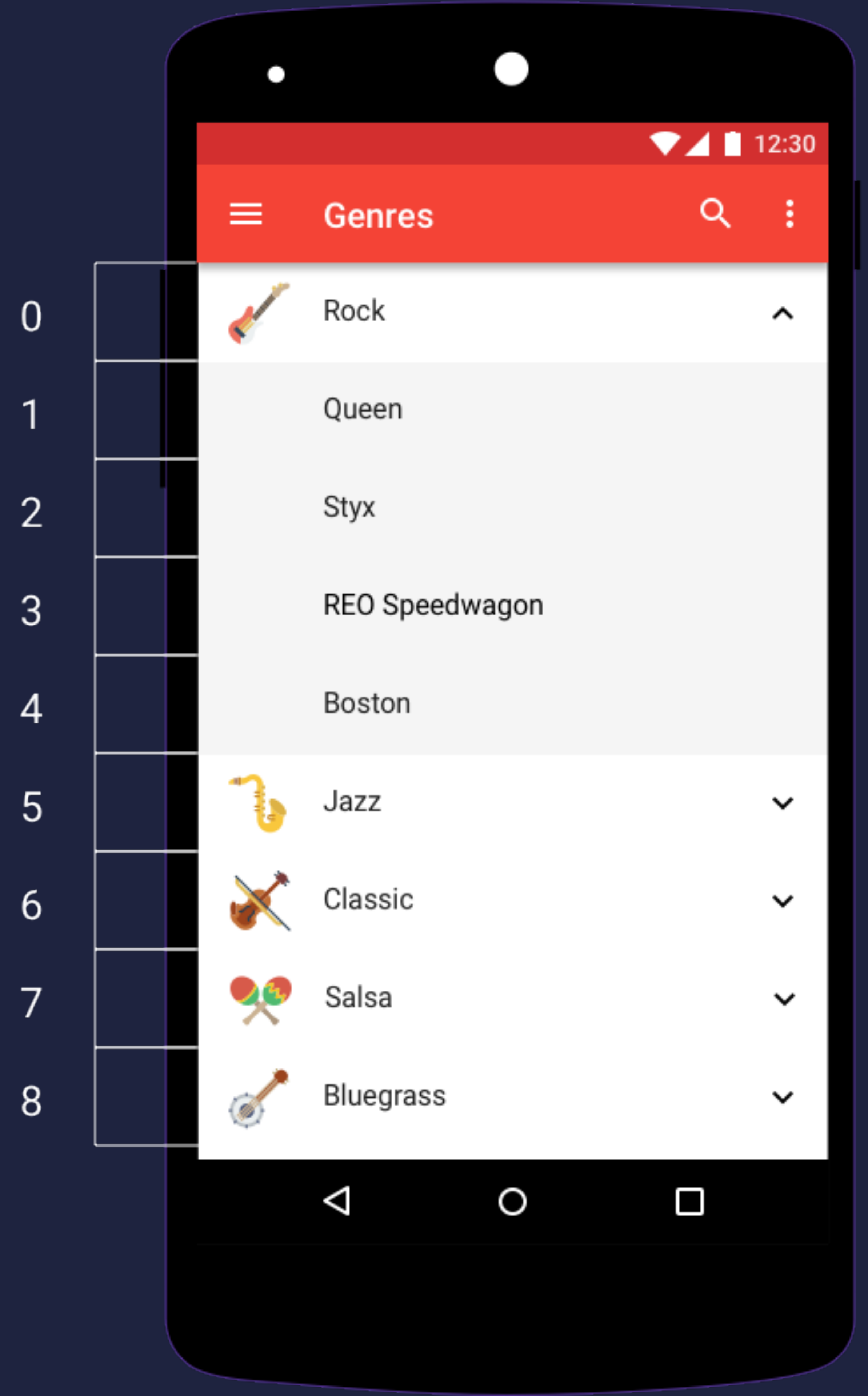
Classic

Salsa

Bluegrass

CLASSIC

6


0
1
2
3
4
5
6
7
8
12:30
Genres
Rock
Queen
Styx
REO Speedwagon
Boston
Jazz
Classic
Salsa
Bluegrass

SAY HELLO

`ExpandableList`

EXPANDABLELIST

Translator BETWEEN THE FLAT LIST POSITION AND THE BACKING DATA

SHOW ME THE ~~MONEY~~ CODE

ExpandableList

```
public ExpandableListPosition getUnflattenedPosition(int flPos) {
  int groupItemCount;
  int adapted = flPos;
  for (int i = 0; i < groups.size(); i++) {
    groupItemCount = numberOfVisibleItemsInGroup(i);
    if (adapted == 0) {
      return ExpandableListPosition.obtain(ExpandableListPosition.GROUP, i, -1, flPos);
    } else if (adapted < groupItemCount) {
      return ExpandableListPosition.obtain(ExpandableListPosition.CHILD, i, adapted - 1, flPos);
    }
    adapted -= groupItemCount;
  }
  throw new RuntimeException("Unknown state");
}
```

