Практическая работа. Панель с прокруткой. Material Design

Многим приложениям нужен эффектный пользовательский интерфейс, быстро реагирующий на действия пользователя.

Вы научились пользоваться разными представлениями: текстовыми представлениями, кнопками и полями выбора, а также применять темы `Material` для внесения масштабных изменений в оформление и поведение приложения. Впрочем, это далеко не все. Теперь вы узнаете, как улучшить скорость реакции вашего приложения применением координирующего макета. Вы создадите панели инструментов с возможностью сворачивания или прокрутки содержимого по вашему усмотрению. Вы познакомитесь с интересными новыми представлениями: флажками, переключателями, плашками и плавающими кнопками действий. Наконец, вы узнаете, как отображать удобные всплывающие сообщения с использованием панелей `Toast` и `Snackbar`.

Material широко используется в мире Androidville

Вы научились пользоваться панелями инструментов, нижними и выдвижными панелями навигации, упрощающими перемещение в приложении, и оформлять их применением тем из библиотеки Material. Как вы, вероятно, помните, Material представляет собой систему дизайна, которая помогает строить приложения, обладающие целостным оформлением и поведением по всем экранам. Библиотека Material не ограничивается панелями инструментов, выдвижными и нижними панелями; стилевое оформление применяется ко всем представлениям в вашем приложении, от кнопок до текстовых представлений. Несколько примеров компонентов и возможностей, использующих Material:

- Панели инструментов с прокруткой и сверткой Панель инструментов может уходить за пределы экрана (сворачиваться) при прокрутке содержимого пользователем.
- Переключатели, флажки и плашки
- Плавающие кнопки действий (FAB)
- Snackbar

Чтобы показать, как использовать эти представления и возможности, мы построим новое приложение.

Приложение Bits and Pizzas

Мы построим новое приложение, которое называется Bits and Pizzas. Основное внимание будет уделено экрану Create Order, который позволяет пользователю оформить заказ на пиццу.

Приложение состоит из активности `MainActivity`, отображающей фрагмент с именем `OrderFragment`. Фрагмент определяет внешний вид и функциональность экрана Create Order.

Рассмотрим последовательность действий для построения приложения.

Что предстоит сделать

Чтобы построить приложение, мы выполним следующие действия:

- Добавление панели инструментов с прокруткой. Мы создадим фрагмент OrderFragment и добавим в его макет панель инструментов, которая уходит с экрана, когда пользователь прокручивает экран вверх, и появляется снова при прокрутке вниз.
- Реализация сворачиваемой панели инструментов. Когда панель инструментов будет поддерживать прокрутку, мы добавим к ней изображение и заставим ее сворачиваться и разворачиваться, когда пользователь прокручивает содержимое экрана.
- Добавление представлений. Пользователь должен иметь возможность оформить заказ на пиццу. Для этого в макет OrderFragment будут добавлены переключатели, плашки и плавающая кнопка действия.
- Программирование реакции FAB на щелчки. Когда пользователь щелкает на FAB, приложение будет отображать всплывающее сообщение с подробным описанием заказа.

Создание проекта Bits and Pizzas

Мы создадим новый проект для приложения `Bits and Pizzas` по схеме, уже знакомой вам по предыдущим главам.

Выберите вариант `Empty Activity`, введите имя проекта «Bits and Pizzas» и имя пакета «com.hfad.bitsandpizzas», подтвердите папку сохранения по умолчанию. Убедитесь в том, что выбран язык Kotlin с минимальным уровнем SDK API 29, чтобы приложение работало на большинстве устройств Android.

Включение завиисимости для библиотеки Material в файл build.gradle приложения.

В этой главе будут использоваться темы, представления и функциональность библиотеки `Material`. Следовательно, мы должны убедиться в том, что она включена в файл `build.gradle` приложения как зависимость. Откройте файл BitsandPizzas/app/build.gradle и убедитесь в том, что раздел dependencies включает следующую строку (выделенную жирным шрифтом):

```
dependencies {
 ...
 implementation 'com.google.android.material:material:1.11.0'
 ...
}
```

Весьма вероятно, что среда Android Studio уже включила в файл эту зависимость за вас. Если она отсутствует, добавьте ее самостоятельно и щелкните на ссылке `Sync Now`, появляющейся в верхней части редактора кода, чтобы синхронизировать изменения с остальными частями проекта. После включения библиотеки `Material` в приложение можно переходить к созданию `OrderFragment`.

