УДК 004

Реализация индикатора загрузки в Android с использованием Material Design

Андриенко Иван Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается процесс создания индикатора загрузки в Android-приложении с использованием компонентов Material Design и анимационных элементов. Описаны основные этапы разработки и реализации трех элементов: CircularProgressIndicator, LinearProgressIndicator, и анимации на основе AnimationDrawable. Приведены примеры настройки и использования этих компонентов для обеспечения плавной и эффективной индикации загрузки, улучшая пользовательский интерфейс.
Ключевые слова: Android, material design, индикатор загрузки, CircularProgressIndicator, LinearProgressIndicator, AnimationDrawable, UI.

Implementation of the loading indicator in Android using Material Design

Andrienko Ivan Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the process of creating a loading indicator in an Android application using Material Design components and animation elements. The main stages of the development and implementation of the three elements are described: Circular Progress Indicator, LinearProgressIndicator, and animations based on AnimationDrawable. Examples of configuring and using these components are provided to ensure smooth and efficient loading indication, improving the user interface.
Keywords: Android, material design, loading indicator, CircularProgressIndicator, LinearProgressIndicator, AnimationDrawable, UI.

1 Введение

1.1 Актуальность

Современные мобильные приложения часто требуют визуальной индикации процесса загрузки или выполнения задач. Использование индикаторов загрузки помогает пользователю понять, что приложение работает и скоро завершит текущую операцию. В условиях растущего ожидания высококачественного пользовательского интерфейса важно использовать надежные и эстетически привлекательные способы

отображения состояния загрузки. В Android есть несколько методов реализации прогресс-индикаторов, и правильный выбор может значительно улучшить пользовательский опыт.

1.2 Обзор исследований

Е.А. Парфенова и О.В. Судаков в своем исследовании рассматривают различные методы экспорта анимации для использования в Web-ресурсах и на платформе Android. В работе уделено внимание сравнению различных инструментов и подходов к созданию анимации, а также их эффективности при внедрении в приложения [1]. А.И. Малышева и Т.Н. Томчинская исследуют особенности анимации персонажей в игровом приложении, акцентируя внимание на требованиях к качеству и плавности анимации. Авторы анализируют применяемые технологии и их влияние на восприятие пользователем [2]. В своей статье M.A.M. Topani и A. Azhari описывают процесс разработки игры-викторины для изучения мандаринского языка, основанной на Android. Авторы представляют Android как платформу для создания обучающих приложений, акцентируя внимание на мультимедийных возможностях и разработке интерфейса [3]. В своей работе А.В. Черных обсуждает переход от традиционной XML-верстки интерфейсов в Android-приложениях к использованию Jetpack Compose. Автор анализирует преимущества новой технологии и её влияние на упрощение процесса разработки UI [4].

1.3 Цель исследования

Цель исследования — разработать и реализовать прогресс-индикаторы в Android-приложении с использованием компонентов Material Design и анимаций, чтобы обеспечить визуально привлекательный и функциональный пользовательский интерфейс.

2 Материалы и методы

Для исследования использовались Android Studio, язык программирования Java, и библиотека Material Components для Android. В процессе разработки были реализованы три различных индикатора загрузки: CircularProgressIndicator, LinearProgressIndicator, а также анимация на основе AnimationDrawable. Эти компоненты были адаптированы и интегрированы в пользовательский интерфейс приложения. Каждый индикатор был настроен для обеспечения визуально приятного отображения прогресса загрузки и повышения удобства взаимодействия с приложением.

3 Результаты и обсуждения

Индикаторы выполнения визуально отображают состояние операции. Они используют движение, чтобы привлечь внимание пользователя к тому, насколько близок к завершению процесс, такой как загрузка или обработка данных. Они также могут указывать на то, что обработка выполняется, без привязки к тому, насколько она может быть близка к завершению.