SAY HELLO

`ExpandableListPosition`

ExpandableListPosition

ExpandableListPosition

- int type

ExpandableListPosition

- int type
- int groupPosition

ExpandableListPosition

- int type
- int groupPosition
- int childPosition

ExpandableListPosition

- int type
- int groupPosition
- int childPosition
- int flatListPosition

ExpandableRecyclerViewAdapter

ExpandableRecyclerViewAdapter

```
@Override
public int getItemViewType(int position) {
  return expandableList.getUnflattenedPosition(position).type;
}
```

ExpandableRecyclerViewAdapter

```
@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
  switch (viewType) {
    case ExpandableListPosition.GROUP:
      GVH gvh = onCreateGroupViewHolder(parent, viewType);
      return gvh;
    case ExpandableListPosition.CHILD:
      CVH cvh = onCreateChildViewHolder(parent, viewType);
      return cvh;
    default:
      throw new IllegalArgumentException("viewType is not valid");
  }
}
```

ExpandableRecyclerViewAdapter

```java
@Override
public void onBindViewHolder(ViewHolder holder, int position) {
  ExpandableListPosition listPos = expandableList.getUnflattenedPosition(position);
  ExpandableGroup group = expandableList.getExpandableGroup(listPos);
  switch (listPos.type) {
    case ExpandableListPosition.GROUP:
      onBindGroupViewHolder((GVH) holder, position, group);
      break;
    case ExpandableListPosition.CHILD:
      onBindChildViewHolder((CVH) holder, position, group, listPos.childPos);
      break;
  }
}
```

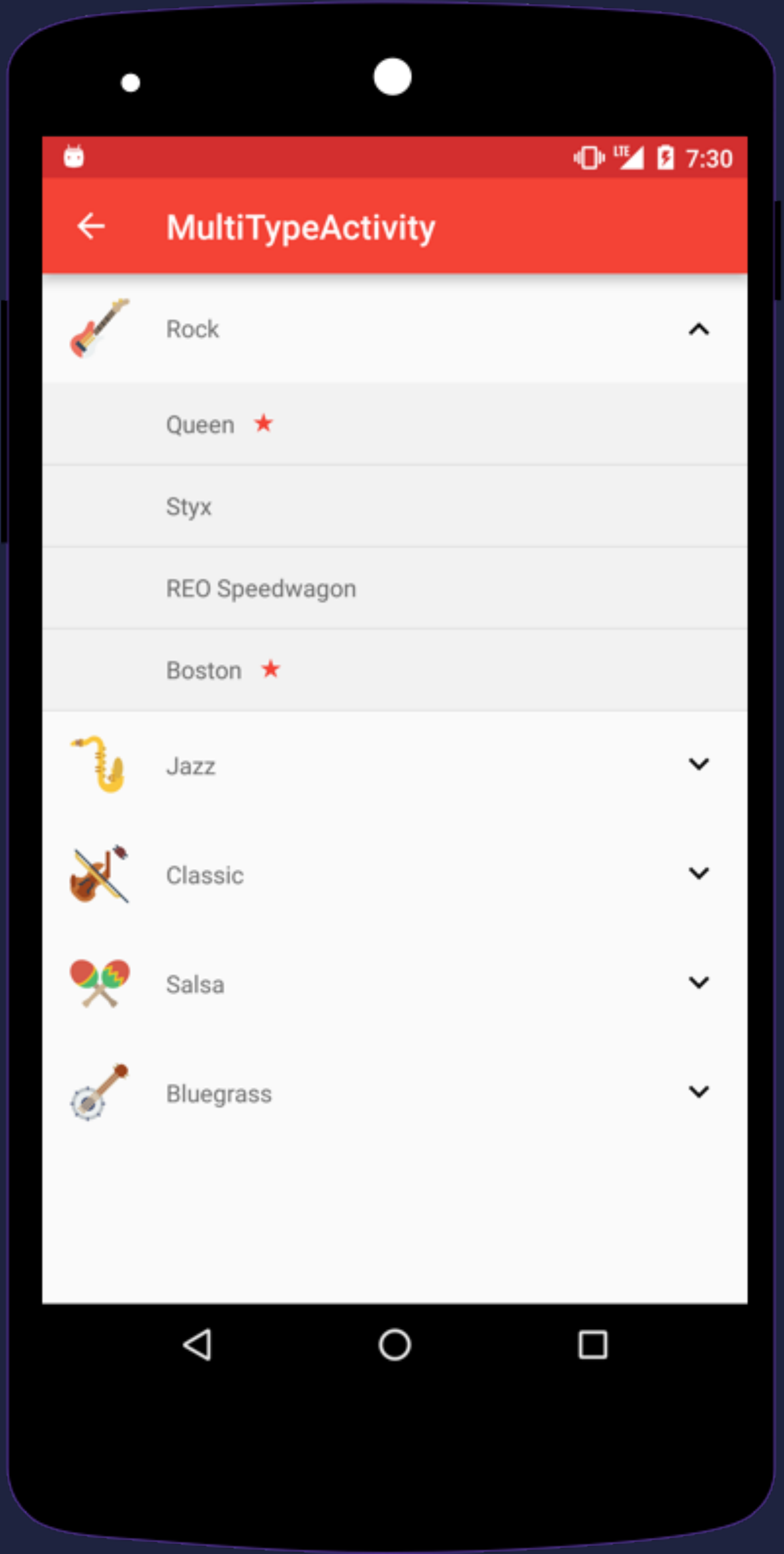
THE WHAT IF (I WANTED MORE)?