Создание OrderFragment

OrderFragment — главный экран приложения Bits and Pizzas, где пользователь будет оформлять заказ на пиццу. Чтобы создать фрагмент, выделите пакет `com.hfad.bitsandpizzas` в папке `app/src/main/java` и выберите команду File→New→Fragment→Fragment (Blank). Введите имя фрагмента «OrderFragment» и имя макета «fragment_order» и убедитесь в том, что выбран язык Kotlin. Затем обновите код OrderFragment.kt и приведите его к следующему виду:

```
package com.hfad.bitsandpizzas
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
class OrderFragment : Fragment() {
 override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
 ): View? {
 return inflater.inflate(R.layout.fragment_order, container, false)
 }
}
```

В оставшейся части мы будем заниматься обновлением `OrderFragment`. Начнем с отображения его в макете `MainActivity`.

Отображение OrderFragment в макете MainActivity

Чтобы включить `OrderFragment` в макет MainActivity, воспользуемся `FragmentContainerView` с указанием имени фрагмента.

Ниже приведен код решения этой задачи; откройте файл `activity_main.xml` и обновите его код (изменения выделены жирным шрифтом):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.hfad.bitsandpizzas.OrderFragment"
    tools:context=".MainActivity" />
```

Наконец, откройте `MainActivity.kt` и убедитесь в том, что код в файле соответствует приведенному ниже:

```
package com.hfad.bitsandpizzas
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
class MainActivity : AppCompatActivity() {
 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
 }
}
```

Теперь в MainActivity отображается OrderFragment. Давайте посмотрим, как заставить панель приложения реагировать на прокрутку.

Замена стандартной панели приложения панелью инструментов

Сейчас мы займемся реакцией панели приложения `Bits and Pizzas` на прокрутку. Первая задача — позаботиться о том, чтобы панель приложения уходила с экрана, когда пользователь прокручивает экран вверх, и появлялась снова, когда экран прокручивается вниз. Для этого необходимо сначала заменить стандартную панель приложения панелью инструментов. Так как стандартная панель приложения закрепляется у верхнего края экрана, она не может прокручиваться. С другой стороны, панель инструментов обладает куда большей гибкостью.

Для этого сначала необходимо заменить тему приложения другой, не содержащей панели приложения. Откройте файл `themes.xml` в `app/src/main/res/values`, обновите его и включите стиль, выделенный ниже жирным шрифтом:

```
<resources>
 <style name="Theme.BitsAndPizzas"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
 ...
 </style>
</resources>
```

Если ваш проект содержит файл `themes.xml` в папке `values-night`, описанное выше изменение также придется применить в этом файле. После того как изменение будет внесено, добавьте панель инструментов в `FragmentOrder`. Для этого обновите код файла `fragment_order.xml`, чтобы он совпадал с приведенным ниже:

<com.google.android.material.appbar.MaterialToolbar android:id="@+id/toolbar" android:layout_width="match_parent" android:layout_height="?attr/actionBarSize" style="@style/Widget.MaterialComponents.Toolbar.Primary" />

У фрагментов нет метода setSupportActionBar()

После того как панель инструментов будет добавлена в проект, необходимо наделить ее поведением обычной панели приложения, в которой выводится имя приложения. Для этого следует вызвать метод `setSupportActionBar()` активности. Но здесь панель инструментов добавляется в фрагмент, а у фрагментов нет метода `setSupportActionBar()`. Чтобы обойти это затруднение, мы получим ссылку на активность, отображающую фрагмент, преобразуем ее к типу `AppCompatActivity` и вызовем метод `setSupportActionBar()`. Задача решается следующим кодом:

```
val toolbar = view.findViewById<MaterialToolbar>(R.id.toolbar)
(activity as AppCompatActivity).setSupportActionBar(toolbar)
```

Ниже приведен полный код `OrderFragment.kt`; включите изменения в свою версию (выделены жирным шрифтом):

