Лабораторная работа №5. Работа со списками

Задание: создать пользовательский список. Например, создать свой список в виде твиттера (картинка и текст), элементы которого просто статически задать в массиве (как и картинки).

Список можно создать, как минимум, двумя способами: с помощью устаревшего элемента ListView, или с помощью более сложного, но более надёжного и избавленного от недостатков элемента RecyclerView, появившегося с выходом Android Lollipop. Также можно использовать элемент GridView, но это уже на любителя. Рассмотрим сначала ListView, потом RecyclerView.

Создайте новый проект с Empty Views Activity, перейдите в файл activity_main.xml и удалите стандартное текстовое поле с фразой «Hello World!». Для начала необходимо создать xml-файл, который будет содержать список. Нажмите правой кнопкой мыши по папке values и создайте файл так, как показано на рисунке 38.

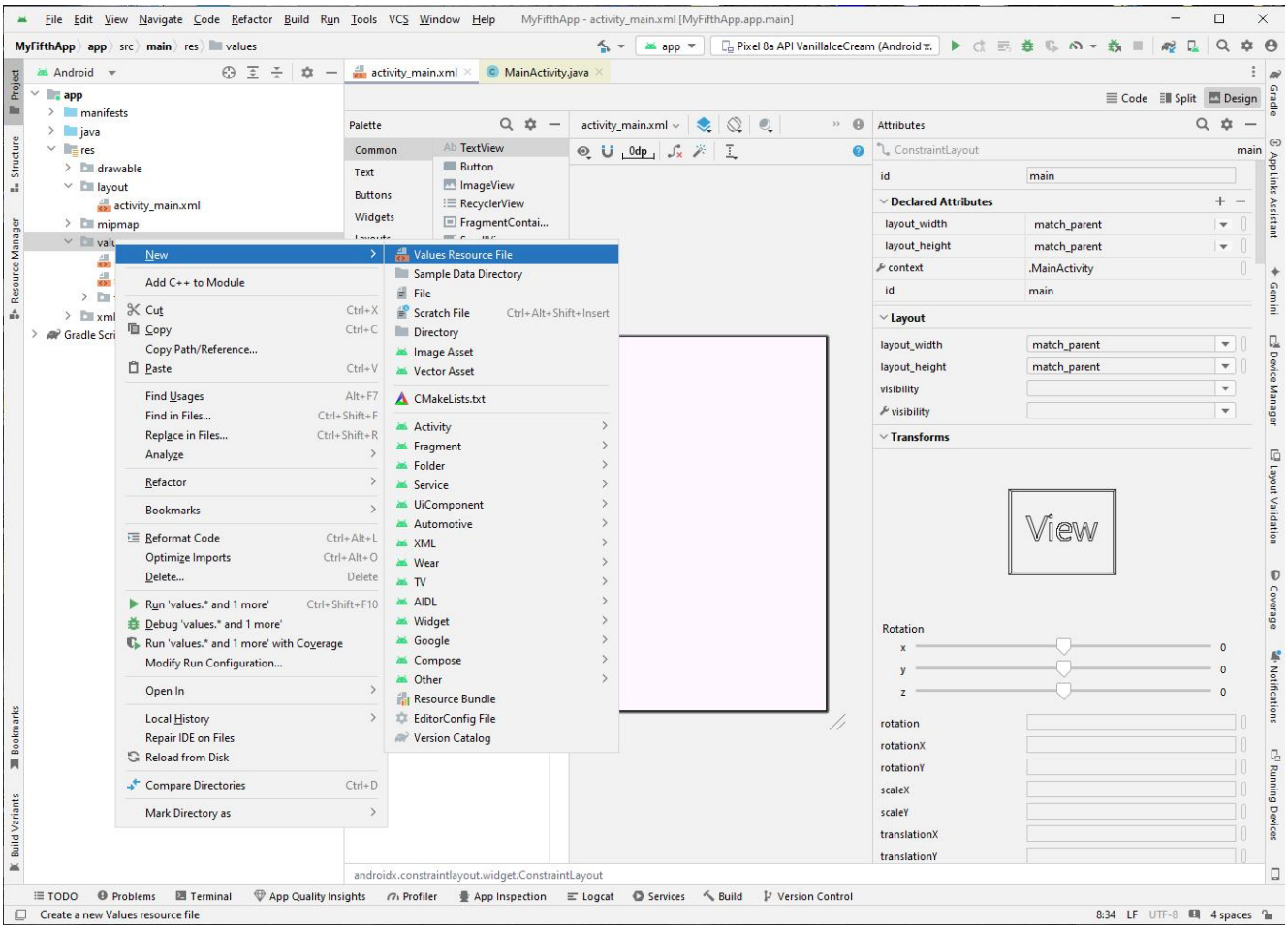

Рисунок 38 – Создание нового файла ресурсов

В появившемся окне оставьте все параметры по умолчанию и введите название файла, например, «images», после чего нажмите OK. После проделанных действий в папке values появился