SAY HELLO

`MultiTypeExpandableRecyclerViewAdapter`

MultiTypeExpandableRecyclerViewAdapter

Allows subclasses to implement multiple different view types for both children and group



7:30

MultiTypeActivity

Rock

Queen ★

Styx

REO Speedwagon

Boston ★

Jazz

Classic

Salsa

Bluegrass

💪

THE CHILD 👶

```
public class Artist {

  private String title;


  public Artist(String title) {
    this.title = title;

  }
}
```

THE CHILD 👶

```
public class Artist {

  private String title;
+ private boolean isFavorite;

  public Artist(String title, boolean isFavorite) {
    this.title = title;
+   this.isFavorite = isFavorite;
  }
}
```

THE FAVORITE CHILD 👶 ❤

```
public class FavoriteArtistViewHolder extends ChildViewHolder {

  private TextView favoriteArtistName;

  public FavoriteArtistViewHolder(View itemView) {
    super(itemView);
    favoriteArtistName = (TextView) itemView.findViewById(R.id.list_item_favorite_artist_name);
  }

  public void bindArtist(Artist artist) {
    favoriteArtistName.setText(artist.getTitle());
  }

}
```

GENRE ADAPTER

MULTITYPE GENRE ADAPTER

MULTITYPE GENRE ADAPTER

```
public class MultiTypeGenreAdapter
     extends MultiTypeExpandableRecyclerViewAdapter<> {
}
```

MULTITYPE GENRE ADAPTER

```
public class MultiTypeGenreAdapter
     extends MultiTypeExpandableRecyclerViewAdapter<GenreViewHolder, ChildViewHolder> {
}
```

MULTITYPE GENRE ADAPTER

```
public class MultiTypeGenreAdapter
     extends MultiTypeExpandableRecyclerViewAdapter<GenreViewHolder, ChildViewHolder> {

  public static final int FAVORITE_ARTIST_VIEW_TYPE = 3;
  public static final int ARTIST_VIEW_TYPE = 4;










}
```

MULTITYPE GENRE ADAPTER

```
public class MultiTypeGenreAdapter
     extends MultiTypeExpandableRecyclerViewAdapter<GenreViewHolder, ChildViewHolder> {

  {...}

  @Override
  public int getChildViewType(int position, ExpandableGroup group, int childIndex) {
    if (((Genre) group).getItems().get(childIndex).isFavorite()) {
      return FAVORITE_ARTIST_VIEW_TYPE;
    } else {
      return ARTIST_VIEW_TYPE;
    }
  }


}
```

MULTITYPE GENRE ADAPTER

```
public class MultiTypeGenreAdapter
     extends MultiTypeExpandableRecyclerViewAdapter<GenreViewHolder, ChildViewHolder> {

  {...}

  {...}

  @Override
  public boolean isChild(int viewType) {
    return viewType == FAVORITE_ARTIST_VIEW_TYPE || viewType == ARTIST_VIEW_TYPE;




}
```