```
package com.example.android.bitsandpizzas
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.appbar.MaterialToolbar
class OrderFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:
Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_order, container, false)
        val toolbar = view.findViewById<MaterialToolbar>(R.id.toolbar)
        (activity as AppCompatActivity).setSupportActionBar(toolbar)
        return view
    }
}
```

Добавили панель инструментов... что дальше?

Мы включили панель инструментов в макет `OrderFragment`, чтобы при запуске приложения панель инструментов отображалась у верхнего края экрана.

Но если вы попытаетесь прокрутить экран, панель инструментов останется неподвижной. Чтобы она реагировала на прокрутку, необходимо внести ряд изменений.

Панель инструментов должна реагировать на прокрутку

Чтобы панель инструментов перемещалась, необходимо добавить дополнительные представления в макет фрагмента. Макет должен иметь следующую структуру:

```
<androidx.coordinatorlayout.widget.CoordinatorLayout ...>
 <com.google.android.material.appbar.AppBarLayout ...>
 <com.google.android.material.appbar.MaterialToolbar .../>
 </com.google.android.material.appbar.AppBarLayout>
 <androidx.core.widget.NestedScrollView ...>
 ...
 </androidx.core.widget.NestedScrollView>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Макет должен включать три составляющие: координирующий макет, макет панели приложения и вложенное представление прокрутки. Их комбинация позволяет панели инструментов реагировать на прокрутку экрана пользователем. Давайте разберемся, что делает каждый из них, начиная с координирующего макета.

Координирующий макет согласовывает анимации между представлениями

Координирующий макет напоминает расширенный фреймовый макет, который используется для координации анимаций между разными представлениями. Например, он может координировать прокрутку основного содержимого макета с выходом панели инструментов за границы экрана. Координирующий макет добавляется в код макета следующим образом:

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 ...>
 ...
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Включите все представления, анимации которых должны координироваться в координирующем макете. Например, в приложении `Bits and Pizzas` необходимо координировать две операции: прокрутку основного содержимого макета пользователем и выход панели инструментов за границы экрана. Это означает, что панель инструментов и основное содержимое экрана необходимо включить в координирующий макет:

Итак, теперь вы знаете, как работает координирующий макет. Перейдем к макету панели приложения.

Макет панели приложения включает анимацию панели инструментов

Макет панели приложения представляет собой разновидность вертикального линейного макета, разработанную для работы с панелями приложений. Его взаимодействие с координирующим макетом

делает возможной анимацию панели инструментов. Макет панели приложения добавляется в код приложения следующим образом:

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 ...>
 <com.google.android.material.appbar.AppBarLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:theme="@style/ThemeOverlay.MaterialComponents.Dark.ActionBar">
 <com.google.android.material.appbar.MaterialToolbar .../>
 </com.google.android.material.appbar.AppBarLayout>
 ...
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Обратите внимание на атрибут `android:theme` в приведенном выше коде макета панели приложения:

```
android:theme="@style/ThemeOverlay.MaterialComponents.Dark.ActionBar"
```

Значение применяет заданный стиль к макету панели приложения и всем его представлениям. В данном случае это означает, что панель инструментов — и все остальное, что будет добавлено в макет панели приложения, — будет оформлена с использованием свойств панели приложения темы `Material`, включая цвета ее фона и текста. Превосходно!Теперь ваша панель инструментов может реагировать на события прокрутки. Впрочем, это еще не все: также необходимо указать, как она должна на них реагировать. Давайте посмотрим, как это делается.