файл images.xml, в котором нужно будет создать список. Переходим в этот файл и внутри тэга <resources> пишем:

```
<string-array name="images">
   <item>Image 1</item>
   <item>Image 2</item>
   <item>Image 3</item>
   <item>Image 4</item>
</string-array>
```

В item можно использовать любые слова и фразы на любом языке. Теперь перейдите в файл MainActivity и исправьте объявление класса:

Java:

```
public class MainActivity extends ListActivity {
```

Kotlin:

```
class MainActivity : ListActivity() {
```

Не забудьте импортировать нужные классы, среда разработки подскажет об этом. Затем добавьте в метод onCreate следующий код:

Java:

```
setListAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
getResources().getStringArray(R.array.images)));
```

Kotlin:

Теперь перейдите в файл activity_main.xml и из раздела Legacy панели Palette перенесите объект ListView на экран. Обратите внимание, что ListView необходимо присвоить id «@android:id/list». Сделать это можно в текстовом представлении xml (вкладка Code), добавив в тэг ListView следующую строчку:

```
android:id="@android:id/list"
```

Запустите приложение и увидите список, который был написан в файле images.xml. Однако, чтобы список был с картинками, нужно воспользоваться Custom Lists. Для этого в папке layout необходимо добавить ещё один файл. Нажмите правой кнопкой мыши по папке layout и создайте новый Layout resource file. В появившемся окне оставьте все параметры по умолчанию и введите название файла «list_item». Переходим в этот файл и помещаем элементы ImageView и TextView на своё усмотрение. Пример расположения элементов в данном файле показан на рисунке 39. Так будет выглядеть один элемент нашего списка.

Рисунок 39 – Пример расположения элементов в файле list_item.xml

Теперь вернитесь в файл MainActivity.java и измените код в методе onCreate на следующий:

setListAdapter(**new** MyAdapter(**this**, android.R.layout.***simple_list_item_1***, R.id.***textView***, getResources().getStringArray(R.array.***images***)));

Далее создайте класс MyAdapter внутри класса MainActivity:

```
    public class MyAdapter extends ArrayAdapter<String> {
  public MyAdapter(Context context, int resource, int textViewResourceId, String[] string){
    super(context, resource, textViewResourceId, string);
  }

  @Override
  public View getView(int position, View convertView, ViewGroup parent){
    LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
```

```
        View row = inflater.inflate(R.layout.list_item, parent, false);
        String[] items = getResources().getStringArray(R.array.images);
        ImageView image = (ImageView) row.findViewById(R.id.imageView);
        TextView text = (TextView) row.findViewById(R.id.textView);
        text.setText(items[position]);
        return row;
    }
}
```

Но это ещё не всё. Необходимо создать конструкцию switch-case, чтобы в каждой строке была соответствующая ей картинка. Перед return row вставьте следующий код:

```
        switch (items[position]) {
    case "Image 1":
        image.setImageResource(R.drawable.image1);
        break;
    case "Image 2":
        image.setImageResource(R.drawable.image2);
        break;
    case "Image 3":
        image.setImageResource(R.drawable.image3);
        break;
    case "Image 4":
        image.setImageResource(R.drawable.image4);
        break;
}
```

Теперь осталось поместить любые изображения в папку drawable. Изображения либо должны называться image1.jpeg, image2.jpeg и т.д., либо в конструкции switch-case необходимо поменять путь до картинок в строке R.drawable.image1 и т.д. Не забудьте про текстовое поле с ФИО. Запустите приложение и увидите собственный список с картинкой и текстом. Пример показан на рисунке 26.