MULTITYPE GENRE ADAPTER

```java
public class MultiTypeGenreAdapter
    extends MultiTypeExpandableRecyclerViewAdapter<GenreViewHolder, ChildViewHolder> {

  public static final int FAVORITE_ARTIST_VIEW_TYPE = 3;
  public static final int ARTIST_VIEW_TYPE = 4;

  @Override
  public boolean isChild(int viewType) {
    return viewType == FAVORITE_ARTIST_VIEW_TYPE || viewType == ARTIST_VIEW_TYPE;
  }

  @Override
  public int getChildViewType(int position, ExpandableGroup group, int childIndex) {
    if (((Genre) group).getItems().get(childIndex).isFavorite()) {
      return FAVORITE_ARTIST_VIEW_TYPE;
    } else {
      return ARTIST_VIEW_TYPE;
    }
  }
```

HOW IT'S MADE 🔨

GOOD OL’ VIEWTYPES

```
public int getChildViewType(int position, ExpandableGroup group, int childIndex)

public int getGroupViewType(int position, ExpandableGroup group)

public boolean isGroup(int viewType)

public boolean isChild(int viewType)
```

EXPANDABLE RECYCLERVIEW ADAPTER

```
@Override
public int getItemViewType(int position) {
  return expandableList.getUnflattenedPosition(position).type;

  }
}
```

MULTITYPE EXPANDABLE RECYCLERVIEW ADAPTER

```
@Override
public int getItemViewType(int position) {
// return expandableList.getUnflattenedPosition(position).type;

  ExpandableListPosition listPosition = expandableList.getUnflattenedPosition(position);
  ExpandableGroup group = expandableList.getExpandableGroup(listPosition);

  int viewType = listPosition.type;
  switch (viewType) {
  case ExpandableListPosition.GROUP:
    return getGroupViewType(position, group);
  case ExpandableListPosition.CHILD:
    return getChildViewType(position, group, listPosition.childPos);
  default:
    return viewType;
  }
}
```

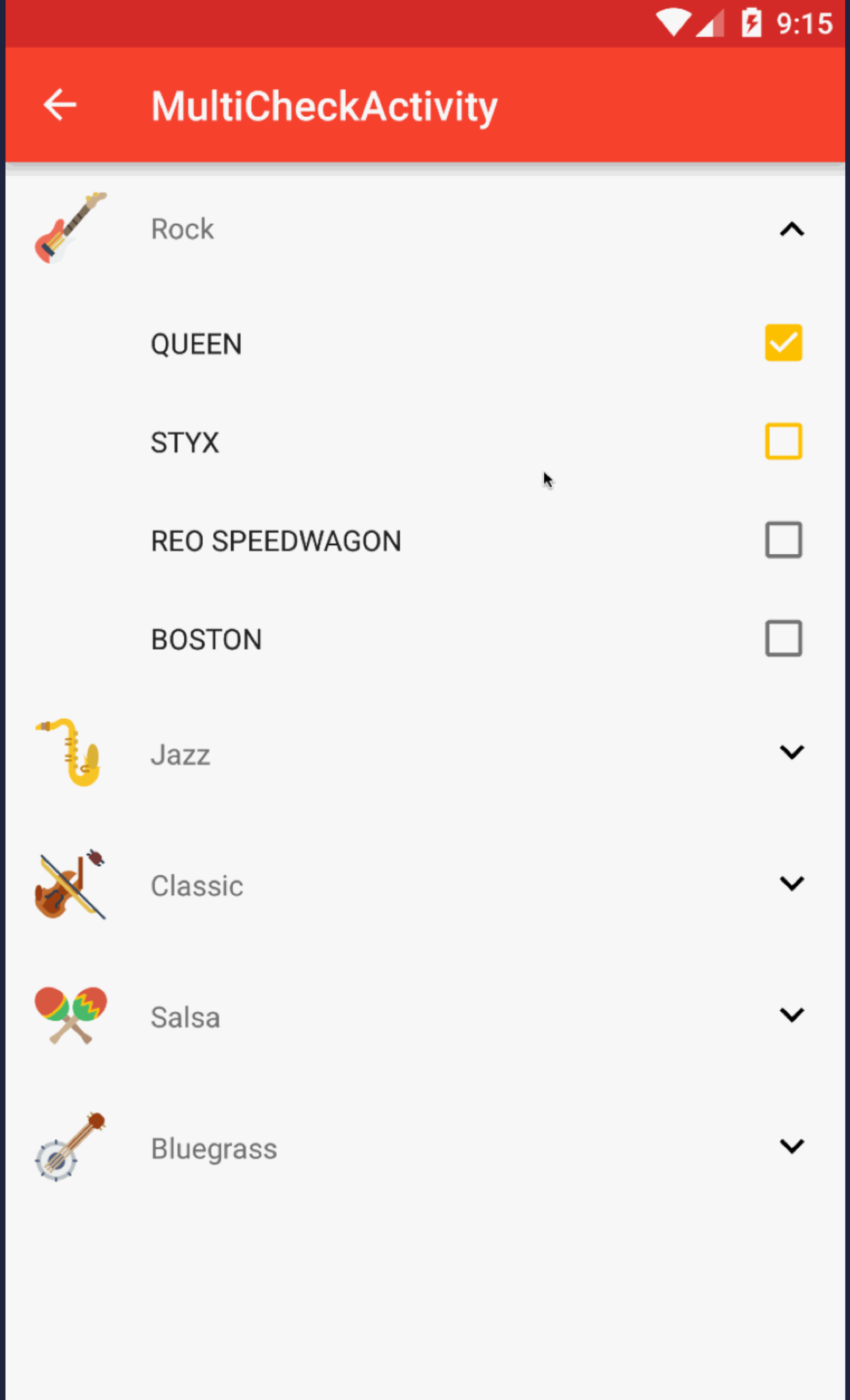
THE WHAT IF (I WANTED *Even* MORE)?