Для реализации индикаторов необходимо добавить зависимости com.google.android.material:material:1.10.0 в проект Android. Это позволяет использовать компоненты и стили из библиотеки Material Design, предоставляемой Google (рис. 1).

```
dependencies {

    implementation 'com.google.android.material:material:1.10.0'


    implementation libs.appcompat
    implementation libs.material
    implementation libs.activity
    implementation libs.constraintlayout
    testImplementation libs.junit
    androidTestImplementation libs.ext.junit
    androidTestImplementation libs.espresso.core
}
```

Рис. 1. Добавление зависимостей в gradle module app

Этот код XML используется для создания и отображения двух типов индикаторов прогресса в Android-приложении, используя библиотеку Material Components. Первый CircularProgressIndicator: круглый индикатор прогресса, который используется для отображения состояния выполнения задачи в виде круга, где прогресс показывается по круговой дорожке. Это позволяет пользователям видеть, что задача выполняется, и дает информацию о примерном этапе загрзки. Второй LinearProgressIndicator: линейный индикатор прогресса, который отображается в виде горизонтальной полосы, показывающей прогресс выполнения задачи. Неопределенный индикатор не отражает, насколько близка к завершению операция. Скорее, он использует движение, чтобы указать пользователю, что обработка продолжается, хотя и без указания какой-либо степени завершения (рис. 2).

```
    <com.google.android.material.progressindicator.CircularProgressIndicator
        android:id="@+id/progress_circular"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        app:indicatorSize="48dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.445"
        app:trackThickness="4dp" />

    <com.google.android.material.progressindicator.LinearProgressIndicator
        android:id="@+id/progress_linear"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:indeterminate="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/progress_circular" />
```

Рис. 2. Создание индикаторов в макете

Пропишем логику для индикаторов. В методе onCreate активируется полноэкранный режим, после чего вызывается метод EdgeToEdge.enable(this), чтобы обеспечить отображение элементов на всю ширину экрана. Далее устанавливается макет activity_main, и для основного представления задаются отступы в соответствии с системными панелями. Линейный индикатор LinearProgressIndicator и круговой CircularProgressIndicator инициализируются и становятся видимыми с помощью метода setVisibility(View.VISIBLE).

Затем используется Handler для обновления прогресса кругового индикатора каждую 50 миллисекунд до достижения значения 100. Этот подход обеспечивает плавное отображение загрузки. При необходимости есть возможность привязать индикатор к какой-либо загрузки (рис. 3).

```
21      @SuppressLint("UseCompatLoadingForDrawables")
22      @Override
23      protected void onCreate(Bundle savedInstanceState) {
24          super.onCreate(savedInstanceState);
25          EdgeToEdge.enable( $this$enableEdgeToEdge: this);
26          setContentView(R.layout.activity_main);
27          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
28              Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
29              v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
30              return insets;
31
32          });
33
34          LinearProgressIndicator progressIndicatorL = findViewById(R.id.progress_linear);
35          progressIndicatorL.setVisibility(View.VISIBLE);
36
37          CircularProgressIndicator progressIndicator = findViewById(R.id.progress_circular);
38          progressIndicator.setVisibility(View.VISIBLE);
39
40          Handler handler = new Handler();
41          int[] progress = {1};
42          handler.postDelayed(new Runnable() {
43              @Override
44              public void run() {
45                  if (progress[0] <= 100) {
46                      progressIndicator.setProgress(progress[0]);
47                      progress[0]++;
48                      handler.postDelayed( r: this,  delayMillis: 50);
49                  }
50              }
51          },  delayMillis: 50);
```

Рис. 3. Реализация индикаторов загрузки в коде

При запуске приложения индикаторы корректно отображаются (рис. 4).

Рис. 4. Индикаторы из Material Design

Реализация анимации с использованием AnimationDrawable также эффективна. Она позволяет создавать динамичные и визуально привлекательные элементы интерфейса. Этот подход особенно полезен для создания уникальных анимаций, которые могут повысить вовлеченность пользователя.

Создадим анимацию песочных часов с помощью AnimationDrawable. Добавим ImageView в макет (рис. 5).

```
33      <ImageView
34          android:id="@+id/hourglass_view"
35          android:layout_width="374dp"
36          android:layout_height="211dp"
37          android:src="@drawable/hourglass_animation"
38          app:layout_constraintBottom_toBottomOf="parent"
39          app:layout_constraintEnd_toEndOf="parent"
40          app:layout_constraintHorizontal_bias="0.486"
41          app:layout_constraintStart_toStartOf="parent"
42          app:layout_constraintTop_toTopOf="parent"
43          app:layout_constraintVertical_bias="0.101" />
44
45
46  </androidx.constraintlayout.widget.ConstraintLayout>
```

Рис. 5. Добавление ImageView в макет

Для отображения анимации необходимо создать файл непрерывной анимации песочных часов, путем последовательного отображения изображений. Файл hourglass_animation.xml представляет собой список анимаций (animation-list), который определяет последовательность изображений для создания анимации. Каждый <item> внутри этого списка соответствует отдельному кадру анимации, для которого указано изображение и продолжительность показа кадра в миллисекундах. Данный файл создается в директории res>drawble (рис. 6).