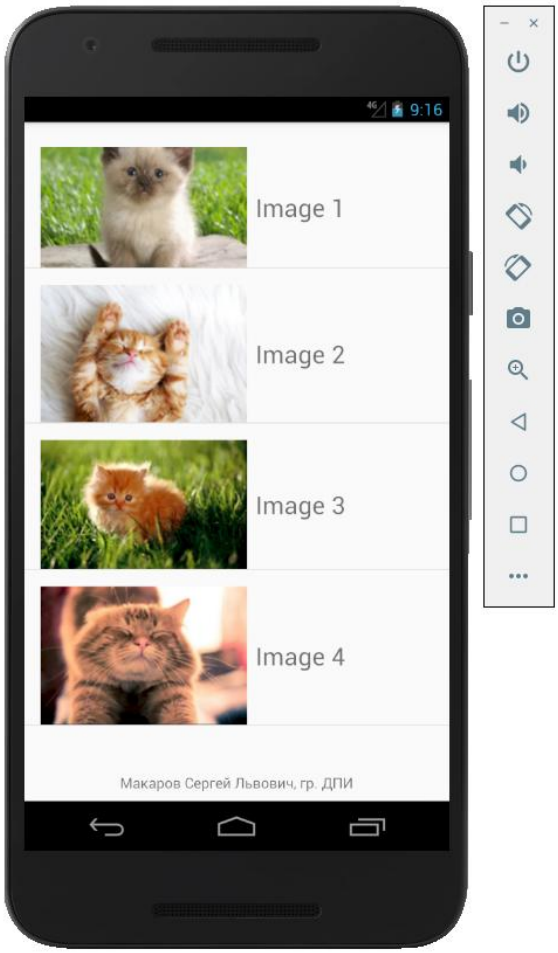

Рисунок 26 – Пример выполнения лабораторной работы №5

Теперь рассмотрим более мощный и современный элемент RecyclerView[1]. Этот элемент создаёт ровно столько элементов списка, сколько видно на экране пользователю, поэтому он значительно экономит память, если список большой – а в приложениях списки, как правило, длинные. Когда пользователь прокручивает список вниз, верхний элемент уходит за пределы экрана и становится невидимым, при этом его содержимое очищается, а сам этот более ненужный на экране элемент помещается вниз экрана и заполняется новыми данными следующих элементов списка, т.е. переиспользуется (is being **recycled**). Общая модель работы RecyclerView изображена ниже.

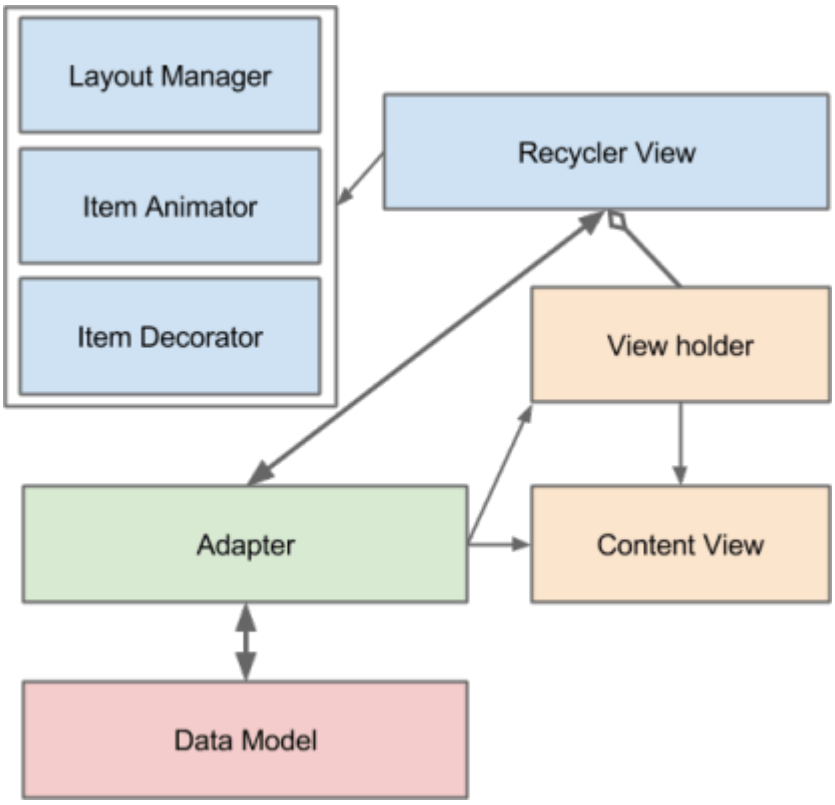

Рисунок 26.2 – Модель работы RecyclerView[2]