SAY HELLO

`ExpandableCheckRecyclerView`

ExpandableCheckRecyclerView

An extension of `expandablerecyclerview` for checking single or multiple children within a group



Rock

QUEEN ☑

STYX □

REO SPEEDWAGON □

BOSTON □

Jazz

Classic

Salsa

Bluegrass

TEACH ME HOW TO ~~DOUGIE~~ IMPLEMENT THAT

THE GROUP 👪

```
  public class Genre extends ExpandableGroup<Artist> {


  public Genre(String title, List<Artist> items) {
    super(title, items);
  }
}
```

THE CHECK GROUP ✔ 👪

```
- public class Genre extends ExpandableGroup<Artist> {
+ public class Genre extends SingleCheckExpandableGroup<Artist> {

  public Genre(String title, List<Artist> items) {
    super(title, items);
  }
}
```

```java
public abstract class CheckedExpandableGroup extends ExpandableGroup {

  public abstract void onChildClicked(int childIndex, boolean checked);

}
```

```java
public class SingleCheckExpandableGroup extends CheckedExpandableGroup {

  @Override
  public void onChildClicked(int childIndex, boolean checked) {
    if (checked) {
      for (int i = 0; i < getItemCount(); i++) {
        unCheckChild(i);
      }
      checkChild(childIndex);
    }
  }
```

BACK TO BUILDING 🔧

THE CHILD 👶

```
public class ArtistViewHolder extends ChildViewHolder {

  private TextView childTextView;

  public ArtistViewHolder(View itemView) {
    super(itemView);
    childTextView = (TextView) itemView.findViewById(R.id.list_item_artist_name);
  }



}
```

THE CHECKABLE CHILD ✔️ 👶

```
+ public class MultiCheckArtistViewHolder extends CheckableChildViewHolder {
- public class ArtistViewHolder extends ChildViewHolder {

  private TextView childTextView;


  public ArtistViewHolder(View itemView) {
    super(itemView);
    childTextView = (TextView) itemView.findViewById(R.id.list_item_artist_name);
  }





}
```

THE CHECKABLE CHILD ✔️ 👶

```
+ public class MultiCheckArtistViewHolder extends CheckableChildViewHolder {


  private TextView childTextView;


  public ArtistViewHolder(View itemView) {
    super(itemView);
    childTextView = (TextView) itemView.findViewById(R.id.list_item_artist_name);
  }

+  @Override
+  public Checkable getCheckable() {
+
+  }
}
```

THE CHECKABLE CHILD ✔️ 👶

```
+ public class MultiCheckArtistViewHolder extends CheckableChildViewHolder {


+  private CheckedTextView childTextView;


  public ArtistViewHolder(View itemView) {
    super(itemView);
    childTextView = (TextView) itemView.findViewById(R.id.list_item_artist_name);
  }

+  @Override
+  public Checkable getCheckable() {
+    return childTextView;
+  }
}
```

THE CHECKABLE CHILD ✔ 👶

```
+ public class MultiCheckArtistViewHolder extends CheckableChildViewHolder {


+  private CheckedTextView childTextView;


  public ArtistViewHolder(View itemView) {
   super(itemView);
-    childTextView = (TextView) itemView.findViewById(R.id.list_item_artist_name);
+    childTextView = (CheckedTextView) itemView.findViewById(R.id.list_item_artist_name);
  }

+  @Override
+  public Checkable getCheckable() {
+  + return childTextView;
+  }
}
```

LOONEY TUNES
Thats all Folks!

THE CODE 💻

https://github.com/thoughtbot/expandable-recycler-view

THE PERSON 👩

@mandybess