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item
        android:drawable="@drawable/a1"
        android:duration="100" />
    <item
        android:drawable="@drawable/a2"
        android:duration="100" />
    <item
        android:drawable="@drawable/a3"
        android:duration="100" />
    <item
        android:drawable="@drawable/a4"
        android:duration="100" />
    <item
        android:drawable="@drawable/a5"
        android:duration="100" />
    <item
        android:drawable="@drawable/a6"
        android:duration="100" />
    <item
        android:drawable="@drawable/a7"
        android:duration="100" />
```

Рис. 6. Создание файла анимации

Далее пропишем код для запуска анимации. Сначала ImageView связывается с View в макете, и к нему устанавливается ресурс анимации из XML-файла (hourglass_animation.xml). Затем создается экземпляр AnimationDrawable, который извлекается из установленного ресурса. Внутри метода post анимация запускается с помощью вызова start(). Этот код инициализирует и запускает анимацию в момент, когда ImageView уже доступен для отображения, обеспечивая плавное начало анимации (рис. 7).

```
    ImageView hourglassView = findViewById(R.id.hourglass_view);
    hourglassView.setImageDrawable(getResources().getDrawable(R.drawable.hourglass_animation));
    AnimationDrawable hourglassAnimation = (AnimationDrawable) hourglassView.getDrawable();
    hourglassView.post(new Runnable() {
        @Override
        public void run() {
            hourglassAnimation.start();
        }
    });


}
```

Рис. 7. Запуск файла анимации

Анимация с использованием AnimationDrawable также успешно протестирована (рис. 8).

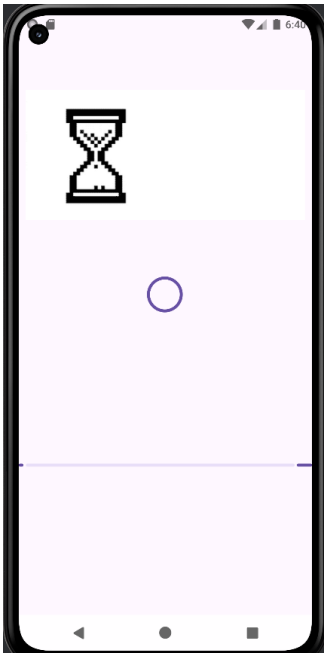

Рис. 8. Тестирование анимации

Выводы

В данной статье рассмотрены различные подходы к реализации индикаторов загрузки в Android-приложениях, включая использование ProgressBar, MaterialProgressBar и AnimationDrawable. Проведенный анализ показал, что каждый из методов имеет свои преимущества и может быть применен в зависимости от требований приложения. Использование этих инструментов позволяет создавать интуитивно понятные и визуально привлекательные интерфейсы, улучшая пользовательский опыт.

Библиографический список

1. Парфенова Е.А., Судаков О.В. Исследование способов экспорта анимации для web-ресурсов и платформы android // В сборнике: КОГРАФ-2022. Сборник материалов 32-й Всероссийской научно-практической конференции по графическим информационным технологиям и системам. Нижний Новгород, 2022. С. 61-67.
2. Малышева А.И., Томчинская Т.Н. Особенности анимации персонажей в игровом приложении // В сборнике: КОГРАФ-2020. Сборник материалов 30-й Всероссийской научно-практической конференции по графическим информационным технологиям и системам. Нижний Новгород, 2020. С. 66-70.
3. Topani M.A.M., Azhari A. Development of mandarin education quiz game using android-based multimedia development life cycle // Mobile and Forensics. 2023. Т. 5. № 2. С. 31-41.

4. Черных А.В. Развитие android ui: переход от xml-верстки к jetpack compose // В сборнике: Совершенствование науки и образования в области естественных и технических исследований. Материалы XXXVI Всероссийской научно-практической конференции. Ставрополь, 2023. С. 269-271.
5. URL: https://developer.android.com/design/ui?hl=en (дата обращения 12.08.2024)