Создадим новый проект с EmptyActivity. Первое, что нужно сделать – это добавить следующую зависимость в gradle-файл уровня модуля:

```
implementation 'androidx.recyclerview:recyclerview:1.2.1'
```

Далее нужно построить макет для отдельного элемента списка, но он у нас уже есть в проекте с ListView и лежит в файле list_item.xml:

[1] https://metanit.com/java/android/5.11.php
[2] http://developer.alexanderklimov.ru/android/views/recyclerview-kot.php

```
<?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:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<ImageView
   android:id="@+id/imageView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginStart="16dp"
   android:layout_marginLeft="16dp"
   android:layout_marginTop="16dp"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toTopOf="parent"
   app:srcCompat="@mipmap/ic_launcher" />

<TextView
   android:id="@+id/textView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginStart="8dp"
   android:layout_marginLeft="8dp"
   android:layout_marginEnd="8dp"
   android:layout_marginRight="8dp"
   android:text="TextView"
   android:textSize="24sp"
   app:layout_constraintBottom_toBottomOf="@+id/imageView"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintHorizontal_bias="0.077"
   app:layout_constraintStart_toEndOf="@+id/imageView"
   app:layout_constraintTop_toTopOf="@+id/imageView" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Также нам понадобится файл images.xml в папке res/values, содержащий названия картинок, и сами эти картинки в папке drawable. Теперь удалим TextView "Hello world" и добавим компонент RecyclerView в activity_main.xml. Чтобы видеть дизайн элемента списка, который мы создали в list_item.xml, добавим значение @layout/list_item в свойство RecyclerView под названием listitem. Кроме того, нужно добавить следующее свойство в код описания RecyclerView:

```
app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
```

Это нужно для того, чтобы RecyclerView знал, как отображать список. Оптимизируем наш предыдущий проект: создадим 2 дополнительных класса, осблуживающих RecyclerView. Первый класс

назовём Image, он будет содержать основные методы для задания необходимых данных элемента списка:

```
public class Image {

   private String name; // название
   private int imageRes;  // картинка


   public Image(String name, int image){

      this.name=name;
      this.imageRes=image;

   }

   public String getName() {
      return this.name;
   }

   public void setName(String name) {
      this.name = name;
   }


   public int getImageResource() {
      return this.imageRes;
   }

   public void setImageResource(int flagResource) {
      this.imageRes = flagResource;
   }

}
```

Второй класс назовём CustomRecyclerAdapter, он нужен для того, чтобы определить свой кастомный адаптер для RecyclerView, и чтобы в итоге связывать нужные данные и отображать список.

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

   private final LayoutInflater inflater;
   private final List<Image> images;

   CustomRecyclerAdapter(Context context, List<Image> images) {
      this.images = images;
      this.inflater = LayoutInflater.from(context);
   }
```

```
    public void onBindViewHolder(CustomRecyclerAdapter.ViewHolder holder, int position) {
        Image image = images.get(position);
        holder.text.setText(image.getName());
        holder.image.setImageResource(image.getImageResource());

    }

    @NonNull
    @Override
    public CustomRecyclerAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view = inflater.inflate(R.layout.list_item, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public int getItemCount() {
        return images.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        final ImageView image;
        final TextView text;
        ViewHolder(View view){
            super(view);
            image = view.findViewById(R.id.imageView);
            text = view.findViewById(R.id.textView);
        }
    }
}
```

И теперь, когда у нас есть всё необходимое, в MainActivity.java достаточно добавить код, наполняющий с помощью кастомного адаптера список RecyclerView. Данные добавляются в методе setData():

```
        public class MainActivity extends AppCompatActivity {

    ArrayList<Image> images = new ArrayList<Image>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setData();
        RecyclerView recyclerView = findViewById(R.id.review1);
        CustomRecyclerAdapter adapter = new CustomRecyclerAdapter(this, images);
        recyclerView.setAdapter(adapter);
    }

    private void setData(){
```

```
        images.add(new Image ("cat1", R.drawable.image1));
        images.add(new Image ("cat2", R.drawable.image2));
        images.add(new Image ("mycat", R.drawable.image3));
        images.add(new Image ("nyancat", R.drawable.image4));

    }
}
```

Таким образом, получим тот же список из котов, но теперь он пролистывается вниз, каждый кот расположен на отдельной странице, и список можно значительно расширить.