Программирование реакции панели инструментов на события прокрутки

После того как панель приложения будет добавлена, нужно сообщить ей, как реагировать на прокрутку. Для этого следует добавить к панели инструментов атрибут `app:layout_scrollFlags` и присвоить ему значение. Значение указывает, как панель инструментов должна реагировать на события прокрутки. В приложении `Bits and Pizzas` панель инструментов должна прокручиваться вверх за пределы экрана, когда пользователь перемещает экран вверх, и быстро возвращаться в исходную позицию, когда пользователь перемещает экран вниз. Для этого следует присвоить атрибуту `app:layout_scrollFlags` панели инструментов значение `"scroll|enterAlways"`, для чего используется код следующего вида:

```
<androidx.coordinatorlayout.widget.CoordinatorLayout ...>
 <com.google.android.material.appbar.AppBarLayout...>
 <com.google.android.material.appbar.MaterialToolbar
 android:id="@+id/toolbar"
 android:layout_width="match_parent"
 android:layout_height="?attr/actionBarSize"
```

```
 app:layout_scrollFlags="scroll|enterAlways"/>
 </com.google.android.material.appbar.AppBarLayout>
 ...
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Строка

```
app:layout_scrollFlags="scroll|enterAlways"
```

задает два аспекта поведения: `scroll` и `enterAlways`. Значение `scroll` означает, что представление может прокручиваться за верхний край экрана при прокрутке экрана вверх. Без этого значения панель инструментов осталась бы закрепленной у верхнего края экрана и не участвовала бы в прокрутке. Значение `enterAlways` означает, что панель инструментов быстро прокручивается вниз до своей исходной позиции, когда пользователь прокручивает экран вниз. Без этого значения панель инструментов все равно прокручивалась бы вниз, но это происходило бы намного медленнее. Почти все! Чтобы панель инструментов `OrderFragment` прокручивалась, осталось сделать последний шаг: добавить вложенное представление прокрутки.

Представление вложенной прокрутки делает возможной прокрутку содержимого макета

Теперь необходимо добавить представление вложенной прокрутки `NestedScrollView`, чтобы основное содержимое макета могло прокручиваться. Такие представления работают так же, как и обычные представления прокрутки, не считая того, что они поддерживают вложенную прокрутку. Это важно, потому что координирующий макет прослушивает события только вложенной прокрутки. Если вы используете обычное представление прокрутки в своем макете, панель инструментов не сможет реагировать на прокрутку экрана пользователем. Для включения представления вложенной прокрутки в макет используется код следующего вида:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout...>
 <com.google.android.material.appbar.AppBarLayout...>
 ...
 </com.google.android.material.appbar.AppBarLayout>
 <androidx.core.widget.NestedScrollView
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:layout_behavior="@string/appbar_scrolling_view_behavior">
 <TextView
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:text="This is the order fragment" />
 </androidx.core.widget.NestedScrollView>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

В приведенном выше примере представление вложенной прокрутки включает дополнительный атрибут с именем `app:layout_behavior`, которому присваивается встроенное строковое значение `"@stringappbar_scrolling_view_behavior"`. Оно гарантирует, что содержимое представления вложенной прокрутки будет размещено под макетом панели приложения и будет перемещаться при его прокрутке.

Обратите внимание: представление вложенной прокрутки может иметь только одного прямого потомка (в приведенном примере это текстовое представление). Если вы захотите добавить в представление вложенной прокрутки более одного представления, вам придется сначала включить их в группу представлений (например, линейный макет), а потом добавить группу в представление вложенной прокрутки. Вот и все, что необходимо знать для обеспечения прокрутки панели инструментов `Bits and Pizzas`. Сделаем следующий шаг и обновим макет `OrderFragment`.

Полный код fragment_order.xml

Ниже приведен полный код `fragment_order.xml`; обновите свою версию файла (изменения выделены жирным шрифтом):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout

    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=".OrderFragment">
    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.MaterialComponents.Dark.ActionBar">
        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"

            app:layout_scrollFlags="scroll|enterAlways" />
    </com.google.android.material.appbar.AppBarLayout>
    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="This is the order fragment" />
    </androidx.core.widget.NestedScrollView>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

При запуске приложения в макете `MainActivity` отображается фрагмент `OrderFragment`. Панель инструментов отображается у верхнего края экрана. Когда вы прокручиваете экран вверх, панель инструментов уходит с экрана за верхний край.

Поздравляем! Вы узнали, как создать панель инструментов, которая реагирует на прокрутку. После следующего упражнения вы узнаете, как преобразовать панель инструментов с прокруткой в панель, которая сворачивается и разворачивается при прокрутке экрана пользователем.

Коммит

- git add .
- git commit -m "панель с прокруткой"